HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr

Danilo Morgado Anglés

Konfigurieren einer Hardware-plattform zur Analyse von Android Schadsoftware

Mittweida, 2016

Fakultät Angewandte Computer- und Biowissenschaften

BACHELORARBEIT

Konfigurieren einer Hardware-plattform zur Analyse von Android Schadsoftware

Autor:
Herr Danilo Morgado Anglés

Studiengang:
Angewandte Informatik - IT-Sicherheit

Seminargruppe:
IF13wI-B

Erstprüfer:
Prof. Dr. rer. nat. Christian Hummert

Zweitprüfer:
Prof. Dr.-Ing. Wilfried Schubert

Einreichung:
Mittweida, 07.06.2016

Verteidigung/Bewertung:
Mittweida, 2016

Zusammenfassung

Android ist eine Plattform und kommt auf den unterschiedlichsten mobilen Endgeräten wie Smartphones, Tablets und vielen anderen zum Einsatz. Aufgrund des prosperierenden Marktes für die auf Android zugeschnittenen Geräte, erlangt diese Plattform eine immer größere Bedeutung im Rahmen der Informationssicherheit. Da die Leistungsfähigkeit der mobilen Endgeräte perpetuell expandiert, nimmt auch die Gefahr zu, dass diese kompromittiert und so durch kriminelle Handlungen zweckentfremdet werden. Zum Schutz vor solchen Bedrohungen ist eine Untersuchung über online Anbieter oder Sicherheitssuiten möglich. Während sich die statische Analyse mit der Mustersuche befasst, wird bei der dynamischen Analyse das Verhalten eines Schadprogramms geprüft. Die Methode der Verhaltensanalyse muss im laufenden Betrieb geschehen, was üblicherweise durch Software Emulation realisiert wird. Die virtuelle Umgebung kann jedoch von Schadprogrammen eruiert werden und um einer Erkennung zu entgehen, führen diese ein eigens dafür programmiertes Verhalten aus. Die vorliegende Arbeit befasst sich mit der Hardwareplattform Wandboard und deren Konfiguration, um im späteren Verlauf Android Schadsoftware zu analysieren. Dabei sollte das Hauptaugenmerk auf die Schadsoftware gelegt werden, die eine Emulation erkennen kann. Dazu werden zunächst die Grundlagen zur Hardware Wandboard und zur Plattform Android gegeben. Um Schadprogramme einer detaillierten Analyse zu unterziehen, muss das Verständnis von Betriebssystem Abläufen sowie den genutzten Dateisystemen mit möglichen Speicherorten, apodiktisch sein. Die Software basierenden Werkzeuge müssen bekannt sein, um eine ubiquitäre Einführung entwickeln zu können. Diese kann dann als Leitfaden für aufbauende Arbeiten zur Disposition stehen. Anhand einer Schadsoftware wird exemplarisch gezeigt, wie eine Analyse der Netzwerkkommunikation eines Android Schadprogramms bewerkstelligt werden kann. Nach einem Überblick wird artikuliert, dass die Hardwareplattform Wandboard eine sehr gute Wahl für eine Analyse von Android Schadsoftware darstellt und die Evaluation der Hardware in demselben Maße von Bedeutung ist, wie die der Software.

Abstract

Android is a platform and it is used in a wide range of mobile devices such as smartphones, tablets and many more. By virtue of the prosperous market for costum Android devices, this platform has gained increasing importance in the context of information security. Since the performance of the mobile devices expands perpetually, the risk is growing up that this will be compromised and so misused by criminal acts. To be protected against such threats an investigation via online providers or security suites is possible. While static analysis deals with the search of patterns, in dynamic analysis the behavior of a malicious software is checked. The method of behavioral analysis must be done during runtime which is usually realised by software emulation. Virtual environment can be determined by malicious software and in order to escape a determination they can perform a specially programmed behavior. This thesis deals with the hardware platform Wandboard and its configuration for a later analysis of Android malicious software. The main focus should be placed on the malicious software that can detect an emulation. First of all, the basics of the hardware Wandboard and the platform Android are given. To undergo detailed analysis of malicious programs, understanding the operating system processes and the used file systems with their possible memory locations must be apodictic. Software-based tools must be known to develop an ubiquitous introduction. This can then be a guide for constructive work. Based on a malicious software it is exemplary shown how an analysis of network communications for Android malicious software can be accomplished. After an overview it is pointed out that the hardware platform Wandboard is a very good choice for an analysis of Android malware and that the evaluation of the hardware is as important as the evaluation of the software.

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADB	*Android Debug Bridge*
API	*Application Programming Interface*
ARM	*Advanced RISC Machines*
CPU	*Central Processing Unit*
DDR	*Double Data Rate*
DHCP	*Dynamic Host Configuration Protocol*
DNS	*Domain Name System*
DVI	*Digital Visual Interface*
DVM	*Dalvik Virtual Machine*
EDM	*Embedded-Design-Modul*
E-Mail	*Electronic Mail*
GB	*Gigabyte*
GNU GPL	*GNU General Public License*
GPIO	*General Purpose Input/Output*
GPU	*Graphics Processing Unit*
HDMI	*High Definition Multimedia Interface*
I^2C	*Inter-Integrated Circuit*
IDE	*Integrated Development Environment*
IPC	*Inter Process Communication*
ISP	*Internet Service Provider*
IT	*Informationstechnik*
JVM	*Java Virtual Machine*
KB	*Kilobyte*
LAN	*Local Area Network*
LPDDR	*Low Power Double Data Rate*

MB	*Megabyte*
MMU	*Memory Management Unit*
NDK	*Native Development Kit*
NTP	*Network Time Protocol*
OS	*Operating System*
RAM	*Random Access Memory*
RISC	*Reduced Instruction Set Computing*
ROM	*Read Only Memory*
SATA	*Serial AT Attachment*
SBC	*Single Board Computer*
SDK	*Software Development Kit*
SD-Karte	*Secure Digital Memory Card*
SMS	*Short Message Service*
SPI	*Serial Peripheral Interface*
SSID	*Service Set Identifier*
S/P DIF	*Sony/Philips Digital Interface*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
URL	*Uniform Resource Locator*
USB OTG	*Universal Serial Bus On-The-Go*
VM	*Virtual Machine*
WLAN	*Wireless Local Area Network*
WWW	*World Wide Web*

1 Einleitung

Smartphones sind nicht nur ein Trend der aktuellen Epoche, sondern inzwischen auch fester Bestandteil des privaten und beruflichen Umfelds des Großteils der Menschheit. Gemäß den Hochrechnungen des Marktforschungsinstituts Gartner wurden im vierten Quartal 2014 weltweit rund 1,9 Milliarden Mobiltelefone verkauft, das sind 3,9 % mehr als im gleichen Zeitraum des Vorjahres, wovon allein rund 1 Milliarde das Android Betriebssystem nutzen [I-Gart15]. Im Vergleich zum traditionellen Mobiltelefon verfügen Smartphones über deutlich leistungsstärkere Hardware und höhere Speicherkapazität, was die Nutzung als digitalen Assistent in jeder Nische erdenklich macht. Durch die Bereitstellung zusätzlicher Anwendungen kann der Funktionsumfang des Smartphones erweitert werden, einzig das verwendete Betriebssystem mit offener Schnittstelle zur Softwareentwicklung ist dafür maßgebend. Die vielseitigen Einsatzmöglichkeiten des Smartphones zur Verwaltung von Kontakten, Terminen und E-Mails, die Bearbeitung von Dokumenten und Bildern sowie die Beratung bei der Navigation, lassen Smartphones zu einer wachsenden Bedeutung im Bereich der IT-Sicherheit führen. Aufgrund der Marktstellung von Android, mit einem 80,7 prozentigem Anteil im vierten Quartal 2014, sind vorrangig Geräte von besonderer Wichtigkeit, welche auf diesem Betriebssystem basieren [I-Gart15].

1.1 Problemstellung

Derzeit existieren eine Menge Anbieter im Netz, welche Dateien oder Internetseiten auf bösartige Software untersuchen. Die Mehrheit dieser Anbieter vertraut bei der Suche nach Signaturen in den zu untersuchenden Objekten, nach bereits bekannten Programm Routinen die auf Schadsoftware schließen lässt. Eine weitere Form der Suche bieten Browser-basierende Scanner, die eine zu untersuchende Anwendung ausführen und deren Verhalten als auswertbare Protokolldatei präsentieren. Hierbei wird außer Acht gelassen, dass die gegenwärtige Schadsoftware erkennen kann, ob diese in einem Emulator gestartet worden ist, beziehungsweise auf einem dafür vorgesehenem Gerät wie einem Smartphone, zur Ausführung gekommen ist oder nicht. So ist es der Schadsoftware möglich, durch eine andere Hardwareumgebung ein abwegiges Verhalten hervorzurufen, um das Resultat der Prüfung zu verfälschen.

An dieser Stelle setzt dieses Projekt an und hat zum Ziel, eine Hardwareplattform zu schaffen, die einer Anwendung vortäuscht, auf einer bestimmten Zielplattform wie beispielsweise einem Samsung Galaxy S6 Edge zu laufen, ohne als Emulator enttarnt werden zu können. Dabei soll die Anwendung parallel auf mehreren Wandboards unter vortäuschen verschiedener Geräteplattformen ausgeführt werden, um somit ihr Verhalten zu protokollieren und zu analysieren.

Nachdem der Benutzer über eine Onlineschnittstelle seine zu prüfende Anwendung hochgeladen hat, wird diese Anwendung im automatisierten Verfahren auf dem Wandboard mit der spezifizierten Geräteumgebung installiert, ausgeführt und im laufenden Betrieb analysiert. Weiterhin kann das Verhalten der Anwendung protokolliert und sämtliche Änderungen am System rekonstruiert werden. Diese Informationen sollen dann in Form eines Berichts, dem Anwender am Frontend präsentiert werden.

1.2 Zielsetzung

Die vorliegende Arbeit befasst sich im Rahmen der Aufgabenstellung mit der Konfiguration der Hardwareplattform Wandboard zur Analyse von Android Schadsoftware. Dazu werden technische Grundlagen vorgestellt und ein Überblick über die aktuellen Softwarewerkzeuge gegeben.

Das Hauptziel ist aber, dass die Funktionalität des Wandboards zur Analyse von Android Schadsoftware hergestellt wird. Dafür ist es notwendig das Betriebssystem Android in den gängigen Versionen als SD-Karten Speicherabbild auf dem Wandboard Einsatz finden zu lassen, das Anwendungen auf dem Wandboard auch ohne einen Markt bereitgestellt und das ausführbare Dateien im RAW Format gestartet und genutzt werden können.

Weiterhin soll es möglich sein, das Wandboard über Fernzugriff zu steuern und mittels Datenaustausch die zu analysierenden Anwendungen auf der Plattform bereitzustellen. Hierfür ist ein Ansatz zu erarbeiten, der es im späteren Verlauf der Umsetzung vollautomatisch gestattet, Anwendungen auf dem Wandboard zu installieren und zu investigieren. Dieser Ansatz bildet die essentielle Basis für sämtliche weiterführenden Arbeiten an der Hardwareplattform Wandboard.

Folgend soll ein Schadprogramm auf dem Wandboard zur Ausführung kommen, das im laufenden Betrieb über ein Fremdsystem via „Man-In-The-Middle“ abgehört wird, um dessen Kommunikation aufzuzeichnen.

Die Bachelorarbeit hat zum Ziel, eine systematische und somit ordnungsgemäße Schritt-für-Schritt Anleitung zu schaffen, um weiterführende Arbeiten am Android basierenden Wandboard, besonders im Zusammenhang mit der Analyse von Schadsoftware, zu ermöglichen. Demnach soll keine abstrakte Auflistung von Durchführungsmethoden für den Verwender des Wandboards erfolgen, sondern vielmehr ein Vademekum mit praxisnahen Handlungsanweisungen zur Verfügung stehen.

1.3 Aufbau der Arbeit

Die Bachelorarbeit untergliedert sich in insgesamt fünf Kapitel, beginnend mit der Einleitung, in der auf die Problemstellung und Zielsetzung eingegangen wird.

Das Kapitel Grundlagen umfasst die Theorie, die zum Verständnis der Bachelorarbeit relevanten Themengebiete notwendig ist. Es gibt den aktuellen wissenschaftlichen Forschungsstand zur Thematik wieder und beschreibt die Darstellung der genutzten Hardware sowie der verwendeten Software und der Werkzeuge zur Installation und Analyse von Anwendungen auf der Hardwareplattform. Eine besondere tiefgehende Beleuchtung gilt dem genutzten Betriebssystem Android, hinsichtlich fundamentaler Aspekte der Administration und dem Verständnis der Systemabläufe. Insgesamt sollen mit der Einführung ausschlaggebende Fragen zur Architektur von Android-Smartphones geklärt werden.

Für den Leitfaden zur Konfiguration der Hardwareplattform werden im Kapitel Methoden sämtliche Schritte einzeln und dokumentarisch beschrieben und erklärt. Die Stärken und Schwächen eines jeden Ansatzes werden betrachtet und schließlich als Grundlage des nächsten Kapitels verwendet.

Alle zuvor erarbeiteten Schrittanleitungen werden im Kapitel Ergebnisse präsentiert. Dies beinhaltet die Evaluierung der Beispielszenarien und die beste zu wählende Methode.

Abgeschlossen wird die Bachelorarbeit mit dem Kapitel Diskussion, in dem abermalig die Problemstellung und Zielsetzung aufgegriffen, das erreichte Ergebnis überprüft und die Arbeit in den Kontext gestellt wird. Ferner wird ein persönliches Fazit hinsichtlich der Konfiguration der Hardwareplattform basierend auf Android offeriert und ein Ausblick auf mögliche Weiterentwicklungen rundet das zugrunde liegende Thema ab.

2 Grundlagen

Dieses Kapitel befasst sich mit den Grundlagen der für die Bachelorarbeit relevanten Aspekte. Smartphones und deren Stellung zum traditionellen Mobiltelefon werden dabei kurz in 2.1 erläutert. In Kapitel 2.2 werden die technischen Details und die Konfiguration der Hardwareplattform Wandboard dargelegt. In 2.3 werden die Grundprinzipien von Android benannt und tief in die Materie des Android Betriebssystems eingetaucht. Indessen geht es nicht um die reine Dokumentation von Android, der Blick wird vielmehr auf die grundlegenden Themen der Systemarchitektur und der internen Abläufe dieser Plattform gerichtet. Zusätzlich wird ein Überblick über die verwendeten Werkzeuge und eine Einführung in den Umgang mit diesen im Kapitel 2.4 gegeben. Abschließend, folgen in 2.5 die Grundlagen zu der Schadsoftware.

2.1 Smartphones

Eine anerkannte und allgemeingültige Definition des Begriffs Smartphones ist nicht zu finden, da sich je nach Autor der Fokus auf einzelne Eigenschaften des Gerätes bezieht.

Der Duden definiert Smartphone als „Mobiltelefon mit Touchscreen und zusätzlichen Funktionen wie GPS und der Möglichkeit, Apps darauf zu installieren.“ [I-Dude16]. Eine erweiterte Definition liefert das Gabler Wirtschaftslexikon, darin heißt es „Mobiltelefon mit erweitertem Funktionsumfang. Dazu zählen neben der Telefonie und Short Message Service (SMS) üblicherweise Zusatzdienste wie Electronic Mail (E-Mail), World Wide Web (WWW), Terminkalender, Navigation sowie Aufnahme und Wiedergabe audiovisueller Inhalte. Auf Smartphones laufen gegenüber herkömmlichen Mobiltelefonen komplexere Betriebssysteme wie etwa Symbian OS, Blackberry OS oder das iPhone OS. Die hierdurch geschaffene Möglichkeit zur Installation weiterer Anwendungen durch den Endnutzer verleiht Smartphones einen erweiterbaren und individualisierbaren Funktionsumfang.“ [I-Gabl16].

Die wesentlichsten Unterschiede liegen im Betriebssystem, welches seinem Endnutzer die Möglichkeit gibt Anwendungen nachzuinstallieren, und einer einem Personal Computer gleichendem leistungsstärkeren Hardware, die so diese Geräte zu Alleskönnern machen.

Die größten technischen Unterschiede, betreffend der Hardware zum konventionellen Mobiltelefon, sind die stark erhöhte Displaygröße mit hoher Auflösung, so das Webseiten, Bilder und Videos detailreich wiedergegeben werden können, die kontinuierliche technische Weiterentwicklung der großen berührungsempfindlichen Displays sowie der Ausbau und die Weiterentwicklung des Mobilfunknetzes für die neuen Kommunikationsschnittstellen der Geräte. Dieser Fortschritt ermöglicht diesem Wunderwerk einen Siegeszug im privaten sowie im beruflichen Umfeld.

2.2 Wandboard

Die Open-Source-Entwicklungsplattform mit Hochleistungs-Multimedia-Fähigkeiten ist basiert auf der ARM Cortex-A9-Architektur. Es ist ein kostengünstiges und Strom schonendes Board bestehend aus einem Kernmodul und einer einfachen Schnittstellenkarte. Auf dem Wandboard arbeitet ein Freescale i.MX6 Cortex-A9 Prozessor, dem je nach gewählter Hardwareversion 512 MB/1 GB/2 GB DDR3-RAM zur Verfügung stehen. Das Board besitzt Gigabit-Ethernet, HDMI, Kamera Anschluss, optisches S/P DIF, Audio Anschluss, USB sowie USB-On-The-Go-Anschluss, zwei Mikro-SD-Karteneinschübe und eine serielle Schnittstelle. Je nach Hardwareversion steht noch 802.11n-WLAN, Bluetooth und ein SATA-Anschluss zur Verfügung und vervollständigen die Ausstattung. An der Kernel-Unterstützung für das Wandboard arbeitet die Linux-Gemeinschaft. Das Wandboard verwendet ein Embedded-Design-Modul (EDM) als Standard Anschluss, jenes ist ein 314 Pin-Anschluss der alle Signale überträgt die von HDMI auf Gigabit-Ethernet übertragen werden, um Pins über GPIO, I²C, SPI und der seriellen Schnittstelle zu steuern. Der Pin-zu-Pin-Standard gilt gleichermaßen für x86- und ARM-Systeme und schafft damit neue Kombinationsmöglichkeiten. Der Embedded-Design-Modul Standard findet Verwendung für Embedded-Anwendungen und Steuerungen, die von vielen Embedded-Unternehmen aktiv entwickelt werden. [I-WanA16]

Mit dem Wandboard und dem entsprechenden Entwicklungstools können Embedded-Linux-Speicherabbilder installiert und von diesen gebootet werden. Mit dem Open-Source-Entwicklungs-Tool kann weiterhin über dem Desktop ordnungsgemäß gepatcht, upgedatet, konfiguriert und rebuildet werden. Bis auf vorkonfigurierte SD-Karten-Speicherabbilder, hat diese jedoch keine Bedeutung und wird in dieser Arbeit nicht weiter beleuchtet. [I-WanA16]

2.2.1 Technische Details

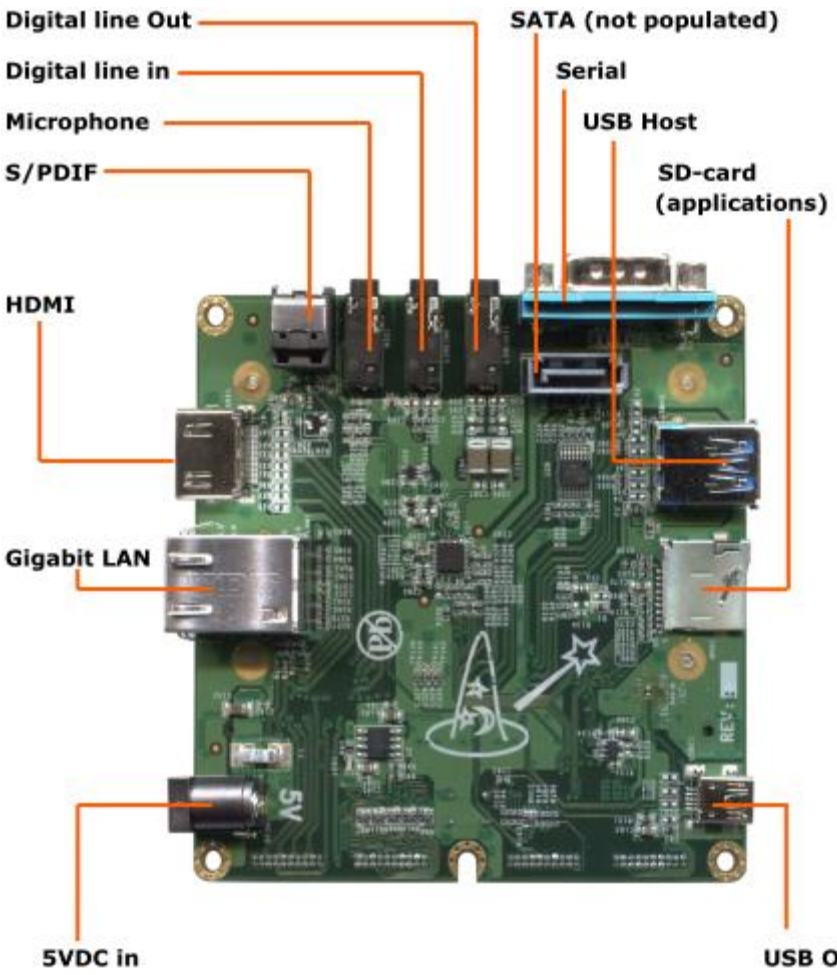
Das Wandboard steht in drei technisch verschiedenen Versionen zur Verfügung:

	Wandboard SOLO	**Wandboard DUAL**	**Wandboard QUAD**
Prozessor	NXP i.MX6 Solo	NXP i.MX6 DualLite	NXP i.MX6 Quad
Kerne	Cortex-A9 Single	Cortex-A9 Dual	Cortex-A9 Quad
GPU	Vivante GC 880 + Vivante GC 320	Vivante GC 880 + Vivante GC 320	Vivante GC 2000 + Vivante GC 355 + Vivante GC 320
RAM	512 MB DDR3	1GB DDR3	2GB DDR3
AUDIO	YES	YES	YES
S/P DIF	YES	YES	YES
HDMI	YES	YES	YES
Kameraanschluss	YES	YES	YES
Mikro-SD Einschub	2	2	2
Serialport	YES	YES	YES
Expansion Header	YES	YES	YES
USB	YES	YES	YES
USB OTG	YES	YES	YES
SATA	-	-	1
LAN	Gigabit	Gigabit	Gigabit
WLAN	-	802.11n	802.11n
Bluetooth	-	YES	YES
Preis	79 $	99 $	129 $

Tabelle 2.2.1-1 Wandboard i.MX6 Spezifikation [I-WanB16]

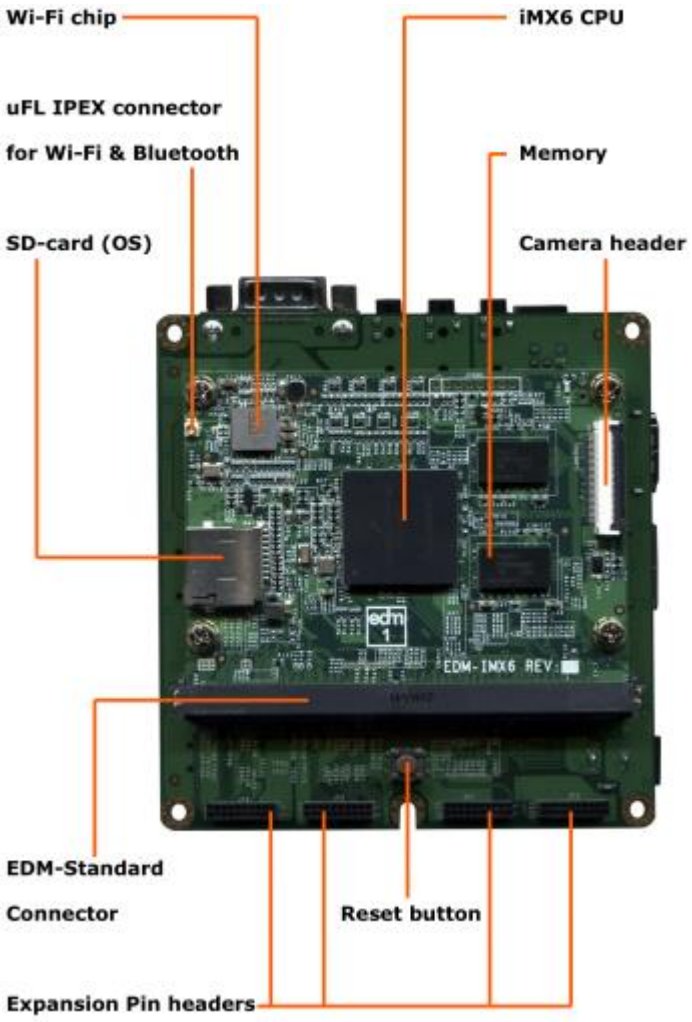
Module

Die ***Schnittstellenkarte*** des Wandboard Quad ist in der folgenden Abbildung dargestellt. Sie dient dem Wandboard als Anschluss sämtlicher Peripherie, mit Ausnahme der Mikro SD-Karte welche zum Booten des Betriebssystems erfordert wird. Wie aus der Tabelle 2.2.1-1 Wandboard i.MX6 Spezifikation [I-WanB16] in Spalte Wandboard QUAD beschrieben, können die Anschlüsse der folgenden Abbildung entnommen werden.



2.2.1-1 Beschriftung zur Wandboard Schnittstellenkarte [I-WanC16, Seite 6]

Das ***Kernmodul*** ist in der folgenden Abbildung dargestellt. Es ist in die Schnittstellenkarte eingesteckt und mit Schrauben fixiert. Zusätzlich befindet sich ein CPU-Kühlgitter über dem i.MX6 CPU zur Wärmeabgabe, dieses wurde auf der Abbildung weg gelassen, um alle Komponenten auf dem Modul zu sichten. Die einzige für dieses Projekt benötigte Peripherie dieses Moduls, ist die Mikro SD-Karte die das Betriebssystem für das Wandboard bereitstellt.



2.2.1-2 Beschriftung zum Wandboard Kernmodul [I-WanC16, Seite 5]

Netzteil

Das Netzteil ist beim Wandboard separat und der Lieferung beigepackt. Es wird ans Stromnetz angeschlossen, transformiert den Strom von 100 Volt bzw. 220 Volt auf 5 Volt und verteilt den Strom über die Schnittstellenkarte an alle Bauteile. Das Netzteil liefert nach der Transformation 3 Ampere. Beim Wechsel dieses Netzteils sollte darauf geachtet werden, dass die Versorgungsleistung gleich bleibt, da sich dies beim Einsatz eines Netzteils mit geringerer Leistung in Schreib- / Lesefehlern auf der SD-Karte, Fehlfunktionen des Wandboards, bis hin zum Einfrieren des gesamten Systems äußern kann.

Netzwerkkabel

Ein Netzwerkkabel mit RJ45 Stecker ist der Lieferung nicht beigepackt. Es ist nötig um das Wandboard mit einem Kabelnetzwerk zu verbinden. Die Konfiguration dieser Schnittstelle zur Internetverbindung würde durch den DHCP Service eingerichtet, welcher bei normalen Heim Routern der Standard ist. Im Fall eines Firmen-, Hochschul- oder Universitätsnetzwerkes können weitere Schritte zur Einrichtung der Internetverbindung notwendig sein.

HDMI

Die Videoausgabe des Wandboards erfolgt über HDMI Adapterkabel auf einem HDMI Display oder via HDMI auf DVI Adapterkabel zu einem Gerät mit DVI Eingang. Das Adapterkabel für die Videoausgaben ist nicht Teil der Wandboard Lieferung. Die Anschlussmethode über HDMI hat den Vorteil, dass das Audiosignal zusammen mit dem Videosignal übertragen wird, was es somit zu einer idealen Anschlussart für Multimedia Anwendungen macht. Sollte das DVI Adapterkabel Verwendung finden, muss das Audiosignal zusätzlich vom Wandboard mittels 3,5 mm Klinkenkabel abgegriffen werden.

WLAN Antenne

Bei der Hardwareplattform Wandboard Duallite und Wandboard Quad wird eine WLAN-Antenne mitgeliefert. Sie bündelt elektromagnetische Wellen, die beim Senden und Empfangen abgestrahlt werden. Die Position zum WLAN-Router ist maßgebend für den Einsatz dieser Antenne und kann bei kurzen Distanzen weggelassen werden.

Gehäuse

Das mitgelieferte Wandboard Gehäuse dient dazu, das Kernmodul und die Schnittstellenkarte in sich aufzunehmen, damit diese nicht ungeschützt, äußerem Einfluss ausgesetzt werden. Im Fall dieser Arbeit wurde durch den ständigen physischen Zugriff auf die Mikro SD-Karten, nur das untere Gehäuse genutzt und das Board ohne Befestigung hineingelegt.

Weitere Komponenten

Je nach Hardwareversion des Wandboards und Einsatzzweck kann es erforderlich sein, weitere Komponenten zu verwenden. Aufgrund der Tatsache, dass der 5 Volt Eingang des Wandboards mit 3 Ampere abgesichert ist, welches das gesamte Wandboard versorgt, wird schnell klar, dass der USB-Port auf eine gewisse Anzahl passiver USB-Hubs beschränkt ist. Es ist zu empfehlen beim Anschluss von Geräten mit höherem Strombedarf, diese an einem aktiven USB-Hub zu betreiben. Weiterhin besitzt das Wandboard um mit einem Rechner verbunden und gesteuert werden zu können, ein USB OTG Anschluss, der mit einem Mikro USB auf USB-Kabel betrieben wird, damit sich Daten austauschen lassen. Bei der Hardwareversion Wandboard Quad ist die Nutzung einer SATA-Schnittstelle möglich, um Datenaustausch mit Festplatten und anderen Speichergeräten durchzuführen. Für das Betreiben der Schnittstelle ist ein SATA-Kabel mit SATA-Stecker notwendig. Weiterhin wird für die erste Inbetriebnahme des Wandboards ein Rechner mit SD-Karten Lesegerät und Internetzugriff vorausgesetzt, um die SD-Karte vorzubereiten.

Anschluss des Wandboards

Für den weiteren Verlauf der Arbeit wird angedacht, dass alle Komponenten zur Hand und an das Wandboard Quad angeschlossen sind, jedoch stromlos geschaltet und keine SD-Karte eingesetzt ist.

2.2.2 SD-Karten und Speicherabbilder

2.2.2.1 SD-Karte

Bedingt durch die Aufgabenstellung ist es erforderlich, das Betriebssystem Android über eine SD-Karte zu starten. Eine SD-Karte ist ein Wechseldatenträger der vielfältig zur Speicherung von Daten eingesetzt werden kann. Bei SD-Karten gibt es drei unterschiedliche Größen Mini SD, Mikro SD und die Standard SD. Für das Projekt wurde durch die technische Vorgabe des Wandboards, eine Mikro SD-Karte mit einer Größe von 11 mm × 15 mm verwendet. Am Wandboard existieren zwei Mikro SD-Karten Einschübe. Der Einschub auf dem Kernmodul dient dem Betriebssystem zum Booten. Die Erweiterung des Speichers wird durch den Einschub der Schnittstellenkarte realisiert. Die Größe der Speicherkarte sollte 4 GB nicht unterschreiten, da die vorkonfigurierten Betriebssysteme eine Datenpartition besitzen, die einen Großteil der Speicherabbildgrößen ausmacht. Es ist auch möglich, die Größe der Datenpartition zu reduzieren, um die Limitierung von 4 GB großen SD-Karten einzuhalten. Bei der Auswahl der Speicherkarte sollte auf die Class geachtet werden, die repräsentativ die Geschwindigkeitseinstufung einer SD-Karte beschreibt. Gemäß der SD-5.0-Spezifikation, sind folgende Geschwindigkeitsklassen als minimale Schreibgeschwindigkeiten definiert:

Minimum Sequential Write Speed	Speed Classes		
	Speed Class	UHS Speed Class	Video Speed Class (New)
90 MB/sec			V90
60 MB/sec			V60
30 MB/sec		3	V30
10 MB/sec	10	1	V10
6 MB/sec	6		V6
4 MB/sec	4		
2 MB/sec	2		

2.2.2-1 Übersicht der Geschwindigkeitsklassen bei SD-Karten [I-Sdc16, Seite 4]

Für die Aufgabenstellung wird mindestens die Class 10 benötigt, damit die Speicherkarte nicht zum limitierenden Faktor bei Schreib- und Lesezugriffen im laufenden Betrieb wird. Eine schnellere Schreibgeschwindigkeit, wie Video Speed Class ab V30, ist für spätere Ansätze der Weiterentwicklung wünschenswerter.

2.2.2.2 Speicherabbild

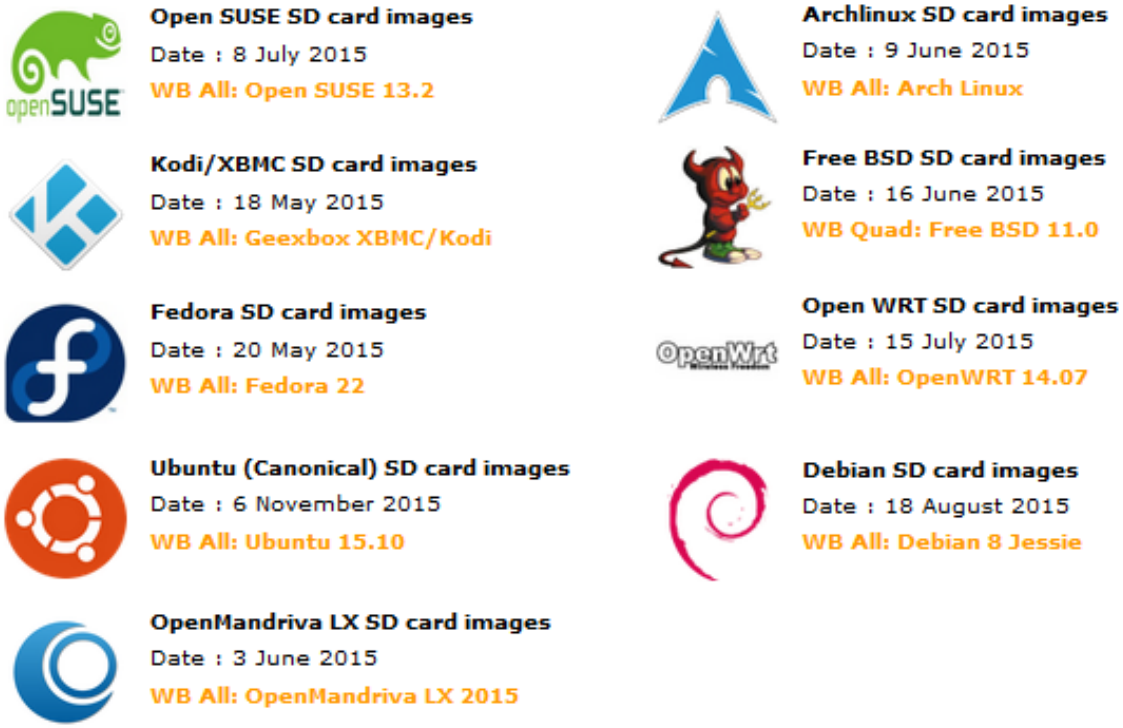
Als Speicherabbild wird eine Datei bezeichnet, die ein komplettes Ebenbild eines Speichermediums ist, inklusive Berechtigungen und Metadaten. Diese Kopie enthält auch Informationen über die Dateisystemstruktur des Originaldatenträgers, einschließlich des Startsektors. Um nun das Android Speicherabbild für das Wandboard zu installieren, muss es zuerst heruntergeladen werden. Es stehen dabei zwei Arten von Betriebssystemabbildern zur Verfügung:

Betriebssystem Speicherabbilder für SD-Karten

2.2.2-2 Betriebssystem Speicherabbilder [I-WanD16]

Diese Betriebssystem Speicherabbilder sind vom Hersteller des Wandboards als lauffähige Distributionen angedacht und können genutzt werden. Unter Umständen müssen einige Abbilder vor der Nutzung im Wandboard, an das Speichermedium Mikro SD-Karte angepasst werden. Diese Anpassung wird im späteren Verlauf dieser Arbeit genauer beschrieben.

Betriebssystemabbilder für SD-Karten der Wandboard Gemeinschaft



2.2.2-3 Betriebssystem Speicherabbilder der Wandboard Gemeinschaft [I-WanD16]

Diese Betriebssystem Speicherabbilder sind von der Wandboard Gemeinschaft im Rahmen der freien Entwicklung entstanden und laufen ebenfalls als stabile Distribution auf dem Wandboard.

In dieser Arbeit kommen die Betriebssystem Speicherabbilder Android 5.0 (Lollipop) vom 5. November 2015 [S-WanA16] und Android 4.4 (Kitkat) vom 3. März 2015 [S-WanB16] zum Einsatz. Ebenfalls werden die Betriebssystem Speicherabbilder Android 6.0 (Marshmallow) vom 28. April 2016 [S-WanD16] und Android 4.3 (Jellybean) vom 2. März 2015 [S-WanC16] für das Wandboard angeboten. Die Betriebssystem Speicherabbilder können als ZIP Dateien, im DVD-Pfad **/SD-Karten_Speicherabbilder,** der Projekt-DVD entnommen werden.

Nach dem erfolgreichen Download muss die Datei mittels einem geeigneten Werkzeug entpackt werden, dafür bietet sich die Anwendung 7-zip an, die über die Quelle [S-Szip16] als kostenlose Version zur Verfügung steht. Bei der resultierenden Datei mit Endung „img“, die rund 3,5 GB bei Android 4.4 und 7,5 GB bei Android 5.0 einnimmt, handelt es sich um das Betriebssystemspeicherabbild. Im Laufe dieser Arbeit wird das Abbild auf die vorhandene SD-Karte geschrieben.

2.2.3 Verbindungsaufbau

Um das Wandboard für die Aufgabenstellung richtig nutzen zu können, ist die Fernsteuerung des Systems und der Datenaustausch mit diesem, von substanzieller Bedeutung. Das Wandboard besitzt, wie in 2.2.1 Technische Details beschrieben, einen USB OTG Anschluss zum Koppeln mit einem weiteren System via USB. Eine weitere Möglichkeit ist die Verbindung über Netzwerkkabel oder WLAN. Hierfür muss entsprechend der Methode die richtige Konfiguration gewählt werden. Im Fall der Netzwerkverbindung über Netzwerkkabel, kann bei Verwendung eines DHCP Servers oder Router mit aktivierter DHCP Funktionalität, die Standard Einstellung belassen werden, ansonsten ist die Konfiguration der aktuellen Netzanforderung anzupassen. Die Netzwerkverbindung über WLAN gleicht der Konfiguration über Netzwerkkabel, indes muss das WLAN Modul aktiviert, das WLAN-Netz über die SSID gewählt und bei gesetzten Sicherheitsoptionen ein Passwort eingegeben werden.

Die Verbindung zum Android Betriebssystem auf dem Wandboard wird am Fernsteuerungsrechner per Android Debug Bridge durchgeführt. Funktionen dieser Anwendung werden in 2.4.4 Android Debug Bridge (ADB) dieser Arbeit genauer beschrieben. Damit eine Interaktion zwischen Fernsteuerungsrechner und Android durchführbar ist, muss das Wandboard ordnungsgemäß angeschlossen, eingeschaltet und mit einer Mikro SD-Karte bestückt sein, auf der sich ein lauffähiges Android Betriebssystem befindet, dieses Vorgehen wird in Abschnitt 3.2.1 Vorbereiten der SD-Karten-Speicherabbilder erklärt. Diese Anleitung fungiert gleichermaßen zum erstmaligen Verbindungsaufbau wie auch für weitere Verbindungen zwischen Fernsteuerungsrechner und Android Betriebssystem. Bevor jedoch die Fernsteuerung starten kann, muss das USB-Debugging im Android aktiviert und Wandboard und Fernsteuerungsrechner müssen mit einem Mikro USB auf USB-Kabel verbunden sein. Dieser Teil ist in Abschnitt 3.2.2 Verbinden des Rechnersystems mit Android über ADB beschrieben.

USB-Debugging aktivieren

Die Entwickleroptionen in Einstellungen sind aus Sicherheitsgründen versteckt, da diese tief in das Android System eingreifen. Diese Hürde hat Google jedoch bewusst nicht allzu sicher gestaltet, um Entwicklern nicht die Arbeit zu erschweren. Das Freischalten der Entwickleroptionen ist in Abschnitt 3.2.2 Verbinden des Rechnersystems mit Android über ADB genau beschrieben.

2.3 Android

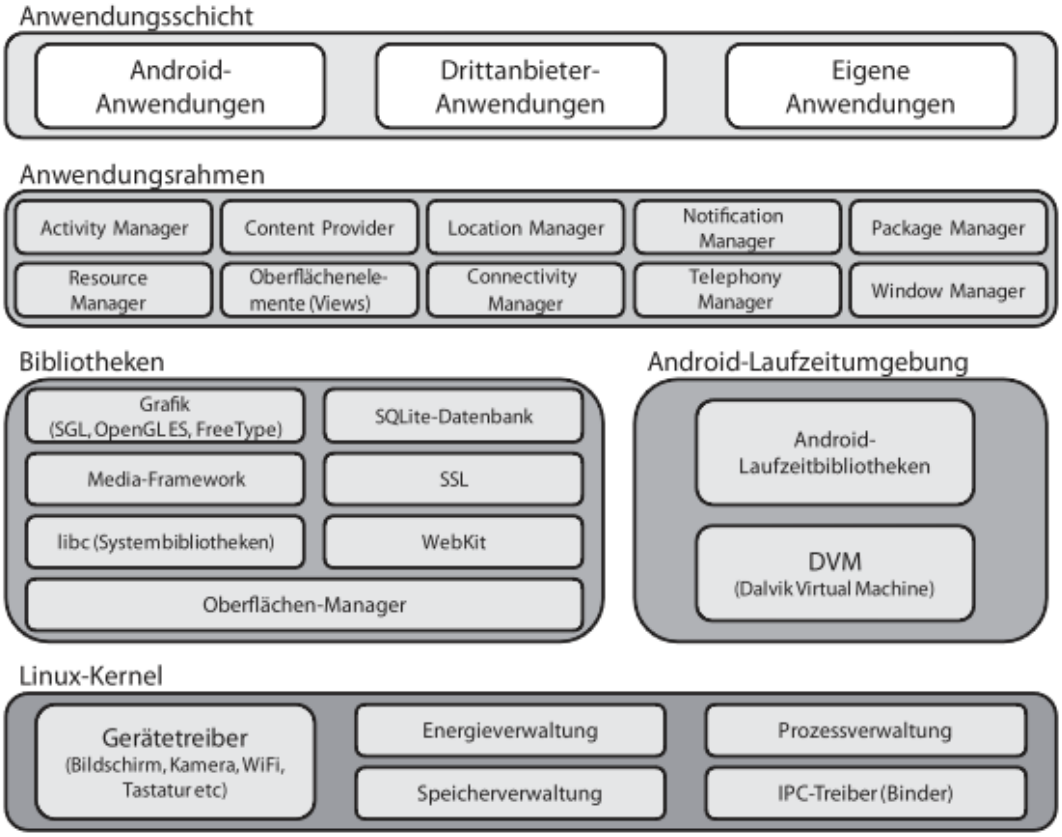
Da die Ressourcen des Wandboards begrenzt sind, wird ein schlankes Betriebssystem benötigt, welches weniger Speicher verbraucht. Android, entwickelt von Google und der Open Handset Alliance, ist die weltweit meist verbreitetste Plattform und kommt bei vielen unterschiedlichen mobilen Endgeräten wie Smartphones, Tablets, Spielekonsolen, Netbooks und E-Book-Readern zum Einsatz. Bei Android handelt es sich um eine freie Software, die quelloffen entwickelt wird. [I-AnSo16] Dieser Teil der Arbeit konzentriert sich auf die Kernthemen von Android, das einen Linux-Kernel enthält, jedoch keine klassische Linux-Distribution ist, Hardwaretreiber zur Verfügung stellt und eine Reihe von freien, quelloffenen Bibliotheken nutzt. Dies bildet die Betriebssystemgrundlage. Die Anwendungen werden in der Programmiersprache Java geschrieben und auf einer von Google entwickelten virtuellen Maschine ausgeführt. Neben den vorinstallierten Anwendungen, können auf dem Android System weitere Programme nachinstalliert werden, die dessen Funktionalität erweitern.

2.3.1 Linux

Linux, ein Unix-ähnliches Betriebssystem, ist die Bezeichnung für ein Betriebssystem-Kernel. Alle Unix-Programme sind ursprünglich in Assembler geschrieben, wurden jedoch in der damals neu entwickelten Programmiersprache namens B, neu programmiert. Im weiteren Verlauf wurde Unix von der wortorientierten Sprache B, in die byte-orientierte Sprache C überführt, da C eine sehr portable und systemnahe Sprache ist. Der Vorteil der damals neuen Programmiersprache C war, dass Programme die in dieser Sprache geschrieben wurden, gut auf neuen Plattformen implementiert werden konnten, um dann auch dort performant zu laufen. Einzig ein Übersetzer für die neue Hardwareplattform wurde benötigt. Aus Unix entwickelten sich dann viele Abkömmlinge, sogenannte Derivate. 1992 wird Linux unter die GNU General Public License (GPL) gestellt. Das Ziel des GNU-Projekts ist die Entwicklung eines komplett freien Betriebssystems. Den Kern dieses Betriebssystems bildet meistens Linux und die wichtigsten Komponenten der Anwender-Software von Linux sind GNU-Programme. Das Linux selbst nur den Kernel umfasst ist nun klar, denn das Ziel war von Anfang an ein völlig freies Unix mit Linux als ersten freien Kernel zu entwickeln. [L-PlWe11, vgl. Seite 53 ff.] Als Linux-Distribution wird somit ein Programm-Paket mit Linux-Kernel, der GNU-Software und eine Reihe von anderen Programmen, die jeder Anwender nutzen kann, bezeichnet.

2.3.2 Architektur

Die grundlegende Architektur des Betriebssystems untergliedert sich in die „Anwendungsschicht“, den „Anwendungsrahmen“, verschiedene „Bibliotheken“, die „Android-Laufzeitumgebung“ und den „Linux-Kernel“ (vgl. Abbildung 2.3.2-1 Die Android-Systemarchitektur [L-BePa10, Seite 15]). Die aufgeführten Ebenen sind für den laufenden Betrieb erforderlich und werden nachfolgend genauer beschrieben.



2.3.2-1 Die Android-Systemarchitektur [L-BePa10, Seite 15]

2.3.2.1 Linux-Kernel

Google verwendet für Android einen modifizierten Linux-Kernel als Fundament des Betriebssystems. Dieser Kernel ist an die Funktionalität von Smartphones angepasst, mittels entfernen unwichtiger Treiber und nutzen von angepassten Treibern. Auch Änderungen an bestimmten Teilen des Kernels waren übliche Anpassungen beim Erstellen dieses Betriebssystems auf Linuxbasis. Es gibt verschiedene Arten für einen Kernel: Monolithkernel, Mikrokernel und Makrokernel. Beim Monolithkernel sind besonders viele Funktionen integriert, was für die Geschwindigkeit vorteilhaft ist jedoch die Komplexität und die Fehleranfälligkeit stark erhöht. Der Mikrokernel lagert so viele Funktionen wie möglich in separate Prozesse aus, was zusätzlicher Kommunikation bedarf. Makrokernel sind Hybride der beiden vorigen Kernels und vereinen die Vorteile dieser beiden Architekturen. Welche Aufgaben nun dem Kernel zugeschrieben werden, hängt von seiner Architektur ab [I-LoLe16]. Da in Android ein Linux-Kernel Verwendung findet, ist die genutzt Architektur monolithisch. Der Kernel dient als Abstraktionsschicht zwischen der Hardware und den darüber liegenden Ebenen, stellt Treiber zur Verfügung und ist für die Energie-, Speicher- und Prozessverwaltung zuständig.

2.3.2.2 Android-Laufzeitumgebung

Die Dalvik Virtual Machine (DVM) bildet den Kern der Android Laufzeitumgebung. Weiterhin besteht die Laufzeitumgebung von Android aus verschiedenen Laufzeitbibliotheken, welche die Kernfunktionalitäten von Java bereitstellen. Anwendungen für Android werden in Java geschrieben, welche zur Ausführung eine virtuelle Maschine benötigen. Google entwarf für diesen Fall eine eigens dafür effiziente und sichere Umgebung, die Dalvik Virtual Machine. Android Anwendungen werden in Java mit dem Java Software Development Kit (SDK) geschrieben und kompiliert. Die Transformation des Java-Bytecode in den DVM-kompatiblen Code übernimmt das dx-Tool, welches im Lieferumfang des Android Development Kit enthalten ist. [L-BePa10, vgl. Seite 15 f.]

2.3.2.3 Bibliotheken

In Android bilden die Bibliotheken die zumeist in C oder C++ programmiert sind, für den Betrieb der Anwendungsschicht und des Anwendungsrahmens die Kernfunktionalität. Diese Bibliotheken sind direkt für die verwendete Hardwarearchitektur kompiliert und laufen nicht auf der virtuellen Maschine, sie stellen erforderliche Funktionen für Datenbank, 3D-Grafik, Webzugriff, Multimedia-Verwaltung und Oberflächengestaltung bereit. Diese Abstraktion macht es möglich, besonders rechenintensive Funktionen auszulagern, um sie schneller ausführen zu können. [L-BePa10, vgl. Seite 16]

2.3.2.4 Anwendungsrahmen

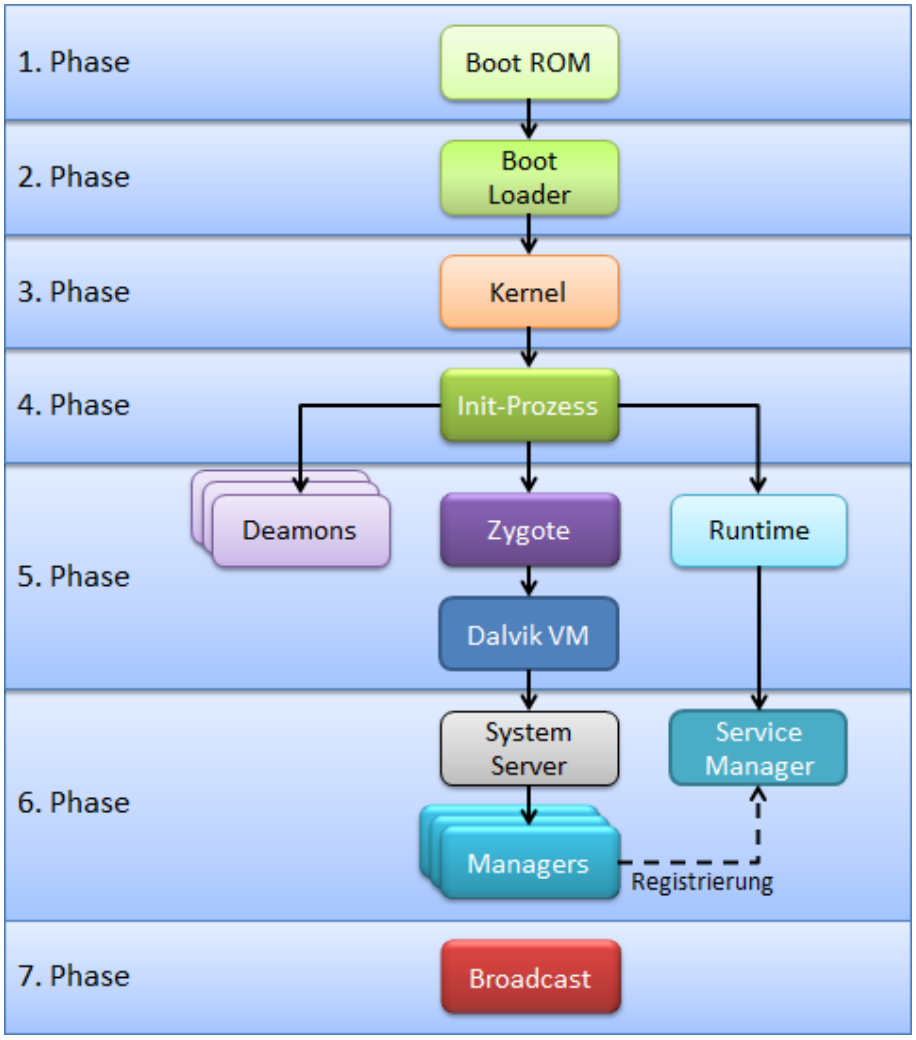
Der Anwendungsrahmen ist für Android-Entwickler die interessanteste Schicht, da diese die Kommunikation zwischen einzelnen Anwendungen sowie zwischen Endanwender und Anwendung realisiert. Diese System-Komponenten stellen Manager-Klassen zur Verfügung, die es erlauben, über bereitgestellte Application Programming Interfaces (APIs) den Zugriff auf die Funktionalitäten der Ressourcen zu erhalten. Mit Hilfe des Anwendungsrahmens werden Hardware und Betriebssystem abstrahiert. [L-BePa10, vgl. Seite 16]

2.3.2.5 Anwendungsschicht

Auf der Ebene der Anwendungsschicht befinden sich die Android Anwendungen, die das Hauptunterscheidungsmerkmal zwischen einem traditionellen Mobiltelefon und einem Smartphone sind. Innerhalb der Anwendungsschicht findet die Kommunikation zwischen Mensch und Maschine sowie Interaktionen zwischen Anwendungen statt. Jede Anwendung bedient sich dabei der darunterliegenden Programmierschnittstelle. Anwendungen in Android setzen sich aus bis zu vier verschiedenen Komponenten zusammen. Diese sind Activities, Services, Content Provider und Broadcast Receiver. Die Activities dienen der Oberflächendarstellung (View), die Services dienen als Hintergrundprozesse (Controller) und die Content Provider verwalten die Daten (Model). Das Entwurfsmuster Model-View-Controller strukturiert die Software für den flexiblen Programmentwurf. Die Broadcast Receiver stellen die Verbindung zum System her, welche durch Intents kommunizieren, die Objekte zur Nachrichtenübermittlung sind. [L-BePa10, vgl. Seite 16 ff.]

2.3.3 Bootvorgang

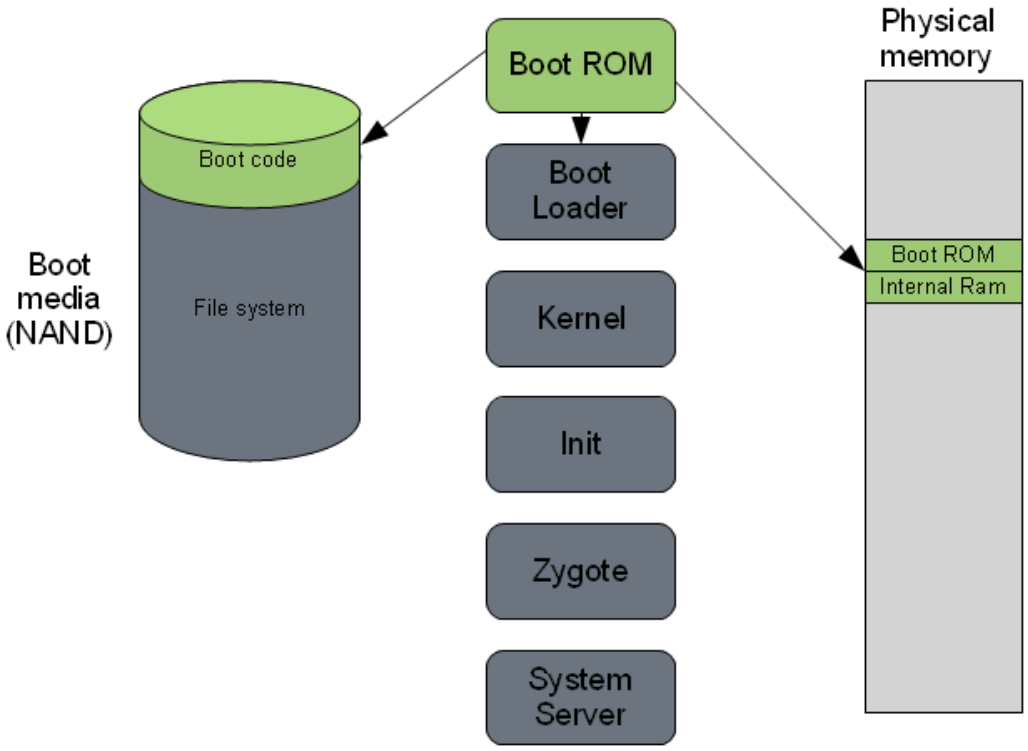
Der Android Bootvorgang, der einen wesentlichen Unterschied zu Desktop-Systemen vorweist, wird in seinen sieben Phasen beginnend vom Einschalten der Spannungsquelle, bis hin zur Android Benutzeroberfläche nachfolgend detaillierter beschrieben [L-Hoog11, vgl. Seite 49 ff.].



2.3.3-1 Phasen der Android Boot Sequenz

2.3.3.1 Phase 1 – Einschalten und Boot-ROM-Code

Mit dem Druck auf die Einschalt-Taste (Power-Button) wird die CPU mit Strom versorgt. Nach dem Einschalten befindet sich die CPU im nicht initialisierten Zustand, mit nicht eingerichteter Uhr, einzig das interne RAM ist verfügbar. Bei stabiler Stromversorgung wird der Boot-Code im Boot-ROM ausgeführt, der sich der Lokalisierung des Bootmediums annimmt. Typischerweise wird als Bootmedium der interne Flash-Speicher genutzt, da dieser nach Abschaltung der Stromzufuhr bestehen bleibt. Im Fall des Wandboards ist dies die Mikro SD-Karte, die in das Kernmodul eingesteckt ist.



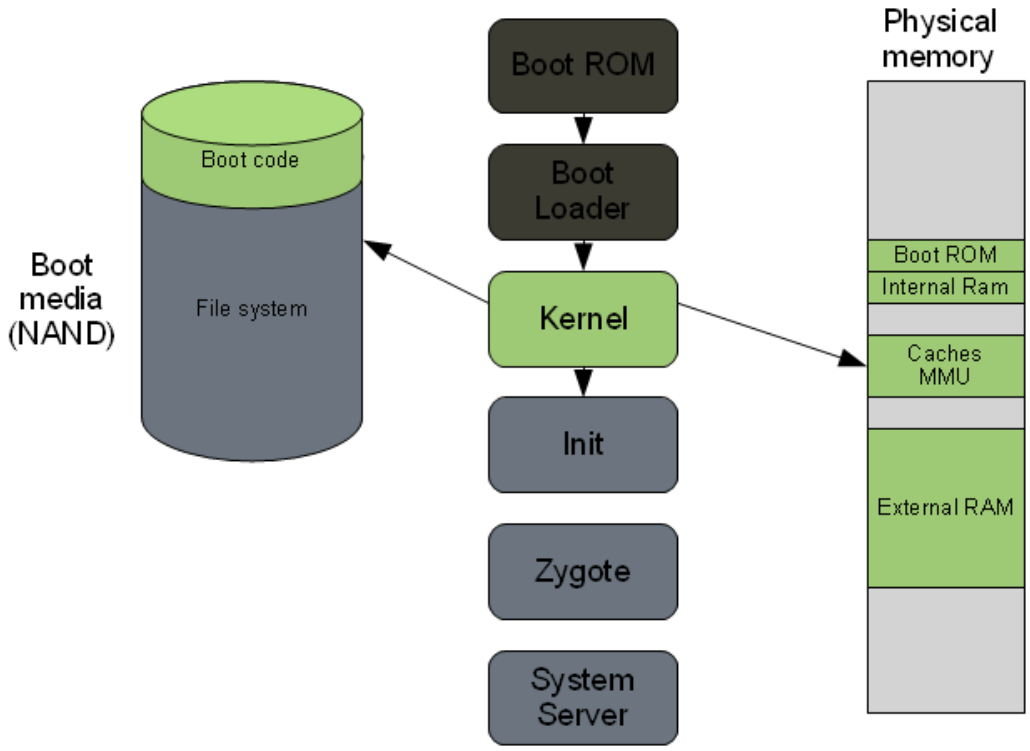
2.3.3-2 Phase 1 der Android Boot Sequenz [L-Hoog11, Seite 51]

2.3.3.2 Phase 2 – Boot Loader

Die Phase des Boot Loaders unterteilt sich in zwei Teile:

Der **primäre Boot Loader** ist ein kleiner Programmteil, der das externe RAM erkennt, einrichtet und mit dieser Hilfe den sekundären Boot Loader lädt.

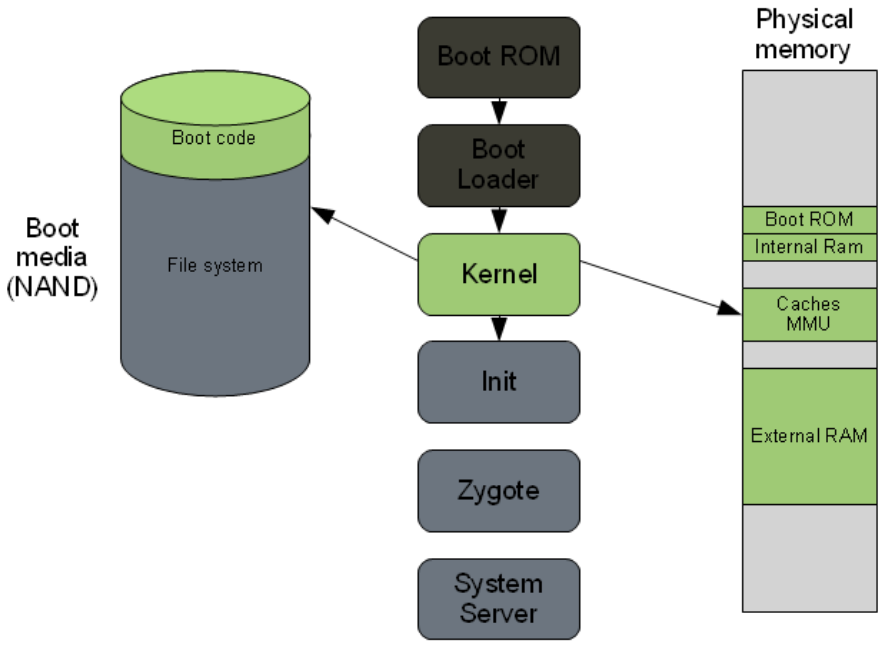
Der **sekundäre Boot Loader** bietet die Möglichkeiten an, einen standardmäßigen Bootvorgang durchzuführen oder einen alternativen Bootmodus auszuwählen. Der standardmäßige Bootvorgang ist für die Einrichtung des Dateisystems, für den zusätzlichen Speicher, die Netzwerkunterstützung und andere Funktionen zuständig. Nach diesem Schritt wird versucht, den Kernel in das externe RAM zu laden, Boot-Parameter zu setzen und einen Sprung auf dem Linux-Kernel durchzuführen.



2.3.3-3 Phase 2 der Android Boot Sequenz [L-Hoog11, Seite 52]

2.3.3.3 Phase 3 – Kernel

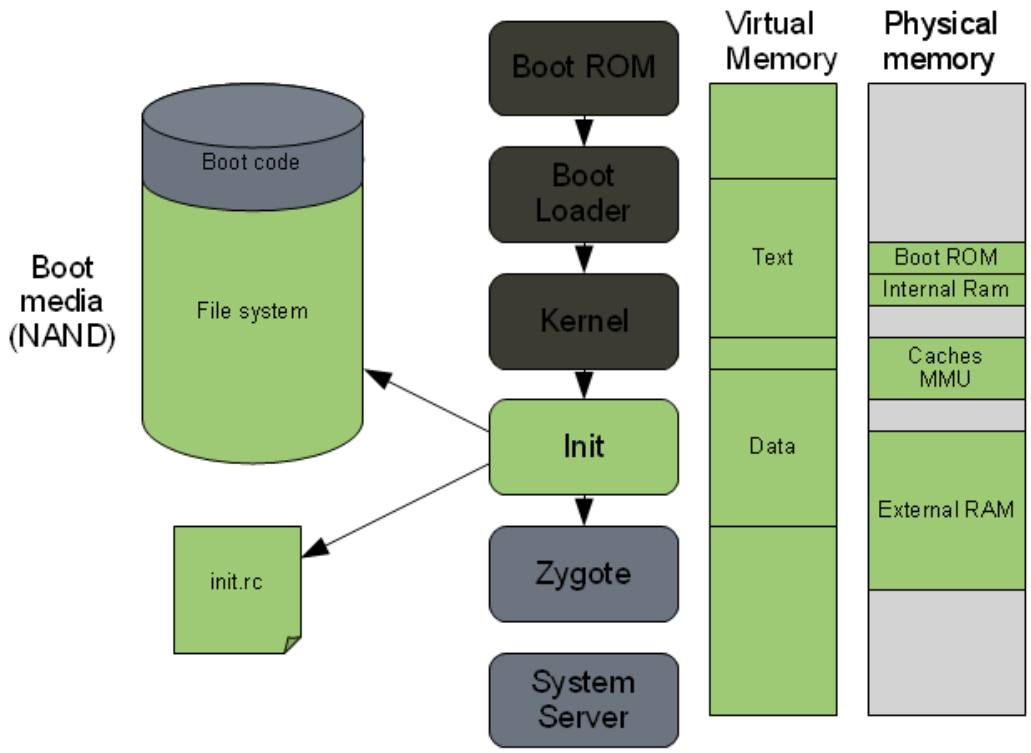
Ab dieser Phase übernimmt der Kernel die Systemverantwortung und ist zuständig für die System-Ressourcen, er eignet sich die Kontrolle über die Hardware an. Es werden weitere Komponenten wie Interrupt-Controller, Speicherschutz und Prozessplaner eingerichtet, um das System zu betreiben. Sind die Caches und Memory Management Units (MMU) eingerichtet, können Prozesse gestartet werden. Weiter wechselt der Kernel in das Root Verzeichnis und startet den init Prozess als Initial Prozess.



2.3.3-4 Phase 3 der Android Boot Sequenz [L-Hoog11, Seite 52]

2.3.3.4 Phase 4 – Init-Prozess

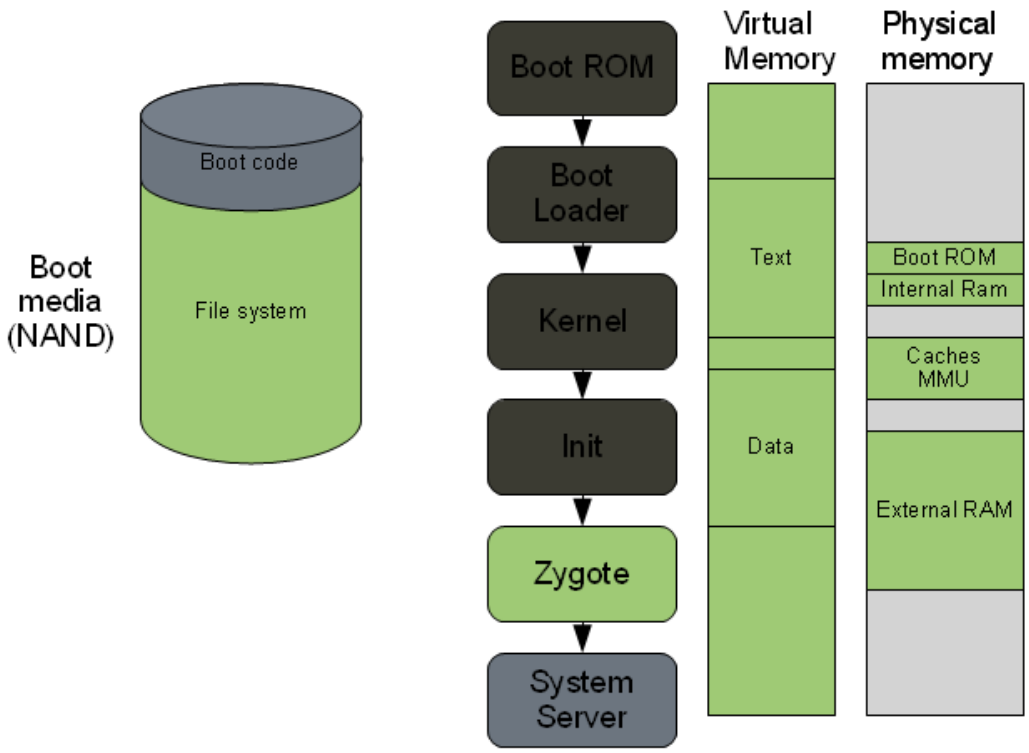
Der init Prozess ist der Ursprung aller Systemprozesse, jeder weitere Prozess ist von ihm oder einer seiner Kinder Prozesse gestartet. Im System-Verzeichnis /system/core/rootdir sucht er das Shell-Skript init.rc und liest dies ein. Das Skript richtet eine Vielzahl von Systemeinstellungen ein und startet die Systemprozesse.



2.3.3-5 Phase 4 der Android Boot Sequenz [L-Hoog11, Seite 55]

2.3.3.5 Phase 5 – Zygote und Dalvik VM

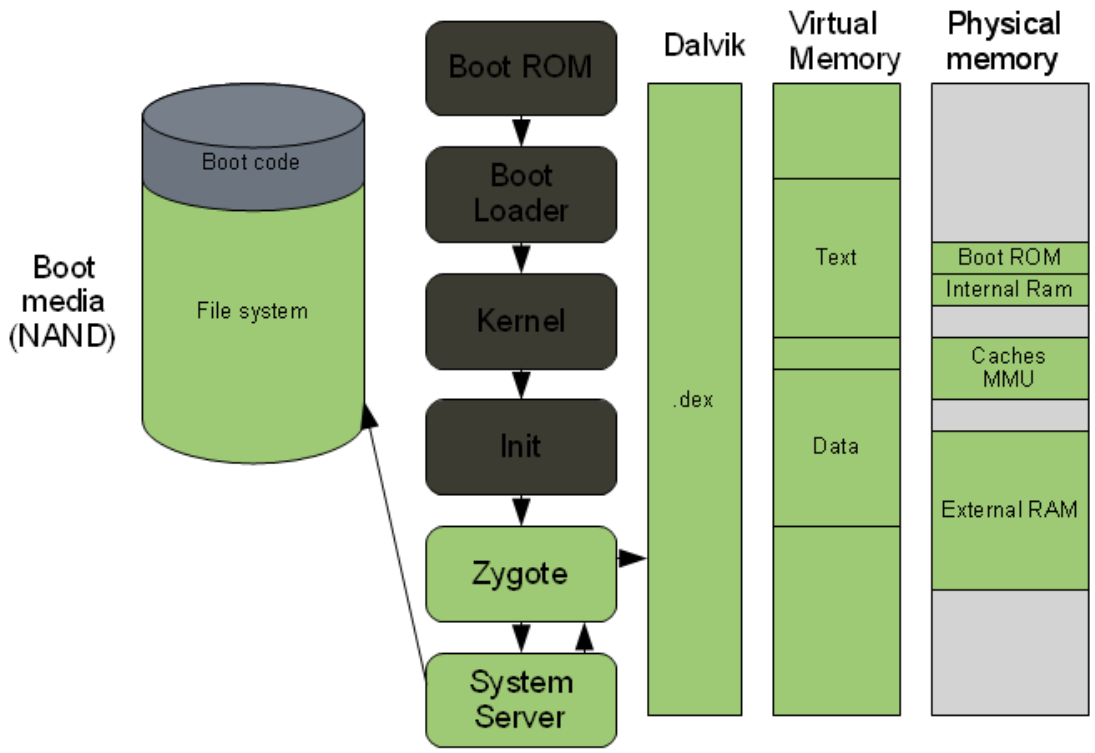
Durch den init Prozess startet Zygote, ein Prozess welcher die Dalvik Virtual Machine initialisiert und startet. Die Dalvik VM dient als Umgebung zur Ausführung einer Anwendung unter Android. Dies bedeutet, dass für jede aufgerufene Anwendung einer Kopie von Zygote, auch eine Dalvik VM erzeugt wird.



2.3.3-6 Phase 5 der Android Boot Sequenz [L-Hoog11, Seite 55]

2.3.3.6 Phase 6 – System Server

Der System-Server ist die erste Java-Komponente die wie in der zuvor beschriebenen Phase gestartet wird. Der System-Server initialisiert die grundlegenden Systemdienste die in der Run-Methode des System-Servers eingetragen sind, er stellt somit die wesentlichen Kernfunktionalitäten des Systems zum Betrieb zur Verfügung.



2.3.3-7 Phase 6 der Android Boot Sequenz [L-Hoog11, Seite 56]

2.3.3.7 Phase 7 – Broadcast

Läuft der System-Server und ist der Bootvorgang beendet, wird eine Broadcast Nachricht ACTION_BOOT_COMPLETED an das System ausgesendet, das System ist nun betriebsbereit. An dieser Stelle setzen registrierte Dienste ein, die nach dem Bootvorgang gestartet werden sollen.

2.3.4 Dateisystem und Partitionierung

2.3.4.1 Dateisystem

Es liegt nahe, dass das verwendete Dateisystem ein Linux-Dateisystem ist, da dem Android-System ein Linux-Kernel zugrunde liegt. Das Dateisystem dient der Organisation des Datenträgers, sodass Dateien gelesen, gespeichert oder gelöscht werden können. Dabei muss das Ordnungs- und Zugriffssystem die Geräteeigenschaften berücksichtigen.

Für die verschiedenen Partitionen auf dem Android Betriebssystem, kommen eine Vielzahl von Dateisystemen zum Einsatz. Die Hauptsächlich genutzten sind dabei das FAT32 Dateisystem das „Root Filesystem“ (Rootfs) und das in Linux-Umgebungen sehr weit verbreitete „Fourth Extended Filesystem“ (Ext4).

FAT

Die FAT ist eine Tabelle mit fester Größe, in der über die belegten und freien Cluster eines FAT-Dateisystems Eintragungen, über genutzte Datenbereiche geführt werden. Die Datenbereiche haben eine feste Anzahl von Clustern.

Rootfs

Das Root-Dateisystem ist ein minimales Dateisystem und sollte in der Regel klein sein, da es sehr kritische Dateien enthält. Es wird vom Kernel unter „/“ eingebunden.

Ext

Im Gegensatz zum FAT-Dateisystem wird der Laufwerksinhalt, in einer Baumstruktur statt in einer Tabelle gespeichert. Die Knoten dieser Baumstruktur bestehen aus Inodes, die Metadaten enthalten und Daten der Datei beziehungsweise Dateilisten von Verzeichnissen referenzieren.

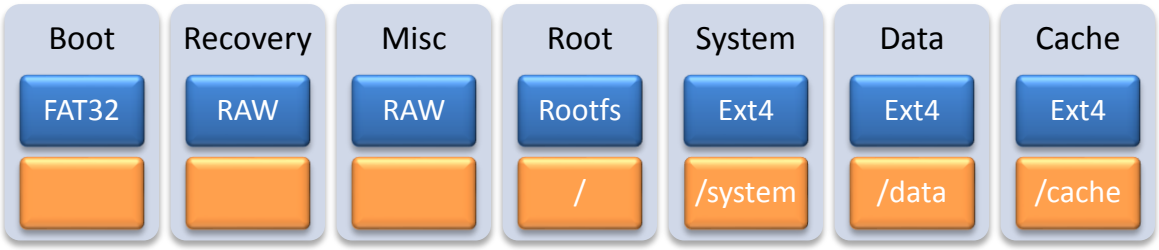
RAW

Das RAW Format abstrahiert Daten nicht über ein Dateisystem und lässt somit den direkten Zugriff auf eine Festplattenpartition zu.

Jedes Dateisystem bietet bestimmte Vor- und Nachteile, auf die in dieser Arbeit nicht weiter eingegangen werden kann. Es wurde in der Kurzbeschreibung auch nicht auf eine spezielle Version der genutzten Dateisysteme verwiesen.

2.3.4.2 Partitionierung

Das Dateisystem des Android Betriebssystems ist in verschieden Partitionen untergliedert und kann nachträglich verändert werden. Das Speicherabbild kann mittels eines geeigneten Werkzeugs eingelesen und verändert werden, dafür bietet sich gparted an. Diese Anwendung ist bereits auf dem Betriebssystem Kali vorinstalliert und wird als kostenlose Version angeboten [S-Kali16]. Wie ein solches Speicherabbild modifiziert werden kann, ist in 3.2.1 Vorbereiten der SD-Karten-Speicherabbilder genau beschrieben. Folgender Abschnitt erklärt grundlegende Partitionen, die auf nahezu allen Android Speicherabbildern vorhanden sind. Die nächste Abbildung stellt dabei Partitionen mit den nachfolgend aufgeführten Dateisystemen (blau) und Einhängepunkten im System (orange) dar [I-DrWi16]:



2.3.4-1 Grundlegende Partitionen auf Android Speicherabbildern

Boot: Die Boot-Partition ist auf fast jedem Linux-System zu finden und enthält Daten, die zum Starten des Linux-Kernels benötigt werden. Aus Kompatibilitätsgründen zu anderen Betriebssystemen wird hier das FAT Dateisystem eingesetzt.

Recovery: Die Recovery-Partition ist die Wiederherstellungspartition und beinhaltet eine Sicherung des Systems, des Weiteren den Bootvorgang im Sicherungsmodus. Die Daten dieser Partition werden ohne Dateisystem abstrahiert und es wird ein direkter Zugriff darauf erlaubt.

Misc: Die Misc-Partition dient der Bereitstellung von Einstellungsdaten die während des Bootvorganges geladen werden, wie bei Recovery wird kein Dateisystem verwendet und der Zugriff auf die Daten erfolgt direkt.

Root: Die Root-Partition ist die Erweiterungspartition und konstituiert den Ausgangspunkt aller weiteren Partitionen.

System: Die System-Partition beinhaltet das Betriebssystem Android welches durch ein NAND-Lock geschützt wird. Beim Löschen dieser Partition wird nur das Betriebssystem gelöscht, ein Bootvorgang ist dennoch weiterhin möglich.

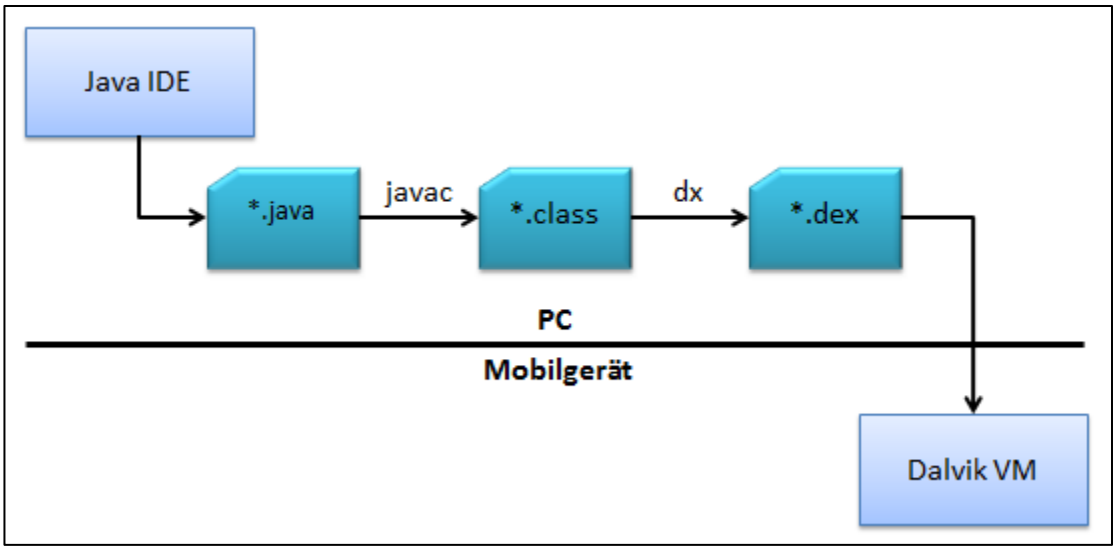
Data: Alle Daten und installierten Anwendungen des Anwenders, werden in der Data-Partition gespeichert.

Cache: Die Cache-Partition dient dem Zwischenspeichern von Anwendungsdaten die im laufenden Betrieb generiert werden.

2.3.5 Dalvik Virtual Machine

Ein wichtiger Teil der Android Laufzeitumgebung ist die Dalvik Virtuelle Maschine. Sie wurde von Dan Bornstein entwickelt und ist nach dem Ort Dalvík in Island benannt, in dem Verwandte von Bornstein lebten. Die Dalvik VM basiert auf der quelloffenen JVM Apache Harmony und wurde in Aufbau und Funktionsumfang an die Anforderungen mobiler Endgeräte angepasst. Als reduzierte virtuelle Maschine mit geringerem Speicherbedarf, unterscheidet sie sich in vielen Punkten von Oracles JVM. Bei der Dalvik VM die als virtuelle Registermaschine konzipiert wurde, werden Instruktionen die auf Variablen arbeiten, direkt im Register gespeichert [L-AsBa02, Seite 18 ff.]. Die JVM im Vergleich zur Dalvik VM arbeitet als Kellerautomat und agiert über einen Stapel [L-AsBa02, Seite 299 ff.]. Weiter wird für die Dalvik VM eine eigene, stark optimierte Version von Byte-Code durch den Präprozessor dx erzeugt und als dex Datei abgelegt. Ein weiterer Vorteil liegt in der genutzten Hardware, da die Dalvik VM sehr gut an die RISC-Architektur der ARM Prozessoren angepasst wurde, um schnell und ressourcenschonend zu laufen. So ist es auch möglich pro Anwendung beziehungsweise pro Prozess eine Dalvik VM zu starten, was sicherheitstechnisch den größten Vorteil an diesem Konzept ausmacht.

Die folgende Abbildung skizziert den Weg des Java-Codes von der Erstellung bis zur Ausführung im Android-Endgerät. Oberhalb der gestrichelten Linie, findet die Entwicklung in der IDE auf dem PC statt. Der nach unten gerichtete Pfeil, deutet den Erstellungsprozess bis hin zum Mobilgerät an. Dieses Vorgehen ist für die Entwicklung von Anwendungen völlig transparent. [L-BePa10, vgl. Seite 17 f.]



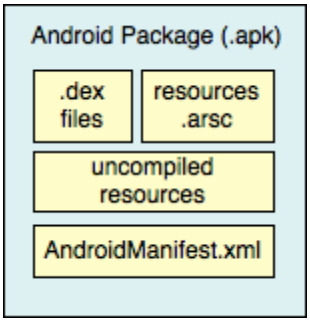
2.3.5-1 Von *.java zu *.dex [L-BePa10, vgl. Seite 17]

2.3.6 Android Paket (APK)

Eine Anwendung kann auf dem Dateisystem vorinstalliert sein oder nachinstalliert werden, lediglich der Speicherort des Android Paketes variiert dabei. Die vom Betriebssystem bereitgestellten Anwendungen, befinden sich in Unterverzeichnissen im System-Verzeichnis /system, nachträglich installierte Anwendungen sind in Unterverzeichnissen unter /data aufzufinden. Da im weiteren Vorgehen dieses Projekts, Anwendungen auf Schadsoftware analysiert werden sollen, ist es von Bedeutung zu wissen, wie eine Anwendung aufgebaut ist. Folgende zwei Abschnitte geben einen Einblick in den Aufbau eines Android Pakets und die Schritte des Erstellungsprozesses.

2.3.6.1 Aufbau

Android Pakete, auch APK-Datei bezeichnet, sind Dateien mit der Dateiendung „apk“. APKs sind ZIP formatierte Dateien, die auf dem Format der JAR-Datei aufbauen und folgende Bestandteile beinhaltet:



2.3.6-1 Aufbau des Android Pakets [I-AnDeA16]

.dex files

Diese Dateien sind das Ergebnis der Transformation des Java-Byte-Codes über den Cross-Compiler dx in dem für Dalvik VM vorgesehenen Format dex-Byte-Code (vgl. Abschnitt 2.3.5 Dalvik Virtual Machine).

resources.arsc

Diese Datei ist eine Tabelle, in der den Ressourcen IDs und Pakete zugeordnet werden. Ressourcen sind keine Code-Dateien, stellen jedoch Informationen für diese bereit.

uncompiled resources

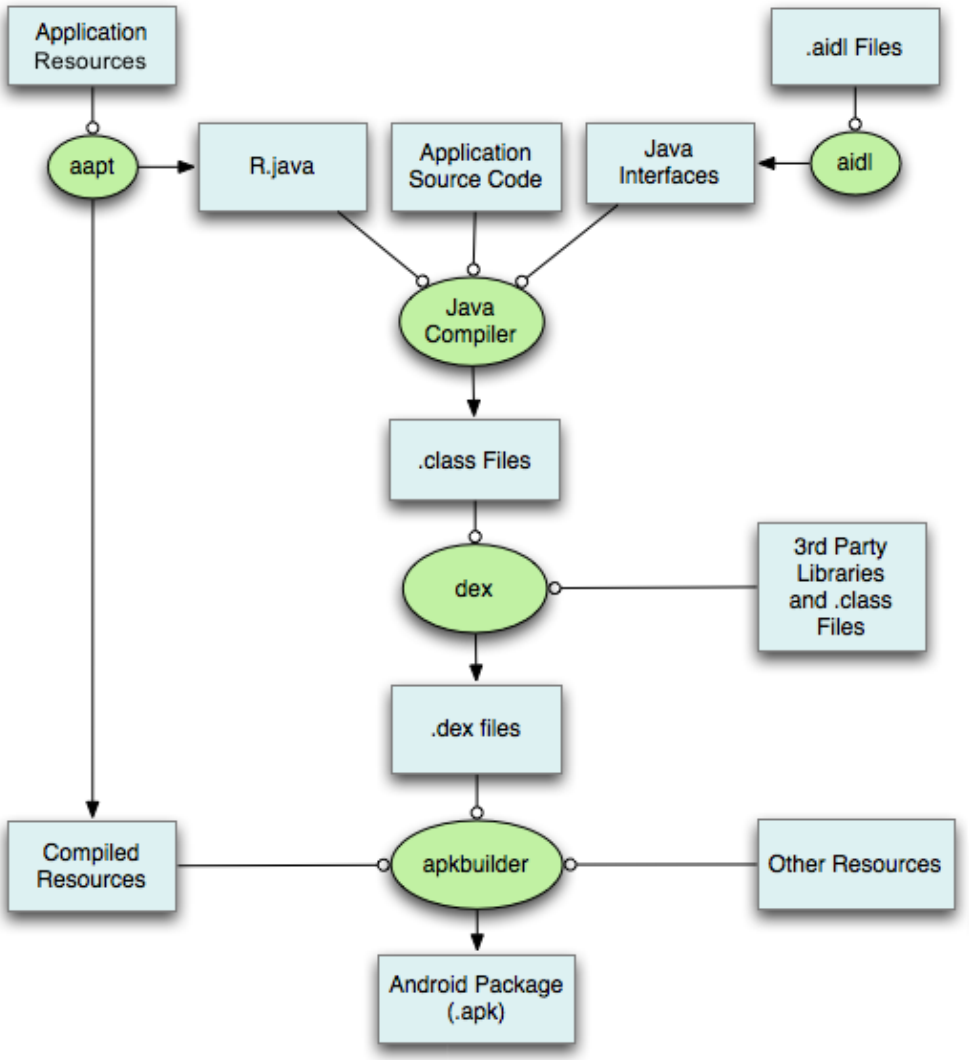
Die nicht kompilierten Ressourcen einer Anwendung, wie Bilder, Audio oder Videos, liegen im res Ordner des ZIP Archives.

AndroidManifest.xml

Das Android Manifest gibt Auskünfte über die Anwendung und dem geplanten Android Zielsystem, auf der es lauffähig sein soll. Jede APK muss eine AndroidManifest.xml Datei beinhalten. Weiter verfügt das Manifest die globalen Definitionen der Anwendung, die implementierten Komponententypen und die Berechtigungen die angefordert werden.

2.3.6.2 Erstellungsprozess (Build Prozess)

Das allgemeine Verfahren für den Erstellungsprozess, wird in der nachfolgenden Abbildung dargestellt, jedoch ist die Signierung der Anwendung zum Schluss des Erstellungsprozesses weggelassen worden.



2.3.6-2 Erstellungsprozess des Android Pakets [I-AnDeB16]

Der Java Compiler erstellt aus der R.java Datei, dem Quellcode und den konvertierten „aidl“ Dateien die kompilierten „class“ Dateien. Diese werden dann zur Benutzung in der Dalvik VM, durch das dex Tool in den Dalvik Bytecode konvertiert. Alle bereits kompilierten Ressourcen werden mit den nicht zu kompilierenden Ressourcen wie beispielsweise Bilder, Audios und Videos und dem Dalvik Dateien „dex“ an den apkbuilder übergeben, der eine APK erzeugt.

2.3.7 Berechtigungen

Android baut auf Linux auf, welches ein Mehrbenutzer-Betriebssystem ist. Anwendungen laufen auf unterster Systemebene mit eindeutigen Systemidentitäten (Benutzer-ID und Gruppen-ID). Somit werden nicht nur Anwendungen, sondern auch ganze Teile des Systems, voneinander getrennt. Dies wird durch den Betriebssystemkern umgesetzt. Zum Thema Linux-Berechtigungsmodel wird in dieser Arbeit nicht weiter eingegangen.

Ein weiterer Sicherheitsmechanismus ist der Android spezifische Berechtigungsmechanismus, der Beschränkungen für spezifische Operationen erzwingt. Diese Berechtigungen lassen sich dabei in drei Kategorien einteilen [I-AnDeJ16]:

Normal

Das Schutzlevel steht für Berechtigungen mit geringem Risiko, da diese Berechtigungen nur Zugriff auf isolierte Funktionen innerhalb der isolierten Anwendung haben.

Gefährlich

Dieses Schutzlevel steht für Berechtigungen mit höherem Risiko, da diese Berechtigungen Zugriff auf Anwenderdaten bieten oder Geräteeinstellungen ändern lassen können.

Signatur

Dieses Schutzlevel bezieht sich auf das Zertifikat, mit dem eine Anwendung signiert wurde. Die angeforderte Berechtigung wird nur vergeben, wenn die fordernde Anwendung mit dem gleichen Zertifikat signiert worden ist, wie die Anwendung welche die Berechtigung vergibt.

Signatur oder System

Dieses Schutzlevel steht für Berechtigungen, die für Anwendungen im Android Betriebssystem-Verzeichnis vergeben werden oder für Anwendungen die mit demselben Zertifikat signiert wurden, wie die Anwendungen die im Systemabbild signiert worden sind.

Für Anwendungen müssen bereits während der Installation alle Berechtigungen angegeben werden, die zur Ausführung nötig sind. Dabei sind die Berechtigungen im Grunde Bezeichner in Form von Zeichenketten, die dem Manifest der Anwendung entnommen werden können (vgl. Abschnitt 2.3.6 Android Paket (APK)). Entwicklern ist es auch möglich, eigene, anwendungsspezifische Berechtigungen zu erstellen, um die Kommunikation mit ihren Komponenten abzusichern. Eine Liste der Berechtigungen für die genutzte Plattform, kann im Internet eingesehen werden. [I-AnDeI16]

2.3.8 Shell-Programmierung in Android

Die Android Shell ist an die Unix Shell angelehnt, dabei ist die Unix Shell-Programmierung in der Literatur umfassend behandelt [L-Hero99, L-Park11]. Die angebotene Funktionalität der Android-Shell verglichen mit der einer Standard Linux-Distribution ist stark vermindert, kann jedoch durch Verwendung von zusätzlicher Software erweitert werden (vgl. Abschnitt 3.2.8 Erweiterung der Android Shell Funktionalität).

2.4 Werkzeuge

Der Umgang mit der Hardwareplattform Wandboard erfordert den Einsatz verschiedener Software basierender Werkzeuge (Tools). Die Selektion der passenden Werkzeuge für die spätere Analyse von Viren, wird maßgeblich durch das Betriebssystem bestimmt. Für die Aufgabenstellung der Analyse von Android Schadsoftware ist der Einsatz des Software Development Kits mit integrierter Android Debug Bridge unumgänglich.

2.4.1 Android Studio

Android Studio ist die offizielle integrierte Entwicklungsumgebung für die Entwicklung von Android Anwendungen. Der Umfang dieser Software umfasst die grafische Oberfläche zur Entwicklung, einen Emulator für virtuelle Geräte, einen Code Editor sowie Code Optimierer, ein Layout Editor, ein Erstellungsprozess System für die verschiedenen Varianten und verschiedenen Fehleranalyseprogramme. Auch Dokumentationen, Beispielcodes mit Übungsbeispielen und einen Manager welcher zur Verwaltung der installierten und installierbaren Tools und Pakete zuständig ist, sind in der Entwicklungsumgebung enthalten. Im Android Studio ist das Android Software Development Kit und die Android Debug Bridge inbegriffen [I-AnDeD16].

2.4.2 Android Software Development Kit (SDK)

Das Android Software Development Kit ist eine Sammlung von Tools und stellt eine umfangreiche Auswahl an Entwicklungswerkzeugen zur Verfügung. Es ist nicht an das Android Studio gebunden und kann über Befehlszeile beziehungsweise Shell oder einer anderen IDE nach Wahl genutzt werden. Das Android SDK umfasst einen Debugger, Bibliotheken, einen Emulator, Dokumentationen, Beispielcodes und Tutorials. Es besitzt Programme zum flashen der Sicherung und Anzeigen der Systemprotokollierung. Somit ist es nicht nur für Entwickler von Anwendungen gedacht, sondern richtet seinen Blick auch an Erfahrene Anwender und Administratoren aus.

2.4.3 Android Native Development Kit (NDK)

Das Android Native Development Kit ist eine Zusammenfassung von Programmen um nativen Code, also Maschinencode geschrieben in C beziehungsweise C++, auf Android Geräten zur Ausführung zu bringen [I-AnDeE16]. Im Gegensatz zum Java-Byte-Code der von der Dalvik VM interpretiert werden muss, wird der kompilierte C/C++ Code direkt auf dem Prozessor ausgeführt, was eine Beschleunigung zur Folge hat und für CPU-intensive Anwendungen wie Spiele-Umgebung, Signalverarbeitung und Physik-Simulation sehr zu bevorzugen ist.

2.4.4 Android Debug Bridge (ADB)

Die Android Debug Bridge ist ein vielseitiges Kommandozeilen-Programm, das zur Kommunikation zwischen ausführendem System und dem angeschlossenen Android Betriebssystem beziehungsweise einem Emulator dient. Es ist eine Client-Server Anwendung. Dabei läuft eine Client Komponente auf der Entwicklerplattform, ein Server auf selbiger als Hintergrundprozess und ein Dienst, der auf jedem Gerät oder Emulator seinen Lauf ebenfalls im Hintergrund vollzieht. Zu beachten ist, dass auf dem zu verbindenden Gerät das „USB debugging“ aktiviert ist (vgl. Abschnitt 2.2.3 Verbindungsaufbau). ADB wird in der Regel über USB verwendet, es ist jedoch auch über WLAN möglich [I-AnDeF16].

Nachfolgend werden alle wichtigen Kommandos von ADB zur korrekten Nutzung erläutert.

ADB-Dienst starten und beenden

adb start-server	Server als Hintergrundprozess starten.
adb kill-server	Serverprozess beenden.

Konnektivität mit einem Gerät herstellen

adb devices	Listet alle angeschlossenen Geräte auf.
adb usb	Setzt die Verbindungsüberwachung des Servers auf USB.
adb tcpip xxxx	Setzt die Verbindungsüberwachung des Servers auf TCP/IP Port xxxx
adb connect 192.168.x.x	Richtet eine Verbindung zum Gerät mit IP 192.168.x.x ein.

Datenaustausch

adb push <Quelle> <Gerät>	Kopiert eine Datei vom ausführenden System auf das Verbundene Gerät.
adb pull <Gerät> <Ziel>	Kopiert eine Datei vom Gerät auf das ausführende System.

Bsp.: Quelle oder Ziel

C:\Users\User\Desktop\test.apk [Windows]

/home/User/Desktop/test.apk [Linux]

Gerät:

/data/local/tmp/test.apk

Anwendung installieren und deinstallieren

adb install <apk> — Installiert eine Android Anwendung auf dem Gerät.

Bsp.: apk
C:\Users\User\Desktop\test.apk [Windows]
/home/User/Desktop/test.apk [Linux]

adb uninstall <name> — Deinstalliert eine auf dem Gerät installierte Anwendung.

Bsp.: name
com.my.testapp

Anwendung ausführen

adb start <apk> — Startet eine Anwendung auf dem Gerät.

Bsp.: apk
/data/local/tmp/test.apk

WEITERE Kommandos

adb reboot — Fährt das Gerät herunter und startet es neu.

adb remount — Systempartition /system mit Schreibrecht einbinden.

adb shell — Startet eine Shell auf dem Gerät im aktiven Modus.

adb shell <command> — Startet eine Shell auf dem Gerät mit explizit folgendem Kommando.

Bsp.: command
id — Zeigt die User-ID an.
ls — Zeigt den Inhalt des aktuellen Verzeichnisses an.

Minimal ADB

Sollte nur die Verwendung von ADB erwünscht sein, ist es auch möglich eine Speicher schonende Version zu installieren, diese beinhaltet lediglich ADB und fastboot, um den Verbindungsaufbau zwischen einem Rechner und einem Gerät zu realisieren. Minimal ADB ist für jede gängige Android Version über das XDA Entwickler Forum [S-XdaF16] als kostenlose Version abrufbar.

2.4.5 Paket-Manager und Anwendungs-Manager über die Android Shell

Da die Shell-Programmierung nicht das Thema dieser Arbeit darstellt, doch im späteren Verlauf viele Methoden über die Shell realisiert werden, wird der Paket-Manager und der Anwendungs-Manager kurz erläutert. Die Kommandos der Manager bieten Zugriffe auf zahlreiche Systemfunktionen wie das Steuern und das Starten der Aktivitäten verschiedener Anwendungen. [I-AnDeH16]

Paket-Manager

Mit dem Paket-Manager (Package Manager) lassen sich alle installierten Anwendungen anzeigen, bereits auf dem Gerät befindliche „apk“ Dateien installieren und generell Anwendungen verwalten. [I-AnDeH16, Meta-Tag #pm]

Anwendungs-Manager

Mit dem Anwendungs-Manager (App Manager) lassen sich Aktivitäten von Apps direkt starten, was etwa dem Antippen von Verknüpfungen auf dem Hauptbildschirm entspräche. Ebenso können Prozesse beendet, verschiedene Systemaktionen ausgeführt, Intents als Broadcast gestartet, die Gerätebildschirmeigenschaften verändert werden und vieles anderes mehr. [I-AnDeH16, Meta-Tag #am]

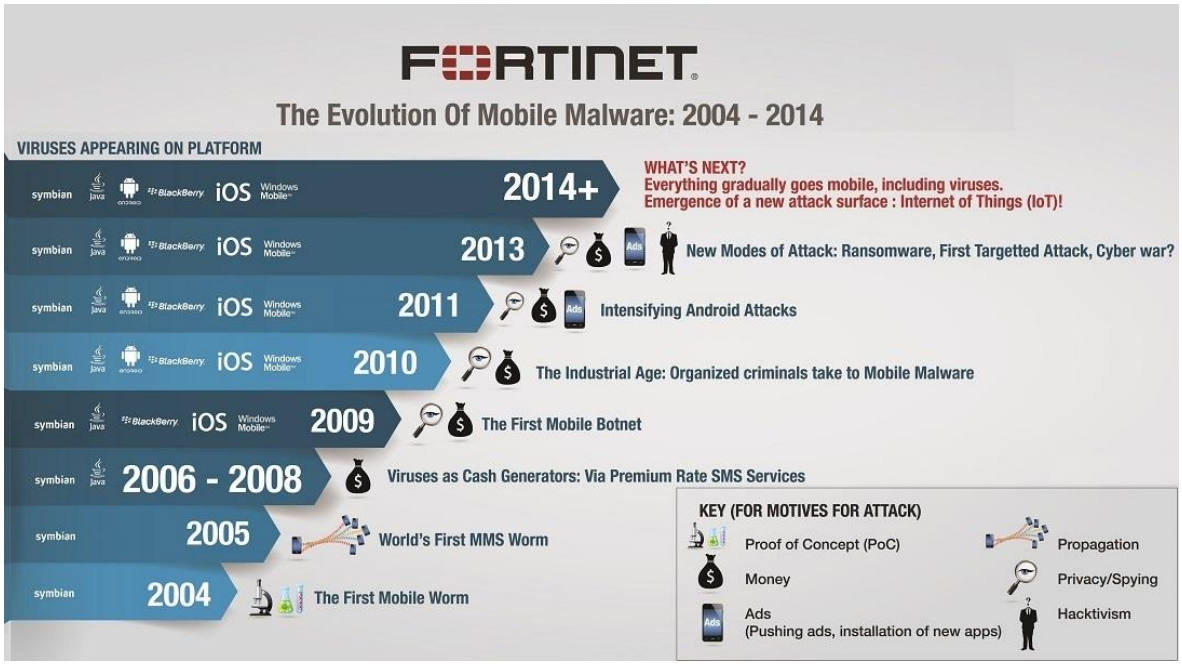
Der Überblick der möglichen Befehle des Paket-Managers und des Anwendungs-Managers, macht die Mächtigkeit dieses Werkzeuges erst klar. Die Verwendungen dieser Werkzeuge im späteren Verlauf des Projektes sind unabwendbar. Eine einfache Handhabung über ein Terminal ist mit Leichtigkeit zu realisieren.

2.5 Schadsoftware

Schadsoftware bezeichnet Computerprogramme die in datenverarbeitende Systeme eingeschleust werden, um deren Benutzer Schaden durch unerwünschte Funktionen zu zufügen. Der Einsatz von Schadsoftware stellt somit eine ernstzunehmende Bedrohung dar. Im folgenden Abschnitt wird zunächst darauf eingegangen, warum Android ein lukratives Ziel für Schadsoftware ist. Weiter wird die Schadsoftware anhand der Verbreitungsmethoden und des Funktionsumfanges klassifiziert, dabei spielt auch der Sicherheitsmechanismus zur Rechtevergabe eine enorme Rolle.

2.5.1 Schadsoftware im Wandel der Zeit

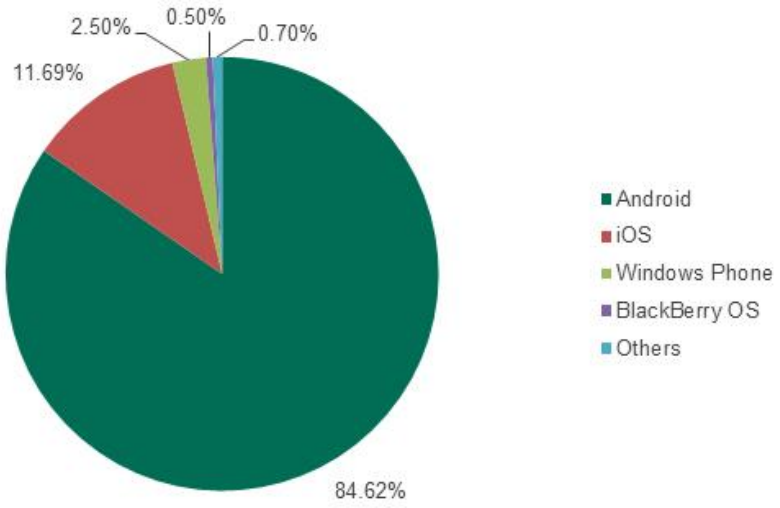
In der Anfangszeit der Computertechnik wurde Schadsoftware oft von Personen, mit eher akademischem Interesse verfasst. Diese Schadprogramme konnten wichtige Dateien löschen oder das Betriebssystem beschädigen. Aktuell handelt es sich bei diesem Personenkreis um IT-Kriminelle, die profitorientiert handeln und gezielt arbeiten [L-Ecke13, vgl. Seite 23]. Die Schadsoftware die heutzutage zum Einsatz kommt, ist so verfeinert, dass ein Geschädigter eine Infektion gar nicht bemerkt. Manche Schadprogramme übernehmen Rechner, um im Rahmen eines Netzwerks diese fernzusteuern. Andere Schadsoftware-Varianten greifen sensible Daten ab, um Online-Betrug vorzubereiten. Ein weiterer Faktor dieses Wandels findet mit der ständigen Anpassung der Systeme, gegen diese Bedrohung statt. Bestehende Sicherheitsmechanismen werden mittels Hardware sowie Software ständig verbessert und durch neue Schutzhürden, wird der Schadsoftware der Zugang zum System erschwert.



2.5.1-1 Evolution mobiler Schadsoftware [I-Digi16]

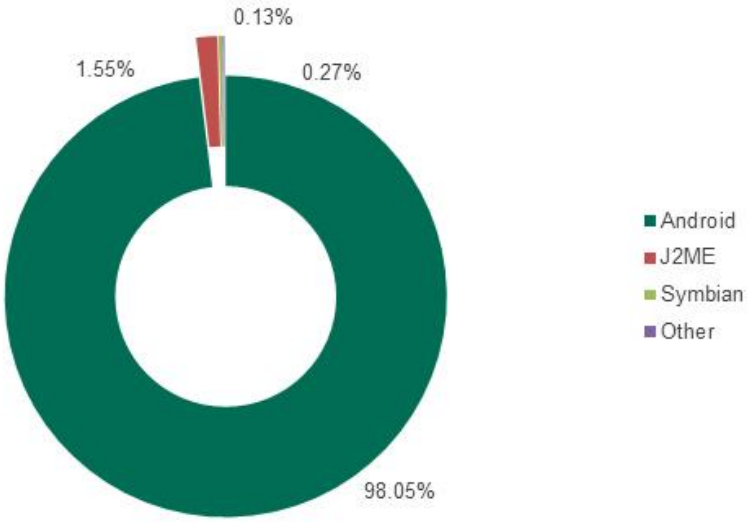
2.5.2 Android als Zielplattform

Aufgrund der Tatsache das Smartphones immer mehr Anwendung im privaten und beruflichen Alltag finden, erreichen die Betriebssysteme dieser Geräte eine immer größere Verbreitung. Folgende Abbildung zeigt die Verteilung von mobilen Betriebssystemen am Weltmarkt.



2.5.2-1 Aufteilung der mobilen Betriebssysteme im 2. Quartal 2014 [I-Sec16]

Durch die stetig wachsende Leistungsfähigkeit dieser Systeme, sind sie nicht nur lohnendes Ziel für Angriffe von IT-Kriminellen, sondern auch von Behörden und Staaten geworden.

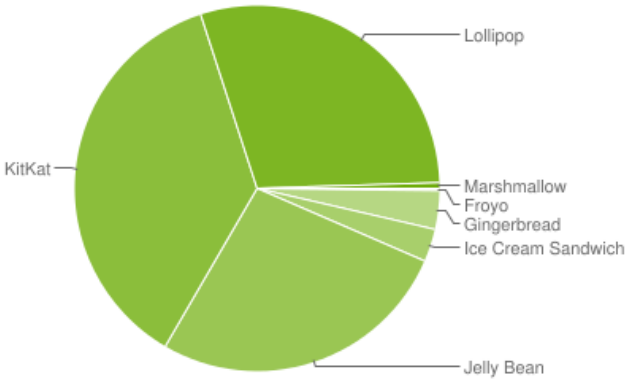


2.5.2-2 Schadsoftwarefunde durch Kaspersky Lap im Jahr 2013 [I-Sec16]

Die Abbildung zeigt die prozentuale Verteilung an Schadsoftware auf mobilen Umgebungen, erkannt durch Kaspersky Lab im Fundzeitraum 2013.

Es ist zu entnehmen, dass sich im Jahr 2013 der größte Teil der Schadsoftware gegen das Betriebssystem Android richtet. Der hohe Schadsoftwareanteil ist aufgrund der starken Marktpräsenz des Android-Systems zu erklären. Die Update Problematik die mit Android einhergeht, ist ein sicherheitskritischer Aspekt. Da Updates nur vom Hersteller des Gerätes kommen und dies erst nach sehr langer Zeit geschieht, können bekannte Sicherheitslücken auf den Geräten ausgenutzt, aber nicht geschlossen werden. Die nachfolgende Abbildung vermittelt dabei deutlich die prozentuale Verteilung der einzelnen Android Versionen am Markt.

Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	3.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.9%
4.1.x	Jelly Bean	16	10.0%
4.2.x		17	13.0%
4.3		18	3.9%
4.4	KitKat	19	36.6%
5.0	Lollipop	21	16.3%
5.1		22	13.2%
6.0	Marshmallow	23	0.5%



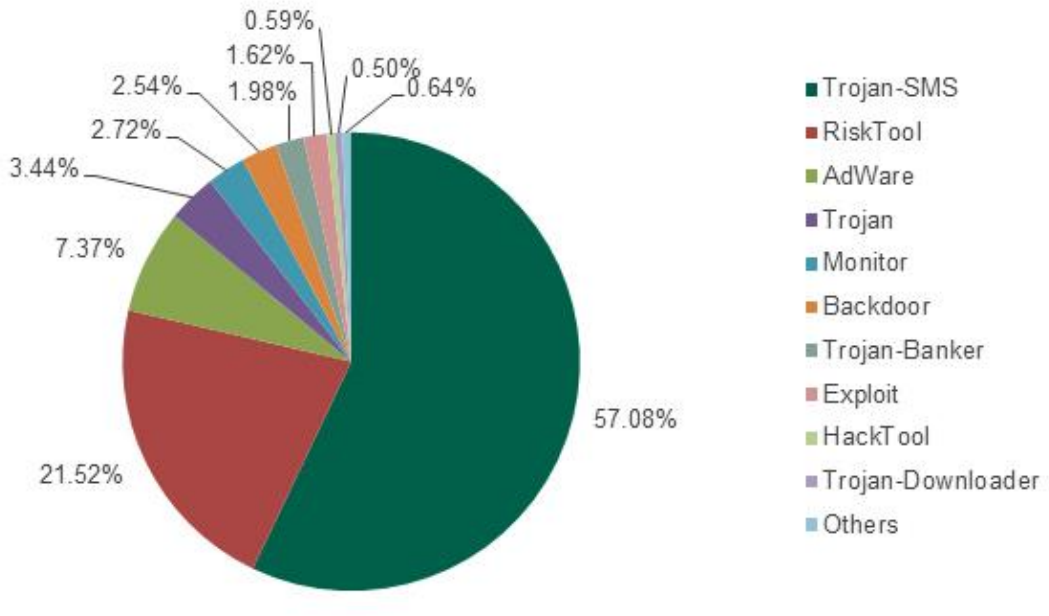
Any versions with less than 0.1% distribution are not shown.

2.5.2-3 Verteilung der Android Plattform Version [I-AnDeZ15]

Es ist unverkennbar, dass die Version KitKat 4.4 mit 36,6 % den größten Anteil annimmt, dicht gefolgt von Jelly Bean mit insgesamt 26,9 %. Dabei ist die Version Lollipop 5.0 am 3. November 2014 am Markt eingeführt wurden. Geschultert ist dies den aufwändigen Anpassungen die ein Hersteller durchführen muss, um Updates für eine neue Android Version publizieren zu können, angepasst an die vielfältigen Geräte.

2.5.3 Klassifizierung

In der Vergangenheit konnten Schadprogramme in Rubriken eingeteilt werden, die bekanntesten Vertreter davon sind Viren, Würmer und Trojaner. Aktuell ist es so, dass diese Klassifizierung weniger Anwendung findet, da die Schadprogramme eine Vielzahl von Funktionen besitzen und so mehr und mehr zu Hybriden werden. Leider ist eine Klassifizierung von Schadprogrammen in keiner Norm aufgeführt, darum finden auch die unterschiedlichsten Modelle zur Taxonomie von Schadsoftware Anwendung. Die nachfolgende Abbildung zeigt die Verteilung von Schadsoftware anhand der Taxonomie von Kaspersky Lab.



2.5.3-1 Die 10 aktivsten Schadprogramm-Typen August 2013 - Juli 2014 [I-Sec16]

Der vermehrte Einsatz von Trojanern, wird aus der Abbildung sichtbar. Trojaner sind Computerprogramme die sich als nützliche Anwendung tarnen und im Hintergrund ohne Wissen des Anwenders, andere Funktionen ausführen. Dies ist auch im Hinblick der aktuellen Sicherheitsmechanismen die ein Betriebssystem bietet, als Präventionsmaßnahme zum Schutz vor einer Infektion plausibel. In den meisten Fällen wird Schadsoftware in Anwendungen integriert, unter einem Decknamen angeboten und durch das Vertrauen des Anwenders ausgeführt.

Im weiteren Verlauf dieses Abschnittes wird keine Klassifizierung im herkömmlichen Sinn durchgeführt, sondern die Schadsoftware anhand der Verbreitungsmethoden und des Funktionsumfangs beschrieben.

2.5.4 Verbreitungsmethoden

Durch immer sicher werdender Kommunikationsprotokolle und Betriebssysteme, ist die Verbreitung von Schadsoftware durch Ausnutzung von Sicherheitslücken zurückgegangen. Aktuell nutzen Schadprogramme reguläre Kommunikationskanäle, wie Instant Messaging Programme und E-Mails, um sich zu verbreiten. Diese Programme sind auf die Mithilfe ihrer Geschädigten angewiesen, die wiederum durch Social Engineering dazu bewegt werden, den Anhang einer E-Mail auszuführen oder einem zugeschickten Link zu folgen. Die Möglichkeiten einer Infektion sind dadurch sehr vielfältig, können jedoch in drei Oberrubriken unterteilt werden:

Installation durch den Anwender

Eine Anwendung wird durch den Benutzer des Gerätes installiert und so legitimiert mit denn zur Installation stattgegebenen Rechten, zu interagieren. Die Anwendung lässt durch Deckname und andere irreführende Beschreibungen, keinen Rückschluss auf ein Schadprogramm zu.

Installation durch Dritte

Eine Anwendung wird durch einen Dritten, der physischen Zugriff auf das System hat, installiert. Der Benutzer des Geräts weiß unter Umständen nicht, dass Schad- oder Spionagesoftware auf dem Gerät installiert wurde.

Ausnutzung von Schwachstellen

Eine Anwendung wird durch Ausnutzung einer Schwachstelle im Betriebssystem installiert. Diese Technik ist auch als Exploit bekannt und gewährt der Anwendung zumeist die höchsten Systemrechte. Schwachstellen ergeben sich hierbei durch die permanent steigende Funktionalität der Betriebssysteme. Auch die Synchronisation mit anderen Geräten kann eine große Rolle spielen. Es bestehen Möglichkeiten, dass bei dem Datenaustausch zwischen Mobilgerät und Fremdgeräten Schwachstellen ausgenutzt und Angriffe in beide Richtungen realisiert werden.

An dieser Stelle ist zu sagen, dass auch Mischformen in Form von „Drive-by-Angriffen“ möglich sind. Dabei wird der Benutzer auf eine Internetseite verwiesen, auf der Schadsoftware für das genutzte Betriebssystem und den genutzten Browser hinterlegt ist, auch hierbei werden Schwachstellen im System ausgenutzt. Noch eine sehr interessante Möglichkeit, ergibt sich mit der Datenumleitung „Man-in-the-Middle“. „Man-in-the-Middle“ ist die Möglichkeit, die Kommunikation zwischen einem Mobilgerät und einem Zielsystem abzuhören und die gesendeten beziehungsweise empfangenen Daten zu manipulieren, um unbemerkt Schadsoftware zu platzieren.

2.5.5 Funktionsumfang und Berechtigungen

Schadsoftware ist auf die vom Betriebssystem zur Verfügung gestellten Funktionen angewiesen. Die Komplexität dieser Programme ist also abhängig von der Anzahl und dem Umfang der genutzten Funktionen, einige dieser sind nachfolgend aufgelistet:

- Ver- und Entschlüsselung der Anwendung zur Laufzeit
- Obfuscating des Quellcodes
- Kommunikation mit einem Command-and-Control-Server
- Nachladen von Code
- Entwenden, manipulieren oder verschlüsseln von privaten Daten
- Umleiten der Kommunikation
- Versenden von SMS oder E-Mails

Schadsoftware unter Android kann nur wirken, wenn diese installiert wurde und die geforderten Berechtigungen erhalten hat. Dieser Sicherheitsmechanismus findet Anwendung über das Android Manifest, welches nur im Binärformat vorliegt und ohne Konvertierung unbrauchbar ist. Um Berechtigungen einer Anwendung lesen oder sogar ändern zu können, ist der Entwicklungsumgebung das Android Asset Packaging Tool (aapt) beigefügt. Der Funktionsumfang kann daher Aufschluss geben, um welche Art von Schadprogramm es sich handelt. Die Zuordnung von Funktionen zu Berechtigungen ist jedoch nicht immer trivial, da die Zuteilung der Rechte in die Berechtigungsgruppen von Google erfolgt [I-AnDeC16, Meta-Tag #perm-groups]. Nachfolgend sind einige Beispiele zu Funktionen und Berechtigungen aufgelistet:

Identität:	GET ACCOUNTS, READ PROFIL, …
Kamera/Mikrofon:	CAMERA, FLASHLIGHT, RECORD VIDEO, …
SMS:	RECEIVE MMS, WRITE SMS, SEND SMS, …
Kontakte:	READ CONTACTS, …
Geräte-ID und A.:	READ PHONE STATE, READ SETTINGS, …
Geräte- und App-Verlauf:	GET TASKS, CHECK LICENSE, …
Telefon:	CALL PHONE, …
Standort:	ACCESS COARSE LOCATION, …
Fotos/Medien/Dateien:	READ EXTERNAL STORAGE, …
Sonstiges:	INTERNET, STATUS BAR, INSTALL SHORTCUT,…

2.5.6 Sonstige Besonderheiten

Unter Schadsoftware ist nicht immer die gesamte Anwendung zu verstehen. So können ahnungslosen Benutzern legitime Anwendungen mit eingebettetem, schadhaftem Codefragment untergeschoben werden. Hierbei ist die Modifizierung einer kostenpflichtigen Anwendung sehr beliebt, da sie über einen alternativen Markt unentgeltlich greifbar ist. Dabei sind die Hauptkomponenten der modifizierten Anwendung, mit denen des Originals gleich. Das Manifest der modifizierten Anwendung ist auf dem Berechtigungslayout der Schadroutinen erweitert. Beim Vergleich der Anwendungen würde eine Manipulation zum Vorschein treten.

3 Methoden

Im folgenden Kapitel wird eine Aufstellung der verwendeten Hardware und Software gegeben, Der Aufbau und die Installation aller Komponenten werden beschrieben. Darüber hinaus präsentiert das Kapitel, anhand des zu konfigurierenden Materials, alle durchgeführten Methoden.

3.1 Material

Nachdem die Grundlagen zum Wandboard und dem zu verwendeten Betriebssystem bekannt sind, müssen noch Vorbereitungen zur Durchführung der einzelnen Methoden getroffen werden. Zu dieser Bereitstellung gehört der Einsatz verschiedener Fremdsysteme, um bestimmte Interaktionen mit der Hardwareplattform Wandboard durchführen zu können.

3.1.1 Verwendete Hardware

Zu der bereits erwähnten Hardwareplattform Wandboard, wird zur Durchführung aller Methoden folgende Hardware benötigt:

1x Rechnersystem mit USB und SD-Karten-Schnittstelle (PC oder Laptop)

1x Mikro SD-Karte (evtl. pro Betriebssystem eine Mikro SD-Karte)

Zur Durchführung einiger Methoden wird folgende Hardware benötigt:

1x Mikro USB auf USB-Kabel

1x optische Maus über USB

1x Tastatur über USB

1x USB-Hub (Passiv)

1x WLAN-Router (mit Konnektivität zum Internet)

Obendrein sollte das zu verwendende Rechnersystem folgende Komponenten besitzen eine, USB Schnittstelle zur ersten Kommunikation mit dem Wandboard, eine SD-Karten-Schnittstelle zum Bereitstellen der Android Betriebssysteme und ein WLAN-Modul für die Kommunikation mit einem Netzwerkgerät. An das Rechnersystem selbst werden keine hohen Ansprüche gestellt, da die meisten Aktionen über die Shell beziehungsweise Eingabeaufforderung erledigt werden und somit kein großes Leistungsniveau vorausgesetzt wird. Es wird mindestens eine Mikro SD-Karte benötigt, die das zu verwendende Betriebssystem für das Wandboard bereitstellt. Hier bietet es sich pro Betriebssystem an, eine weitere Mikro SD-Karte vorrätig zu haben. Dies macht es beim Experimentieren mit den verschiedenen Android Versionen geradezu einfacher, da das Beschrei-

ben der Mikro SD-Karten mit dem Betriebssystem Speicherabbild sehr viel Zeit in Anspruch nimmt. Dem Wandboard liegt kein Mikro USB auf USB Verbindungskabel bei, das zu ergänzen ist. Zur Steuerung des Android Betriebssystems eignet sich eine USB Maus und eine USB Tastatur sehr gut. An dieser Stelle sei vermerkt, dass die Steuerung auch über ein Touchscreen erfolgen kann. Die USB Maus und die USB Tastatur müssen über einen USB-Hub angeschlossen werden, da das Wandboard nur einen USB-Port besitzt, hierbei jedoch zwei Steuerungsgeräte Verwendung finden. Der USB-Hub kann hier ein passiver sein, da Maus und Tastatur keine hohen Leistungsverbraucher darstellen. Beim Einsatz von weiteren USB Medien an den USB-Hub ist zu bedenken, dass ein Tausch der Passiv-Komponente mit der Aktiv-Komponente nötig sein könnte, um sich nicht der Leistung des Systems zu bedienen. Überdies wird die Kommunikation mit dem Internet durch einen benötigten WLAN-Router mit DHCP-Funktion geregelt.

3.1.2 Verwendete Software

Das in Gebrauch zu nehmende Betriebssystem für das Rechnersystem ist unabhängig von der Durchführung und sollte dem Benutzer bekannt sein, sodass eine Verwendung ohne Probleme vollzogen werden kann. Die für diese Arbeit angewandten Betriebssysteme sind:

Microsoft Windows 8.1 Pro 64-bit

Kali Linux 2016.1 32-bit

Weiter wird noch folgende Software verwendet:

Android Studio 2.1

Oracle Virtual Box 5.0.14

Win32 Disk Imager 0.9.5

Wireshark 1.12.8

Es ist für die auf dem Rechnersystem befindliche Betriebssystem-Umgebung zu empfehlen, eine Linux Distribution zu wählen, denn die Dateisysteme der in den Speicherabbildern befindlichen Betriebssysteme sind Android basierend und nutzen somit Dateisysteme, die unter anderen Betriebssystemen nicht erkannt werden können. Explizit heißt das, dass die Dateisysteme bestimmter Partitionen der Mikro SD-Karte unter Microsoft Windows nicht richtig interpretiert werden können.

3.1.3 Aufbau der Testumgebung

Die Verbindungen der Komponenten des Rechnersystems sowie der Komponenten des Routers, sind nach den mitgelieferten Benutzerhandbüchern dieser Geräte erfolgt. Das Anschließen und Einstecken der einzelnen Komponenten, Kabel und Schnittstellen am Wandboard ist dem Abschnitt 2.2.1-1 Beschriftung zur Wandboard Schnittstellenkarte [I-WanC16, Seite 6] und dem Abschnitt 2.2.1-2 Beschriftung zum Wandboard Kernmodul [I-WanC16, Seite 5] zu entnehmen.

3.1.4 Installation der Testumgebung

Die Testumgebung die für diese Arbeit Verwendung findet, lässt sich wie folgt detaillieren:

Das Rechnersystem ist mit einem voll funktionsfähigen Microsoft Windows 8.1 Pro 64-bit Betriebssystem ausgestattet und nach Installationsanleitung eingerichtet [I-MicrA16]. Des Weiteren sind folgende Programme installiert:

Android Studio 2.1, Installationshinweis unter [I-AnDeG16]

Oracle Virtual Box 5.0.14, Installationsanleitung unter Kapitel 2 [I-Orac16]

Win32 Disk Imager 0.9.5, Installationshinweis unter [I-MicrB16]

Wireshark 1.12.8, Installationsanleitung unter Kapitel 2 [I-Wire16]

Über Oracle Virtual Box 5.0.14 wurde eine virtuelle Maschine erzeugt, die ein Rechnersystem mit dem Kali Linux 2016.1 32-bit Betriebssystem beinhaltet. Die Installation einer virtuellen Maschine ist in Kapitel 3 [I-Orac16] beschrieben. Die WLAN Verbindung ist über das Rechnersystem eingerichtet und funktionsbereit [I-MicrC16]. Der angemeldete Benutzer am Microsoft Windows Rechnersystem und am Linux System hat volle Administrationsrechte beziehungsweise Root-Rechte oder kann diese, über die Benutzerkontensteuerung des jeweiligen Systems, ohne Einschränkung anfordern.

An dieser Stelle ist anzumerken, dass die Testumgebung auch sehr viel einfacher gestaltet werden kann, indem nur die Distribution Kali Linux zum Einsatz kommt, mit einer Installation von minimal Android Debug Bridge.

3.1.4.1 Einrichten der SD-Karten-Schnittstelle für die virtuelle Maschine

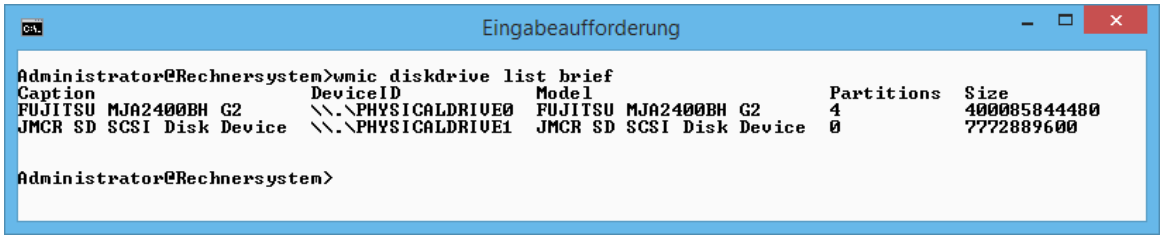
Da Microsoft Windows nicht alle Dateisysteme unterstützt und der Umgang mit den für diesem Projekt benötigten Dateisystemen notwendig ist, wird in diesem Abschnitt eine Konfiguration erläutert, die den direkten Zugriff auf eine im SD-Karten-Lesegerät steckende SD-Karte über die virtuelle Maschine realisiert.

1. Identifizierung der Geräte ID für das SD-Karten Lesegerät

Über das von Microsoft Windows mitgelieferte Programm cmd.exe (Eingabeaufforderung) kann die Geräte ID mit folgendem Befehl ermittelt werden:

```
wmic diskdrive list brief
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.1.4-1 cmd.exe: Geräte ID unter Microsoft Windows ermitteln

Die zwei Einträge der ausgegebenen Tabelle sind:

- Festplatte des Rechnersystems

- eingesetzte SD-Karte

Somit lautet die Geräte ID für die SD-Karte **\\.\PHYSICALDRIVE1**. Außerdem kann hier auch schon die Größe des Datenträgers mit 7772889600 Byte ermittelt werden.

2. Spezial Link über Virtual Box erstellen

Über ein im Leistungsumfang integriertes Programm von Virtual Box, wird ein spezieller Hardwarelink für den direkten Zugriff auf die SD-Karte erstellt. Die Software ist im Installationsverzeichnis von Virtual Box, das standardmäßig nach „C:\Program Files\Oracle\VirtualBox“ installiert wird, unter VBoxManage.exe zu finden.

```
VBoxManage.exe internalcommands createrawvmdk -filename
  "%USERPROFILE%/Desktop/sdcard.vmdk" -rawdisk "\\.\PHYSICALDRIVE1"
```

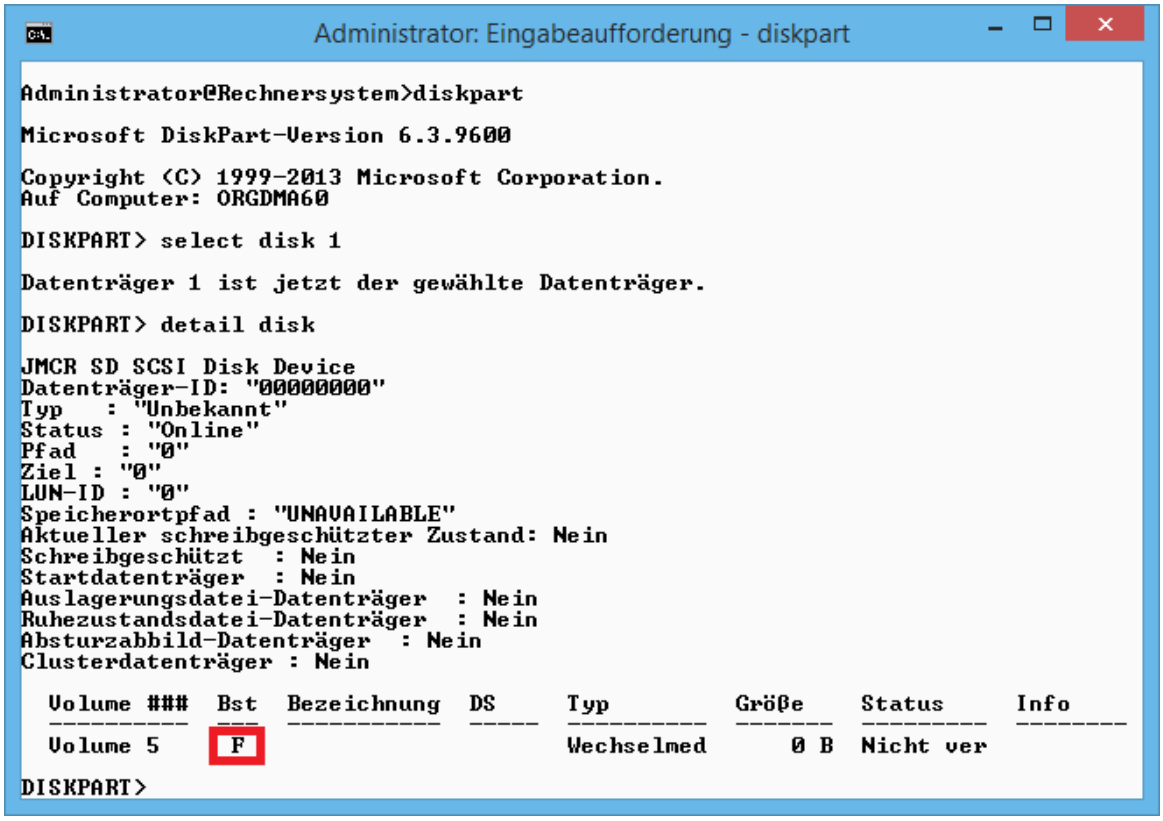
Die so erstellte Datei sdcard.vmdk, die sich auf dem Desktop des angemeldeten Benutzers befindet, lässt den direkten Zugriff auf die SD-Karte über eine virtuelle Maschine zu.

3. Virtuelle Größe der tatsächlichen SD-Karten Größe anpassen

Über das Programm cmd.exe (Eingabeaufforderung) wird nun die Sektor Größe der SD-Karte bestimmt. Da die Geräte ID \\.\PHYSICALDRIVE1 lautet, trägt die SD-Karte die Bezeichnung Datenträger 1 im Programm DiskPart, folglich kann das Medium mit „disk 1“ angesprochen werden. Der Laufwerksbuchstabe wird mit nachstehender Befehlsfolge ermittelt:

```
diskpart
select disk 1
detail disk
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diese Befehle.



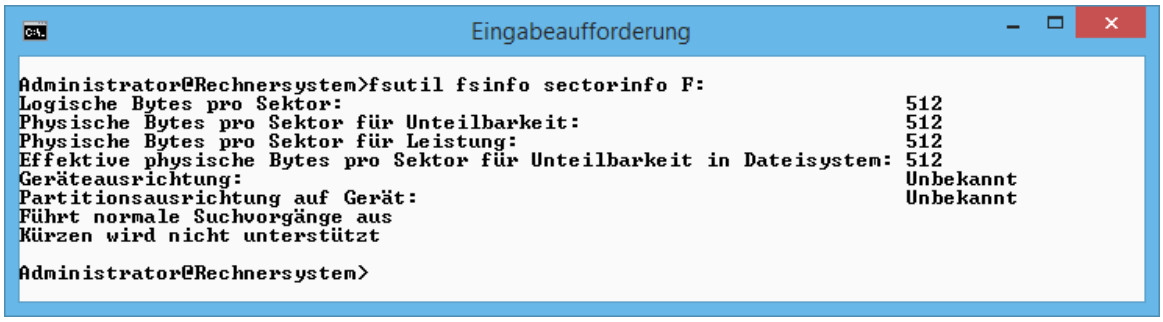
3.1.4-2 cmd.exe: Laufwerksbuchstabe für SD-Karte ermitteln

Der Laufwerksbuchstabe der Disk 1 ist mit „F“ (roter Rahmen) klar bezeichnet.

Zudem sollte die Sektor Größe ermittelt werden, dies ist mit folgendem Befehl über das Programm cmd.exe möglich:

```
fsutil fsinfo sectorinfo F:
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.

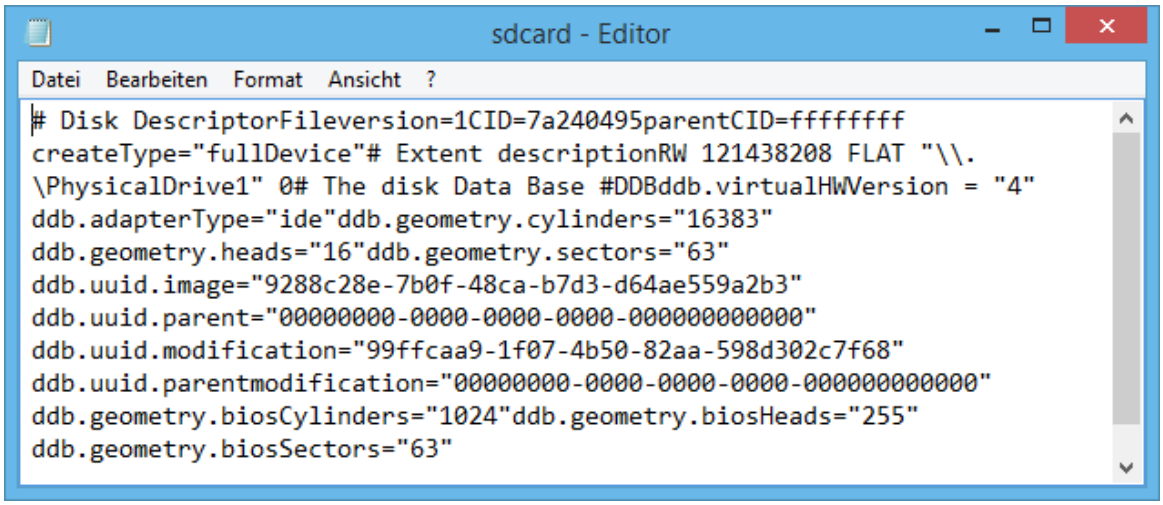


3.1.4-3 cmd.exe: Sektor Größe eines Datenträgers ermitteln

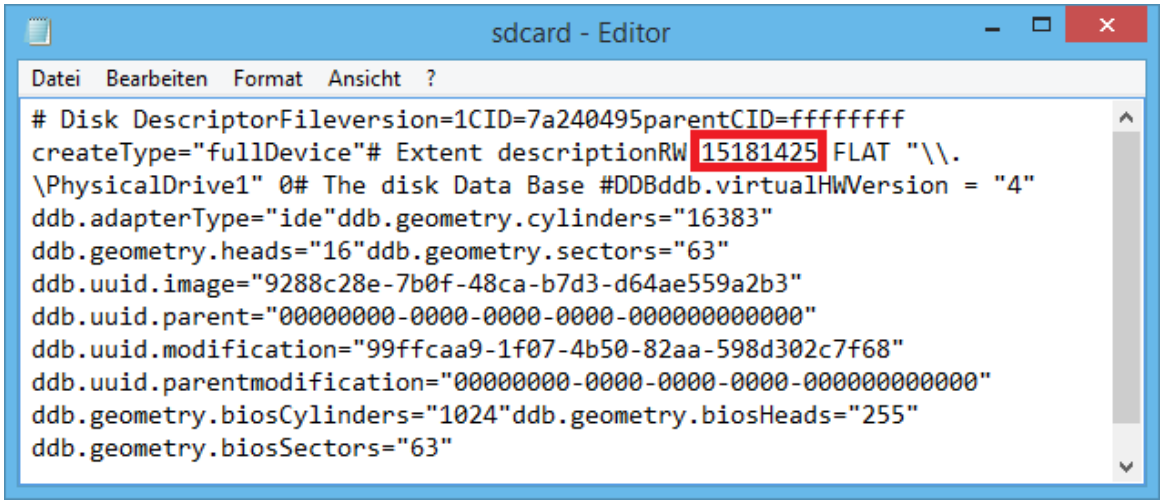
Zu sehen sind die logischen Byte pro Sektor mit einer Größe von 512 Byte. Mit den so gewonnenen Informationen über die SD-Karte kann die tatsächliche Größe ermittelt werden. Eine Rechnung mit den zuvor ermittelten Werten berechnet die Größe in 512 Byte Einheiten:

7772889600 Byte / 512 Byte = **15181425**

Die im vorherigen Schritt erstellte Datei sdcard.vmdk, wird nun mit einem Editor geöffnet und an der Stelle „Extent descriptionRW “ mit der berechneten Größe geändert:



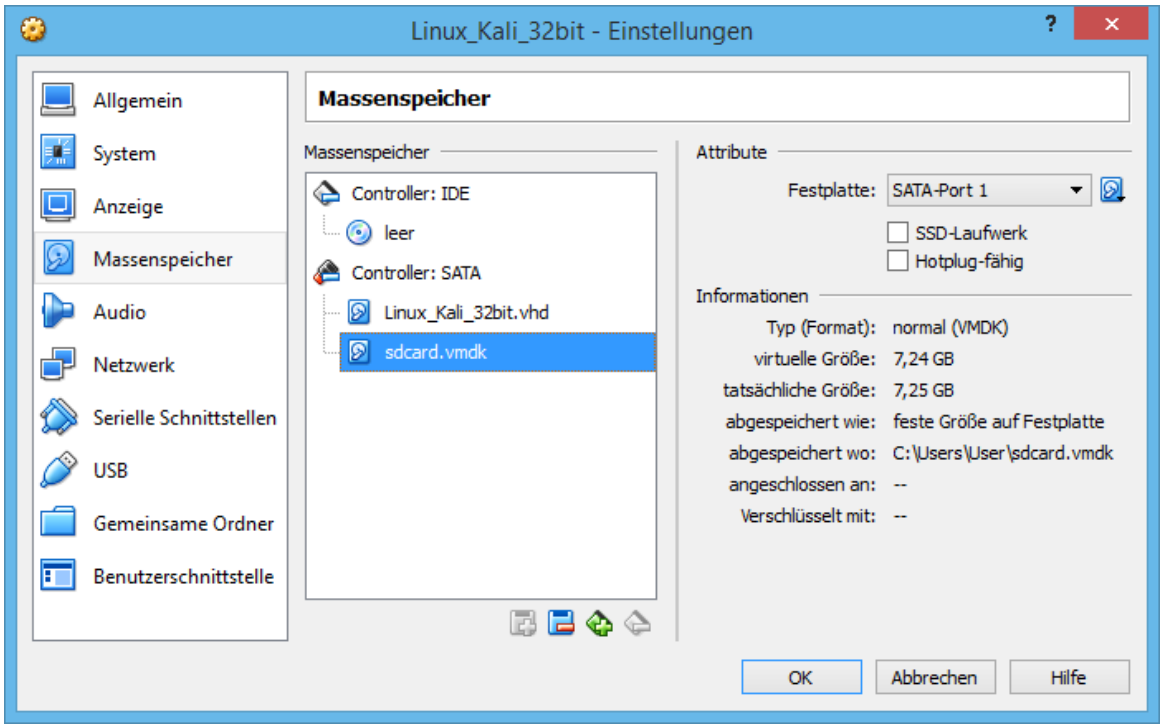
3.1.4-4 notepad.exe: sdcard.vmdk vor der Bearbeitung



3.1.4-5 notepad.exe: sdcard.vmdk nach der Bearbeitung

4. Einrichten des Massenspeichermediums an der virtuellen Maschine

Als letzte Maßnahme wird die im Schritt 2 erstellte Datei sdcard.vmdk, als Massenspeichermedium der virtuellen Maschine angehangen. Dafür muss die Virtual Box gestartet und über die Option „Ändern“, die virtuelle Maschine modifiziert werden. Unter dem Punkt Massenspeicher wird die Datei als Speichermedium hinzugefügt. Folgende Abbildung zeigt die virtuelle Maschine mit der als Speichermedium eingerichteten Datei.



3.1.4-6 Virtual Box: Einrichten des Massenspeichers

3.1.4.2 Einrichten eines Hotspots am Rechnersystem

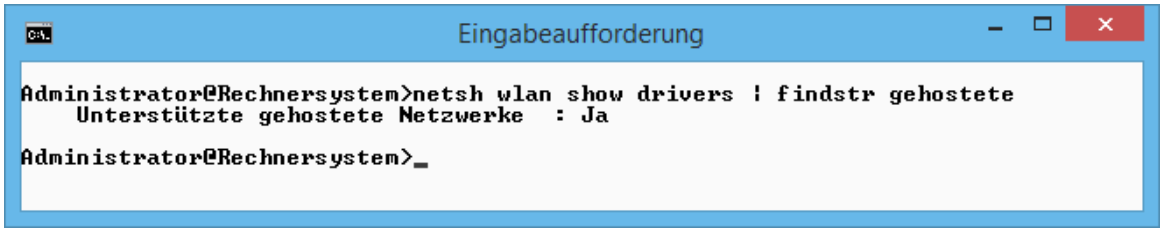
Um im späteren Verlauf dieser Arbeit den Netzwerkverkehr der Hardwareplattform mitschneiden zu können, wird die Kommunikation über das Rechnersystem geleitet. Dies kann über einen Hotspot realisiert werden. In diesem Abschnitt wird eine Konfiguration kommentiert, die es dem Rechnersystem ermöglicht über das Betriebssystem Microsoft Windows [ab Windows 8] einen Hotspot bereitzustellen, um Netzwerkgeräte den Zugriff auf das Internet zu gewähren.

1. Unterstützung gehosteter Netzwerke ermitteln

Über das von Microsoft Windows mitgelieferte Programm cmd.exe (Eingabeaufforderung) kann die Unterstützung gehosteter Netzwerke eingesehen werden:

```
netsh wlan show drivers | findstr gehostete
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.1.4-7 cmd.exe: Unterstützung für gehostete Netzwerke ermitteln

Die Funktion einen Hotspot über das im Rechnersystem vorhanden WLAN Moduls anbieten zu können, wird mit einem „Ja“ nach dem Doppelpunkt ermittelt. Die Abbildung zeigt für die Testumgebung „Ja“ an.

2. Hotspot erstellen

Weiter kann nun über das Programm cmd.exe (Eingabeaufforderung) der Hotspot mit folgendem Befehl erstellt werden:

```
netsh wlan set hostednetwork mode=allow ssid=NETZ key= abcd!1234
        keyUsage=persistent
```

Dabei sollte für die SSID (ssid) eine in dieser Lokalität nicht verwendete Zeichenfolge gebraucht und als Kennwort (key) eine sichere Zeichenfolge, bestehend aus einer zufälligen Kombination von Buchstaben, Zahlen und Symbole, mit einer Mindestlänge von 12 Zeichen, genutzt werden.

Für die Testumgebung lautet die SSID:	NETZ
und das Kennwort:	abcd!1234

Die Ausgabe an der Befehlszeile bestätigt die Einrichtung des Hotspots mit den Textzeilen:

```
Der Modus für das gehostete Netzwerk ist so festgelegt, dass das gehostete Netzwerk zugelassen wird.
Die SSID des gehosteten Netzwerks wurde erfolgreich geändert.
Die Benutzerschlüsselpassphrase des gehosteten Netzwerks wurde erfolgreich geändert.
```

3. Hotspot starten

Der Hotspot ist nun erstellt und wird für die Verwendung mit folgendem Befehl gestartet:

```
netsh wlan start hostednetwork
```

Die Ausgabe an der Befehlszeile bestätigt die Aktivierung des Hotspots mit dem Text:

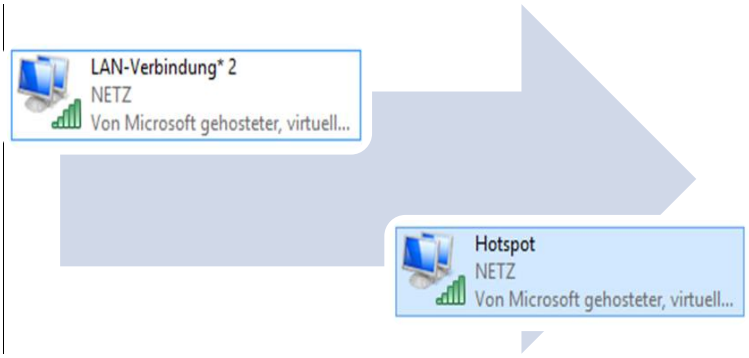
```
Das gehostete Netzwerk wurde gestartet.
```

4. Netzwerk zur Verwendung freigeben

Der Hotspot ist nun erstellt und aktiv, doch für die Verwendung muss noch die Schnittstelle zum Internet gewählt sowie der Zugriff auf dieses durch andere Geräte aktiviert werden. Für diesen Vorgang müssen über "Netzwerk- und Freigabecenter" die Adaptereinstellungen, des zuvor erstellten Netzwerkadapters und des Adapters welcher die Internetverbindung realisiert, geändert werden.

a) Umbenennen des erstellten Netzwerkadapters

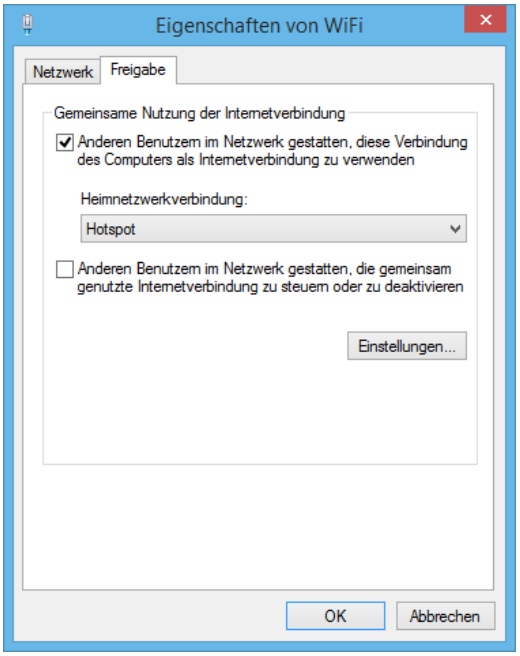
Zum einfachen Umgang wird geraten den Netzwerkadapter der den Hotspot bereitstellt umzubenennen, dieser Teilschritt ist nicht zwingend vorgeschrieben.



3.1.4-8 Microsoft Windows Netzwerkadapter umbenennen

b) Freigabe für den Hotspot einstellen

Über das Kontextmenü des Adapters der die Internetverbindung bereitstellt, muss die Auswahl „Eigenschaften“ gewählt, im Reiter „Freigabe“ der Kontrollkasten „Anderen Benutzern im Netzwerk gestatten, diese Verbindung des Computers als Internetverbindung zu verwenden“ aktiviert und im Kombinationsfeld der Hotspot ausgewählt und mit „OK“ bestätigt werden.



3.1.4-9 Microsoft Windows Netzwerkadapter: Eigenschaften von WiFi

Der Hotspot ist nun aktiv und kann genutzt werden. Die Adapter-IP wird auf 192.168.137.1/24 gesetzt und sollte nicht geändert werden. Der Client kann mit einer IP im Bereich 192.168.137.2 bis 192.168.137.254 konfiguriert werden.

3.2 Durchführung

Der folgende Abschnitt dient als praxisnahe Handlungsanweisung bei der Konfiguration der Hardwareplattform. Es wird damit begonnen, die Betriebssystem Speicherabbilder auf die dafür zur Verfügung gestellten SD-Karten zu schreiben und bestehende Fehler zu beheben. Dem folgend, wird die gewählte Hardwareplattform vorbereitet und konfiguriert. Weiter wird die Funktionalität des Android Betriebssystems dahingehend aufgewertet, dass eine spätere voll automatisierte Analyse von Android Schadsoftware durchgeführt werden kann. Diese Aufwertung wird durch zusätzliche Software in Form neuer Anwendungen oder ausführbarer Dateien bewerkstelligt. Den Schluss bildet die Installation eines Schadprogramms, gefolgt von einer ersten explorativen Analyse.

3.2.1 Vorbereiten der SD-Karten-Speicherabbilder

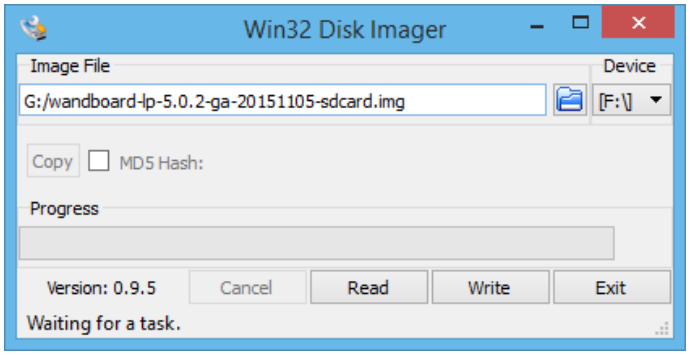
Nachdem alle Vorbereitungen zur Nutzung des Wandboards getroffen wurden, trägt der Abschnitt das Ziel, mehr über die SD-Karten-Speicherabbilder zu erfahren. Das Vorgehen zum Beschreiben des Betriebssystems auf die SD-Karte sowie die Modifikation von Partitionen in diesem, wird exemplifiziert.

Auf dem Rechnersystem werden die Speicherabbilder vorausgesetzt, die bereits in 2.2.2 SD-Karten und Speicherabbilder benannt wurden. Es handelt sich hierbei um die Betriebssystem Speicherabbilder Android 5.0 (Lollipop) vom 5. November 2015 [S-WanA16] und Android 4.4 (Kitkat) vom 3. März 2015 [S-WanB16] für das Wandboard. Die heruntergeladenen Dateien müssen mittels eines entsprechenden Entpackprogramms dekomprimiert werden, dafür bietet sich wieder die Anwendung 7-zip an [S-Szip16]. Zugleich ist es nötig, die für das Android Betriebssystem reservierte Mikro SD-Karte, in den SD-Karten-Leser des Rechnersystems direkt oder durch Zuhilfenahme eines SD-Karten-Adapters zu stecken. Dieser ist hierbei nur angebracht, wenn das Rechnersystem keinen Mikro SD-Karten-Anschluss zur Verfügung stellt, sondern nur eine Schnittstelle für Standard SD-Karten.

3.2.1.1 Speicherabbild auf die SD-Karte schreiben

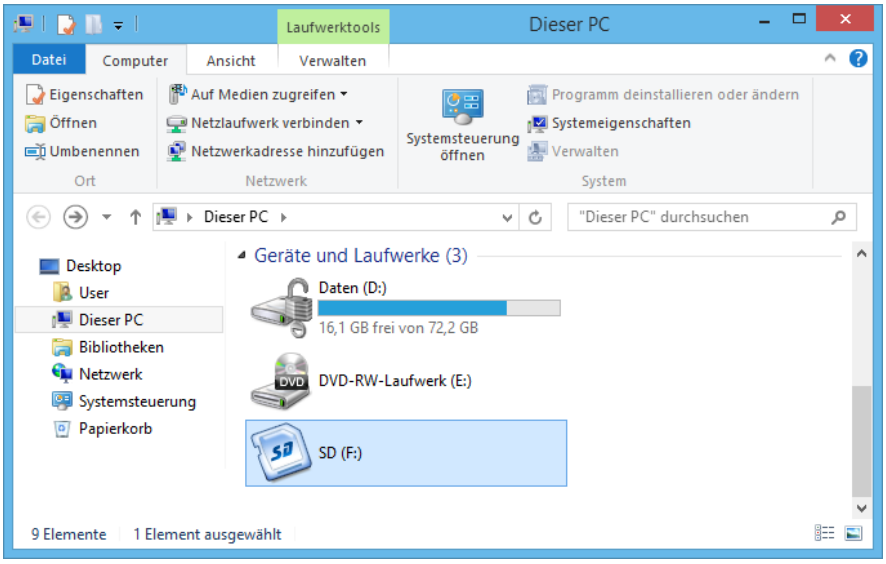
Vorgehen mit Microsoft Windows

Da Microsoft Windows keine mitgelieferte Funktionalität bietet, um die bereits heruntergeladenen Speicherabbilder auf die dafür reservierten SD-Karten zu kopieren, wird das Programm Win32 Disk Imager priorisiert. Dieses Werkzeug muss heruntergeladen [S-SoFo16], installiert und anschließend gestartet werden.



3.2.1-1 Win32 Disk Imager

Die Programmoberfläche bietet ein Feld zur Auswahl des korrekten Wechsellaufwerkes (*Device*), um einen bestimmten Wechseldatenträger zu beschreiben. Es sollte zuvor geprüft werden, dass es sich um das richtige Laufwerk handelt.



3.2.1-2 Microsoft Windows Explorer zur Laufwerkssichtung

Die Abbildung zeigt die SD-Karte unter „F:“ an. In der Abbildung 3.2.1-1 Win32 Disk Imager sind folgende Knöpfe zu sehen:

„Read“ Knopf
- erstellt ein Speicherabbild von einem Wechseldatenträger

„Write“ Knopf
- schreibt das Speicherabbild auf den Wechseldatenträger

„Exit“ Knopf
- beendet das Programm

Darüber hinaus zeigt die Abbildung 3.2.1-1 Win32 Disk Imager, noch folgendes Feld:

Speicherabbild (*Image File*)
- wird im Anschluss auf den gewählten Wechseldatenträger geschrieben

Für den Vorgang an dieser Stelle, muss der „Write“ Knopf betätigt werden.

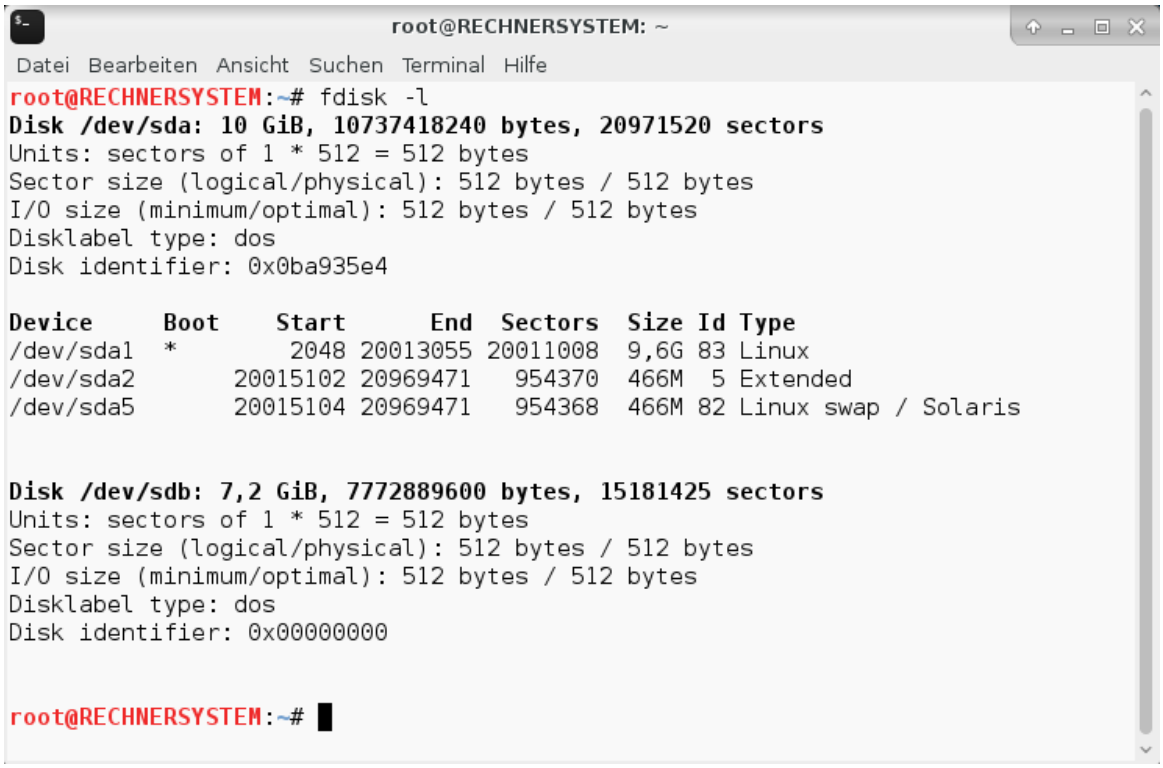
Die SD-Karte sollte während des Schreibvorganges nicht entfernt werden, da dies Datenverlust bis hin zu irreparable Schäden an der SD-Karte zur Folge haben kann. Nach Abschluss des Schreibvorgangs kann die SD-Karte aus dem SD-Karten-Lesegerät entfernt und in das Wandboard eingesetzt werden. Dies schließt das Schreiben des Speicherabbilds auf die SD-Karte ab. Das Wandboard kann nun an die Stromversorgung angeschlossen werden, um das Android Betriebssystem zu starten.

Vorgehen mit Linux

Über die Linux-Shell (Terminal) kann mittels folgenden Befehls die SD-Karte ermittelt werden:

```
fdisk -l
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.

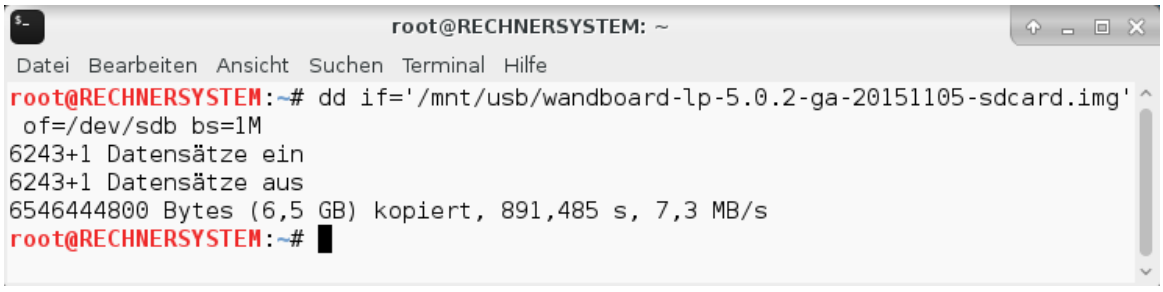


3.2.1-3 Shell: Ermitteln der SD-Karte

Zu sehen ist die Größe der SD-Karte in Bytes sowie deren Zuordnungseinheiten pro Sektor. Die SD-Karte kann über /dev/sdb angesprochen werden. Der Pfad zu dem modifizierenden Speicherabbild lautet **/mnt/usb/WB/wandboard-lp-5.0.2-ga-20151105-sdcard.img** und wird mit der Kurzschreibweise **/…/5.0.2.img** abgekürzt. Der folgende Befehl schreibt das Speicherabbild auf die SD-Karte:

```
dd if=/…/5.0.2.img of=/dev/sdb bs=1M
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.1-4 dd: Beschreiben der SD-Karte mit dem Speicherabbild

3.2.1.2 Erstellung des Speicherabbilds der SD-Karte (Sicherung)

Vorgehen mit Microsoft Windows

Es ist möglich mit dem Win32 Disk Imager Programm, Sicherungen von einer SD-Karte zu erstellen. Soll eine Sicherung angelegt werden, ist eine SD-Karte in das SD-Karten Lesegerät einzustecken und der „Read“ Knopf aus Abbildung 3.2.1-1 Win32 Disk Imager zu betätigen. Im Dialogfeld „Image File“ ist hierzu ein nicht existierender Dateipfad einzugeben und das entsprechende Wechsellaufwerk über „Device“ zu selektieren. Nach dem Klick auf „Read“ wird das Auslesen der SD-Karte gestartet und anschließend in die neu angelegte Datei geschrieben. Nach der Fertigstellung wird die Datei die Größe der verwendeten SD-Karte haben. Eine Wiederherstellung kann ohne weitere Änderungen, nur auf eine solch große oder größere SD-Karte zu Stande kommen. Das Wiederherstellen ist synchron zum Abschnitt 3.2.1.1 Speicherabbild auf die SD-Karte schreiben.

Vorgehen mit Linux

Das Befehlszeilen basierende Programm dd für die Linux-Shell kann ebenso zur Sicherung genutzt werden. Dafür ist nur eine kleine Modifikation am Befehl nötig. Folgender Befehl kann in der Linux-Shell (Terminal) eingegeben werden:

```
dd if=<Quelle> of=<Ziel> <Optionen>
```

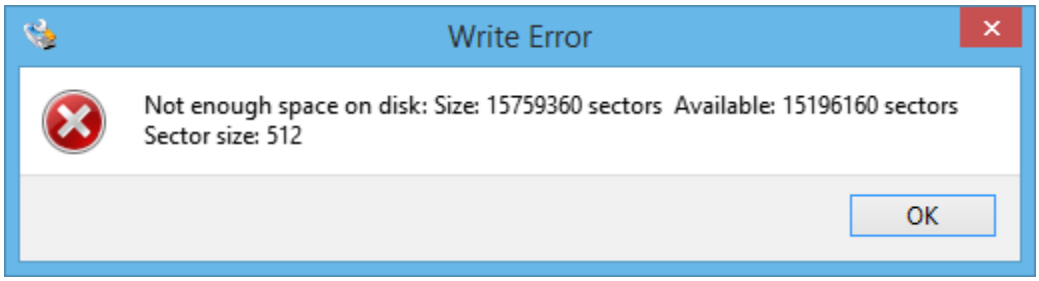
Bsp.: Quelle
/dev/sdb
Ziel
/…/backup.img
Optionen
bs=1M [bs steht für BlockSize=1 MB]

Das Programm wird ohne weitere Sicherheitsabfragen ausgeführt und überschreibt potentiell vorhandene Daten auf dem Ziel.

3.2.1.3 Speicherabbild Schreibfehler

Fehlermeldung mit Microsoft Windows

Bei dem Versuch ein Speicherabbild auf eine SD-Karte mit weniger Platz als das zu lokalisierende Abbild zu schreiben, generiert das Win32 Disk Imager Programm eine Fehlermeldung. Im Fall des Betriebssystem Speicherabbilds Android 5.0 (Lollipop) vom 5. November 2015 ist der Fehler der folgenden Abbildung zu entnehmen.



3.2.1-5 Win32 Disk Imager: Schreibfehler beim Speicherabbild Android 5.0

Fehlermeldung mit Linux

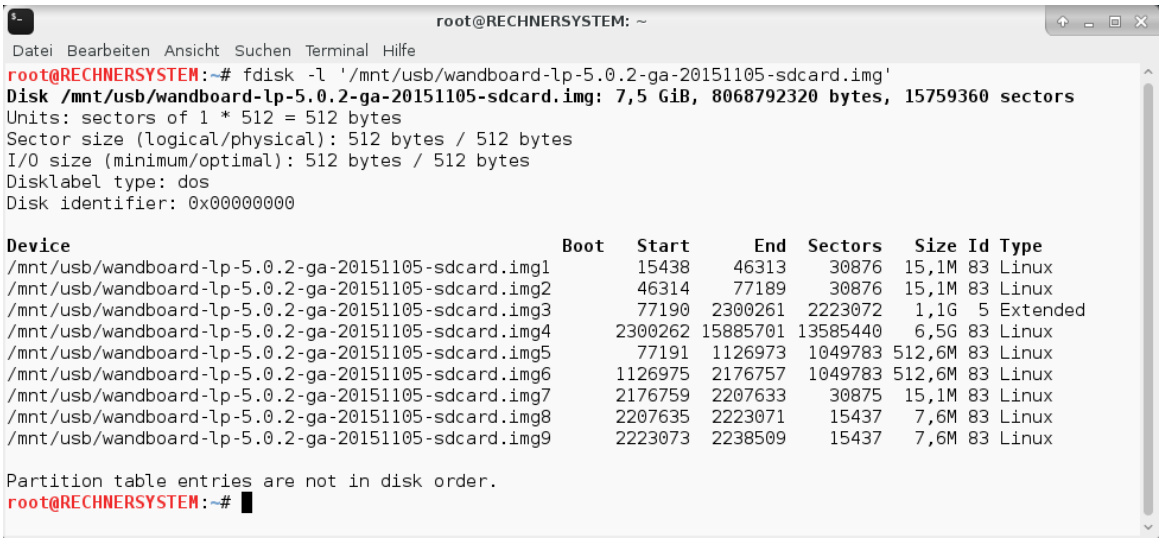
Bei dem Versuch ein Speicherabbild auf eine SD-Karte mit weniger Platz als das zu lokalisierende Abbild zu schreiben, wird unter Verwendung des Programms dd eine Fehlermeldung generiert. Der Pfad zu dem modifizierenden Speicherabbild lautet **/mnt/usb/WB/wandboard-lp-5.0.2-ga-20151105-sdcard.img** und wird mit der Kurzschreibweise **/…/5.0.2.img** abgekürzt. Im Fall des Betriebssystem Speicherabbilds Android 5.0 (Lollipop) vom 5. November 2015 ist der Fehler der folgenden Abbildung zu entnehmen.

3.2.1-6 dd: Schreibfehler beim Speicherabbild Android 5.0

Der internen Partitionierung des Speicherabbilds Android 5.0 (Lollipop) vom 5. November 2015 ist zu entnehmen, dass die letzte Partition mehr Speicherplatz allokiert (mit 15885701 Sektoren) als das Speicherabbild selbst in der Gesamtgröße zur Verfügung stellt (mit 15759360 Sektoren). Dies ist mit dem folgenden Befehl in der Linux-Shell (Terminal) ermittelbar:

```
fdisk -l /…/wandboard-lp-5.0.2-ga-20151105-sdcard.img
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.1-7 fdisk: Partitionierung des Lollipop Speicherabbilds

Zum Testen der Partitionen des Speicherabbilds auf Zulässigkeit, kann folgender Befehl in der Linux-Shell (Terminal) genutzt werden:

```
partprobe /…/wandboard-lp-5.0.2-ga-20151105-sdcard.img
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.

3.2.1-8 partprobe: Prüfung der Partitionen des Speicherabbilds

Aus Abschnitt 3.1.4.1 geht die Größe der SD-Karte mit 7772889600 Byte hervor. Der Schreibfehler kann durch eine Modifikation der internen Partitionierung des Speicherabbilds behoben werden und wird in dem folgenden Abschnitt beschrieben.

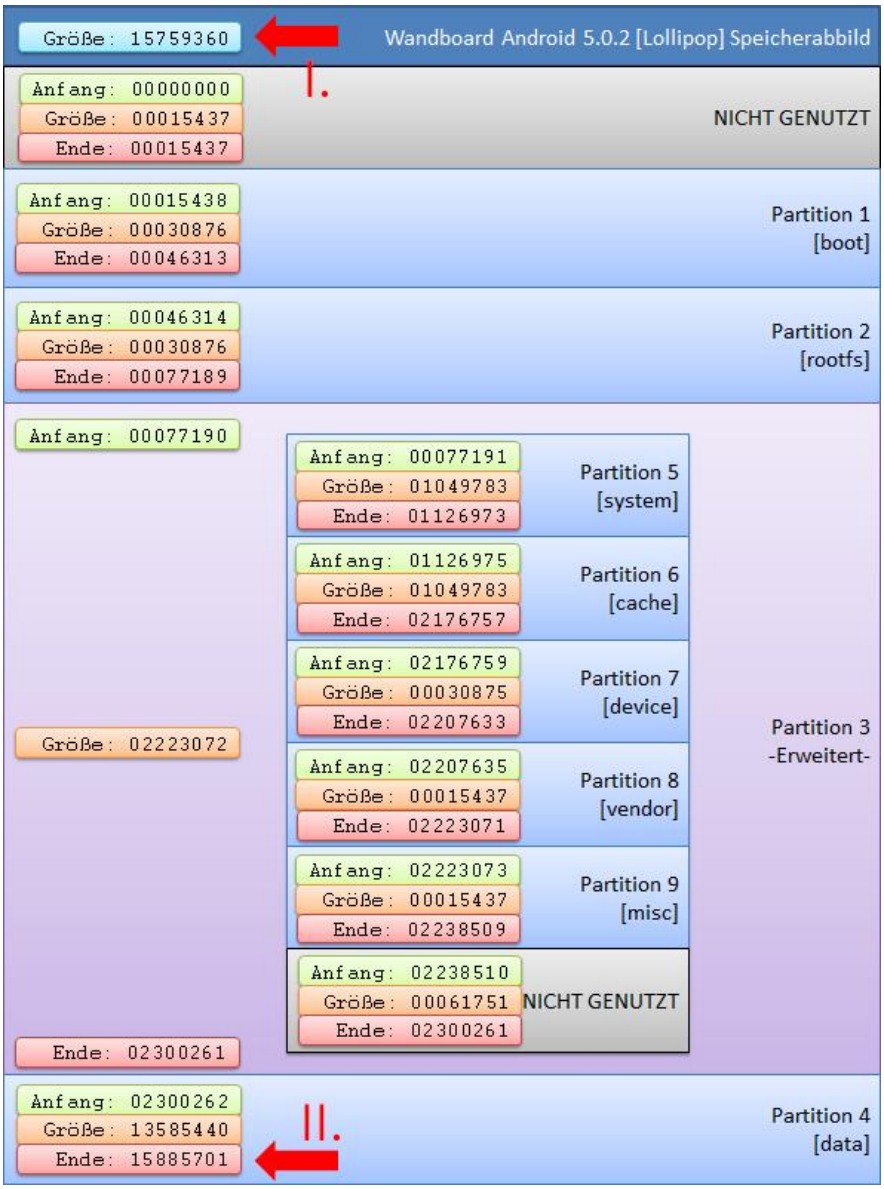
3.2.1.4 Modifikation von Partitionen eines Speicherabbilds

Vorgehen mit Microsoft Windows

Da Microsoft Windows die verwendeten Dateisystem im Speicherabbild nicht unterstützt und eine Realisierung der Modifikation von Partitionen eines Speicherabbilds nicht problemlos und unkompliziert umzusetzen wäre, wird die Modifikation von Partitionen eines Speicherabbilds nur unter Verwendung eines Linux artigen Betriebssystems erklärt.

Vorgehen mit Linux

Um den Fehler im Speicherabbild beheben zu können, sollte erst das Verständnis der Partitionierung im Speicherabbild verstanden werden. Die folgende Abbildung stellt die numerischen Werte aus der Abbildung 3.2.1-7 fdisk: Partitionierung des Lollipop Speicherabbilds als Grafik zum besseren Verständnis dar.



3.2.1-9 Grafische Veranschaulichung der internen Partitionierung

Die Abbildung zeigt grafisch die Größe in Sektoren (hellblau) des Speicherabbilds Android 5.0 (Lollipop) vom 5. November 2015 und alle Partitionen an, mit jeweils den zugehörigen Start-Sektoren (grün), End-Sektoren (rot) und der Größe in Sektoren (orange). Der Vergleich von I. mit II. in der Abbildung lässt erkennen, dass die letzte Partition über die Grenzen des Speicherabbilds hinaus geht und beim erneuten laden der Partitionierungstabelle den Fehler (vgl. Abbildung 3.2.1-8 partprobe: Prüfung der Partitionen des Speicherabbilds) auslöst. Weiterhin zeigt der Vergleich I. der Abbildung mit der aus Abschnitt 3.1.4.1 Einrichten der SD-Karten-Schnittstelle für die virtuelle Maschine bekannten Größe von 15181425 Sektoren, das die Speicherabbildgröße über die Grenzen der SD-Karte hinaus geht und beim Schreibversuch auf diese einen Schreibfehler auslöst (vgl. Abbildung 3.2.1-6 dd: Schreibfehler beim Speicherabbild Android 5.0). Diese Fehler können durch nachfolgende Korrekturen behoben werden. Für diesen Abschnitt ist der Pfad zum modifizierenden Speicherabbild immer **/mnt/usb/WB/wandboard-lp-5.0.2-ga-20151105-sdcard.img** und wird in den Befehlen mit Kurzschreibweise **/…/5.0.2.img** abgekürzt. Auf der Abbildung 3.2.1-9 der vorigen Seite ist signifikant, dass die vierte und letzte physisch angeordnete Partition eine Größe von **15885701** Sektoren hat. Daraus folgt, dass dies auch die minimale Größe für das Speicherabbild ist. Um das Speicherabbild regelkonform bearbeiten zu können, wird es zunächst auf die richtige Größe gebracht. Folgender Befehl muss in der Linux-Shell (Terminal) genutzt werden:

```
truncate --size=$[(15885701+10)*512] /…/5.0.2.img
```

Dieser Befehl setzt die Größe des Speicherabbilds auf 15885711 Sektoren, dabei wird Speicher am Ende des Speicherabbilds hinzugefügt. Um nun mit dem Speicherabbild weiter arbeiten zu können, wird dieses als Loop-Datei in das System eingebunden, da Änderungen nicht direkt an der Abbilddatei angewandt werden. Zum Einbinden des Speicherabbilds muss folgender Befehl in der Linux-Shell (Terminal) eingegeben werden:

```
losetup /dev/loop0 /…/5.0.2.img
```

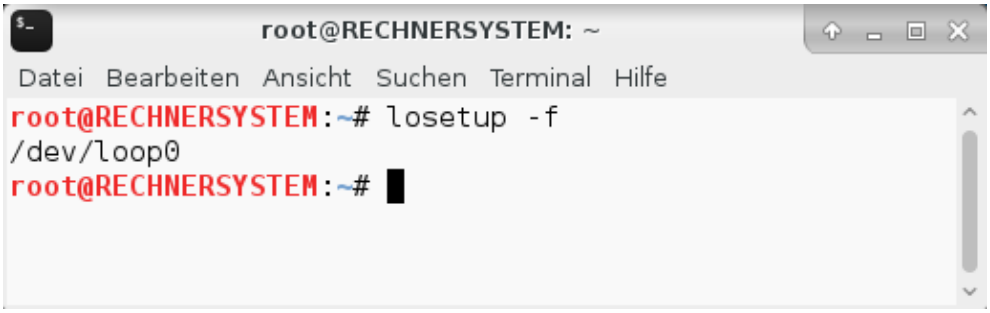
Fortan ist das Speicherabbild über **/dev/loop0** ansprechbar. Für die jeweiligen Partitionen muss dieser Beschreibung nur „p“ für Partition mit der entsprechenden Partitionsnummer angehangen werden.

Für die Verwendung des richtigen Loop-Gerätes ist es unter Umständen notwendig, das Erste freie Loop-Gerät zum Einbinden von regulären Dateien oder Blockgeräten zu identifizieren.

Zur Umsetzung dieser Prozedur muss folgender Befehl in der Linux-Shell (Terminal) ausgeführt werden:

```
losetup -f
```

Folgende Abbildung zeigt eine mögliche Ausgabe.



3.2.1-10 losetup: Anzeigen des ersten freien Loop-Geräts

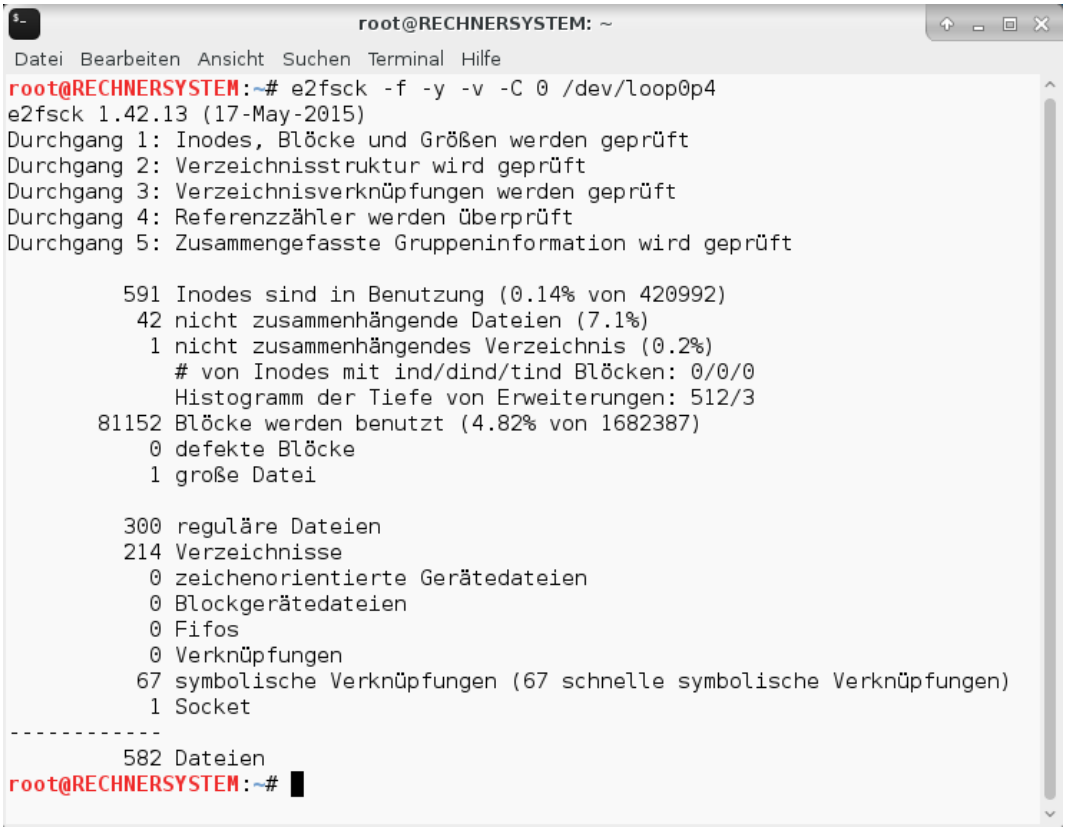
Ein anschließend erneutes Laden der Partitionierungstabelle, um weitere Änderungen durchführen zu können, muss mit dem folgenden Befehl durchgeführt werden:

```
partprobe /dev/loop0
```

Dem Schritt der Größenänderung muss eine Dateisystemprüfung vorangehen, um weiter an dem Dateisystem arbeiten zu können. Für die Dateisystemprüfung muss folgender Befehl in der Linux-Shell (Terminal) eingegeben werden:

```
e2fsck -f -y -v -C 0 /dev/loop0p4
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.

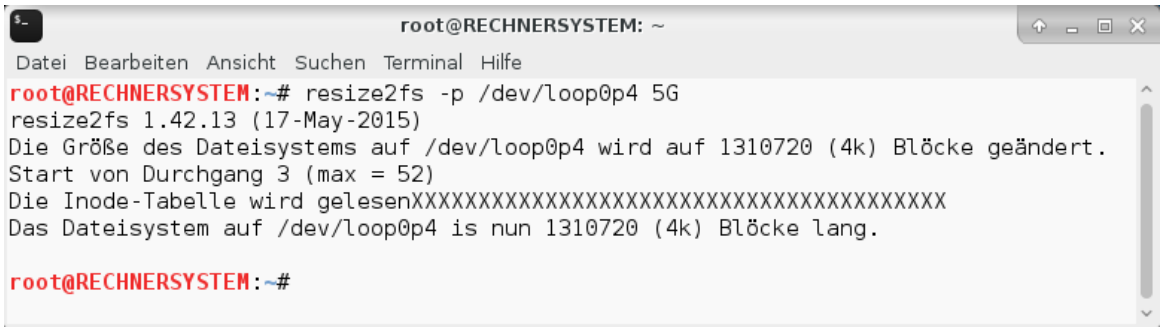


3.2.1-11 e2fsck: Überprüfung der Partition 4 des Speicherabbilds

Anschließend kann das Dateisystem auf die Gewünschte Größe geändert werden, einzige Restriktionen sind die minimale Größe von ca. 300 MB durch schon vorhandene Daten und die maximale Größe der SD-Karte. In diesem Abschnitt wird für die Partition 4 (Daten Partition) im Speicherabbild die feste Größe von **5 GB** verwendet. Folgender Befehl muss in der Linux-Shell (Terminal) ausgeführt werden:

```
resize2fs -p /dev/loop0p4 5G
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.1-12 resize2fs: Größenanpassung des Dateisystems der Partition 4

Ein anschließend erneutes Laden der Partitionierungstabelle muss wieder mit folgendem Befehl durchgeführt werden:

```
partprobe /dev/loop0
```

Da nun die Größe des Dateisystems angepasst ist, muss noch die Partitionierungstabelle des Speicherabbilds an diese Größe angeglichen werden. Für diese Angleichung ist die Größe der Partition 4 mit **13585440** Sektoren notwendig, die aus Abbildung 3.2.1-9 hervorgeht und die neu zu setzende Größe mit **5 GB**. Der nun zu berechnende Wert ist $5\,GB * 1024^3$ $\left(GB \xrightarrow{1} MB \xrightarrow{2} KB \xrightarrow{3} Byte\right)$ geteilt durch 512 (512 Byte pro Sektor). Es ergeben sich **10485760** Sektoren. Zur Umsetzung muss folgender Befehl in der Linux-Shell (Terminal) ausgeführt werden:

```
sfdisk -d /dev/loop0 | sed -e 's/13585440/10485760/g' | sfdisk /dev/loop0
```

Dieser Befehl besteht aus einer Doppel Pipeline, wobei der erste Teil die aktuelle Partitionierungstabelle ermittelt, der zweite Teil die Größe der Partition 4 in der Ausgabe ändert und der dritte und letzte Teil die Partitionierungstabelle mit der Änderung als neue Partitionierungstabelle setzt. Die Ausgabe des Befehls kann in der Anlage, Teil A eingesehen werden.

Ein anschließend erneutes Laden der Partitionierungstabelle, muss wieder mit folgendem Befehl durchgeführt werden:

```
partprobe /dev/loop0
```

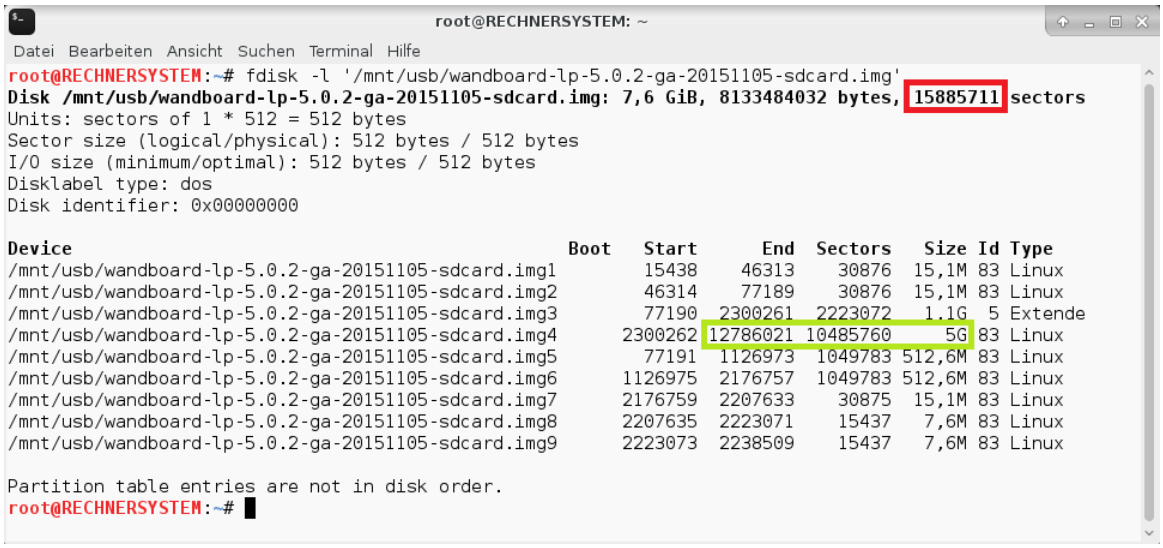
Nun muss noch das Speicherabbild aus dem Loop-Gerät entfernt werden, dies ist durch folgenden Befehl in der Linux-Shell (Terminal) zu realisieren:

```
losetup -d /dev/loop0
```

Dabei ist zu beachten, dass bei Verwendung eines anderen Loop-Gerätes dieses hier freigegeben werden sollte. Ein Blick in das Speicherabbild zeigt nun die angewendeten Änderungen an der Partition 4 ohne Datenverlust (grüner Rahmen). Folgender Befehl muss in der Linux-Shell (Terminal) ausgeführt werden:

```
fdisk -l /…/5.0.2.img
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.1-13 fdisk: Änderungen an der Partition 4 des Speicherabbilds

Im letzten Schritt muss noch das Speicherabbild an die Größe der SD-Karte beziehungsweise die minimale Größe, bedingt durch die interne Speichernutzung, angepasst werden. Für den in diesen Abschnitt getroffenen Speicherbereich, wird der interne Platzverbrauch der aus der vorangehenden Abbildung mit **12786021** Sektoren hervorgeht (grüner Rahmen), genutzt.

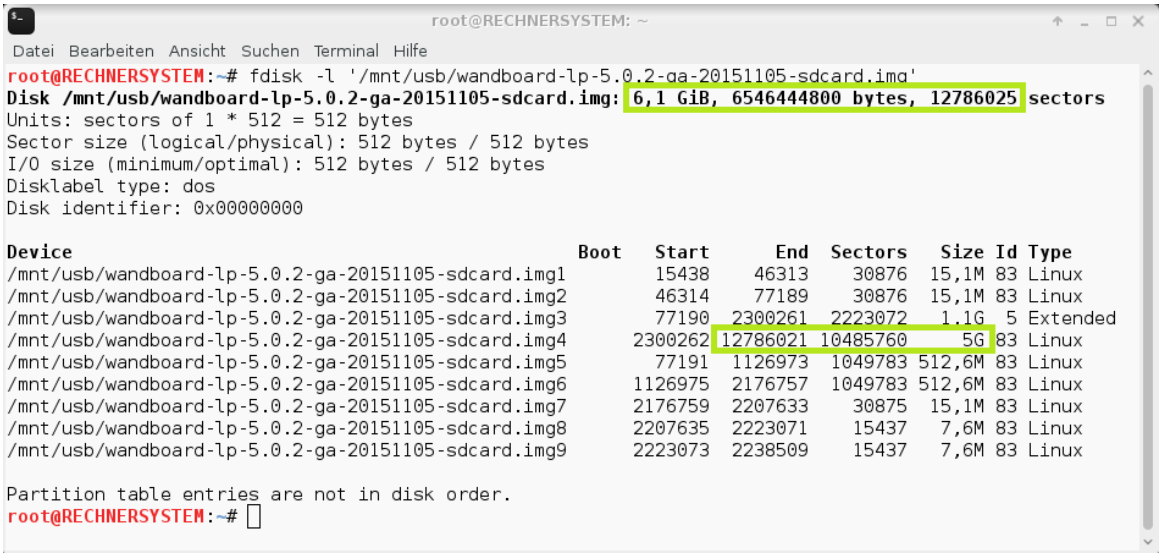
Für diesen Schritt muss folgender Befehle in der Linux-Shell (Terminal) eingegeben werden:

```
truncate --size=$[(12786021+4)*512] /…/5.0.2.img
```

Dieser Befehl setzt die Größe des Speicherabbilds auf **12786025** Sektoren, dabei werden alle Daten am Ende der Partition abgeschnitten. Eine erneute Überprüfung des geänderten Speicherabbilds Android 5.0 (Lollipop) vom 5. November 2015 ist mit dem folgenden Befehle in der Linux-Shell (Terminal) durchführbar:

```
fdisk -l /…/5.0.2.img
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.1-14 fdisk: Änderung am Speicherabbild

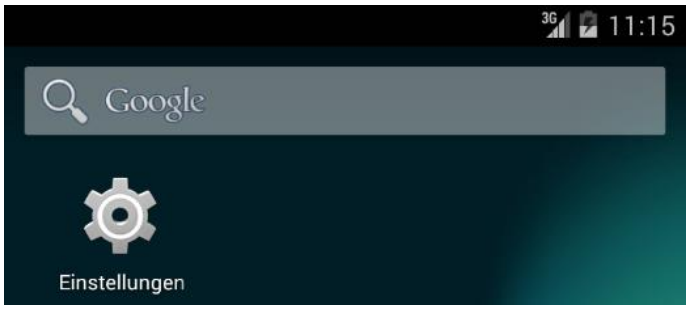
Die Änderungen am Speicherabbild und an der Partition 4, sind durch grüne Rahmen in der letzten Abbildung gekennzeichnet. Nun ist das Speicherabbild zur Nutzung konfiguriert und für den Schreibvorgang betriebsbereit. Das Abbild kann als ZIP Datei, mit dem Namen **wandboard-lp-5.0.2-ga-20151105-sdcard_MODIFIED.zip** im DVD-Pfad **/SD-Karten_Speicherabbilder,** der Projekt-DVD entnommen werden.

3.2.2 Verbinden des Rechnersystems mit Android über ADB

Für diesen Abschnitt wird das Rechnersystem (Microsoft Windows) benötigt, der Pfad zur Android Debug Bridge lautet **C:\SDK\platform-tools**. Wie bisher muss das Wandboard ordnungsgemäß angeschlossen, eingeschaltet und mit einer Mikro SD-Karte bestückt sein, auf der sich ein lauffähiges Android Betriebssystem befindet, dieses Vorgehen wird in Abschnitt 3.2.1 Vorbereiten der SD-Karten-Speicherabbilder erklärt. Weiter wird für diesen Abschnitt das Betriebssystem Android 4.4 (Kitkat) vom 3. März 2015 verwendet. Die Vorgehensweise für die Einrichtung der Verbindung der anderen Android Betriebssysteme, ist die Gleiche wie folgend beschrieben.

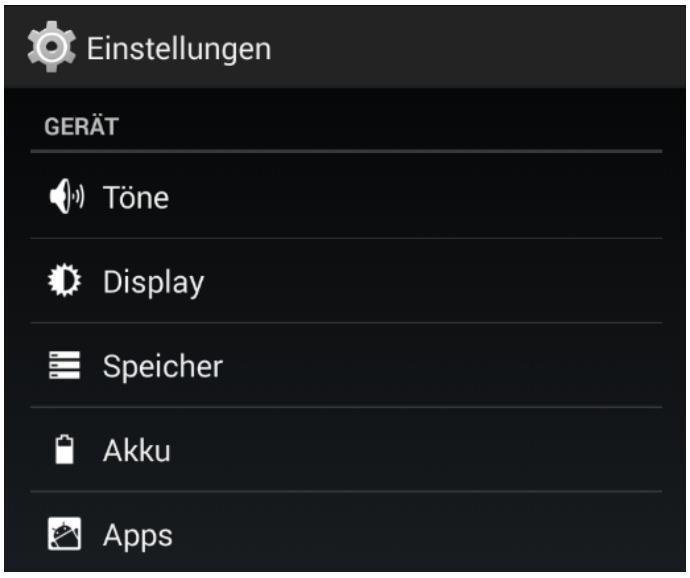
3.2.2.1 Konfiguration am Android Betriebssystem

Unter Umständen ist es Notwendig, die Entwickleroptionen am Android Betriebssystem zu aktivieren, sollte dies nicht schon bei der Erstellung des Betriebssystem Speicherabbilds durch den Entwickler erfolgt sein. Über die Android Benutzeroberfläche muss „Einstellungen“ (*Settings*) gewählt werden.



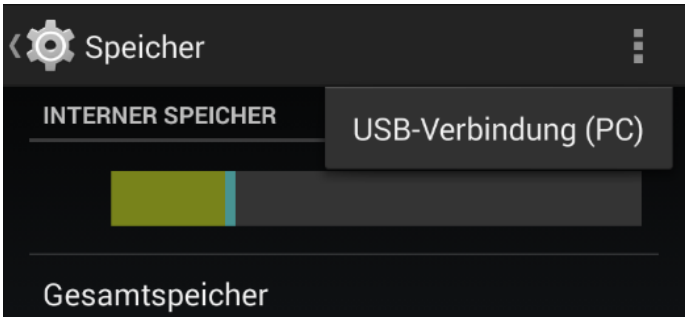
3.2.2-1 Android: Benutzeroberfläche Hauptbildschirm

Über die „Einstellungen“ (*Settings*) ist nun „Speicher“ (*Storage*) zu wählen.



3.2.2-2 Android: Geräteeinstellungen

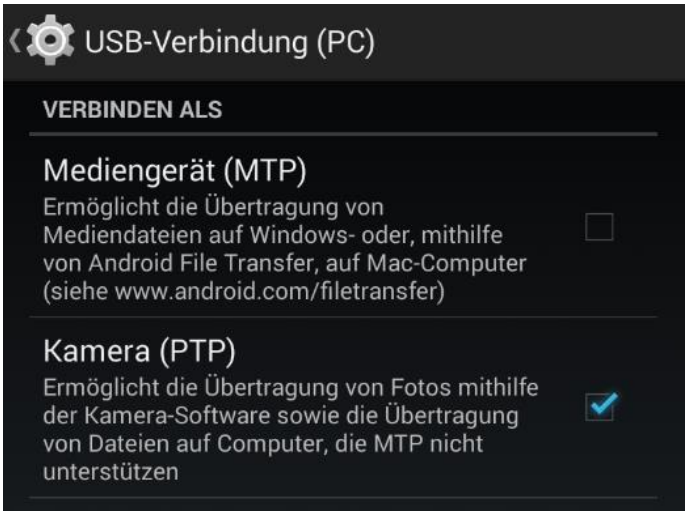
Wird dieser ausgewählt, werden Informationen über sämtliche Speichermedien und deren Platzbelegungen ausgegeben. Im Menü muss nun „USB-Verbindung (PC)“ (*USB computer connection*) gewählt werden, da die Standardkonfiguration das Gerät noch nicht als Speichermedium zur Verbindung frei gibt.



3.2.2-3 Android: Speichereinstellungen

Ist diese Option bedingt durch das Betriebssystem Android 6.0 (Marshmallow) nicht vorhanden, kann unter Verwendung der Benachrichtigungsleiste, die Option „USB zum Aufladen“ (*USB for charging*) auf „Fotoübertragung (PTP)“ (*Photo transfer (PTP)*) geändert werden.

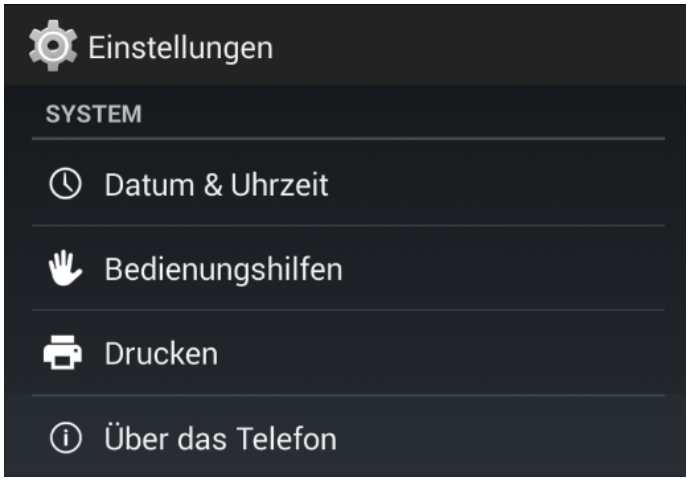
Um nun eine Verbindung über USB realisieren zu können, muss der Modus „Kamera (PTP)“ (*Camera (PTP)*) gewählt werden wie in der unteren Abbildung zu sehen ist.



3.2.2-4 Android: USB-Verbindungs-Modus

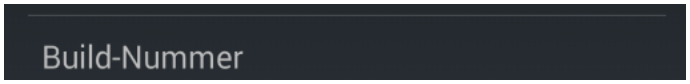
In dem Moment wo der Modus von Android zu „Kamera (PTP)“ gewechselt wird und die Hardwareplattform am Rechnersystem über USB verbunden ist, kann am Rechnersystem ein Verbindungsaufbau über die USB-Schnittstelle des angeschlossenen Systems erkannt werden.

Weiter muss nun über die „Einstellungen“ (*Settings*) in die „Entwickleroptionen“ (*Developer options*) gewechselt werden. Falls diese nicht sichtbar sind, ist eine Aktivierung wie folgend Beschrieben möglich:



3.2.2-5 Android: Systemeinstellungen

Hierzu wird „Über das Telefon“ (*About phone*) oder „Über das Tablet“ (*About tablet*) gewählt und siebenmal die „Build Nummer“ (*Build number*) geklickt. Ein kleiner Countdown zeigt die verbleibenden Klicks an.

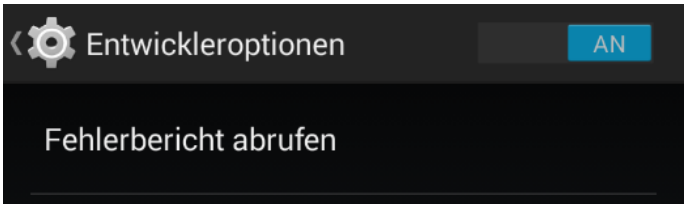


3.2.2-6 Android: Über das Telefon/Tablet

Nach dem letzten Klick erscheint die Information „Sie sind jetzt ein Entwickler“ (*You have enabled development settings!*), somit sind die Entwickleroptionen freigeschaltet.

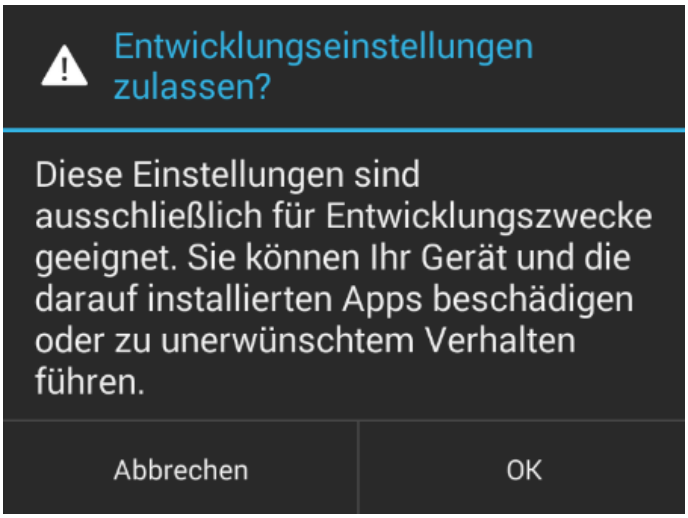
3.2.2-7 Android: Systemeinstellungen

Über die Entwickleroptionen müssen diese nun aktiviert werden durch verschieben des Schiebereglers auf „AN“ (*ON*).



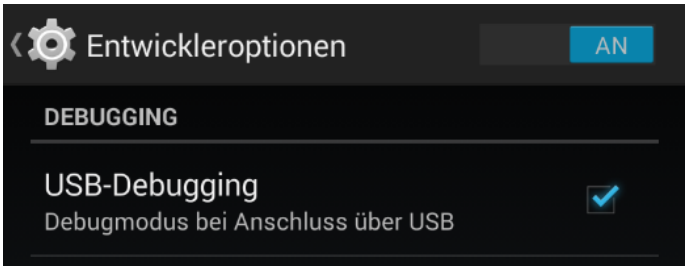
3.2.2-8 Android: Entwickleroptionen

Bei der Aktivierung der Entwickleroptionen muss die generierte Systemmeldung mit „OK“ bestätigt werden.



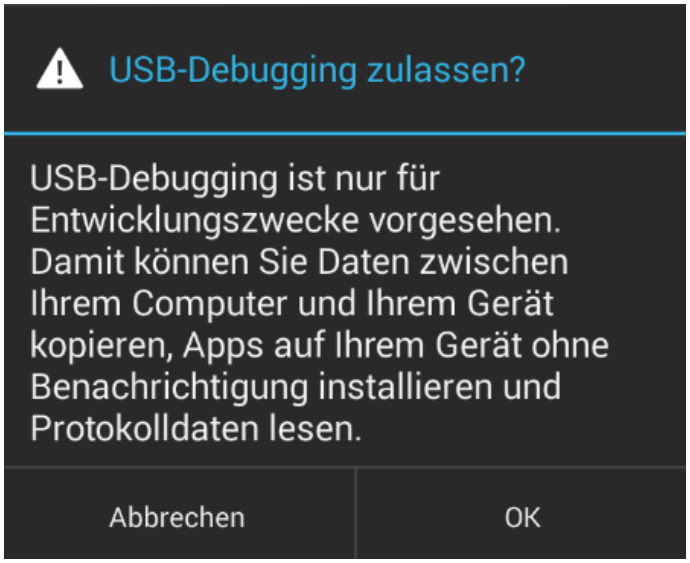
3.2.2-9 Android: Warnhinweis zu Entwicklungseinstellungen

Im letzten Schritt muss nun noch der Kontrollkasten „USB-Debugging“ (*USB debugging*) unter den Debugging Optionen aktiviert werden.



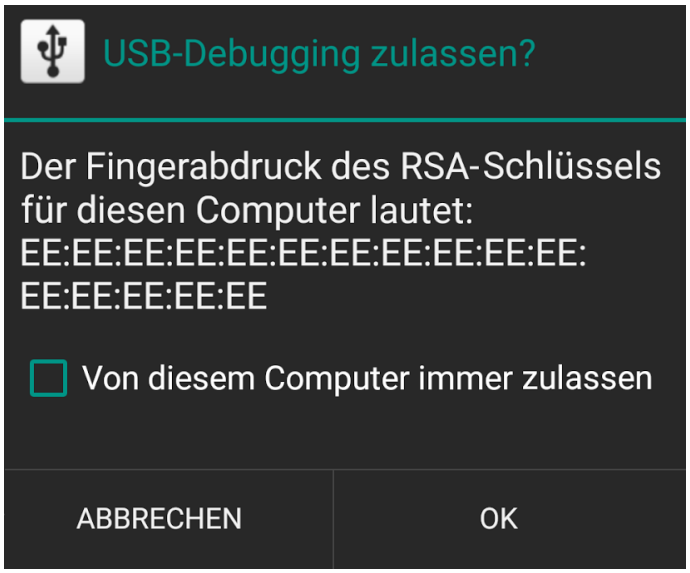
3.2.2-10 Android: USB-Debugging aktivieren

Auch hier muss eine vom System generierte Meldung mit „OK“ bestätigt werden



3.2.2-11 Android: Warnhinweis zu USB-Debugging

Ist das angeschlossene Rechnersystem durch diese Prozedur das erste Mal mit dem Android Betriebssystem verbunden, meldet dies dem Android Betriebssystem ab der Version 4.2.2 durch ein Anfrage über die RSA-Schlüssel Signatur. Folgende Abbildung zeigt dies am Beispiel.



3.2.2-12 Android: USB-Debugging Vertrauensstellung zum Rechnersystem

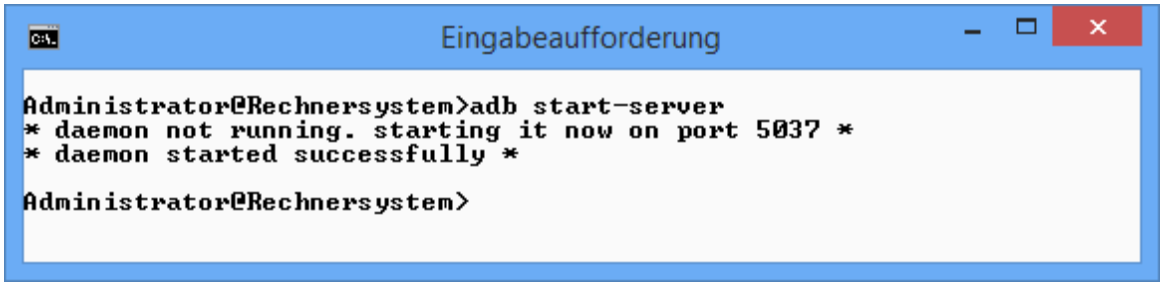
Die Abfrage für eine erfolgreiche Verbindung ist mit „OK“ zu quittieren. Soll dem Rechnersystem diese Verbindung immer wieder zur Verfügung gestellt werden, ist zusätzlich der Kontrollkasten zu aktivieren. Die Verbindung vom Android Betriebssystem zum Rechnersystem ist nun für die Kommunikationsschnittstelle USB am Android Betriebssystem konfiguriert.

3.2.2.2 Konfiguration am Rechnersystem

Um nun die Android Debug Bridge ohne Einschränkung nutzen zu können, muss der Hintergrundprozess mit folgendem Befehl über das Programm cmd.exe gestartet werden:

```
adb start-server
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.2-13 adb: Start des ADB Serverprozess

An dieser Stelle ist nun zu wählen über welche Schnittstelle mit dem Android Betriebssystem kommuniziert werden soll.

Zur Auswahl steht die Verbindung über USB und die Verbindung über WLAN, die in den folgenden zwei Abschnitten genauer erläutert wird.

Für dieses Projekt ist anzustreben, dass sämtliche Kommunikation über WLAN möglich ist und lediglich die erste Konfiguration über die USB Schnittstelle zu Stande gebracht wird. Die Vertrauensstellungen zu den Fernsteuerungsrechnern kann in einer Betriebssystem Sicherung der jeweiligen Android Version hinterlegt werden und muss so, nicht nach jeder Installation erneut konfiguriert werden.

Kommunikation über USB

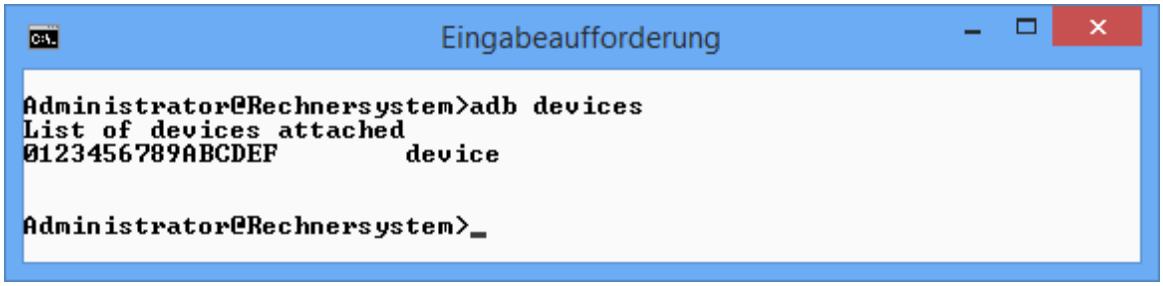
Für den ersten Verbindungsaufbau mit dem Gerät ist es unausweichlich die Verbindung über USB zu realisieren, um eine Vertrauensstellung mit dem Rechnersystem einzugehen. Mit folgendem Befehl über das Programm cmd.exe wird die Schnittstelle über USB aktiviert:

```
adb usb
```

ADB gibt die Meldung „restarting in USB mode“. Als Referenz zur angeschlossenen Hardwareplattform wird die Seriennummer des zu konfigurierenden Systems benötigt, da bei Verbindungen mit mehreren Hardwareplattformen eine Unterscheidung erfolgen muss. Zur Sichtung sämtlicher angeschlossenen und über ADB anzusteuernde Geräte, muss folgender Befehl über das Programm cmd.exe ausgeführt werden:

```
adb devices
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.2-14 adb: Anzeigen der angeschlossenen Geräte

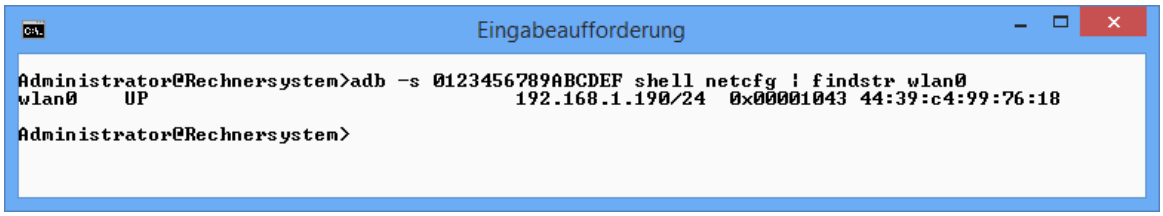
Ist das Android Betriebssystem erst nach dem Wechsel des Verbindungsmodus „Kamera (PTP)“ und der Aktivierung des „USB-Debugging“ an das Rechnersystem angeschlossen worden, wird spätestens jetzt vom Android Betriebssystem auf der Benutzeroberfläche die Abfrage zum Verbindungsaufbau mit Vertrauensstellung generiert und muss vom Benutzer manuell bestätigt werden. Ab diesem Zeitpunkt ist die Ansteuerung von Android über die USB Schnittstelle mittels der Android Debug Bridge möglich.

Kommunikation über WLAN

Wurde das Gerät schon einmal über USB verbunden und über WLAN konfiguriert, ist zudem das Netzwerk erreichbar und das verwendete Rechnersystem dem Android Betriebssystem bekannt, so lässt sich die Verbindung über WLAN herstellen. Mit folgendem Befehl über das Programm cmd.exe kann bei bestehender USB Verbindung, die IP-Adresse des Gerätes ermittelt werden:

```
adb -s 0123456789ABCDEF shell netcfg | findstr wlan0
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.2-15 adb: IP-Adresse des Android System über ADM ermitteln

Mit folgendem Befehl wird die Schnittstelle über WLAN und einem Port nach Wahl aktiviert:

```
adb tcpip 5555
```

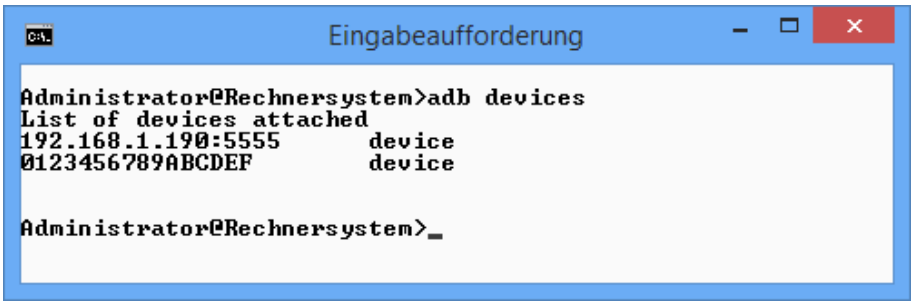
Darauf meldet ADB „restarting in TCP mode port: 5555“. Als Referenz zur angeschlossenen Hardwareplattform dienen die zuvor ermittelte IP-Adresse 192.168.1.190 und der gewählte Port 5555. Folgender Befehl richtet die Verbindung zum Android Betriebssystem über WLAN her:

```
adb connect 192.168.1.190:5555
```

Darauf meldet ADB „connected to 192.168.1.190:5555“ und die Verbindung wurde etabliert. Zur Sichtung sämtlicher angeschlossenen und über ADB anzusteuernden Geräte dient folgender Befehl:

```
adb devices
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



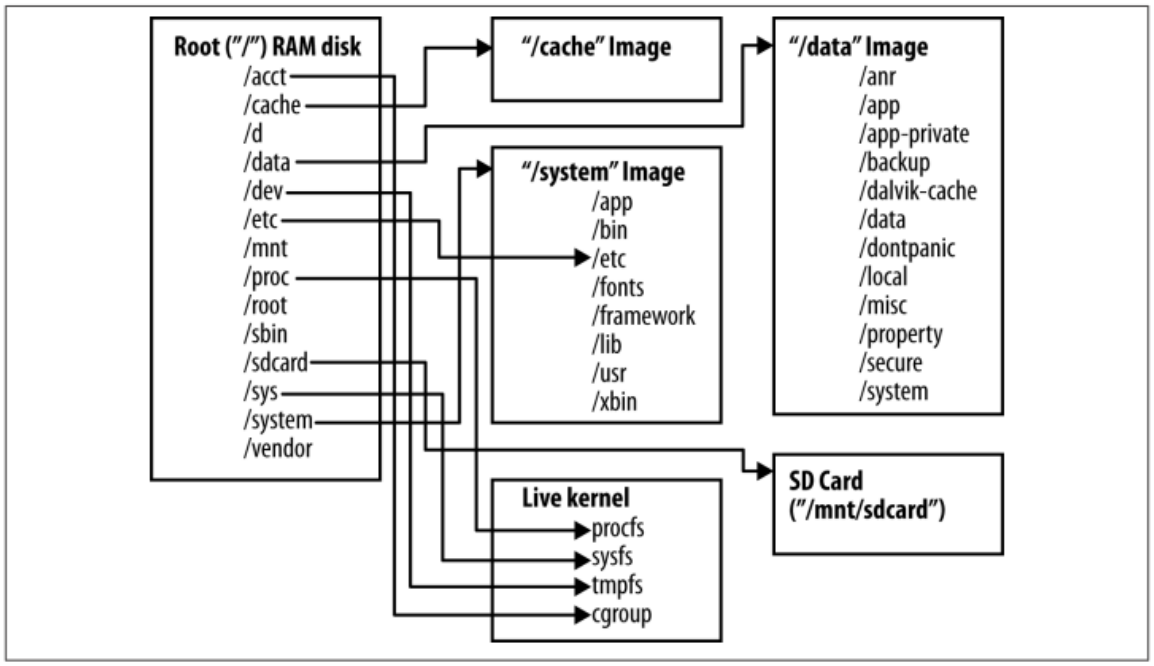
3.2.2-16 adb: Anzeigen der angeschlossenen Geräte

3.2.3 Einhängen des Dateisystems mit bestimmten Zugriffsrecht

Um Dateien im Dateisystem des Android Betriebssystems zu platzieren oder auch bestehende Dateien modifizieren zu können, ist das Verständnis über das verwendete Dateisystem und der Zugriff auf das Speichermedium von übermäßiger Bedeutung.

3.2.3.1 Android Dateisystem

Die einzelnen Verzeichnisse des Android Dateisystems werden typischerweise separat in das System eingehängt, da jedes Verzeichnis durch die differierende Art der Speichervorrichtung oder Technologie anders verwendet wird.



3.2.3-1 Android Root-Dateisystem [L-Yagh13, Seite 176]

Die Abbildung zeigt die Beziehungen zwischen den Verzeichnissen zur Laufzeit des Systems bei einem Android 4.2 (Jellybean), welches im Wesentlichen zu den Android Betriebssystemen die in dieser Arbeit Verwendung finden, gleichgestellt ist.

Die Systemanwendungen können im laufenden Betrieb nicht deinstalliert werden, da das Verzeichnis **/system** standardmäßig schreibgeschützt (read-only) eingebunden ist. Diese Restriktion gilt nicht für das Verzeichnis **/data**, denn hier werden alle Anwendungen im Laufe der Zeit durch den Android Benutzer installiert.

In den oben genannten Verzeichnissen gibt es jeweils noch mindestens ein Unterverzeichnis, welches die installierten Anwendungen beinhaltet. Folgende Pfade sind möglich:

System

/system/app
- Standard Anwendungsverzeichnis für das System

/system/priv-app
- Standard Anwendungsverzeichnis für Anwendungen mit Signatur oder System (vgl. Abschnitt 2.3.7 Berechtigungen)

Benutzer

/data/app
- Standard Anwendungsverzeichnis für den Benutzer

/data/app-asec
- Anwendungsverzeichnis für verschlüsselte Anwendungen

/data/app-lip
- Verzeichnis von lip‘s installierter Anwendungen

/data/app-private
- Legitimitätsprüfung von Anwendungen

Für den Abschnitt 3.2.4 Einrichten des Play Stores auf dem SD-Karten Speicherabbild ist nun auch klar, dass der Google Play Store nach **/system/priv-app** installiert werden sollte, da dieser durch Signaturen auf Basis der Android Version legitimiert wird. Es gibt mehrere verschiedene Einrichtungsmöglichkeiten um den Play Store auf dem Android Betriebssystem nutzen zu können. Im Abschnitt 3.2.4 werden zwei Möglichkeiten detailliert beschrieben.

3.2.3.2 Bestehendes Dateisystem mit bestimmten Zugriffsrecht ins System einbinden

Um das Dateisystem von nur lesend (read-only), auf nun schreibend (writable) neu einzubinden, muss folgender Befehl über das Programm cmd.exe ausgeführt werden:

```
adb -s 0123456789ABCDEF remount
```

Da der Ersteller des Betriebssystem Speicherabbilds schon den Root-Zugriff über die Android Debug Bridge eingerichtet hat, meldet ADB „remount succeeded“, andernfalls würde ADB „permission denied“ melden. Die Umsetzung dass das Dateisystem mit schreibenden Zugriffsrechten neu eingebunden wird, jedoch auch über das Android Betriebssystem selbst erfolgen.

Um sich mit Root-Rechten am Android Betriebssystem anzumelden, muss folgender Befehl über das Programm cmd.exe ausgeführt werden:

```
adb root
```

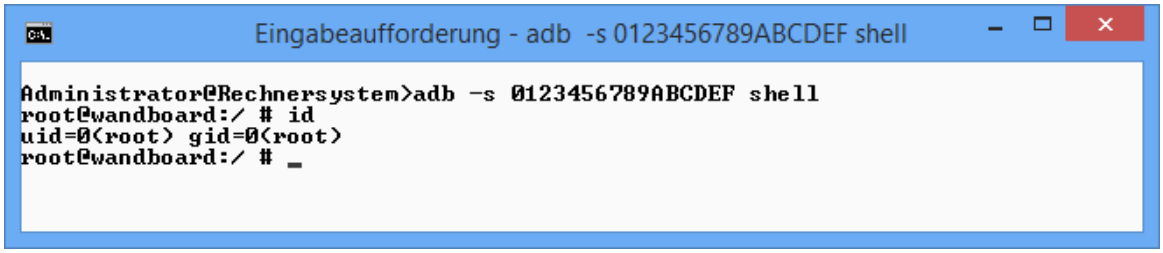
Darauf meldet ADB „restarting adbd as root“, will heißen das der Server Prozess der als Hintergrunddienst läuft, neugestartet wird und fortan als „root“ agiert. Weiter kann dann eine Shell auf dem Android gestartet und über die Android Debug Bridge gesteuert werden. Die Anmeldung am Android Betriebssystem wird mit folgendem Befehl gewährt:

```
adb -s 0123456789ABCDEF shell
```

Die Shell am Android Betriebssystem ist vom Benutzer „root“ geöffnet und hat bei jeder Aktion auch dessen Berechtigungen. Mit dem folgenden Befehl kann die Benutzer- und Gruppen-ID des aktiven Benutzers am System eingesehen werden:

```
id
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diese Befehle.



3.2.3-2 adb: Anmeldung am Android Betriebssystem als root

Der Aktive Benutzer am System hat die Benutzer-ID 0 und hat somit die höchsten Rechte.

Nach diesem Schritt durch folgenden Befehl, das Dateisystem lesend oder schreibend eingebunden werden:

Lesend

mount -o ro,remount /system

Schreibend

mount -o rw,remount /system

Der letzte Befehl sorgt für das erneute Einbinden eines bestehenden Dateisystems im System mit bestimmtem Zugriffsrecht. Dabei stehen die Kürzel:

ro	für read-only (nur lesend)
rw	für writable (schreibbar)

3.2.4 Einrichten des Play Stores auf dem SD-Karten Speicherabbild

Der Google Play Store ist auf Android basierenden Betriebssystemen vorinstalliert und kann Smartphones und Tablets vorgefunden werden. Im Fall der vorkonstruierten Android Speicherabbilder ist dies jedoch nicht der Fall und muss bei deren Verwendung, nachinstalliert werden. Für die Betriebssysteme Android 6.0 (Marshmallow) vom 28. April 2016 [S-AnFiC16], Android 5.0 (Lollipop) vom 5. November 2015 [S-AnFiA16], Android 4.4 (Kitkat) vom 3. März 2015 [S-AnFiB16] und Android 4.3 (Jellybean) vom 2. März 2015 [S-MeFiA16] muss der Play Store über das XDA Entwickler Forum heruntergeladen werden oder ist als ZIP Datei der beiliegenden Projekt-DVD im DVD-Pfad **/Methoden/Install_googleapps**, dieser Arbeit zu entnehmen.

Nach dem erfolgreichen Download müssen die Dateien mittels einem geeigneten Werkzeug entpackt werden, hierfür bietet sich wieder die Anwendung 7-zip an [S-Szip16].

Die benötigten Anwendungen für die jeweilige Betriebssystem Versionen sind folgende:

Android 6.0 (Marshmallow) vom 28. April 2016

PrebuiltGmsCore.apk	[dynamic\PrebuiltGmsCore\arm\priv-app\…]
GoogleLoginService.apk	[system/priv-app/…]
GoogleServicesFramework.apk	[system/priv-app/…]
Phonesky.apk	[system/priv-app/…]

Android 5.0 (Lollipop) vom 5. November 2015

PrebuiltGmsCore.apk	[optional/gms/438/priv-app/…]
GoogleLoginService.apk	[system/priv-app/…]
GoogleServicesFramework.apk	[system/priv-app/…]
Phonesky.apk	[system/priv-app/…]

Android 4.4 (Kitkat) vom 3. März 2015

PrebuiltGmsCore.apk	[optional\gms\038\priv-app /…]
GoogleLoginService.apk	[system/priv-app/…]
GoogleServicesFramework.apk	[system/priv-app/…]
Phonesky.apk	[system/priv-app/…]

Android 4.3 (Jellybean) vom 2. März 2015

GoogleLoginService.apk	[system/app/…]
GoogleServicesFramework.apk	[system/app/…]
Phonesky.apk	[system/app/…]

Die Anwendungen **PrebuiltGmsCore.apk** die für das Einrichten des Play Stores für Android 6.0, 5.0 und 4.4 notwendig sind, wurden für den Abschnitt 3.2.4.1 und 3.2.4.2, in **GmsCore.apk** umbenannt.

3.2.4.1 Einrichten der Anwendungen direkt über die SD-Karte

Für diesen Abschnitt wird die virtuelle Maschine (Kali Linux) benötigt und der Pfad zu den Android Anwendungen lautet **/mnt/usb/*.apk**. Die Einrichtung wird an der Android Version 5.0 (Lollipop) vollführt. Um die eingesetzte SD-Karte in der virtuellen Maschine fehlerfrei nutzen zu können, kann es unter Umständen sein die Partitionierungstabelle erneut zu Laden, folgender Befehl ist in der Linux-Shell (Terminal) zu nutzen:

```
partprobe /dev/sdb
```

Der Zugriff auf die Anwendungen ist nötig, da die korrekte Funktion des Google Play Stores durch diese Anwendungen implementiert wird. Dabei ist der Zugriffspunkt frei wählbar, jedoch in diesem Abschnitt über **/mnt/os/** realisiert. Dies kann mit folgendem Befehl in der Linux-Shell (Terminal) durchgeführt werden:

```
mount /dev/sdb5 /mnt/os
```

Nach dem fehlerfreien Einhängen des Dateisystems in unser Rechnersystem, müssen die Anwendungen in Form von APK Dateien, in das Verzeichnis **/priv-app** des Android Betriebssystems dupliziert werden. Befinden sich nur die benannten Anwendungen im Verzeichnis **/mnt/usb**, so kann der folgende Befehle in der Linux-Shell (Terminal) benutzt werden, andernfalls müssen die Anwendungen einzeln kopiert werden:

```
cp -v /mnt/usb/*.apk /mnt/os/priv-app/
```

Der Befehl realisiert das Kopieren sämtlicher Anwendungsdateien (.apk) vom Quellverzeichnis ins Zielverzeichnis.

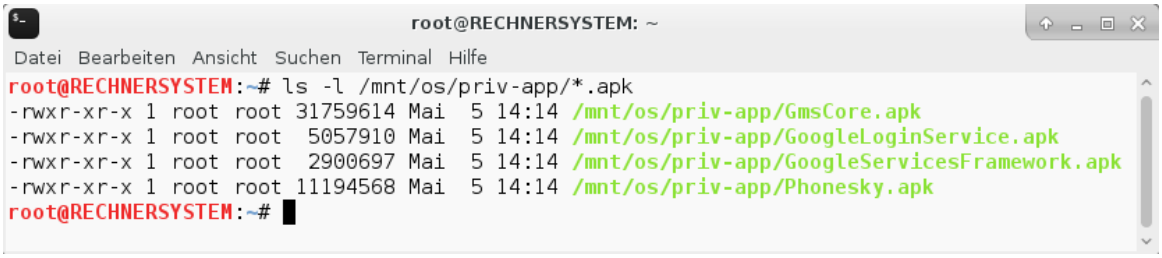
Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.

3.2.4-1 cp: Kopieren der Anwendungen auf das Android Betriebssystem

Ein Blick auf die kopierten Dateien, gibt eine Übersicht darüber, dass Besitzer und Rechte schon richtig konfiguriert wurden und ein weiteres Handeln unnötig ist. Dies ist mit dem Befehl in der Linux-Shell (Terminal) möglich:

```
ls -l /mnt/os/priv-app/*.apk
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.

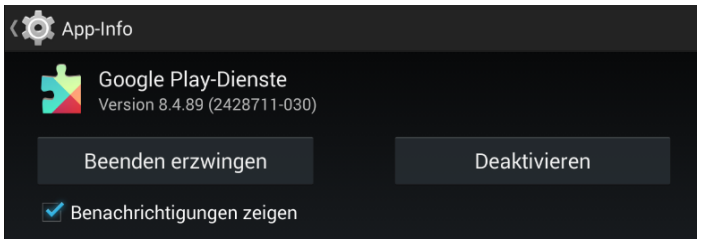


3.2.4-2 ls: Überprüfung des Besitzes und der Rechte der kopierten Anwendungen

Die SD-Karte muss nun lediglich noch in das Wandboard eingesetzt werden, dafür muss das eingehängte Dateisystem vom Rechnersystem entkoppelt werden, folgender Befehl in der Linux-Shell (Terminal) ist dafür nötig:

```
umount /mnt/os
```

Nun kann die SD-Karte in das Wandboard eingesetzt werden. Die Anwendungen werden vom Android Betriebssystem während des Bootvorgangs, nach dem Start des System Services, automatisch initialisiert und gestartet. Dabei wird das Verzeichnis **/system/priv-app** rekursiv durchsucht und noch nicht installierte Anwendungen werden initialisiert. Das Android Betriebssystem startet direkt neu und der Google Play Store kann verwendet werden. Bei dieser Methode ist klar, dass der so eingerichtete Play Store nicht wieder über die Android Benutzeroberfläche deinstalliert werden kann, da bei System Anwendungen nur die Option „Deaktivieren“ vorhanden ist.



3.2.4-3 Android: Anwendungsmanager

Der Play Store kann nur über ein Google Konto genutzt werden, hierfür bietet sich der soeben eingerichtete Google Play Dienst zum Erstellen und Verknüpfen mit einem Google Konto an.

3.2.4.2 Einrichten der Anwendungen im laufenden Betrieb über ADB

Für diesen Abschnitt wird das Rechnersystem (Microsoft Windows) benötigt, der Pfad zur Android Debug Bridge lautet **C:\SDK\platform-tools** und der Pfad zu den Android Anwendungen lautet **F:**. Außerdem muss das Wandboard ordnungsgemäß angeschlossen, eingeschaltet und mit einer Mikro SD-Karte bestückt sein, auf der sich ein lauffähiges Android Betriebssystem befindet, dieses Vorgehen wird in Abschnitt 3.2.1 Vorbereiten der SD-Karten-Speicherabbilder erklärt. Die Einrichtung wird an der Android Version 4.4 (Kitkat) vollführt. Weiter ist die Verbindung zum Android Betriebssystem eingerichtet und kann, wie in Abschnitt 2.2.3 Verbindungsaufbau beschrieben worden ist, verwendet werden.

Als Referenz zur angeschlossenen Hardwareplattform wird die Seriennummer des zu konfigurierenden Systems benötigt, da bei Verbindungen mit mehreren Hardwareplattformen eine Distinktion erfolgen muss. Zur Sichtung sämtlicher angeschlossener und über ADB ansteuerbarer Geräte, muss folgender Befehl über das Programm cmd.exe ausgeführt werden:

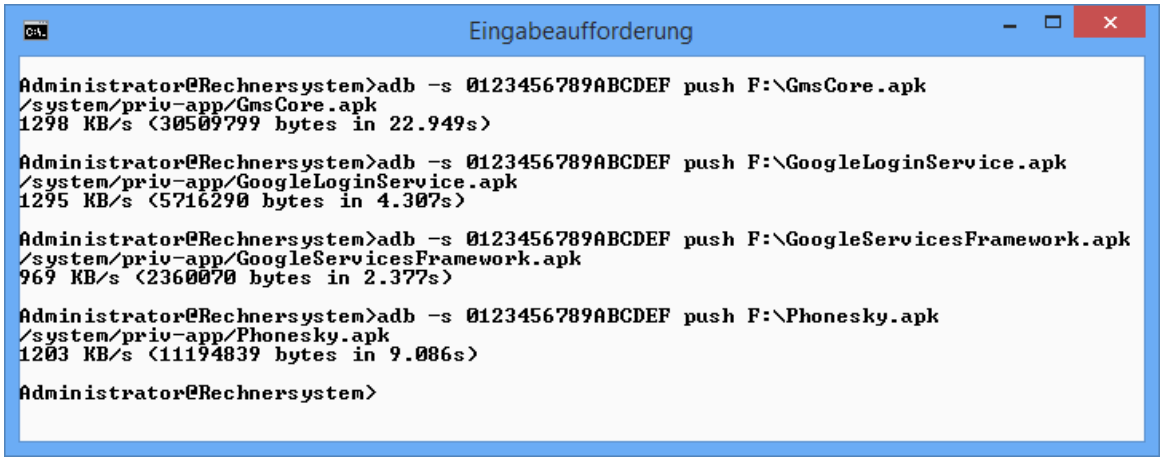
```
adb devices
```

Die Abbildung 3.2.2-14 adb: Anzeigen der angeschlossenen Geräte, zeigt eine mögliche Ausgabe für diesen Befehl an.

Im ersten Schritt muss das Dateisystem von nur lesend (read-only) auf schreibend (writable) neu eingebunden werden, dies ist in 3.2.3.2 Bestehendes Dateisystem mit bestimmten Zugriffsrecht ins System einbinden, detailliert beschrieben. Jetzt beginnt der primäre Schritt, die Einrichtung der Anwendungen auf dem Android Betriebssystem. Dazu sind die Lokalitäten der Anwendungen von Bedeutung. Damit die Anwendungen mit Systemrechten laufen, müssen diese in das System-Verzeichnis des Android Betriebssystems kopiert werden, folgende Befehle sind hierbei unerlässlich:

```
adb -s 0123456789ABCDEF push F:\GmsCore.apk
        /system/priv-app/GmsCore.app

adb -s 0123456789ABCDEF push F:\GoogleLoginService.apk
        /system/priv-app/GoogleLoginService.app

adb -s 0123456789ABCDEF push F:\GoogleServicesFramework.apk
        /system/priv-app/GoogleServicesFramework.app

adb -s 0123456789ABCDEF push F:\Phonesky.apk
        /system/priv-app/Phonesky.app
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.

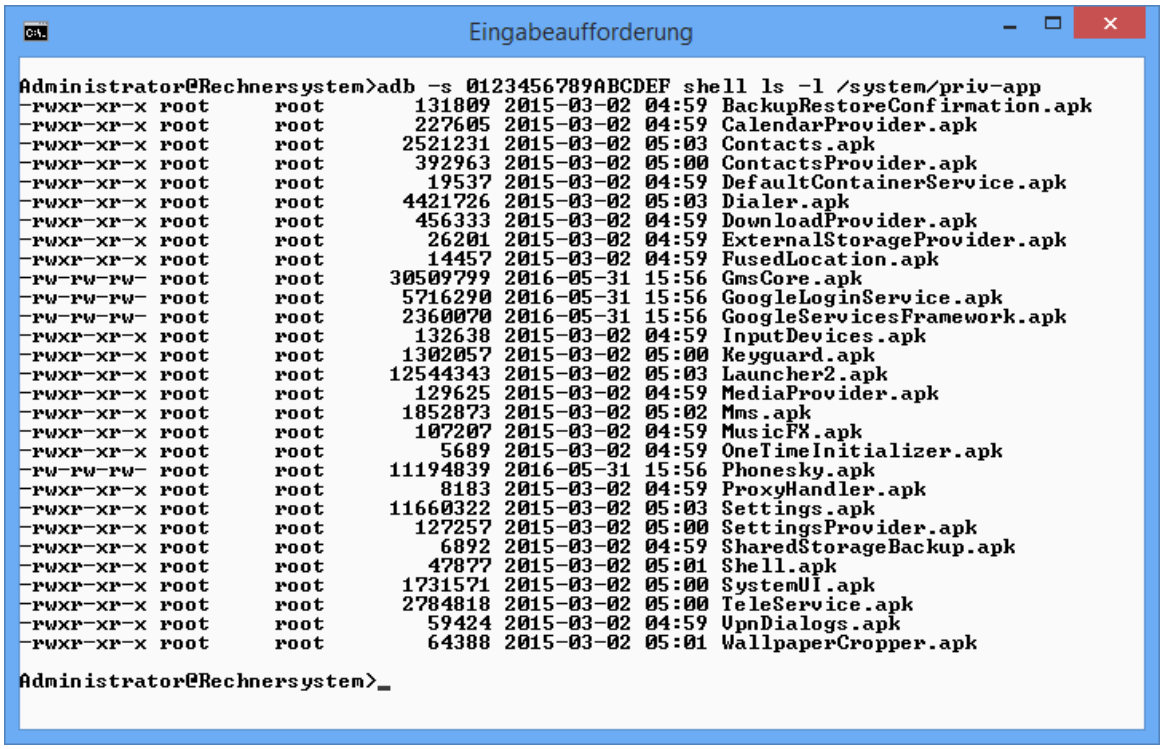


3.2.4-4 adb: Kopieren der Anwendungen in das System-Verzeichnis

Wenn alle Anwendungen erfolgreich kopiert worden sind, können diese und die dazugehörigen Berechtigungen, mit folgendem Befehl über das Programm cmd.exe angezeigt werden:

```
adb -s 0123456789ABCDEF shell ls -l /system/priv-app
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.4-5 adb: Auflisten der kopierten Anwendungen

Auffällig sind die idealen Besitzrechte der Anwendungen, aber da das Ausführungsrecht ungesetzt ist, können diese nicht gestartet werden.

Dies muss mit folgenden Befehlen über das Programm cmd.exe geändert werden:

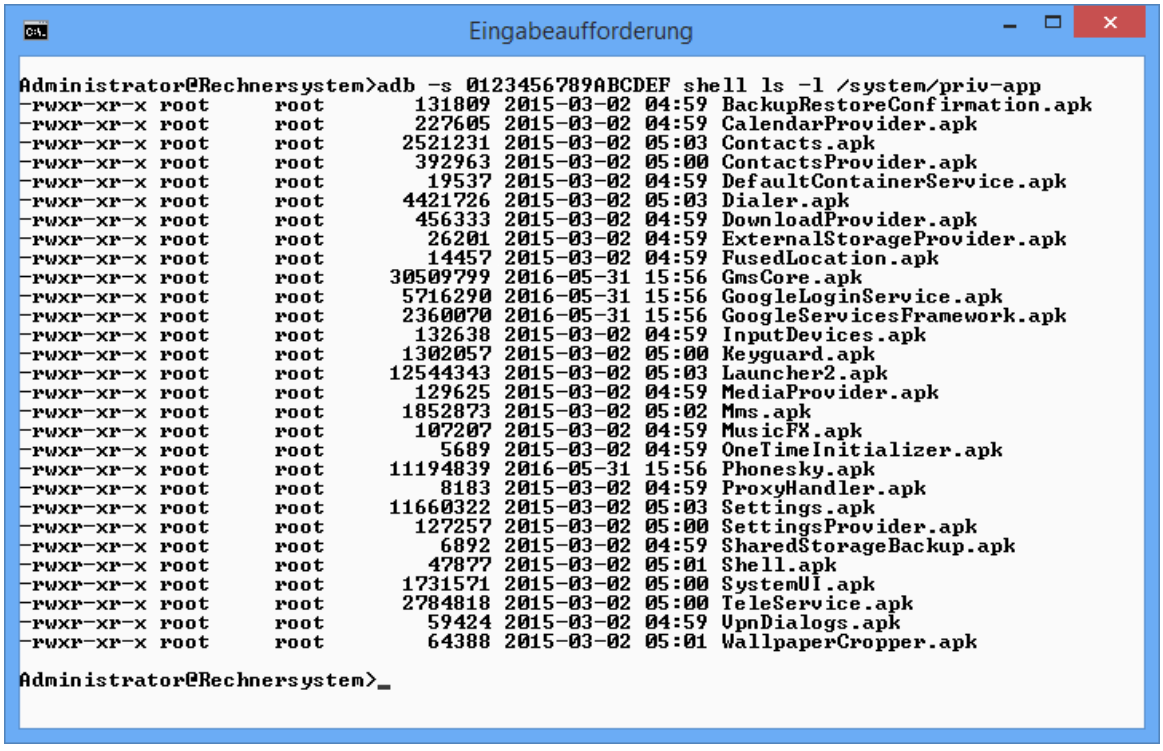
```
adb -s 0123456789ABCDEF shell chmod 0755
        /system/priv-app/GmsCore.app

adb -s 0123456789ABCDEF shell chmod 0755
        /system/priv-app/GoogleLoginService.app

adb -s 0123456789ABCDEF shell chmod 0755
        /system/priv-app/GoogleServicesFramework.app

adb -s 0123456789ABCDEF shell chmod 0755
        /system/priv-app/Phonesky.app
```

Das **chmod 0755** im Befehl sorgt dafür, das alle Benutzer des Android Betriebssystems lesen sowie ausführen dürfen und der Eigentümer „root“, obendrein schreiben darf.

Die Kontrolle mittels dem folgendem Befehl zeigt die Änderungen an:

```
adb -s 0123456789ABCDEF ls -l /system/priv-app
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.4-6 adb: Auflisten der kopierten Anwendungen nach Modifizierung der Rechte

Ein Neustart des Android Betriebssystem muss stattfinden. Die Initialisierung der Anwendungen geschieht während des Bootvorgangs des Android Betriebssystem automatisch.

Dies ist mit dem folgenden Befehl über das Programm cmd.exe möglich:

```
adb -s 0123456789ABCDEF reboot
```

Das Android Betriebssystem startet augenblicklich und der Google Play Store kann verwendet werden. Bei dieser Methode ist klar, dass ein Deinstallieren des Play Stores nicht über den Anwendungsmanager der Benutzeroberfläche passieren kann, da bei System Anwendungen nur die Option zur Deaktivierung gegeben ist (vgl. Abbildung 3.2.4-3 Android: Anwendungsmanager).

Der Play Store kann nur über ein Google Konto genutzt werden, hierfür bietet sich der soeben eingerichtete Google Play Dienst zum Erstellen und Verknüpfen mit einem Google Konto an.

3.2.5 Installation von Anwendungen ohne Nutzung des Play Stores

Die Vorgehensweise für die Installation weiterer Anwendungen ist grundsätzlich konform, wie in den Abschnitten 3.2.4.1 und 3.2.4.2 bereits beschrieben, nur das die Lokalität entsprechend den Benutzerkriterien angepasst werden sollte. Mehr zu den Lokalitäten für Benutzeranwendungen ist dem Abschnitt 3.2.3.1 Android Dateisystem zu entnehmen. Weiter ist es auch möglich Anwendungen die nicht als Systemanwendungen laufen sollen, über die Android Debug Bridge oder die Benutzeroberfläche selbst zu installieren. Die Software Avast.apk, die in diesem Abschnitt eingerichtet wird, ist im DVD-Pfad **/Methoden/Install_app**, der beiliegenden Projekt-DVD, zu finden.

3.2.5.1 Installation der Anwendungen über die Android Debug Bridge

Für diesen Abschnitt wird das Rechnersystem (Microsoft Windows) benötigt, der Pfad zu der zusätzlichen Software nennt sich **F:\Software**. Weiter muss das Wandboard ordnungsgemäß angeschlossen, eingeschaltet und mit einer eingesetzten Mikro SD-Karte mit dem Android Betriebssystem 4.4 (KitKat) betriebsbereit sein. Die Verbindung mit der Android Debug Bridge ist wie im Abschnitt 3.2.2 Verbinden des Rechnersystems mit Android über ADB konfiguriert und zur Verwendung bereit. Um nun eine Android Anwendung auf dem Android Betriebssystem zu installieren, ist folgender Befehl über das Programm cmd.exe auszuführen:

```
adb -s 0123456789ABCDEF install F\Software\Avast.apk
```

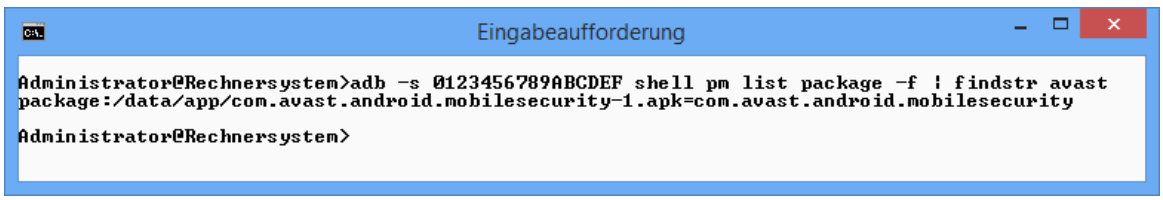
Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.

3.2.5-1 adb: Installation der Referenzanwendung

Die Kontrolle ob die Anwendung installiert wurde sowie deren Installationsverzeichnis, kann wie folgt abgerufen werden:

```
adb -s 0123456789ABCDEF shell pm list packages -f | findstr avast
```

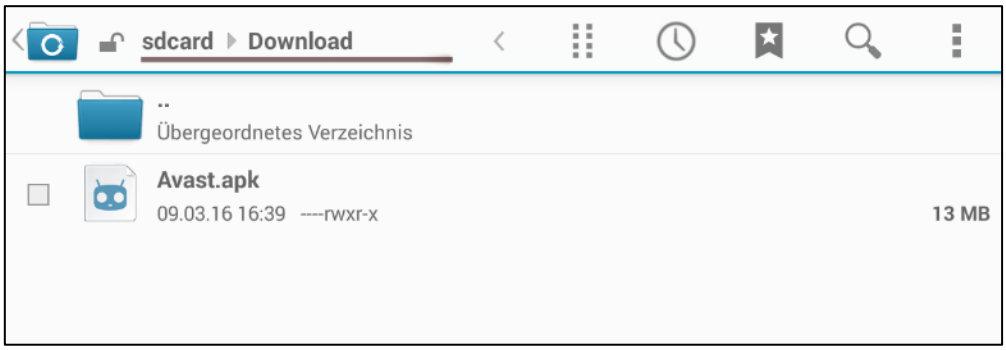
Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.5-2 adb: Prüfen der korrekten Installation der Anwendung

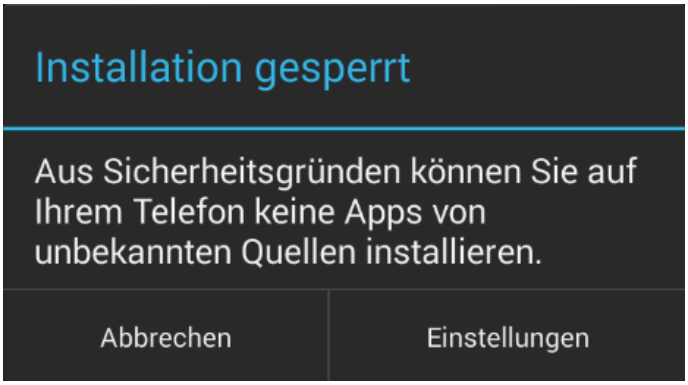
3.2.5.2 Installation der Anwendungen über die Android Benutzeroberfläche

Weiter ist es auch möglich, Anwendungen die nicht als Systemanwendungen laufen sollen, direkt über die Benutzeroberfläche des Android Betriebssystems zu installieren. Dafür sind ein Dateimanager nach Wahl sowie die zu installierende Anwendung notwendig. In diesem Abschnitt ist der Dateimanager der *CMFileManager* und die Anwendung *Avast.apk* wird über den internen Speicher mit dem Dateipfad **/sdcard/Download** bereitgestellt.



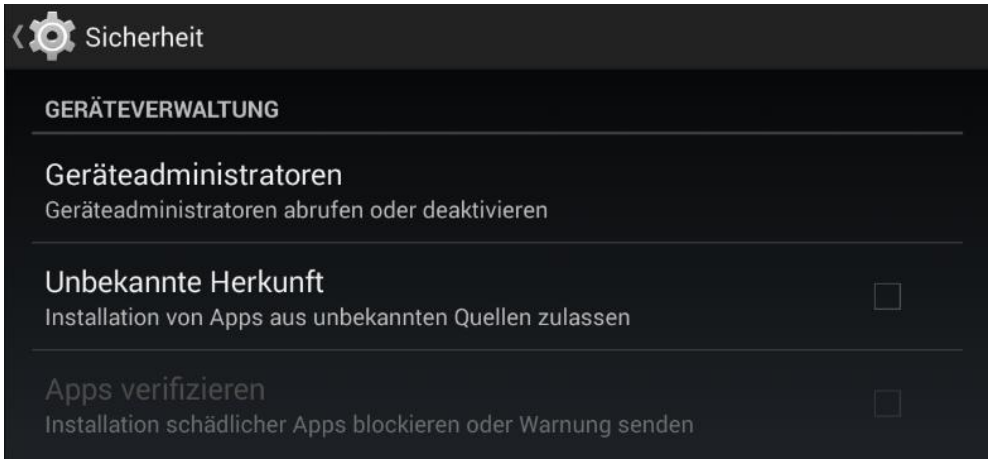
3.2.5-3 CMFileManager: Dateiverzeichnis Download mit der Anwendung Avast.apk

Wird die Anwendung angeklickt, ist es nicht möglich diese ohne Umwege zu installieren. Da die Anwendung nicht über den Play Store installiert wird, fällt sie somit in den Sicherheitsmechanismus von Android für Unbekannte Herausgeber. Gleichwohl ist es möglich diesen Mechanismus zu deaktivieren, entweder über Einstellungen >> Sicherheit >> „Geräteverwaltung“ oder direkt über den folgenden Warnhinweis der beim Installationsversuch vom System generiert wird.



3.2.5-4 Android: Warnhinweis bei der Installation von unbekannten Quellen

Im Einstellungsmenü „Sicherheit“ gibt es einen Abschnitt für die Geräteverwaltung. Der Unterpunkt „Unbekannte Herkunft“ muss an dieser Stelle aktiviert werden, um die Anwendung installieren zu können.



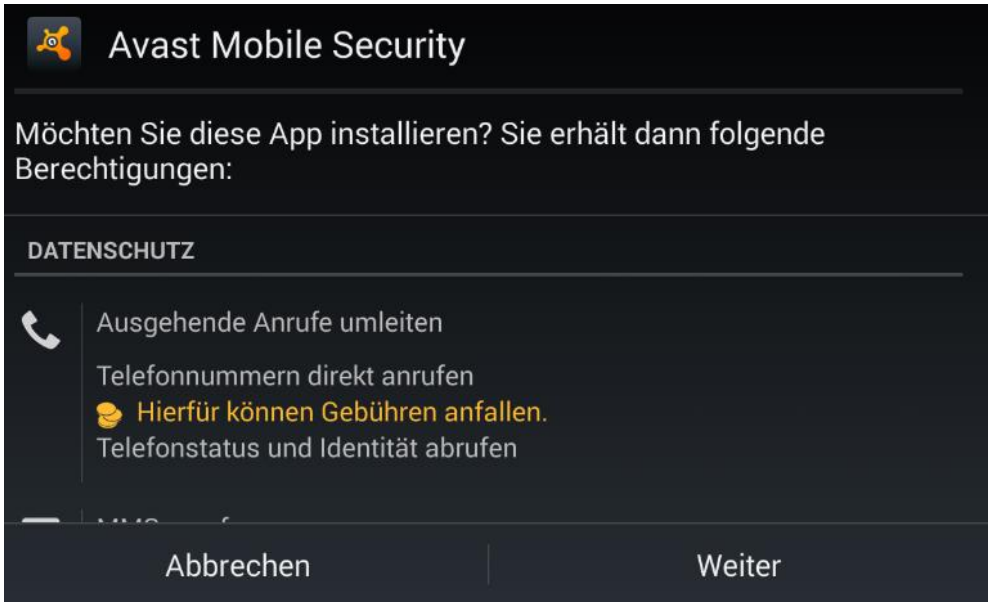
3.2.5-5 Android: Geräteverwaltung der Sicherheitseinstellungen „Unbekannte Herkunft“

Beim aktivieren dieser Option, wird eine weitere Sicherheitsabfrage vom System generiert, die mit „OK“ bestätigt werden muss. „Unbekannte Herkunft“ ist nun aktiviert.

3.2.5-6 Android: Sicherheitsabfrage zur Aktivierung "Unbekannte Herkunft"

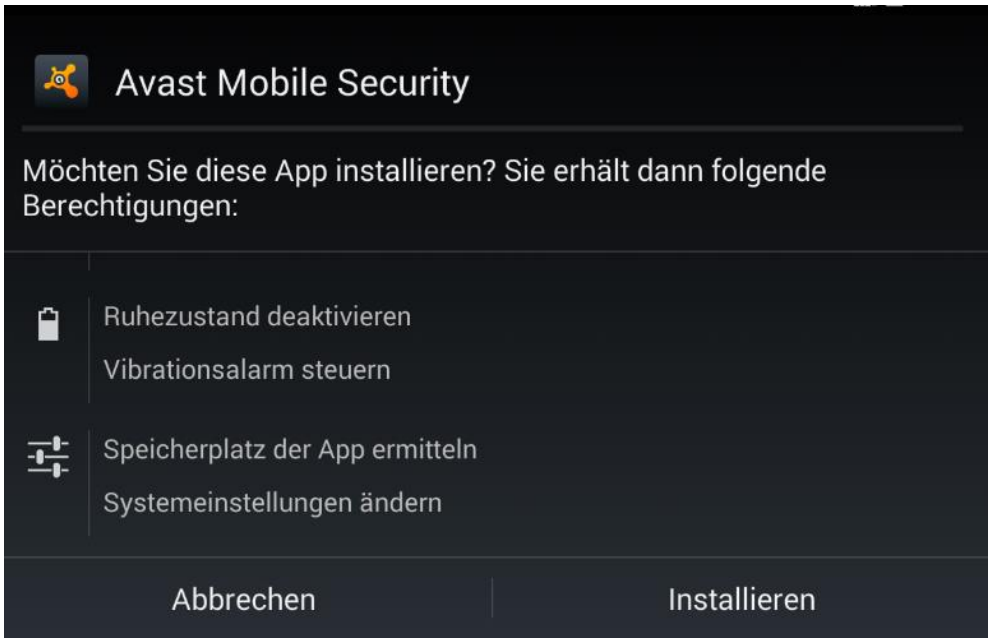
Ab diesem Schritt ist der Sicherheitsmechanismus, zum Schutz der Hard- und Software vor nicht vertrauenswürdigen Anwendungen deaktiviert und Anwendungen unbekannter Herkunft, können ohne weiteres installiert werden.

Nun muss nur noch die Anwendung Avast.apk installiert werden. Dies geschieht durch einen Klick auf die gegebene Option „Weiter“ des Dialogs zur Einsicht der geforderten Berechtigungen für die Anwendung.



3.2.5-7 Android: Einsicht in die Berechtigungen der Anwendung

Wurden alle geforderten Berechtigungen eingesehen, so kann am Ende durch einen Klick auf „Installieren“, die Anwendung installiert werden.



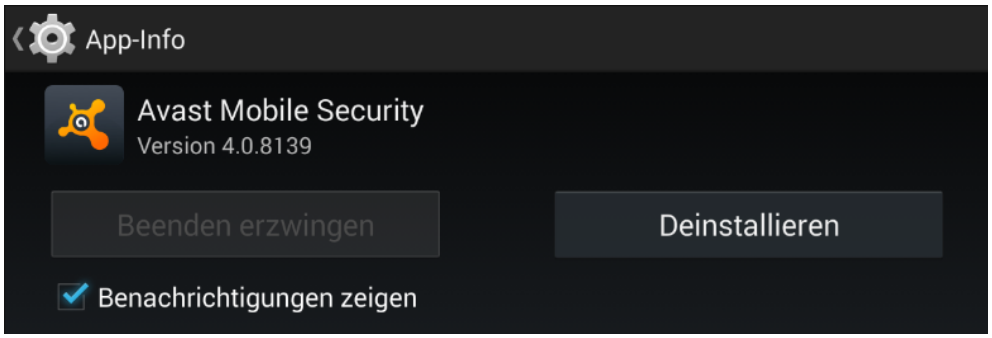
3.2.5-8 Android: Installation der Anwendung

Je nach Größe und Komplexität der Anwendung, benötigt das System eine unbestimmte Zeitspanne zur Installation der Anwendung.

3.2.5-9 Android: Installationszeit der Anwendung

Ist die Installation beendet, wird dies vom System durch „App wurde installiert“ angezeigt und die Anwendung kann über die Option „Öffnen“, sofort vom Benutzer gestartet werden.

Wird die Anwendung über den Anwendungsmanager eingesehen, ist diese durch „Deinstallieren“ wieder vom System entfernbar und somit nicht auf der **/system** Partition des Android Betriebssystems installiert. Das Zugriffsrecht für den Benutzer der Android Benutzeroberfläche, beschränkt sich auf das Lesen.



3.2.5-10 Android: Sichtung der Anwendung über den Anwendungsmanager

3.2.6 Deinstallation von Anwendungen ohne Nutzung des Play Stores

Die im Verlauf des Nutzungszyklus installierten Anwendungen auf dem Android Betriebssystem, können durch verschiedene Möglichkeiten auch wieder vom System entfernt werden. Dieser Abschnitt erläutert kurz die Durchführung der Deinstallation einer installierten Android Anwendung. In diesem Abschnitt wird die Anwendung Avast.apk, die in Abschnitt 3.2.5 Installation von Anwendungen ohne Nutzung des Play Stores installiert wurde, als Referenz Anwendung zur Deinstallation verwendet. Die Anwendung ist der beiliegenden Projekt-DVD im DVD-Pfad **/Methoden/Install_app**, zu entnehmen.

3.2.6.1 Deinstallation direkt über die SD-Karte

Für diesen Abschnitt wird die virtuelle Maschine (Kali Linux) benötigt und Informationen über die Lokalität der installierten Anwendung. Weiter muss Zugriff auf die Mikro SD-Karte mit dem Android Betriebssystem 4.4 (Kitkat) möglich sein, um die eingesetzte SD-Karte in der virtuellen Maschine korrekt nutzen zu können. Es kann unter Umständen sein, dass die Partitionierungstabelle erneut zu Laden ist und folgender Befehl in der Linux-Shell (Terminal) zur Handhabe kommt:

```
partprobe /dev/sdb
```

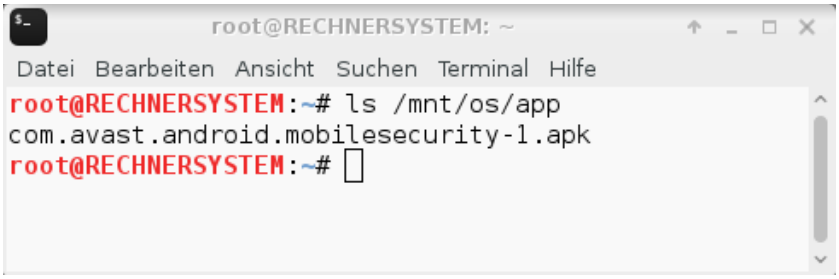
Um nun die installierte Anwendung vom Android Betriebssystem zu deinstallieren, ist der Zugriff auf dieses Notwendig. Dabei ist der Zugriffspunkt frei Wählbar, jedoch in diesem Abschnitt über **/mnt/os/** realisiert. Dies kann mit folgendem Befehl in der Linux-Shell (Terminal) durchgeführt werden:

```
mount /dev/sdb4 /mnt/os
```

Nach dem korrekten Einhängen des Dateisystems in unser Rechnersystem muss die Anwendungen Avast.apk im Verzeichnis **/app** des Android Betriebssystems ausfindig gemacht werden. Zur Sichtung der Anwendung im Verzeichnis kann folgender Befehl in der Linux-Shell (Terminal) ausgeführt werden:

```
ls /mnt/os/app
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.6-1 ls: Prüfen auf Vorhandensein der Anwendung

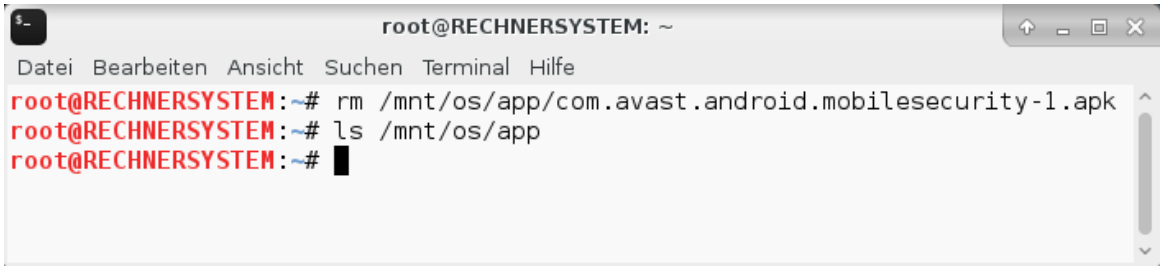
Folgender Befehl löscht dann die Anwendung vom Android Dateisystem:

```
rm /mnt/os/app/com.avast.android.mobilesecurity-1.apk
```

Auch hier kann wieder zur Sichtung des korrekten Löschens der Anwendung im Verzeichnis, folgender Befehl in der Linux-Shell (Terminal) ausgeführt werden:

```
ls /mnt/os/app
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.6-2 rm: Löschen der Anwendungen auf dem Android Betriebssystem

Die SD-Karte muss nun nur noch in das Wandboard eingesetzt werden, dafür muss das eingehängte Dateisystem vom Rechnersystem ausgehangen werden, folgender Befehl in der Linux-Shell (Terminal) ist dafür nötig:

```
umount /mnt/os
```

Nun kann die SD-Karte in das Wandboard eingesetzt werden und die Anwendung ist nach dem Bootvorgang des Android Betriebssystem nicht mehr verfügbar.

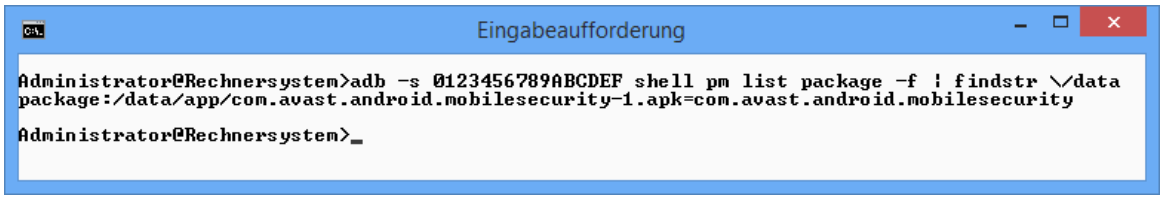
3.2.6.2 Deinstallation im laufenden Betrieb über die Android Debug Bridge

Eine weitere sehr elegante Möglichkeit ist die Deinstallation über die Android Debug Bridge. Hierfür wird das Rechnersystem (Microsoft Windows) benötigt. Weiter muss das Wandboard ordnungsgemäß angeschlossen, eingeschaltet und mit einer eingesetzten Mikro SD-Karte mit dem Android Betriebssystem 4.4 (KitKat) betriebsbereit sein. Die Verbindung mit der Android Debug Bridge ist wie im Abschnitt 3.2.2 Verbinden des Rechnersystems mit Android über ADB konfiguriert und zur Verwendung bereit. Um nun eine Android Anwendung vom Android Betriebssystem zu deinstallieren, ist zunächst die Informationsbeschaffung der installierten Anwendungen von Vorteil, folgender Befehl über das Programm cmd.exe muss dazu ausgeführt werden:

Die Liste der installierten Anwendungen kann jedoch sehr lang sein, so ist es von Vorteil Informationen über die installierte Anwendung, vorrätig zu haben. Ist die Anwendung eine System Anwendung oder vom Benutzer installiert, so können die Verzeichnisse **/system** und **/data** unterschieden werden. Im Beispiel für Avast, wurde die Anwendung durch den Benutzer installiert, somit kann folgender Befehl verwendet werden:

```
adb -s 0123456789ABCDEF shell pm list packages -f | findstr \/data
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.6-3 adb: Informationsbeschaffung der installierten Nutzeranwendungen

Die Ausgabe zeigt Zeilenweise alle Benutzeranwendungen, nach dem „:“ den Installationspfad zur Anwendung und nach dem „=“ den zugehörigen Namen der Anwendung wie er im Android Betriebssystem verwendet wird. Aus dieser Information kann unter Verwendung des folgenden Befehls, die Anwendung im laufenden Betrieb deinstalliert werden:

```
adb -s 0123456789ABCDEF uninstall com.avast.android.mobilesecurity
```

Ist die Anwendung vom Android Betriebssystem erfolgreich deinstalliert worden, meldet ADB „Success“.

3.2.6.3 Deinstallation über die Android Benutzeroberfläche

Wie im Abschnitt 3.2.5.2 Installation der Anwendungen über die Android Benutzeroberfläche angesprochen, kann die Anwendung in ähnlicher Weise vom Android Betriebssystem, über die Benutzeroberfläche deinstalliert werden. Hierfür muss der Knopf „Deinstallieren“ (*Uninstall*), wie in der Abbildung 3.2.5-10 Android: Sichtung der Anwendung über den Anwendungsmanager zu sehen ist, betätigt werden und eine Sicherheitsabfrage „Möchten Sie diese App deinstallieren?“ (*Do you want to uninstall this app?*) mit „OK“ bestätigt werden. Daraufhin deinstalliert das Android Betriebssystem die Anwendung und meldet die Fertigstellung mit folgender Einblendung: „Deinstallation abgeschlossen.“ (*Uninstall finished.*).

3.2.7 Einrichten des Root-Zugriffs für Android Anwendungen

Der folgende Teil erklärt die Einrichtung des Root-Zugriffs, für die erweiterte Rechteanforderung von Android Anwendungen über die Android Benutzeroberfläche. Dabei bezeichnet „Root“ einen Benutzer, der vollen Zugriff auf das System hat und überall Schreibend agieren darf. Der Vorteil des Root-Zugriffs über die Steuerung der Benutzeroberfläche ist dabei, Android Anwendungen mit Systemrechten (Root-Rechten) zu starten, um Aktionen wie beispielsweise „Debugging andere Anwendungen“ durchführen zu können. Dabei wird der Root-Zugriff durch eine Root-Anwendung geregelt und durch den Benutzer manuell freigegeben.

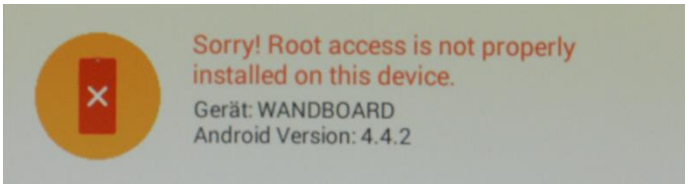
3.2.7.1 Installation für das Speicherabbild Android 5.0 (Lollipop)

Für das Speicherabbild Android 5.0 vom 5. November 2015 ist noch keine Möglichkeit bekannt, einen solchen Root-Zugriff über die Android Benutzeroberfläche zu realisieren.

3.2.7.2 Installation für das Speicherabbild Android 4.4 (Kitkat)

Für den Root-Zugriff des Speicherabbilds Android 4.4 (Kitkat) vom 3. März 2015 ist noch zusätzliche Software erforderlich, diese kann von der Quelle GitHub [S-GiHuA16] heruntergeladen und entpackt werden. Als Alternative bietet sich auch der Weg an, die zusätzliche Softwar der beiliegenden Projekt-DVD im DVD-Pfad **/Methoden/Install_root_on_Android4.4**, dieser Arbeit zu entnehmen. Das Rechnersystem (Microsoft Windows) wird benötigt, einschließlich des Pfads zu der zusätzlichen Software **F:\Software**. Weiter muss das Wandboard richtig angeschlossen, eingeschaltet und mit einer eingesetzten Mikro SD-Karte, auf dem sich das Android Betriebssystem 4.4 (Kitkat) befindet, betriebsbereit sein. Die Verbindung mit der Android Debug Bridge steht wie im Abschnitt 3.2.2 Verbinden des Rechnersystems mit Android über ADB beschrieben, konfiguriert und zur Verwendung bereit.

Zur Überprüfung ob Root-Rechte angefordert werden, kann optional die Anwendung RootChecker5.6.1.apk die ebenfalls im Pfad der zusätzlichen Software vorhanden ist, wie in Abschnitt 3.2.5 Installation von Anwendungen ohne Nutzung des Play Stores, auf dem Android Betriebssystem installiert werden. Die Datei ist aus einem bestehenden Android Betriebssystem extrahiert, hat die minimale Anforderung von Android 3.0 und ist für das Zielsystem Android 5.0 erstellt. Wird diese Anwendung gestartet, kann das Android Betriebssystem auf Root-Rechte geprüft werden. Durch den Klick auf den Knopf „BESTÄTIGE ROOT“ (*VERIFY ROOT*) meldet die Anwendung das Root-Rechte nicht angefordert werden konnten.



3.2.7-1 Android: RootChecker "Bestätige Root" vor Modifizierung

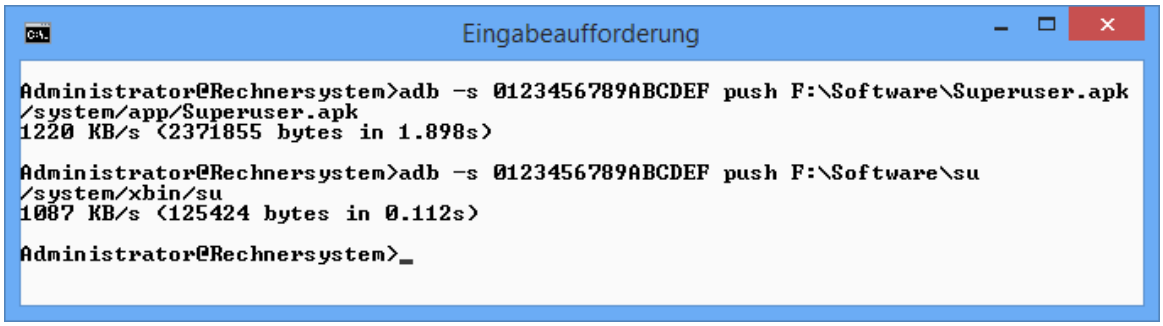
Um nun sämtliche benötigte Software, ordnungsgemäß auf das Android Betriebssystem zu kopieren, ist das Dateisystem von Android mit Schreibrechten einzubinden. Der Abschnitt 3.2.3.2 Bestehendes Dateisystem mit bestimmten Zugriffsrecht ins System einbinden, beschreibt diesen Vorgang detailgenau.

Im Anschluss muss sämtliche Software auf das Android Dateisystem kopiert werden, folgende Befehle sind über das Programm cmd.exe abzusetzen:

```
adb -s 0123456789ABCDEF push F:\Software\Superuser.apk /system/app/Superuser.apk

adb -s 0123456789ABCDEF push F:\Software\su /system/xbin/su
```

Die Lokalitäten der Dateien ergeben sich aus Kenntnissen zu Linux-Derivaten. Dabei muss die Anwendungsdatei Superuser.apk, in das für Anwendungen gültige Verzeichnis und die ausführbare Datei „su“, in das Verzeichnis für Systemprogramme für essentiell wichtige Aufgaben kopiert werden.

Folgende Abbildung zeigt eine mögliche Ausgabe für diese Befehle.



3.2.7-2 adb: Kopieren der zusätzlichen Software

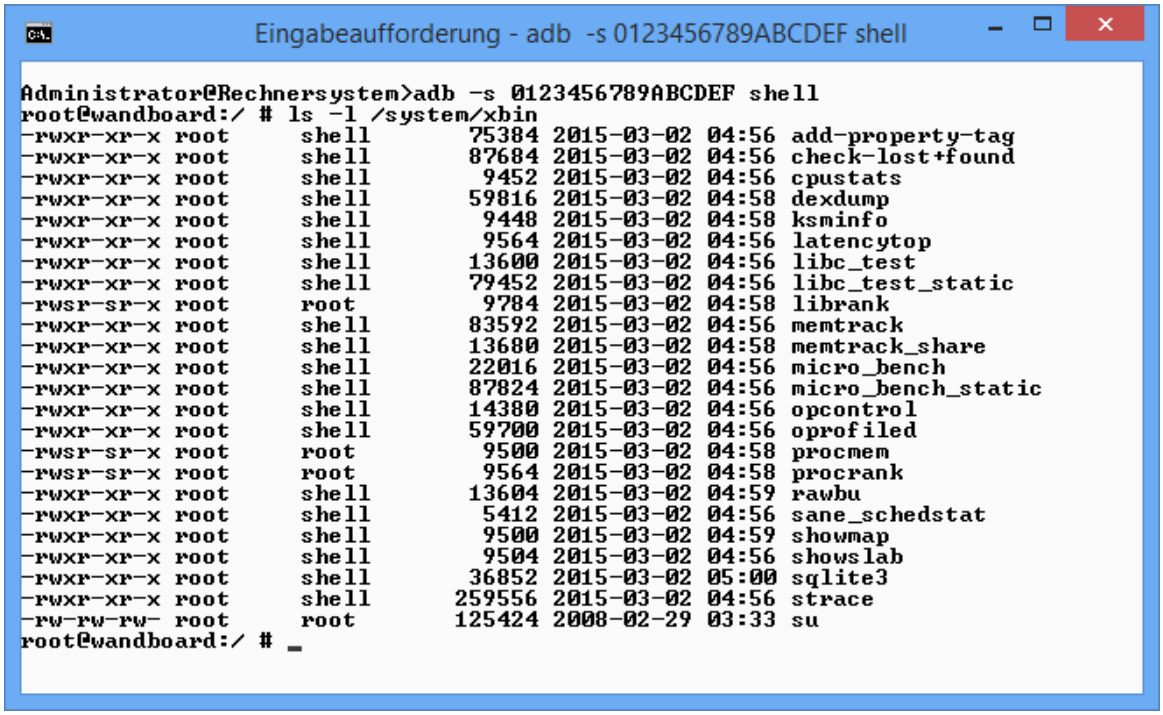
Nach dem Kopiervorgang kann nun, über die Shell am Android Betriebssystem weiter gearbeitet werden. Dies dezimiert die Längen der verwendeten Befehle durch Reduzierung von „adb -s 0123456789ABCDEF shell“. Folgender Befehl ist über das Programm cmd.exe auszuführen:

```
adb -s 0123456789ABCDEF shell
```

Im System-Verzeichnis **/system/xbin** kann das nicht gesetzte Ausführungsrecht für die kopierte Datei, mit folgendem Befehl eingesehen werden:

```
ls -l /system/xbin/
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.7-3 adb: Android Betriebssystem-Verzeichnis xbin

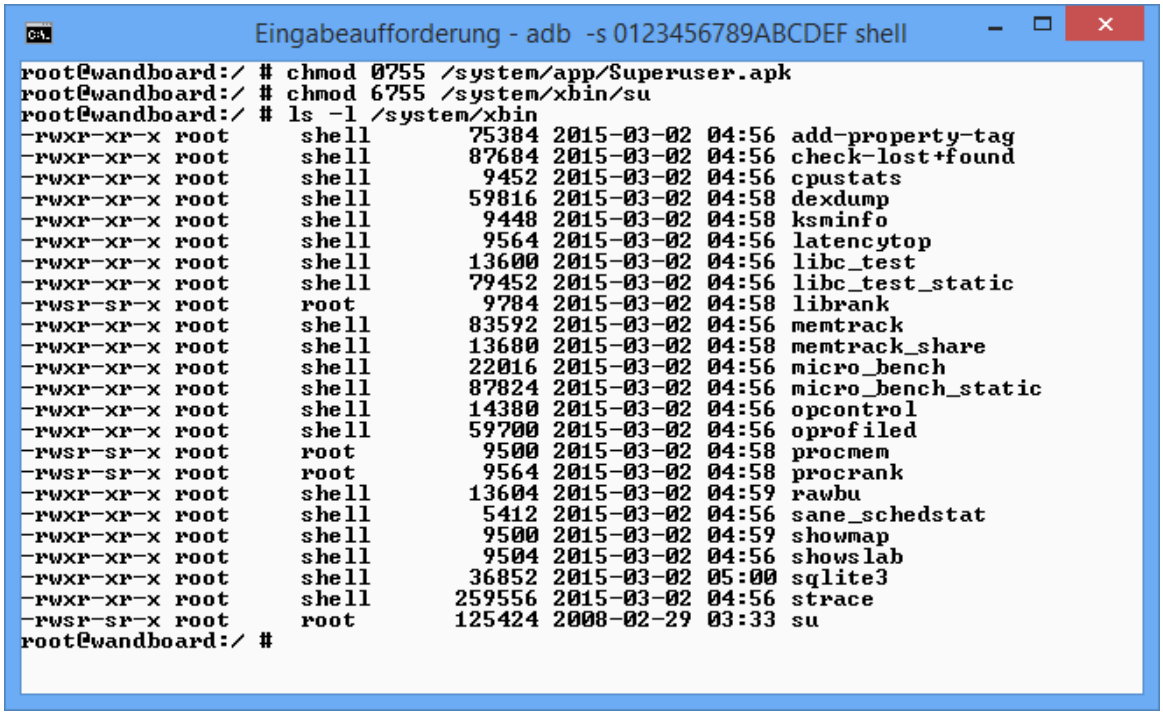
Um nun diesen durch das Kopieren hervorgerufenen Effekt zu korrigieren, welcher ein Sicherheitsmechanismus des Betriebssystems ist, müssen die Rechte der Dateien mit folgendem Befehl angepasst werden:

```
chmod 0755 /system/app/Superuser.apk

chmod 6755 /system/xbin/su
```

Eine erneute Sichtung des System-Verzeichnis **/system/xbin** mit folgendem Befehl, zeigt die Änderungen:

```
ls -l /system/xbin/
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diese Befehle.



3.2.7-4 adb: Android Betriebssystem-Verzeichnis xbin nach Modifizierung der Rechte

Dem folgend, müssen zwei Verzeichnisse angelegt werden. Nachkommende Befehle sind dazu nötig:

mkdir /system/bin/.ext

mkdir /system/etc/init.d

Da der Befehl mkdir, Verzeichnisse mit vollen Berechtigungen für Jedermann erstellt, ist auch hier eine Rechte Modifizierung von Nöten. Nöten. Hierbei ist die Ausführung folgender Befehle vorausgesetzt:

chmod 0755 /system/bin/.ext

chmod 0755 /system/etc/init.d

Nun muss die zuvor kopierte Datei „su“, an verschiedenen Stellen im Dateisystem dupliziert werden. Dieser Teilschritt ist mit folgenden Befehlen auszuführen:

cp /system/xbin/su /system/xbin/daemonsu

cp /system/xbin/su /system/bin/.ext/.su

Der Kopierbefehl kopiert die original Dateien, also ohne Modifizierung des Besitzers und der Rechte. Da aber für die Kopie der Datei „daemonsu“ andere Rechte zwingend sind, müssen diese mit folgendem Befehl angepasst werden:

```
chmod 0755 /system/xbin/daemonsu
```

Nun sind noch Dateien zu erstellen, die über den Init Prozess beim Bootvorgang gestartet und ausgeführt werden. Zur Erstellung der Dateien sind folgende Befehle auszuführen:

Datei: 99SuperSUDaemon

```
echo \#\!\/system\/bin\/sh > /system/etc/init.d/99SuperSUDaemon

echo \/system\/xbin\/daemonsu --auto-daemon \&
        >> /system/etc/init.d/99SuperSUDaemon
```

Datei: .installed_su_daemon

```
echo 1 > /system/etc/.installed_su_daemon
```

Datei: install-recovery.sh

```
echo \#\!\/system\/bin\/sh > /system/etc/install-recovery.sh

echo \/system\/xbin\/daemonsu --auto-daemon \& >> /system/etc/install-recovery.sh

echo \/system\/etc\/install-recovery-2.sh >> /system/etc/install-recovery.sh
```

Dabei dient das Zeichen „\“ als Konvertierungszeichen, sodass das erste nachfolgende Zeichen nicht mehr als Systemzeichen in dem jeweiligen Befehl Beachtung findet. Auch hier ist zu bedenken, dass das Erstellen einer Datei mit dem Befehl echo zur Auswirkung hat, dass diese nicht das Ausführungsrecht besitzt. Folgende Befehle ändern die Rechte der erstellten Dateien:

```
chmod 0755 /system/etc/init.d/99SuperSUDaemon

chmod 0755 /system/etc/.installed_su_daemon

chmod 0755 /system/etc/install-recovery.sh
```

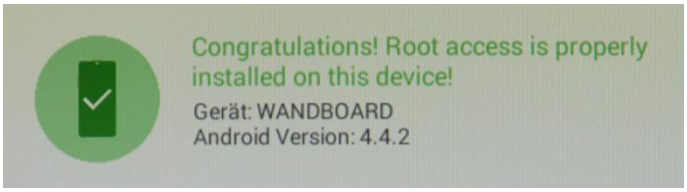
Nach dem alle Dateien mit den entsprechenden Rechten im Dateisystem des Android Betriebssystems installiert und konfiguriert sind, muss das System mit folgendem Befehl neu gestartet werden, um diese Dateien während des Bootvorgangs auszuführen:

```
reboot
```

Nach dem Neustart des Android Betriebssystems kann nochmals geprüft werden, ob Root-Rechte angefordert werden können, dazu ist nochmalig die Anwendung RootChecker5.6.1.apk nötig, die optional installiert werden kann. Nach dem Start dieser Anwendung kann über den Knopf „BESTÄTIGE ROOT“ (*VERIFY ROOT*), kann versucht werden, die Root-Rechte anzufordern. In diesem Fall übernimmt Anwendung Superuser.apk die Kontrolle und fordert vom Benutzer die Bestätigung für diese Aktion an.

3.2.7-5 Android: Root-Zugriff Anfrage von RootChecker über Superuser

Ein Klick auf den Knopf „Erlauben“ (*Grant*) gibt der Anwendung RootChecker die benötigten Berechtigungen, fortan als Root zu agieren. Bestätigt wird dies mit der folgenden Meldung der Anwendung RootChecker5.6.1.apk.



3.2.7-6 Android: RootChecker "Bestätige Root" nach Modifizierung

Die Einrichtung des Root-Zugriffs von Android Anwendungen ist somit abgeschlossen und kann über die Anwendung **Superuser.apk** gesteuert werden.

3.2.8 Erweiterung der Android Shell Funktionalität

Aus lizenzrechtlichen Gründen ist die Verwendung von Code, der unter der GPL lizenziert ist, verboten [I-AnSo16]. Android Betriebssysteme bieten daher, nur rudimentäre Shells mit einigen wenigen Unix-Befehlen an. Ergänzend verwendet Android eine eigene C-Bibliothek, die von Google für Android modifiziert wurde. Aus diesem Grund ist ein Nachrüsten der Unix-Befehle durch zusätzliche Software unumgänglich. Für die Betriebssystem Speicherabbilder aus 2.2.2.2 Speicherabbild, gilt diese Restriktion nicht, da diese Systeme schon angepasst wurden und eine Toolbox, eine BusyBox oder beides zur Verfügung stellen. Einer Nachrüstung bedarf es nur, wenn bei neueren Android Versionen diese Funktionalität fehlt und verlangt wird. Folgender Abschnitt stellt einige Programme vor und konkretisiert die Installation an einem Beispiel.

Toolbox

Die Android Toolbox ist ein Multifunktions-Programm, welches viele gängige Linux-Befehle in einem ausführbaren Programm kapselt. Jedoch sind die Funktionalitäten im Vergleich zu den Linux-Befehlen sehr eingeschränkt. Die Dokumentation dieser Software gibt dabei Aufschluss über die Funktionen.

BusyBox

BusyBox verbindet viele Versionen herkömmlicher Unix-Dienstprogramme in einer einzigen, kleinen ausführbaren Datei. Somit erlaubt sie den Ersatz für diese Dienstprogramme, die in der Regel in GNU fileutils zu finden sind. Busybox wurde nicht auf Geschwindigkeit optimiert, sondern auf geringen Speicherverbrauch. Es stellt eine vollständige Umgebung zur Verfügung und verhält sich in der Funktionalität sehr ähnlich wie das GNU Pendant [I-Busy16].

Im Abschnitt 3.2.9, wird die Installation und Verwendung von ausführbaren Dateien im Android Betriebssystem erklärt, dies geschieht am Beispiel von busybox.

Dropbear

Dropbear ist ein relativ kleiner SSH Server und Client, der auf einer POSIX basierenden Plattform läuft [I-Drop16].

3.2.9 Einrichten und Ausführen von ausführbaren Dateien

Für diesen Abschnitt wird das Rechnersystem (Microsoft Windows) benötigt, der Pfad zur Android Debug Bridge ist **C:\SDK\platform-tools** und der Pfad zu der ausführbaren Datei ist **F:\Software**. Wie bisher muss das Wandboard ordnungsgemäß angeschlossen, eingeschaltet und eine mit einem lauffähigen Android Betriebssystem konfigurierte Mikro SD-Karte eingesetzt sein (vgl. Abschnitt 3.2.1 Vorbereiten der SD-Karten-Speicherabbilder). Da die Wahl der zu installierenden ausführbaren Datei busybox ist und das Speicherabbild Android 4.4 (KitKat) nur die toolbox zur Verfügung stellt, kann die Einrichtung und Ausführung an diesem Beispiel gezeigt werden. Weiter ist die Verbindung wie in Abschnitt 2.2.3 Verbindungsaufbau, zum Android Betriebssystem eingerichtet und kann verwendet werden. Die ausführbare Datei busybox kann heruntergeladen [I-Busy16] und entpackt werden, auch hier bietet sich wieder die Anwendung 7-zip an [S-Szip16]. Außerdem kann die ausführbare Datei busybox, auch der beiliegenden Projekt-DVD im DVD-Pfad **/Methoden/Install_busybox**, dieser Arbeit entnommen werden.

Die ausführbare Datei busybox muss zunächst auf das Android Betriebssystem kopiert werden, dazu ist das Dateisystem von Android mit Schreibrechten einzubinden, der Abschnitt 3.2.3.2 Bestehendes Dateisystem mit bestimmten Zugriffsrecht ins System einbinden, beschreibt diesen Vorgang im Detail. Anschließend muss die Datei auf das Android Dateisystem vervielfältigt werden, folgender Befehl ist über das Programm cmd.exe auszuführen:

```
adb -s 0123456789ABCDEF push F:\Software\busybox-arm5l /system/xbin/busybox
```

Auch hier ergibt sich die Lokalität der Datei aus Kenntnissen zu Linux-Derivaten. Dabei muss die ausführbare Datei busybox, in das Verzeichnis für Systemprogramme für essentiell wichtige Aufgaben kopiert werden.

Worde die Datei erfolgreich kopiert, kann über die Shell am Android Betriebssystem weiter gearbeitet werden. Dies dezimiert die Längen der verwendeten Befehle durch Reduzierung von „adb -s 0123456789ABCDEF shell“. Folgender Befehl ist über das Programm cmd.exe auszuführen:

```
adb -s 0123456789ABCDEF shell
```

Aus Abschnitt 3.2.7.2 Installation für das Speicherabbild Android 4.4 (Kitkat) geht hervor, dass das Ausführungsrecht für die kopierte Datei, noch nicht gesetzt ist. Dies wird mit folgendem Befehl korrigiert:

```
chmod 0755 /system/xbin/busybox
```

Ein Blick auf die ausführbare Datei kann mit folgendem Befehl vollzogen werden:

```
busybox
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.

3.2.9-1 busybox: Anzeigen sämtlicher Funktionen für busybox

Jetzt gibt es zwei Varianten, die ausführbare Datei für Aktionen mit der Shell über das Android Betriebssystem zu benutzen.

Variante 1 – Befehle durch busybox Funktionen

Am Android Betriebssystem müssen keine Änderungen mehr vorgenommen werden. Vielmehr lässt sich jetzt jede genutzte Funktion, über die ausführbare Datei busybox ansprechen. Am Beispiel von **chattr**, was für change attribute steht und bestimmte Attribute auf Dateien oder Ordner setzt, sieht der Befehl zur Änderung des Attributes „unveränderlich“ (immutable) an einer imaginären Datei test, im Pfad /data folgendermaßen aus:

```
busybox chattr +i /data/test
```

Hierbei wird das Attribut „unveränderlich“ gesetzt und die Datei kann nicht mehr verändert werden. Diese Änderung ist mit folgendem Befehl rückgängig zu machen:

```
busybox chattr -i /data/test
```

Variante 2 – Installation von Befehlslinks zur busybox

Am Android Betriebssystem muss eine Änderung vorgenommen werden. Um im späteren Verlauf Befehle wie an einem Linux Terminal ausführen zu können, werden zu den einzelnen Funktionen der busybox, Softlinks im System-Verzeichnis für essentiell wichtige Programme erstellt. Dies ist mit folgendem Befehl möglich:

```
busybox --install /system/xbin
```

Fortan ist die Verwendung der Funktionen die aus 3.2.9-1 busybox: Anzeigen sämtlicher Funktionen für busybox ersichtlich werden, ohne die Verwendung der Textpassage „busybox“ im Befehl möglich. Am Beispiel von **chattr**, was in Variante 1 schon genutzt wurde, sehen die Befehle zum Ändern bestimmter Attribute auf Dateien oder Ordner folgendermaßen aus:

```
chattr +i /data/test
```

Hierbei wird das Attribut „unveränderlich“ gesetzt und die Datei kann nicht mehr verändert werden. Auch diese Änderung ist mit folgendem Befehl rückgängig zu machen:

```
chattr -i /data/test
```

Bei der Verwendung von ausführbaren Dateien für das Android Betriebssystem ist darauf zu achten, dass die verwendeten Dateien auch unter ARM Prozessoren lauffähig sind, da das Wandboard wie in 2.2.1 Technische Details beschrieben, mit einem ARM Prozessor ausgestattet ist.

Sollte ab diesem Zeitraum keine weitere auf das Dateisystem schreibende Aktion (writable) ausgeführt werden, muss das Dateisystem wieder mit lesendem Recht (read-only) eingebunden werden, dass wie im Abschnitt 3.2.3.2 Bestehendes Dateisystem mit bestimmten Zugriffsrecht ins System einbinden, detailliert beschrieben ist. Anstelle dessen, ist auch ein Neustart des Android Betriebssystems möglich.

3.2.10 Einrichten von Terminal und FTP-Server auf Android

Das Einrichten eines eigenständigen Terminals oder FTP-Servers am Android Betriebssystem ist nicht nötig, da wie in den Abschnitten 3.2.2 bis 3.2.9 beschrieben, zum Fernsteuern des Android Betriebssystems und zum Datenaustausch zwischen Fernsteuerungsrechner und Android Betriebssystem ausschließlich die Android Debug Bridge eingesetzt wurde. Die Einrichtung am Fernsteuerungsrechner ist auf eine Installation von minimal ADB (vgl. Abschnitt 2.4.4 Android Debug Bridge (ADB)) begrenzt und zur Realisierung dieser Aufgaben wird die Shell (Eingabeaufforderung/Terminal) des jeweiligen Betriebssystems genutzt.

3.2.11 Installation eines Schadprogramms

Aus den Abschnitten 3.2.4.1 Einrichten der Anwendungen direkt über die SD-Karte, 3.2.4.2 Einrichten der Anwendungen im laufenden Betrieb über ADB, 3.2.5.1 Installation der Anwendungen über die Android Debug Bridge und 3.2.5.2 Installation der Anwendungen über die Android Benutzeroberfläche, gehen mehrere Methoden zur Installation einer Anwendung auf dem Android Betriebssystem hervor. Die Methoden sind jedoch nicht alle vollautomatisch durchführbar, somit verbleiben lediglich die Methoden aus Abschnitt 3.2.4.2 und Abschnitt 3.2.5.1. Da bei der Analyse eines Schadprogramms das volle Potenzial dieser Anwendung analysiert werden soll, ist die Methode 3.2.4.2 besser geeignet, denn Schadprogramme können als Systemanwendung agieren und somit auch alle Funktionen ausführen.

Für diesen Abschnitt wird das Rechnersystem (Microsoft Windows) benötigt, der Pfad zur Android Debug Bridge ist **C:\SDK\platform-tools** und der Pfad zu dem Schadprogramm lautet **F:\Software**. Wie bisher muss das Wandboard ordnungsgemäß angeschlossen, eingeschaltet und eine mit einem lauffähigen Android Betriebssystem konfigurierte Mikro SD-Karte eingesetzt sein (vgl. Abschnitt 3.2.1 Vorbereiten der SD-Karten-Speicherabbilder). Da sich Android 4.4 (KitKat), momentan als ein etabliertes Betriebssystem erweist [I-AnDeZ15], wird dieses hier auch Verwendung finden. Die zu verwendende Anwendung Malware.apk, kann der beiliegenden Projekt-DVD im DVD-Pfad **/Methoden/Install_malware**, dieser Arbeit entnommen werden.

Das Schadprogramm muss zunächst auf das Android Betriebssystem installiert werden, der Abschnitt 3.2.4.2 Einrichten der Anwendungen im laufenden Betrieb über ADB, beschreibt diesen Vorgang. Ein anschließender Neustart des Android Betriebssystems initialisiert und startet dann dieses Schadprogramm. Zu beachten ist hierbei, dass das Schadprogramm direkt nach dem Bootvorgang des Android Betriebssystems aktiv ist, der Neustart sollte bis zum Beginn der Aufzeichnung verzögert werden.

3.2.12 Belauschen eines aktiven Schadprogramms im laufenden Betrieb

Für diesen Abschnitt wird das Rechnersystem (Microsoft Windows) benötigt, mit aktiviertem Hotspot wie im Abschnitt 3.1.4.2 Einrichten eines Hotspots am Rechnersystem, beschrieben ist. Weiter ist das Programm Wireshark aus Abschnitt 3.1.2 Verwendete Software, einsatzbereit und wird im „Promiscuous mode“ für das zu belauschende Netzwerkgerät betrieben. Der „Promiscuous mode“ bezeichnet den Empfangsmodus für die Netzwerkschnittstelle, wobei hier der gesamte ankommende Datenverkehr mitgelesen wird und an das Betriebssystem weiter gegeben wird [I-Wire16, vgl. Seite 43 f.]. Das Wandboard muss ordnungsgemäß angeschlossen, eingeschaltet und eine mit einem lauffähigen Android Betriebssystem konfigurierte Mikro SD-Karte eingesetzt sein (vgl. Abschnitt 3.2.1 Vorbereiten der SD-Karten-Speicherabbilder). In diesem Abschnitt wird das modifizierte Betriebssystem Android 5.0 (Lollipop) verwendet. Als Speicherort für die aufgezeichnete Kommunikation (Capture Files), dient das Verzeichnis des Rechnersystems **F:**. Diese Dateien sind der beiliegenden Projekt-DVD im DVD-Pfad **/Methoden/Capture_malware** zu entnehmen und können zur Kontrolle eingesehen werden.

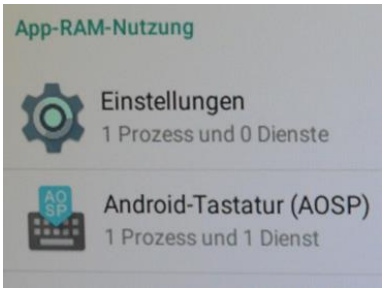
In diesem Abschnitt ist es nötig die Sicherheitsoption „Unbekannte Herkunft“ wie in Abschnitt 3.2.5.2 ab Abbildung 3.2.5-5 Android: Geräteverwaltung der Sicherheitseinstellungen „Unbekannte Herkunft“ zu aktivieren.

Zum Vergleich des Netzwerkverkehrs des Android Betriebssystems mit und ohne dem Schadprogramm, wird die Hardwareplattform mit dem modifiziertem Android 5.0 (Lollipop) Speicherabbild aus Abschnitt 3.2.1 Vorbereiten der SD-Karten-Speicherabbilder gestartet.

Die Aufzeichnung der Netzwerkkommunikation wird durch das Klicken „Start a new live capture“, des Knopfes auf der Wireshark Benutzeroberfläche am Rechnersystem gestartet und 2 Minuten nach dem Start der Android Benutzeroberfläche durch das Klicken „Stop the running live capture“, des Knopfes beendet und durch das Klicken „Save this capture file“, des Knopfes unter folgendem Dateinamen gespeichert:

cap_android502_fromBootTo2minutes_noMalware.pcapng

An dieser Stelle kann in den Einstellungen (*Settings*) unter Apps „AKTIV“ (Apps „RUNNING“) eine Einsicht über die aktuell laufenden Dienste und Anwendungen eingesehen werden. Folgende Abbildung zeigt eine mögliche Teilausgabe für die Android Benutzeroberfläche:



3.2.12-1 Android: Einstellungen "Aktive Anwendungen"

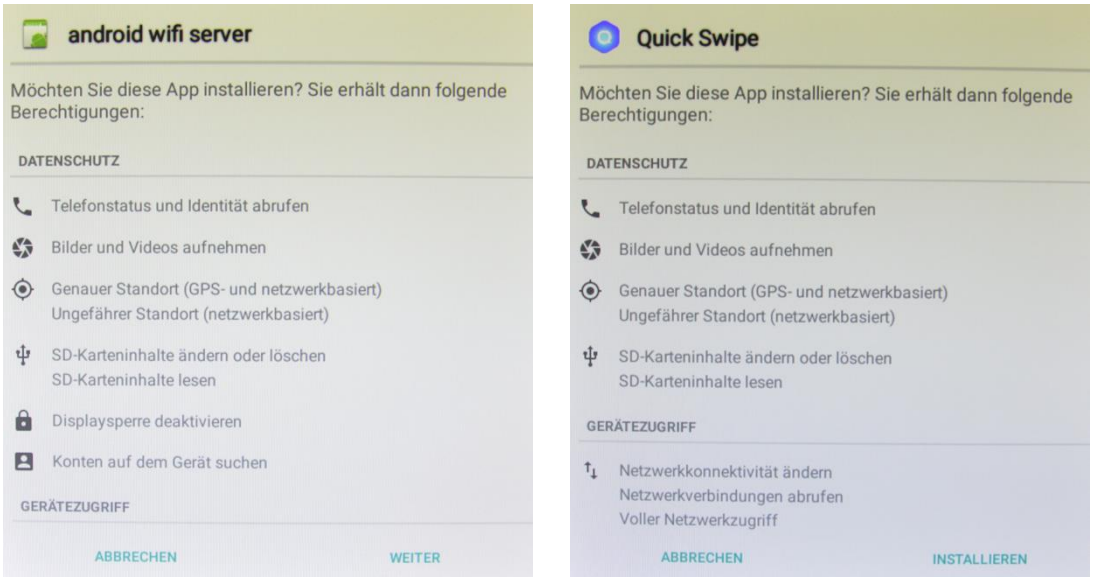
Nach dem Systemstart läuft derzeit nur die Anwendung Android Tastatur (AOSP) die Eingaben über eine virtuelle Tastatur bereitstellt. Die Anwendung Einstellungen hingegen läuft nur, da im Moment der Aufnahme, dieses Programm die Auflistung der aktiven Anwendungen realisiert.

Dem folgend, wird das Schadprogramm wie in Abschnitt 3.2.11 Installation eines Schadprogramms Installation eines auf dem Android platziert. Durch einen Neustart des Android Betriebssystems wird das Schadprogramm dann optimiert, gestartet und ist direkt nach dem Bootvorgang aktiv. Der Neustart allerdings sollte bis zum Start der Aufzeichnung aufgeschoben werden.

Erst wird mit der Aufzeichnung der Netzwerkkommunikation begonnen und danach wird das Android Betriebssystem neu gestartet. Die Aufzeichnung wird erst beendet, wenn der Nachladevorgang des Schadprogramms abgeschlossen ist. Gespeichert wird die Aufzeichnung unter folgendem Dateinamen:

cap_android502_fromBootTo2minutes_firstBootWithMalware.pcapng

Das Schadprogramm hat ein Softwarepaket nachgeladen und fordert über das Android Betriebssystem zur Installation dieser nachgeladenen Schadprogramme auf. Folgende Abbildung zeigt das Verhalten der Installationsaufforderungen des Schadprogramms.



3.2.12-2 Android: Nachladen von Schadprogrammen

Wieder wird zuerst die Aufzeichnung gestartet, erst dann wird die Installation des nachgeladenen Schadprogramms fortgeführt und nachdem sich der Netzwerkverkehr beruhigt hat, wird die Aufzeichnung beendet. Gespeichert wird die Aufzeichnung unter folgendem Dateinamen:

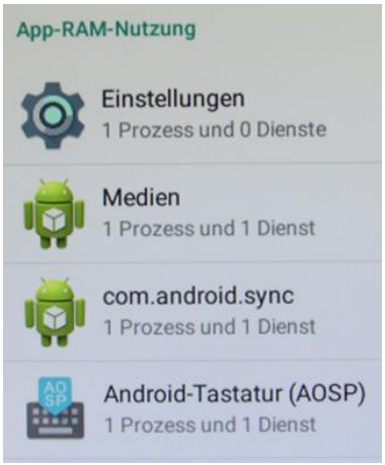
cap_android502_installLoadedMalware.pcapng

Zuletzt wird noch das Netzwerkverhalten des nachgeladenen Schadprogramms aufgezeichnet. Im Fall dieses Schadprogramms wurden die Anwendungen „android wifi server“ und „Quick Swipe“ nachgeladen. Da zu diesem Zeitpunkt schon bekannt ist, das immer „android wifi server“ vom Schadprogram nachgeladen wird, ist dessen Netzwerkkommunikation mit der folgenden Aufzeichnung gesichert. Dafür muss wiederum die Aufzeichnung gestartet werden. Das Schadprogramm muss über die Android Benutzeroberfläche oder den Anwendungs-Manager wie in Abschnitt 2.4.5 beschrieben, ebenfalls gestartet werden. Nachdem eine unbestimmte Aktionszeit des gestarteten Schadprogramms vorüber ist, muss die Aufzeichnung gestoppt und unter folgen-den Dateinamen gespeichert werden:

cap_android502_runningLoadedMalwareWiFiServer.pcapng

Während der Ausführung der Anwendung „android wifi server“ ist zu erkennen, dass die Verknüpfung (Shortcut) im Menü für alle Anwendungen verschwindet und mehrmals eine Benutzeroberfläche ohne Menüs und Einstelloptionen für wenige Sekunden eingeblendet wird.

An dieser Stelle kann wieder in den Einstellungen (*Settings*) unter Apps „AKTIV“ (Apps „RUNNING“) eine Einsicht über die aktuell laufenden Dienste und Anwendungen eingesehen werden. Folgende Abbildung zeigt eine mögliche Teilausgabe für die Android Benutzeroberfläche:



3.2.12-3 Android: Einstellungen "Aktive Anwendungen" mit Schadprogramm

Folgende Anwendungen und Dienste laufen:

Android Tastatur (AOSP)
- Stellt die Eingaben über eine virtuelle Tastatur bereit

Einstellungen
- Listet die aktuelle Anwendungen und Dienste auf

2 Systemdienste
- Anwendung Malware.apk
- Nachgeladene Schadsoftware

Im weiteren Verlauf der explorativen Analyse, ist der Dienst unter der Bezeichnung com.android.sync von großer Bedeutung.

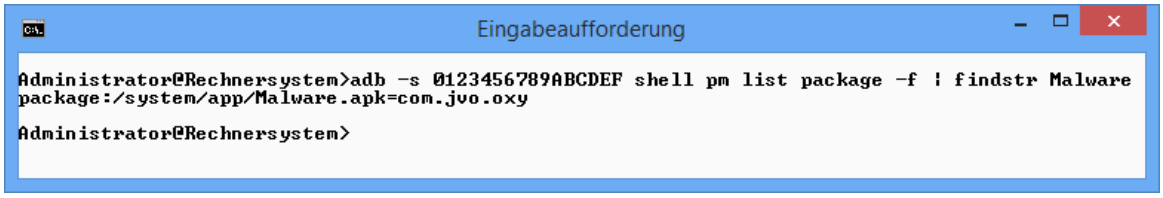
In einem zusätzlichen Schritt können die nachgeladenen Schadprogramme vom Android Betriebssystem gesichert werden, um eine spätere Analyse dieser Anwendungen durchzuführen, folgende Prozedur ist dafür anzuwenden.

Das Rechnersystem (Microsoft Windows) wird benötigt, der Pfad zur Android Debug Bridge ist **C:\SDK\platform-tools** und der Pfad zum Sicherungsspeicher lautet **F:**. Die nachgeladenen Schadprogramme können dieser Arbeit entnommen werden und liegen der Projekt-DVD im DVD-Pfad **/Methoden/Install_loadedMalware**, bei.

Zuerst wird in der Auflistung aller auf dem Android Betriebssystem installierten Anwendungen, die Lokalität der installierten Schadprogramme ermittelt. Folgender Befehl ist dafür am Rechnersystem über das Programm cmd.exe auszuführen:

```
adb -s 0123456789ABCDEF shell pm list packages -f
```

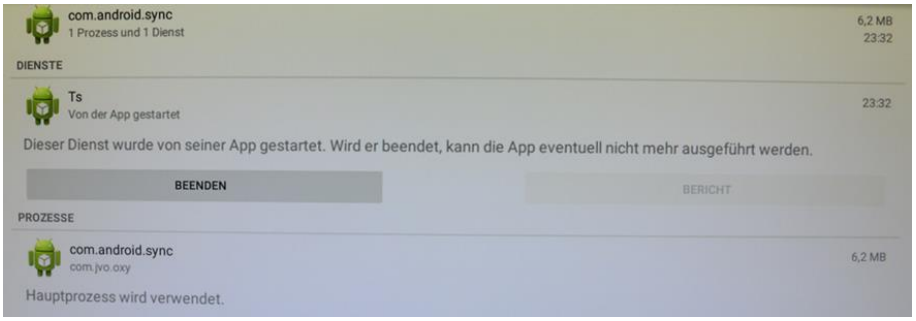
Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.



3.2.12-4 adb: Auflistung aller installierten Anwendungen

Aus dieser Abbildung geht hervor, dass das Schadprogramm unter dem Bezeichner **com.jvo.oxy** angesprochen werden kann.

Bei genauer Betrachtung des Dienstes „com.android.sync“ über die Android Benutzeroberfläche ist festzustellen, dass dieser das Schadprogramm Malware.apk mit dem Bezeichner com.jvo.oxy ist.

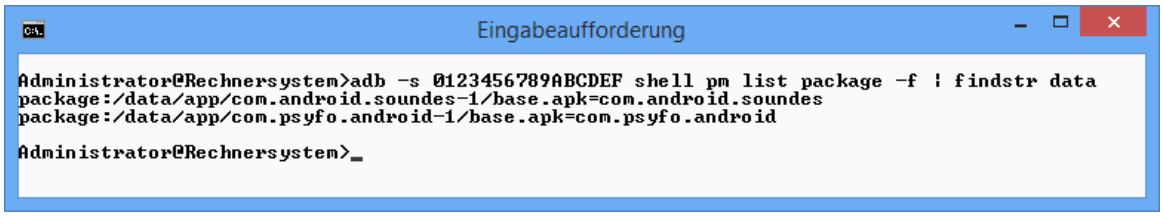


3.2.12-5 Android: Dienst " com.android.sync "

Da das Android Betriebssystem vor der Installation des Schadprogramms keinerlei Nutzeranwendungen aufweist und die Installation der nachgeladenen Anwendungen über den Benutzer vollführt wurde, können mit folgendem Befehl sämtliche Anwendungen des Benutzers aufgelistet werden:

```
adb -s 0123456789ABCDEF shell pm list packages -f | findstr data
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diesen Befehl.

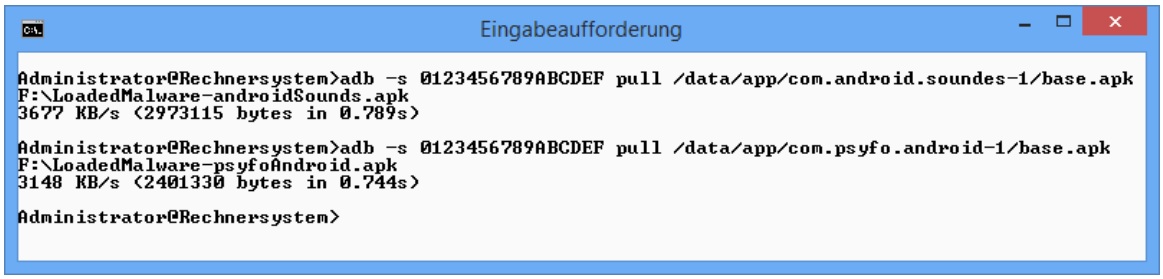


3.2.12-6 adb: Auflistung aller installierten Benutzeranwendungen

Mit dieser Information über die Lokalitäten der nachgeladenen installierten Anwendungen, ist es mit folgenden Befehlen möglich jene zu sichern:

```
adb -s 0123456789ABCDEF pull /data/app/com.android.sounds-1/base.apk
        F:\LoadedMalware-androidSounds.apk

adb -s 0123456789ABCDEF pull /data/app/com.psyfo.android-1/base.apk
        F:\LoadedMalware-psyfoAndroid.apk
```

Folgende Abbildung zeigt eine mögliche Ausgabe für diese Befehle.



3.2.12-7 adb: Sichern der nachgeladenen und installierten Schadprogramme

Um eine Anwendung als Schadprogramm identifizieren zu können, ist nicht nur der Netzwerkverkehr, sondern auch das Verhalten welches die Anwendung zur Laufzeit erbringt, von Bedeutung. Die Analyse des Anwendungsverhaltens ist bis auf die Ausnahme, dass das Schadprogramm weitere Schadprogramme nachlädt somit in die Schadsoftwareeinteilung der Dropper fällt [L-Ayco06, vgl. Seite 17 f.], für diese Arbeit nicht relevant.

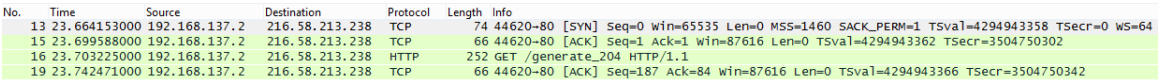
3.2.13 Explorative Analyse der gesammelten Aufzeichnungen

Mit den gesammelten Aufzeichnungen kann eine erste explorative Analyse stattfinden, indem Ziel IP-Adressen untersucht werden, zu dem sich das Android Betriebssystem verbunden hat. Aus Abschnitt 3.1.4.2 Einrichten eines Hotspots am Rechnersystem, gehen die IP-Adressen 192.168.137.2 für das Android Betriebssystem und 192.168.137.1 für das Rechnersystems zur Überwachung hervor. Daraus folgt die Protokollierung jeglicher Verbindungen von 192.168.137.2, zu Zielsystemen die nicht im gleichen Netzwerksegment liegen (192.168.137.0/24). DNS (Domain Name System) und NTP (Network Time Protocol) Anfragen können ebenfalls ausgeschlossen werden.

Die Auswertung der Aufzeichnung cap_android502_fromBootTo2minutes_noMalware.pcapng, kann mit folgendem Filter im Programm Wireshark eingesehen werden:

ip.src==192.168.137.2 and ip.dst!=192.168.137.0/24 and not dns and not ntp

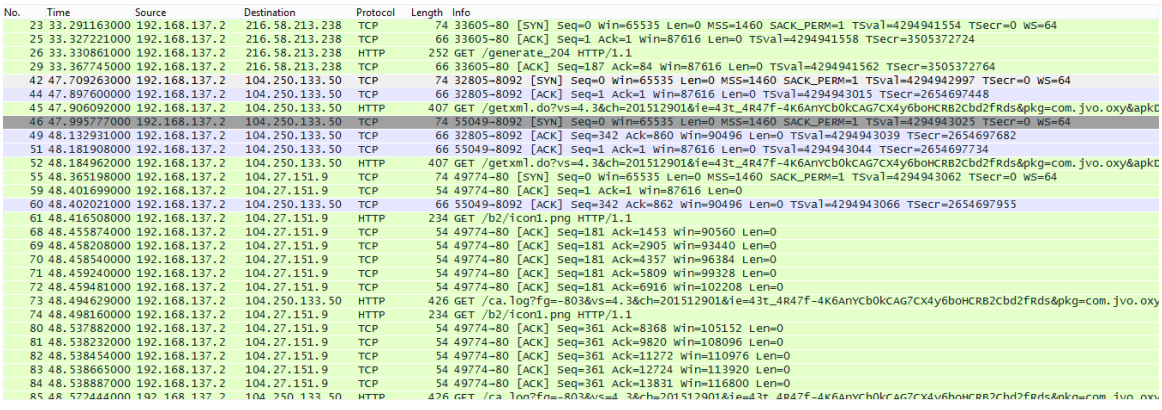
Folgende Abbildung zeigt die Ausgabe an der Wireshark Benutzeroberfläche.



No.	Time	Source	Destination	Protocol	Length	Info
13	23.664153000	192.168.137.2	216.58.213.238	TCP	74	44620→80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=4294943358 TSecr=0 WS=64
15	23.699588000	192.168.137.2	216.58.213.238	TCP	66	44620→80 [ACK] Seq=1 Ack=1 Win=87616 Len=0 TSval=4294943362 TSecr=3504750302
16	23.703225000	192.168.137.2	216.58.213.238	HTTP	252	GET /generate_204 HTTP/1.1
19	23.742471000	192.168.137.2	216.58.213.238	TCP	66	44620→80 [ACK] Seq=187 Ack=84 Win=87616 Len=0 TSval=4294943366 TSecr=3504750342

3.2.13-1 Wireshark: Aufzeichnung des Android Starts

Während dem Bootvorgang authentisiert sich das Android Betriebssystems bei einem Google Server. Dies ist durch das Ziel (*Destination*) mit der IP 216.58.213.238, einem von Google Inc. betriebenen Server [I-Tcp16], gegeben. Weiterer Netzwerkverkehr ist nicht vorhanden. Zum Vergleich der vorigen Aufzeichnung ist der Start des Android Betriebssystems mit dem Schadprogramm zu wählen. Beim Vergleich fallen verschiedene Verbindungen zu Fremdserversystemen auf. Folgende Abbildung zeigt die Ausgabe an der Wireshark Benutzeroberfläche.



No.	Time	Source	Destination	Protocol	Length	Info
23	33.291163000	192.168.137.2	216.58.213.238	TCP	74	33605→80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=4294941554 TSecr=0 WS=64
25	33.327221000	192.168.137.2	216.58.213.238	TCP	66	33605→80 [ACK] Seq=1 Ack=1 Win=87616 Len=0 TSval=4294941558 TSecr=3505372724
26	33.330861000	192.168.137.2	216.58.213.238	HTTP	252	GET /generate_204 HTTP/1.1
29	33.367745000	192.168.137.2	216.58.213.238	TCP	66	33605→80 [ACK] Seq=187 Ack=84 Win=87616 Len=0 TSval=4294941562 TSecr=3505372764
42	47.709263000	192.168.137.2	104.250.133.50	TCP	74	32805→8092 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=4294942997 TSecr=0 WS=64
44	47.897600000	192.168.137.2	104.250.133.50	TCP	66	32805→8092 [ACK] Seq=1 Ack=1 Win=87616 Len=0 TSval=4294943015 TSecr=2654697448
45	47.906092000	192.168.137.2	104.250.133.50	HTTP	407	GET /getxml.do?vs=4.3&ch=201512901&ie=43t_4R47f-4K6AnYCb0kCAG7CX4y6boHCRB2Cbd2fRds&pkg=com.jvo.oxy&apkD
46	47.995777000	192.168.137.2	104.250.133.50	TCP	74	55049→8092 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=4294943025 TSecr=0 WS=64
49	48.132931000	192.168.137.2	104.250.133.50	TCP	66	32805→8092 [ACK] Seq=342 Ack=860 Win=90496 Len=0 TSval=4294943039 TSecr=2654697682
51	48.181908000	192.168.137.2	104.250.133.50	TCP	66	55049→8092 [ACK] Seq=1 Ack=1 Win=87616 Len=0 TSval=4294943044 TSecr=2654697734
52	48.184962000	192.168.137.2	104.250.133.50	HTTP	407	GET /getxml.do?vs=4.3&ch=201512901&ie=43t_4R47f-4K6AnYCb0kCAG7CX4y6boHCRB2Cbd2fRds&pkg=com.jvo.oxy&apkD
55	48.365198000	192.168.137.2	104.27.151.9	TCP	74	49774→80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=4294943062 TSecr=0 WS=64
59	48.401699000	192.168.137.2	104.27.151.9	TCP	54	49774→80 [ACK] Seq=1 Ack=1 Win=87616 Len=0
60	48.402021000	192.168.137.2	104.250.133.50	TCP	66	55049→8092 [ACK] Seq=342 Ack=862 Win=90496 Len=0 TSval=4294943066 TSecr=2654697955
61	48.416508000	192.168.137.2	104.27.151.9	HTTP	234	GET /b2/icon1.png HTTP/1.1
68	48.455874000	192.168.137.2	104.27.151.9	TCP	54	49774→80 [ACK] Seq=181 Ack=1453 Win=90560 Len=0
69	48.458208000	192.168.137.2	104.27.151.9	TCP	54	49774→80 [ACK] Seq=181 Ack=2905 Win=93440 Len=0
70	48.458540000	192.168.137.2	104.27.151.9	TCP	54	49774→80 [ACK] Seq=181 Ack=4357 Win=96384 Len=0
71	48.459240000	192.168.137.2	104.27.151.9	TCP	54	49774→80 [ACK] Seq=181 Ack=5809 Win=99328 Len=0
72	48.459481000	192.168.137.2	104.27.151.9	TCP	54	49774→80 [ACK] Seq=181 Ack=6916 Win=102208 Len=0
73	48.494629000	192.168.137.2	104.250.133.50	HTTP	426	GET /ca.log?fg=-803&vs=4.3&ch=201512901&ie=43t_4R47f-4K6AnYCb0kCAG7CX4y6boHCRB2Cbd2fRds&pkg=com.jvo.oxy
74	48.498160000	192.168.137.2	104.27.151.9	HTTP	234	GET /b2/icon1.png HTTP/1.1
80	48.537882000	192.168.137.2	104.27.151.9	TCP	54	49774→80 [ACK] Seq=361 Ack=8368 Win=105152 Len=0
81	48.538232000	192.168.137.2	104.27.151.9	TCP	54	49774→80 [ACK] Seq=361 Ack=9820 Win=108096 Len=0
82	48.538454000	192.168.137.2	104.27.151.9	TCP	54	49774→80 [ACK] Seq=361 Ack=11272 Win=110976 Len=0
83	48.538665000	192.168.137.2	104.27.151.9	TCP	54	49774→80 [ACK] Seq=361 Ack=12724 Win=113920 Len=0
84	48.538887000	192.168.137.2	104.27.151.9	TCP	54	49774→80 [ACK] Seq=361 Ack=13831 Win=116800 Len=0
85	48.572444000	192.168.137.2	104.250.133.50	HTTP	426	GET /ca.log?fg=-803&vs=4.3&ch=201512901&ie=43t_4R47f-4K6AnYCb0kCAG7CX4y6boHCRB2Cbd2fRds&pkg=com.jvo.oxy

3.2.13-2 Wireshark: Aufzeichnung des Android Starts mit Schadprogramm

Da 3617 Daten Pakete ermittelt werden können, ist die Darstellung der Abbildung auf einige wenige Pakete am Anfang der Aufzeichnung begrenzt.

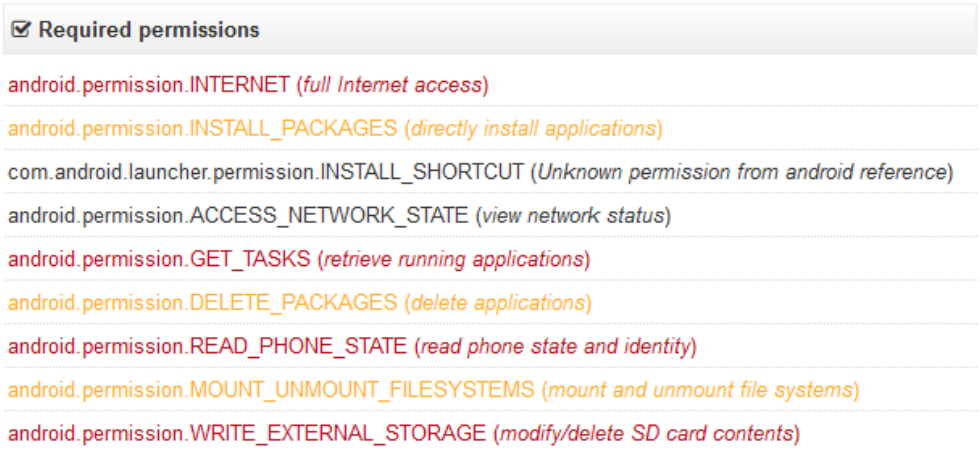
Die Ausgabe kann durch einen Klick der Spalte „Destination“ an der Wireshark Benutzeroberfläche nach Zieladressen sortiert werden. Folgende verschiedene IP-Adressen wurden registriert:

Zieladresse	Pakete	ISP	Lokalität (Stadt, Land, Staat)
216.58.213.238	4	Google Inc.	Mountain View, California, United States
104.250.133.50	25	GorillaServers, Inc.	Los Angeles, California, United States
104.27.151.9	3588	CloudFlare, Inc.	San Francisco, California, United States

Tabelle 3.2.13-1 IP-Adressen-Auswertung des Schadprogramms [I-Tcp16]

Das Schadprogramm lädt ohne die Zustimmung und das Wissen des Benutzers, eine weitere Anwendung von einem Cloud basierenden Datenspeicher herunter. Im Fall dieser Schadsoftware ist der Anbieter „CloudFlare“.

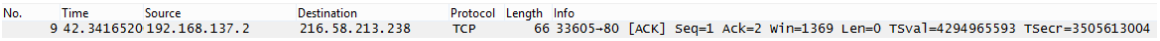
Das Schadprogramm wird von VirusTotal, einem kostenlosen Dienstanbieter der verdächtige Dateien und URLs analysiert und das Erkennen von Bedrohungen jeglicher Art von Schadsoftware ermöglicht, als potentielle Schadsoftware erkannt. Dem Bericht [I-ViToA16] sind folgende geforderten Berechtigungen des Schadprogramms zu entnehmen:



3.2.13-3 VirusTotal: Teilbericht der geforderten Berechtigungen des Schadprogramms

Diesen Berechtigungen ist das Ausmaß über das Verhalten der Anwendung bei vollem Einsatz zu entnehmen. So kann die Anwendung Daten aus dem Internet herunterladen, Anwendungen installieren sowie deinstallieren, das Dateisystem beliebig ins System einbinden oder trennen und auf die externen Speichermedien mit schreibendem Recht zugreifen.

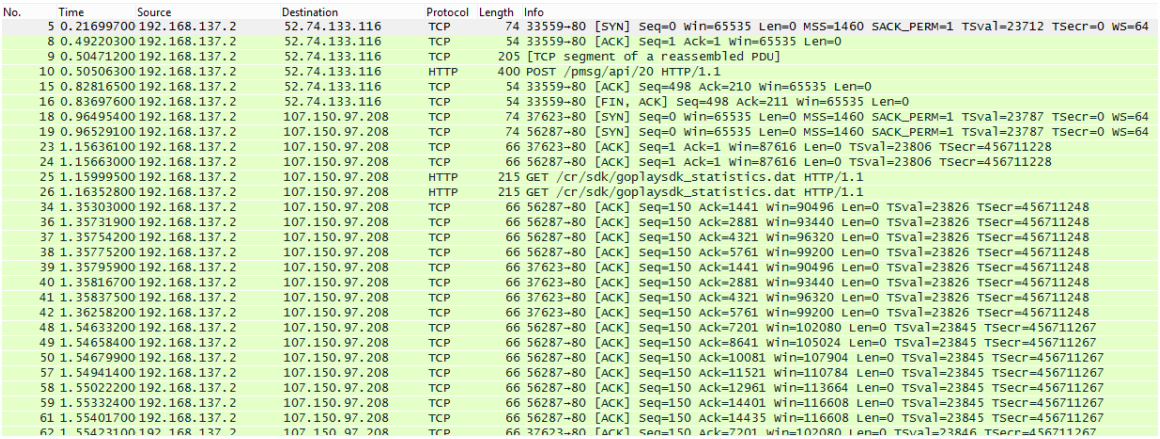
Während der Installation des nachgeladenen Schadprogramms „android wifi server“, sind keine relevanten Verbindungen von Bedeutung, folgende Abbildung zeigt die Ausgabe an der Wireshark Benutzeroberfläche.



No.	Time	Source	Destination	Protocol	Length	Info
9	42.3416520	192.168.137.2	216.58.213.238	TCP	66	33605→80 [ACK] Seq=1 Ack=2 Win=1369 Len=0 TSval=4294965593 TSecr=3505613004

3.2.13-4 Wireshark: Installation des nachgeladenen Schadprogramms

Wird das nachgeladene und installierte Schadprogramm gestartet, so fallen bemerkenswert viele verschieden Verbindungen zu Fremdserversystemen auf. Folgende Abbildung zeigt die Ausgabe an der Wireshark Benutzeroberfläche.



No.	Time	Source	Destination	Protocol	Length	Info
5	0.21699700	192.168.137.2	52.74.133.116	TCP	74	33559→80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=23712 TSecr=0 WS=64
8	0.49220300	192.168.137.2	52.74.133.116	TCP	54	33559→80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
9	0.50471200	192.168.137.2	52.74.133.116	TCP	205	[TCP segment of a reassembled PDU]
10	0.50506300	192.168.137.2	52.74.133.116	HTTP	400	POST /pmsg/api/20 HTTP/1.1
15	0.82816500	192.168.137.2	52.74.133.116	TCP	54	33559→80 [ACK] Seq=498 Ack=210 Win=65535 Len=0
16	0.83697600	192.168.137.2	52.74.133.116	TCP	54	33559→80 [FIN, ACK] Seq=498 Ack=211 Win=65535 Len=0
18	0.96495400	192.168.137.2	107.150.97.208	TCP	74	37623→80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=23787 TSecr=0 WS=64
19	0.96529100	192.168.137.2	107.150.97.208	TCP	74	56287→80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=23787 TSecr=0 WS=64
23	1.15636100	192.168.137.2	107.150.97.208	TCP	66	37623→80 [ACK] Seq=1 Ack=1 Win=87616 Len=0 TSval=23806 TSecr=456711228
24	1.15663000	192.168.137.2	107.150.97.208	TCP	66	56287→80 [ACK] Seq=1 Ack=1 Win=87616 Len=0 TSval=23806 TSecr=456711228
25	1.15999500	192.168.137.2	107.150.97.208	HTTP	215	GET /cr/sdk/goplaysdk_statistics.dat HTTP/1.1
26	1.16352800	192.168.137.2	107.150.97.208	HTTP	215	GET /cr/sdk/goplaysdk_statistics.dat HTTP/1.1
34	1.35303000	192.168.137.2	107.150.97.208	TCP	66	56287→80 [ACK] Seq=150 Ack=1441 Win=90496 Len=0 TSval=23826 TSecr=456711248
36	1.35731900	192.168.137.2	107.150.97.208	TCP	66	56287→80 [ACK] Seq=150 Ack=2881 Win=93440 Len=0 TSval=23826 TSecr=456711248
37	1.35754200	192.168.137.2	107.150.97.208	TCP	66	56287→80 [ACK] Seq=150 Ack=4321 Win=96320 Len=0 TSval=23826 TSecr=456711248
38	1.35775200	192.168.137.2	107.150.97.208	TCP	66	56287→80 [ACK] Seq=150 Ack=5761 Win=99200 Len=0 TSval=23826 TSecr=456711248
39	1.35795900	192.168.137.2	107.150.97.208	TCP	66	37623→80 [ACK] Seq=150 Ack=1441 Win=90496 Len=0 TSval=23826 TSecr=456711248
40	1.35816700	192.168.137.2	107.150.97.208	TCP	66	37623→80 [ACK] Seq=150 Ack=2881 Win=93440 Len=0 TSval=23826 TSecr=456711248
41	1.35837500	192.168.137.2	107.150.97.208	TCP	66	37623→80 [ACK] Seq=150 Ack=4321 Win=96320 Len=0 TSval=23826 TSecr=456711248
42	1.36258200	192.168.137.2	107.150.97.208	TCP	66	37623→80 [ACK] Seq=150 Ack=5761 Win=99200 Len=0 TSval=23826 TSecr=456711248
48	1.54633200	192.168.137.2	107.150.97.208	TCP	66	56287→80 [ACK] Seq=150 Ack=7201 Win=102080 Len=0 TSval=23845 TSecr=456711267
49	1.54658400	192.168.137.2	107.150.97.208	TCP	66	56287→80 [ACK] Seq=150 Ack=8641 Win=105024 Len=0 TSval=23845 TSecr=456711267
50	1.54679900	192.168.137.2	107.150.97.208	TCP	66	56287→80 [ACK] Seq=150 Ack=10081 Win=107904 Len=0 TSval=23845 TSecr=456711267
57	1.54941400	192.168.137.2	107.150.97.208	TCP	66	56287→80 [ACK] Seq=150 Ack=11521 Win=110784 Len=0 TSval=23845 TSecr=456711267
58	1.55022200	192.168.137.2	107.150.97.208	TCP	66	56287→80 [ACK] Seq=150 Ack=12961 Win=113664 Len=0 TSval=23845 TSecr=456711267
59	1.55332400	192.168.137.2	107.150.97.208	TCP	66	56287→80 [ACK] Seq=150 Ack=14401 Win=116608 Len=0 TSval=23845 TSecr=456711267
61	1.55401700	192.168.137.2	107.150.97.208	TCP	66	56287→80 [ACK] Seq=150 Ack=14435 Win=116608 Len=0 TSval=23845 TSecr=456711267
62	1.55423100	192.168.137.2	107.150.97.208	TCP	66	37623→80 [ACK] Seq=150 Ack=7201 Win=102080 Len=0 TSval=23846 TSecr=456711267

3.2.13-5 Wireshark: Ausführung des nachgeladenen und installierten Schadprogramms

Da auch hier sehr viele Pakete ermittelt werden können, in der Stückzahl genau 2397 Pakete, ist die Darstellung der Abbildung auf einige wenige Pakete am Anfang der Aufzeichnung begrenzt.

Die Ausgabe kann wieder durch einen Klick der Spalte „Destination“ an der Wireshark Benutzeroberfläche nach Zieladressen sortiert werden. Folgende verschiedene IP-Adressen wurden registriert:

Zieladresse	**Pakete**	**ISP**	**Lokalität (Stadt, Land, Staat)**
40.127.97.239	48	Microsoft Corporation	Dublin, Leinster, Ireland
52.20.99.13	108	Amazon.com, Inc.	Ashburn, Virginia, United States
52.32.91.76	6	Amazon.com, Inc.	Boardman, Oregon, United States
52.34.22.173	9	Amazon.com, Inc.	Boardman, Oregon, United States

52.40.28.160	18	Amazon.com, Inc.	Boardman, Oregon, United States
52.49.84.66	9	Amazon.com, Inc.	Dublin, Leinster, Ireland
52.73.115.54	32	Amazon.com, Inc.	Ashburn, Virginia, United States
52.74.133.116	96	Amazon.com Tech Telecom	Singapore, Central Singapore Community Development Counc, Singapore
52.85.173.179	331	Amazon.com, Inc.	Seattle, Washington, United States
52.85.173.235	1254	Amazon.com, Inc.	Seattle, Washington, United States
52.86.243.225	18	Amazon.com, Inc.	Ashburn, Virginia, United States
52.89.73.121	12	Amazon.com, Inc.	Boardman, Oregon, United States
54.154.125.59	6	Amazon.com, Inc.	Dublin, Leinster, Ireland
54.169.208.189	6	Amazon.com Tech Telecom	Singapore, Central Singapore Community Development Counc, Singapore
54.247.168.31	13	Amazon.com, Inc.	Dublin, Leinster, Ireland
107.150.97.208	**361**	**Zenlayer Inc**	**Shanghai, Shanghai, China**
107.20.198.216	11	Amazon.com, Inc.	Ashburn, Virginia, United States
107.22.213.188	11	Amazon.com, Inc.	Ashburn, Virginia, United States
139.162.7.203	**16**	**Linode, LLC**	**Singapore**
143.95.150.84	6	Colo4, LLC	Los Angeles, California, United States

176.34.120.72	7	Amazon.com, Inc.	Dublin, Leinster, Ireland
176.34.236.148	14	Amazon.com, Inc.	Dublin, Leinster, Ireland
192.95.127.14	5	TheWay Holdings, LLC	Hudson, Florida, United States

Tabelle 3.2.13-2 IP-Adressen-Auswertung des nachgeladenen Schadprogramms [I-Tcp16]

Bei dieser Tabelle ist zu berücksichtigen, dass die rot markierten Einträge keine Domains stellen. Weiter wird auch klar, dass die Anwendung Verbindungsversuche startet und dabei systematisch aus einer bekannten Liste aus Servern, diese versucht zu erreichen. Zu beachten ist, dass die Ordnung der Tabelle nicht mit dem des Verbindungsversuchs aus der Aufzeichnung übereinstimmt.

Das Schadprogramm com.psyfo.android alias „android wifi server“, wird von VirusTotal als potentielle Schadsoftware erkannt. Die Liste der Berechtigungen des Berichts [I-ViToB16] ist sehr lang, einige werden folgend kurz benannt.

Die Anwendung hat vollen Zugriff zum Internet, kann die Tastensperre deaktivieren, kann Anwendungen deinstallieren, hat vollen Zugriff auf externe Speichermedien und Kamera, kann Systemmeldungen generieren, kann das Dateisystem beliebig ins System einbinden oder trennen und vieles mehr. Der Bericht ist um einiges länger als der zuvor genannte. Auch die bei der Signatursuche festgestellten Treffer der DEX Datei, sind mit 19 von 57 Funden höher, als die beim vorigen Bericht gefundenen 2 von 53 Funden.

Das Potenzial dieses Schadprogramms kann nicht genau bestimmt werden, jedoch ist es um einiges höher als das Potenzial des Schadprogramms, dass diese nachgeladen hat.

Als Quintessenz kann unter Verwendung der Klassifizierung nach [L-Ayco06], die Anwendung Malware.apk, als Dropper und die nachgeladene Anwendung „android wifi server“, als Trojaner bezeichnet werden. Ein Dropper ist für das Nachladen weiterer Schadprogramme bekannt, ein Trojaner hingegen tarnt sich als nützliche Anwendung und führt schädliche Funktionen aus [L-Ayco06, vgl. Seite 11 ff.].

4 Ergebnisse

Im folgenden Kapitel werden die durchgeführten Methoden als Ergebnis ausgewertet und dargestellt.

4.1 Auswahl des SD-Karten Speicherabbilds

Die breit gefächerte Auswahl an vorkonstruiert bestehenden Speicherabbildern, angeboten vom Hersteller und der Wandboard Gemeinschaft ist eine Möglichkeit ein Betriebssystem zu installieren. Eine Weitere ist die Implementierung eingebetteter Android Betriebssysteme durch gepatchte Linux-Kernels und angepasste Android Betriebssysteme. Für die Aufgabenstellung ist die Nutzung eines Android Betriebssysteme mit hoher Verbreitung zwingend nötig. Dafür eignet sich nach [I-AnDeZ15] ganz besonders das Speicherabbild Android 4.4 (Kitkat) vom 3. März 2015 mit einer Verbreitung von 36,6 % und das modifizierte Speicherabbild Android 5.0 (Lollipop) vom 5. November 2015 mit einer Verbreitung von 16,3 %. Zusammen sind dies immerhin schon 52,9 % und damit schon über die Hälfte der Marktabdeckung aller verwendeten Android Betriebssysteme weltweit.

4.2 Auswahl der Verbindungstechnologie

Die Realisierung der Verbindung an der Hardwareplattform könnte per „USB über IP“ (Zusatz Modul) realisiert werden, dabei wird die USB Kommunikation über ein TCP Netzwerk gekapselt. Diese Handlungsweise ist für die erste Etablierung zur Hardwareplattform nötig oder bei fehlerhaften Konfiguration, neuer Installation des Betriebssystems oder Störungen mit Datenverlust der Vertrauensstellung zum Fernsteuerungsrechner. Da die permanente Kommunikation mit der Hardwareplattform zur Analyse von Android Schadsoftware unverzichtbar ist, kann nur die schnellere und zuverlässigere Verwendung der Kommunikationsschnittstelle Ethernet, unter Betrachtung folgender Verbindungskriterien, Zustimmung finden. Die Verbindungskriterien sind zum Ersten die Geschwindigkeit. Da der Unterschied der Ethernet Schnittstelle mit 1000 Megabit pro Sekunde zur WLAN Schnittstelle (nur eine Antenne) mit 150 Megabit pro Sekunde beträgt. Zum Zweiten die Stabilität. Da Ethernet durch Interferenz sowie Funkstörungen weniger beeinflussbar als eine Funkverbindung ist.

4.3 Nutzung des Play Stores für Android

Der Play Store (Google Play) ist ein Softwarearchiv zur Verteilung und Installation von Android Anwendungen, diese Idee wird von einigen Linux-Distributionen schon seit langer Zeit genutzt. Solch eine Softwareausgabe ist auch durch Software-Repositories oder dezentrale Verteilung möglich, bietet aber eine bequemere Verbreitung von Anwendungen für Benutzer und zur Verfügung stellende Firmen oder Organisationen. Für die Analyse von Android Schadsoftware ist der Play Store nicht geeignet, sondern stellt höchstens Anwendung zum weiteren Gebrauch dar. Ist der Play Store im Android Betriebssystem eingerichtet, so können bestimmte Funktionen zur Aufgabenbewältigung mit dem System bereitgestellt werden.

4.4 Installation von Anwendungen ohne Play Store

Das gewählte Verfahren zur Installation von Anwendungen auf dem Android Betriebssystem per Android Debug Bridge wird maßgeblich dadurch bestimmt, das während der Analyse der Android Schadsoftware diese automatisiert auf die Plattform übertragen und installiert werden soll. Interaktionen mit der Benutzeroberfläche sind der Sache dabei nicht hilfreich. Auch die Entnahme der Mikro SD-Karte für jede Untersuchung ist für das Erreichen einer schnellen Analyse nicht möglich.

4.5 Deinstallation von Anwendungen ohne Play Store

Auch die Bewerkstelligungen zur Deinstallation von Anwendungen auf dem Android Betriebssystem per Android Debug Bridge werden nur dadurch bestimmt, dass nach der Analyse von Android Schadsoftware und der Ausgabe dieser, eine Benutzerinteraktion mit der Android Oberfläche dem Automatisierungsprozess entgegensteht.

4.6 Root-Zugriff für Android Anwendungen

Für die Durchführung der Analyse von Schadsoftware kann es nötig sein, den Root-Zugriff für Android Anwendungen einzurichten. Dies stellt eine Erweiterung des Root-Zugriffs über die Android Debug Bridge dar und da das Android Betriebssystem über einen Steuerserver fernsteuerbar ist, stehen einem somit auch die Möglichkeiten der Überwachung, der Protokollierung und der Analyse zur Verfügung. Von bedeutendem Vorteil ist die Kenntnis über die Einrichtung des Root-Zugriffs, welche der Anwendung das Recht über die Android Benutzeroberfläche gibt, die es benötigt.

4.7 Nutzung von ausführbaren Dateien im RAW Format

Im Gegensatz zum Root-Zugriff für Android Anwendungen ist die Nutzung von ausführbaren Dateien im RAW Format unumgänglich. Da im Laufe der Prüfung von Android Anwendungen auf Schadsoftware differierende Ansätze verfolgt werden und für jeden dieser Ansätze verschiedenste Software genutzt wird und keine Software bekannt ist, die alle möglichen Analyseansätze abdeckt, muss für jeden Ansatz auch eine eigens dafür erstellte Software, auf dem zu untersuchenden System zur Verfügung gestellt werden. Die Ausführung von Software in Form von ausführbaren Dateien im RAW Format ist somit auch Bestandteil der Analyse.

4.8 Fernsteuerung und Datenaustausch mit Android

Im Verlauf dieser Arbeit wurden mehrere Anwendungen mit den Funktionalitäten eines Fernterminals, eines FTP-Servers oder beidem getestet und miteinander verglichen. Dabei wurde besonderes Augenmerk auf eine automatisierbare Verwendung, ressourcenschonenden Einsatz, auf verwendbare Schnittstellen und eine Softwaredokumentation gelegt. Der Datenaustausch zwischen Fernsteuerungsrechner und Android Betriebssystem sowie die Fernsteuerung selbigen Systems, lassen nach schlussendlichem Vergleich die Verwendung der Android Debug Bridge als einzig gute Lösung zu. Fast alle getesteten Android Anwendungen haben vorwiegend kommerziellen Inhalt, nutzen zu viele Ressourcen bei dem das System ausgebremst wird oder bieten nicht die mächtige Funktionalität, die bei der Android Debug Bridge so im Vordergrund steht.

4.9 Installationsmethode der Schadsoftware

Die unterschiedlichen Ansätze die ein Schadprogramm verfolgt, macht die Installation auf dem Android Betriebssystem sehr vielfältig, dabei ist zu unterscheiden, ob ein Schadprogramm tiefergehende Systemrechte benötigt oder nicht. Es ist beispielsweise möglich, dass bei der Installation einer Anwendung, diese fragt ob der Geräteadministrator für diese Anwendung aktiviert werden soll, damit diese Anwendung zusätzliche Funktionen ausführen kann, die systeminterne Einstellungen betreffen. Im Falle der Automatisierung der Analyse von Schadsoftware, ist die Installation jedoch auf zwei Methoden zu beschränken. Die erste Methode ist in Abschnitt 3.2.5.1 Installation der Anwendungen über die Android Debug Bridge beschrieben und eignet sich für Schadsoftware die keine erweiterten Berechtigungen bedarf, da sie über den Benutzer installiert wird. Solche Schadsoftware ist jedoch selten, da sie weder private Daten anderer Anwendungen auslesen kann, noch andere Anwendungen steuern kann, jedoch ist der Zugriff auf die in den Berechtigungen zur Installation gestellten Ressourcen oder die Überdeckung von anderen Anwendungen möglich und eignet sich somit als Analyseanwendungen oder Werbeanwendungen. An dieser Stelle ist aber zu bedenken, das neuere Android Betriebssysteme bessere Sicherheitsmechanismen bereitstellen, um Anwendungen dahingehend zu kontrollieren. Beispielsweise ab Android 6.0 (Marshmallow) können die Berechtigungen der Anwendungen einzeln verändert werden, zudem kann genau gesteuert werden, welche Anwendungen andere Anwendungen verdecken dürfen. Die zweite Methode ist in Abschnitt 3.2.4.2 Einrichten der Anwendungen im laufenden Betrieb über ADB beschrieben. Diese Arbeitsweise ist aus Sicht des Schadprogramms wesentlich effektiver, da durch den Eingriff des Schadprogramms ins System wesentlich mehr Funktionalitäten bereitgestellt werden können. Das Schadprogramm kann so über die Grenzen der Dalvik VM, selbst über die Grenzen der Nutzertrennung auf Systemebene hinweg agieren und somit enormen Schaden verursachen. Im Verlauf der Automatisierung der Analyse von Schadsoftware auf dem Android Betriebssystem, wird der zuvor beschriebene Weg als Installationsstrategie gewählt.

4.10 Analyse des Netzwerkverkehrs des Schadprogramms

Das auf dem Android Betriebssystem installierte Schadprogramm, kann über ein Fremdsystem des zur Verfügung gestellten Hotspots für das Android, netzwerktechnisch überwacht werden. Durch aufzeichnen der Netzwerkkommunikation über dieses Fremdsystem, ist es dem Schadprogramm fast unmöglich einer derartigen Analyse zu entgehen. Diese erste explorative Analyse, ist jedoch nur eine triviale rudimentäre Vorstufe einer Analysekette, die im Laufe dieses Projektes bei der Analyse von Android Schadsoftware weiterentwickelt wird.

5 Diskussion

Im abschließenden Kapitel werden die bisher gewonnenen Ergebnisse zusammengefasst und eine Bewertung der Leistung aus Sicht des Autors vorgenommen. Weiterentwicklungspotenziale für tiefgehende Eingriffe in das Android Betriebssystem, durch Modifizierung von Systemdiensten und einbinden eigener Skripte sowie die Erstellung eigener Android Betriebssysteme für die Hardwareplattform Wandboard, geben einen Ausblick auf bevorstehende Arbeiten. Die Verbesserung der ersten explorativen Analyse zu einer vollwertigen automatischen Schadsoftware Analyse mit Verhaltenserkennung, Musteranalyse und Auswertung der Protokolldateien hat vorrangiges Potenzial.

5.1 Bewertung der Arbeit

Die Intention dieser Bachelorarbeit war es die Hardwareplattform Wandboard für eine Android Schadsoftware Analyse zu konfigurieren. Dieses Ziel wurde erreicht. Weiter wurde die Hardwareplattform Wandboard näher betrachtet und das verwendete Betriebssystem Android genauer erörtert. Die erarbeiteten Handlungsanweisungen des Kompendiums, ermöglicht die systematische Durchführung der Konfiguration der Hardwareplattform, als vorbereitende Maßnahme für spätere Analysen.

Diese Arbeit zeigt, dass im laufenden Betrieb des Wandboards, eine erste explorative Analyse von Android Schadprogrammen möglich ist. Mit den passenden Strategien einer erweiterten Analyse ist es diskutabel, die Auswertung synchron auf mehreren Wandboards durch Emulation unterschiedlicher Geräte realistisch und produktiv durchzuführen. Es sollte in Erwägung gezogen werden, dass ein Sammeln und Zusammenführen von Ergebnissen an einer zentralen Stelle erfolgt. Die Analysemethoden von Android Schadprogrammen dieser Arbeit, nimmt Fokus auf Schadprogramme die eine emulierte Umgebung erkennen und durch vortäuschen falscher Anwendung Verhaltensmuster versuchen, eine Erkennung zu verhindern.

Nachfolgend sollen nun noch Hardwarevergleiche und Softwarevergleiche die richtige Wahl der Verwendung des Wandboards zur Analyse von Android Schadprogrammen untermauern.

5.1.1 Vergleichen von Hardwareplattformen

Seit der Veröffentlichung der ersten ARM basierenden Single Board Computer (SBC) ist ein Hype um diese Hardware entstanden. Die konstitutiven Punkte für diesen Durchbruch sind die vollständige Funktionalität eines Computers auf kleinster Fläche, die Benutzer programmierbaren GPIO-Pins zur Ansteuerung und die geringen Kosten gegenüber traditionellen Computern. Da der Markt für die Single Board Computer deutlich am Wachsen und mannigfaltig ist, stehen nicht nur eine sondern mehrere Plattformen zur Auswahl. Zu den gängigsten Geräten mit der Möglichkeit ein Android Betriebssystem betreiben zu können, zählen gegenwärtig außer dem Wandboard noch das BeagleBone, das Odroid, das Raspberry und das Udoo. Die Liste der verfügbaren Boards ist sehr viel länger, dies jedoch sind die Hardwareplattformen die in diesem Abschnitt technisch näher betrachtet werden.

Die nachstehenden Tabellen vergleichen die Hardware von jeweils zwei Single Board Computern. Dabei ist das Wandboard Quad die Referenz-Plattform die mit einer zweiten Plattform verglichen wird, die technischen Details des Wandboard Quad können der Tabelle 2.2.1-1 Wandboard i.MX6 Spezifikation [I-WanB16] entnommen werden. Für den Hardwarevergleich wurden die Attribute so gewählt, dass die Realisierung einer parallelen Analyse von Android Schadsoftware auf mehreren gleichen Hardwareplattformen, mit unterschiedlich emulierten Umgebungen stattfinden soll. Im späteren Betrieb müssen die Interaktionen zwischen Server und Plattform schnell und zuverlässig zur Ausführung kommen, um daraus Analysen zu erstellen. Dies besagt, dass nicht nur die Leistungsanforderung an das Gerät sehr hoch ist, sondern auch die Kommunikationsschnittstellen keine Engpässe darstellen dürfen. Somit wird bei allen Vergleichen das Hauptaugenmerk auf den Prozessor, den Arbeitsspeicher und die vorhandenen Schnittstellen zum Verbindungsaufbau gelegt.

Wahl der Hardwareplattform

Der nachfolgende Hardwarevergleich zwischen Wandboard Quad, BeagleBone Black Rev C, Odroid XU4, Raspberry Pi 3 Model B und dem Udoo Quad, lässt unter den Gesichtspunkten einer schnellen Analyse von Android Schadsoftware und verschiedenen Möglichkeiten zum Verbindungsaufbau mit dieser Plattform, nur das Wandboard Quad als Hardwareplattform zu. Das Wandboard Quad bietet für den im Verhältnis zu den anderen Plattformen hohen Preis, eine immens gute Leistung und sehr viele Funktionen. Die beachtliche Leistung wird durch einen gegenwärtig sehr guten Single Board Computer Prozessor, unter Verwendung von reichlich Arbeitsspeicher erwirkt. Das breit gefächerte Angebot an Kommunikationsschnittstellen und das Nutzen zusätzlicher Eigenschaften, wie die Verwendung von SATA steigern die Funktionalität zusätzlich. Für dieses Projekt ist der Support des Wandboards durch deren Gemeinschaft, ein weiterer Grund diese Plattform zur Umsetzung der Aufgabenstellung und zur Zielerreichung zu wählen.

5.1.1.1 Wandboard vs. BeagleBone

Die nachfolgende Tabelle zeigt den technischen Unterschied der BeagleBone Black Rev. C [I-ExTe16] Plattform zur Wandboard Quad [I-WanB16] Plattform.

	Wandboard Quad	**BeagleBone Black Rev. C**
Prozessor	NXP i.MX6 Quad	AM3358
Kerne	ARM Cortex A9	ARM Cortex A8
Kerne / Takt	4x 1,0 GHz	1x 1,0 GHz
GPU	+ Vivante GC 2000 + Vivante GC 355 + Vivante GC 320	SGX530
RAM	2GB DDR3	512MB DDR3L
SATA	1	-kein-
SD-Karten Slot	2x Mikro	1x 4GB OnBoard 1x Mikro
Netzwerk	Gigabit	Megabit
WLAN	802.11n	-kein-
Bluetooth	4.0	-kein-
USB	1x 3.0	1x 2.0
USB OTG	1	-kein-
Video	HDMI	mikro HDMI
Audio	via HDMI	via HDMI
Fläche in mm	95 x 95	89 x 55
Benutzereingabe	Reset Button	Reset Button Boot Button Power Button
Händler	http://www.wandboard.org	http://www.exp-tech.de
Preis	112,00 €	62,99 €

Tabelle 5.1.1-1 Hardwarevergleich BeagleBone Black Rev. C [I-ExTe16]

Für das BeagleBone Black Rev. C ist das Android Betriebssystem Version 5.1 (Lollipop) verfügbar [S-GiHuB16].

Was bei dem Vergleich schnell klar wird, ist das schwache Leistungsniveau des BeagleBone Black Rev. C gegenüber dem Wandboard Quad. Durch den Einsatz eines Prozessors von geringer Leistungsfähigkeit, die Bereitstellung von zu wenig Arbeitsspeicher und das Fehlen der Netzwerkschnittstelle über WLAN, ist die Plattform zur Aufgabenumsetzung keinesfalls geeignet.

5.1.1.2 Wandboard vs. Odroid

Die abgebildete Tabelle zeigt den technischen Unterschied der Odroid XU4 [I-HaKe16] Plattform zur Wandboard Quad [I-WanB16] Plattform.

	Wandboard Quad	**Odroid XU4**
Prozessor	NXP i.MX6 Quad	Exynos5422
Kerne	ARM Cortex A9	ARM Cortex A15 ARM Cortex A7
Kerne / Takt	4x 1,0 GHz	4x 2,0 GHz 4x 1,3 GHz
GPU	+ Vivante GC 2000 + Vivante GC 355 + Vivante GC 320	Mali-T628
RAM	2GB DDR3	2GB LPDDR3
SATA	1	-kein-
SD-Karten Slot	2x Mikro	1x Mikro
Netzwerk	Gigabit	Gigabit
WLAN	802.11n	-kein-
Bluetooth	4.0	-kein-
USB	1x 3.0	2x 3.0 1x 2.0
USB OTG	1	-kein-
Video	HDMI	HDMI
Audio	via HDMI	via HDMI
Fläche in mm	95 x 95	82 x 58
Benutzereingabe	Reset Button	Power Button
Händler	http://www.wandboard.org	http://www.hardkernel.com
Preis	112,00 €	64,35 €

Tabelle 5.1.1-2 Hardwarevergleich Odroid XU4 [I-HaKe16]

Für das Odroid XU4 C ist das Android Betriebssystem Version 4.4 (Kitkat) verfügbar [S-Odro16].

Dieser Gegenüberstellung zeigt eine gute Alternative zum Wandboard Quad, wenn auf eine Netzwerkschnittstelle über WLAN verzichtet werden kann. Die Odroid XU4 Plattform ist ein sehr leistungsstarkes Modell, welches den Einsatz von Netzwerkkabeln zur Fernsteuerung unumgänglich macht, somit ist diese Plattform zur Aufgabenumsetzung bedingt geeignet.

5.1.1.3 Wandboard vs. Raspberry

Die Tabelle veranschaulicht den technischen Unterschied der Raspberry Pi 3 Model B [I-Rasp16] Plattform zur Wandboard Quad [I-WanB16] Plattform.

	Wandboard Quad	**Raspberry Pi 3 Model B**
Prozessor	NXP i.MX6 Quad	ARMv8-A
Kerne	ARM Cortex A9	ARM Cortex A8
Kerne / Takt	4x 1,0 GHz	4x 1,2 GHz
GPU	+ Vivante GC 2000 + Vivante GC 355 + Vivante GC 320	Broadcom Dual Core VideoCore IV
RAM	2GB DDR3	1GB DDR3
SATA	1	-kein-
SD-Karten Slot	2x Mikro	1x Mikro
Netzwerk	Gigabit	-vorhanden-
WLAN	802.11n	802.11n
Bluetooth	4.0	4.1
USB	1x 3.0	4x 2.0
USB OTG	1	-kein-
Video	HDMI	HDMI
Audio	via HDMI	via HDMI
Fläche in mm	95 x 95	93 x 64
Benutzereingabe	Reset Button	-kein-
Händler	http://www.wandboard.org	https://www.conrad.de
Preis	112,00 €	44,99 €

Tabelle 5.1.1-3 Hardwarevergleich Raspberry Pi 3 Model B [I-Rasp16]

Für das Raspberry Pi 2 C ist das Android Betriebssystem Version 6.0 (Marshmellow) verfügbar [S-MeFiB16]. Da das Raspberry Pi 3 dem Markt erst 3 Monate zur Verfügung steht, ist die Verwendung mit einem Android Betriebssystem derzeit nicht möglich.

Bei diesem Vergleich ist nicht nur die fehlende Netzwerkschnittstelle „Ethernet“ zu bemängeln, sondern auch die geringe Leistung des Arbeitsspeichers und somit ein Engpass für die Umsetzung einer schnelleren Analyse. Diese Plattform ist zur Umsetzung nicht geeignet.

5.1.1.4 Wandboard vs. Udoo

Die nachfolgende Tabelle zeigt den technischen Unterschied der Udoo Quad [I-Udoo16] Plattform zur Wandboard Quad [I-WanB16] Plattform.

	Wandboard Quad	**Udoo Quad**
Prozessor	NXP i.MX6 Quad	Freescale i.MX6 Quad
Kerne	ARM Cortex A9	ARM Cortex A9
Kerne / Takt	4x 1,0 GHz	4x 1,0 GHz
GPU	+ Vivante GC 2000 + Vivante GC 355 + Vivante GC 320	+ Vivante GC 2000 + Vivante GC 355 + Vivante GC 320
RAM	2GB DDR3	1GB DDR3
SATA	1	1
SD-Karten Slot	2x Mikro	1x Mikro
Netzwerk	Gigabit	Gigabit
WLAN	802.11n	-vorhanden-
Bluetooth	4.0	-
USB	1x 3.0	2x 2.0
USB OTG	1	2
Video	HDMI	HDMI
Audio	via HDMI	via HDMI
Fläche in mm	95 x 95	110 x 85
Benutzereingabe	Reset Button	Reset Button Power Button
Händler	http://www.wandboard.org	http://shop.udoo.org/
Preis	112,00 €	118,00 €

Tabelle 5.1.1-4 Hardwarevergleich Udoo Quad [I-Udoo16]

Für das Udoo Quad C ist das Android Betriebssystem Version 4.4 (Kitkat) verfügbar [S-Udoo16].

Bei diesem Vergleich ist der Funkstandard für die Udoo Quad Plattform leider unklar. Nichtsdestotrotz ist auch hier der geringe Arbeitsspeicher der Engpass für die Umsetzung einer schnellen Analyse, dadurch ist die Plattform zur Umsetzung nicht geeignet.

5.1.2 Hardwareplattform versus Softwareemulator

Da im Abschnitt 5.1.1 Vergleichen von Hardwareplattformen verschiedene Hardwareplattformen verglichen wurden, sollte auch ein Vergleich der Problemrealisierung auf Hardware und Software erfolgen. Nachfolgend wird für die Hardware und die Software jeweils eine Pro- und Kontraliste angelegt und abschließend bewertet.

5.1.2.1 Hardwareplattform

PRO	KONTRA
Einsatz angepasster Android Betriebssysteme oder selbst erstellter Betriebssysteme möglich	Konfigurations- oder Fernsteuerungssystem unter Umständen nötig
Anspruch an den Benutzer ist mäßig (bei einfacher Verwendung)	Plattform ist kostenpflichtig (einmalig)
Kontrolle über das Gerät wird durch Entwicklerwerkzeuge umsetzbar	Begrenzte Auswahl vorinstallierter Betriebssystem Speicherabbilder
Stromersparnis gegenüber Software Emulation auf einem Rechnersystem	Gehobener Anspruch an den Benutzer wird vorausgesetzt (bei erweiterter Funktionalität)
Prinzipiell kann jedes Gerät emuliert werden	Anwendungen sind an die Prozessor Architektur gebunden (ARM)
Weitgehend besteht keine Einschränkung beim Emulieren von Betriebssystemen	

Tabelle 5.1.2-1 Pro-Kontraliste zur Hardware

Beim Erstellen der Pro-Kontraliste wurde als Hauptreferenz das Wandboard gewählt. Da das Wandboard grundsätzlich, durch Einsetzen einer mit einem Betriebssystem beschriebenen Mikro SD-Karte und verbinden der nötigen Schnittstellen, einfach konfiguriert und bedient werden kann, ist die Handhabung sehr leicht. Der Anspruch an den Benutzer wächst mit der Aufgabenstellung an die Plattform. So muss der Benutzer tiefgehende Hintergrundinformationen haben, um ein eigenes Betriebssystem für diese Plattform zu erstellen. Bei der Funktionalität mit den GPIO Pins sollte der Benutzer im Umgang mit der logischen Programmierung geschult sein. Das Wandboard selbst ist durch Entwicklerwerkzeuge, bereitgestellt durch NXP [I-Nxp16], konfigurierbar und bietet die Möglichkeit verschiedensten Betriebssysteme lauffähig zu machen. Ferner kann es auch zur Emulation für verschiedenste Geräte verwendet werden.

5.1.2.2 Softwareemulator

PRO	KONTRA
Stehen überwiegend als freie und kostenlose Versionen zur Verfügung	Benötigt Hardware auf der es lauffähig ist
Weitgehend Plattformunabhängig (Linux, Microsoft Windows, etc.)	Läuft nicht in einer virtuellen Umgebung
Große Auswahl an Geräten für die Emulation	Bestimmte Gerätefunktionen nur gegen Entgelt (Pro-Version)
Einfache Handhabung bei Installation und Umgang	Android Versionen sind vorkonfiguriert
Emuliert viele Android Betriebssysteme	Gerätefunktion durch Betreiber limitiert
Anspruch an den Benutzer ist gering	

Tabelle 5.1.2-2 Pro-Kontraliste zur Software

Beim Erstellen der Pro-Kontraliste wurde als Hauptreferenz Genymotion [I-Geny16] gewählt. Die meisten dieser Emulatoren sind als kostenlose Versionen verfügbar und bieten eine große Auswahl an Funktionen, jedoch muss für spezielle Funktionen auf eine kostenpflichtige Version umgestellt werden. Da Software zur Ausführung immer eine Hardwareplattform voraussetzt, ist auch die resultierende Leistung der Software abhängig von der zur Verfügung gestellten Hardwareplattform. Weiter ist eine freie Konfiguration der Betriebssystemabbilder oder der verwendbaren Geräte, auf eine vom Anbieter vorgegebene Liste begrenzt. Der leichte Umgang mit einer solchen Software ist auf Erfahrungen des jeweiligen Anbieters am Markt zurückzuführen.

5.1.2.3 Auswertung des Vergleichs

Es gibt viele Perspektiven ein Android Betriebssystem virtuell zur Verfügung zu stellen. Viele dieser Möglichkeiten wurden schon im größeren Rahmen genutzt. Für dieses Projekt soll jedoch ein herstellerunabhängiges Mittel geschaffen werden, jedes gängige Android Betriebssystem auf der Hardwareplattform lauffähig zu machen. Hierbei wird aus einer unlimitierten Liste von Geräten dieses Betriebssystem emuliert. Nun sind aber die Softwarelösungen von den jeweiligen Anbietern abhängig und demgemäß fällt die Wahl auf eine Hardwarelösung. Die Abhängigkeit zum Hardwarehersteller ist weit wenige relevant als die zum Softwarehersteller.

5.1.3 Android Debug Bridge versus Play Store Anwendungen

Am Markt existieren zahlreiche Anwendungen um die Verbindung des Smartphones via USB, Bluetooth oder WLAN zu realisieren. Dabei ist es wichtig zu unterscheiden, wozu diese Verbindung Verwendung finden soll.

Realisiert eine Anwendung wie MyPhoneExplorer [S-MyPh16] oder AirDroid [S-AiDr16] beziehungsweise TeamViewer [S-TeVi16] die Kommunikation von einem Rechnersystem zum Android Betriebssystem, zur Fernsteuerung oder zum Datenaustausch, so ist die Installation der Android Debug Bridge [S-MyPh16] oder die Installation einer zusätzlichen Anwendung auf dem Android Betriebssystem [S-AiDr16, S-TeVi16] eine grundsätzliche Bedingung zum korrekten Betrieb solch einer Software. Bei dem Versuch auf eine Installation von zusätzlicher Software auf dem Rechnersystem komplett zu verzichten, um mit Bordmitteln des Betriebssystems eine Verbindung zu realisieren, ist der Einsatz weiterer Software auf dem Android Betriebssystem notwendig. Diese Software realisiert dann die netzwerktechnische Portfreigabe, um sich mittels Fernterminal auf dem System anzumelden oder die Freigabe eines Netzwerkordners für den Datenaustausch. Bei diesem Ansatz ist jedoch zu berücksichtigen, dass die Anwendung Ressourcen des Android Betriebssystems blockiert, die für eine schnellere Analyse besser genutzt werden könnten. Auch die Unterbrechung dieser Anwendung durch ein zur Ausführung kommendes Schadprogramm, wäre diesbezüglich möglich.

Somit ist von diesem Vorgang abzuraten und die Methode über den adbd Dienst zu realisieren, der in jedem Android Betriebssystem zur Ausführung kommt. Gleichzeitig wird durch die Verwendung von minimal ADB die beste Lösung dargeboten. Im Rahmen einer automatisierten Konfiguration, ist die Verbindung zur Hardwareplattform über die Android Debug Bridge das beste Weiterentwicklungspotenzial zu zuordnen.

5.1.4 Einordnung in die aktuelle Forschung

Die Idee Android Anwendungen auf Schadsoftware zu analysieren ist nicht neu, dies jedoch auf der Hardwareplattform Wandboard durchzuführen schon. Diese Inspiration, die nicht nur dem Zeitgeist geschuldet ist, soll sich von der breiten Masse der schon vorhandenen Analysemethoden stark abgrenzen.

Eine dieser Methoden ist die Analyse von Android Anwendung auf Schadsoftware durch Analyse der Manifest-Datei, dabei ist das Verständnis der Android Berechtigungen und die Untersuchung auf bösartigen Eigenschaften von großer Bedeutung [T-DuGiVi15]. Eine weitere Methode ist eine durch maschinelles Lernen automatisierte Klassifizierung der Anwendung im laufenden Betrieb, die Android Schadsoftware während der Laufzeit analysiert und in Schadsoftware Familien unterscheidet [T-Dash16]. Um Ähnlichkeiten in der Semantik zu finden, ist der Vergleich eines aktiven Schadprogramms zur bereits bekannten eine weitere Methode [T-Alam16]. Die letzte hier aufgeführte Methode ist die Nachkonstruktion des Android Schadprogramms mit einem separaten Programm durch Untersuchung der Strukturen, Zustände und Verhaltensweisen [T-Rast16].

Die ersten zwei Methoden [T-DuGiVi15, T-Dash16] lassen das Schadprogramm im Emulator laufen, was zur Folge haben kann, dass das Schadprogramm dies bemerkt. Dadurch könnte das Schadprogramm ein programmiertes Ausweichverhalten ausführen oder bei einer modifizierten Anwendung, durch einbetten der Schadroutinen in einer vertrauenswürdigen Anwendung, auf das eigentliche Verhalten dieser Anwendung zurückgreifen. Schlussfolgernd ist ein falsches Ergebnis der Analyse zu erwarten. Der Schadprogramm Vergleich der Methode [T-Alam16] setzt voraus, dass erstens das Schadprogramm bekannt ist und zweitens ein gewisser Schwellenwert der semantischen Gleichheit erreicht werden muss. Die letzte Methode [T-Rast16] betrachtet nur den Aspekt der Analyse vom Quellcode, der bei der Codeinspektion entstanden ist. Das Schadprogramm kann Daten oder Programmcodes nachladen, ist dies nicht in der Analyse integriert so wird das Ergebnis ebenso verfälscht werden. Anzumerken ist demnach, dass die benannten Arbeiten einen ganz anderen Fokus in Bezug auf die Analysemethode haben.

5.1.5 Vergleichbare Arbeiten und alternative Ansätze

Die Ansätze der Analysemethoden gehen von statischen Untersuchungen bis hin zu dynamischen Untersuchungen. Dabei wird nicht nur das Verhalten einer Anwendung analysiert, sondern auch getätigte Systemaufrufe, nachgeladene Bibliotheken und das Verhalten der zu untersuchenden Anwendung bei Benutzerinteraktionen mit anderen Anwendungen [T-TcSa13] überwacht. Wenn die Verwendung eines Emulators vernachlässigt wird, kommt der Beitrag [T-Dash16] dieser Arbeit sehr nahe.

Das Android Schadprogramm wird im laufenden Betrieb durch Überwachung des Verhaltens analysiert. Dies geschieht durch Beobachtung der getätigten Systemaufrufe, der internen Prozesskommunikation (IPC), der geöffneten Dateien, der verwendeten Netzwerk Ports und genutzter Ressourcen die vom Android Betriebssystem zur Verfügung gestellt wurden. Der Unterschied zu dieser Arbeit liegt darin, dass nicht Schadsoftwareklassen eingeteilt werden, sondern lediglich auf potentielle Schadsoftware hingewiesen werden soll und kein Softwareemulator zum Einsatz kommt. Weiter verfolgt die Arbeit [T-Dash16] das Ziel, das Schadprogramme im maschinellen Lernverfahren automatisch klassifiziert werden.

Alternativ besteht natürlich die Möglichkeit, eine zu analysierende Anwendung während des automatischen Analyseprozesses parallel oder als vorbereitende Maßnahme, bei einem oder mehreren Online Anbietern prüfen zu lassen und die Resultate in das Ergebnis mit einfließen zu lassen. Da die in dieser Arbeit angedachte Analysemethode nicht den Ansatz verfolgt, eine Signaturdatenbank für eine statische Analyse bereit zu stellen, ist diese Alternative auch eine starke Abgrenzung zur gewünschten Analysemethode.

5.2 Fazit

Die Hardwareplattform Wandboard ist für eine Analyse von Android Schadprogramme, durch den externen Zugriff auf das System und alle seine Komponenten über die verschiedenen Schnittstellen, sehr gut geeignet. Durch eine automatisierte dynamische Analyse von Android Anwendungen, ist die Untersuchung komplexer Schadsoftware möglich und Ergebnisse können schnell und einfach zur Verfügung gestellt werden.

Die Konfiguration der Hardwareplattform Wandboard stellt mit dem Hintergrundwissen über das Betriebssystem Android, keine unüberwindbare Hürde dar. Der Zugriff auf das System ist durch technisch einfache Mittel gesichert, die vom Entwickler des Android Betriebssystem zur Verfügung gestellt werden. Diese Zugriffe sind für die Fernsteuerung des Systems und des Datenaustausches mit diesem von primärer Bedeutung. Durch diese Mittel ist eine vollautomatische Konfiguration der Hardwareplattform und Analyse von Anwendungen, im laufenden Betrieb möglich.

Bei der Leitfadenerarbeitung zur Konfiguration der Hardwareplattform Wandboard, wird allen Beteiligten die Chance eingeräumt, diese ohne großartigen Aufwand einzurichten. Gleichzeitig sollte ein tiefgehendes Verständnis für die Plattform und das zu betreibende Betriebssystem entwickelt werden.

5.3 Ausblick

Die in dieser Bachelorarbeit entstandenen Durchführungsmethoden sollen künftigen Benutzern des Wandboards einen leichteren Einstieg für diese Hardwareplattform geben. Im Laufe dieser Arbeit kristallisierten sich weitere Punkte heraus, die in Form der Weiterentwicklung des Projektes umgesetzt werden können.

Das Android Betriebssystem könnte so modifiziert werden, dass es möglich ist aus einer Liste bekannter Geräte, eins zu wählen und in diesem Gerätemodus zu starten.

Weiter ist auch wünschenswert das Android Betriebssystem so zu modifizieren, dass der Start von Skripten und eigens erstellten Systemdiensten während beziehungsweise nach dem Bootvorgang, möglich ist. Denkbar wäre auch ein gesondert für diese Plattform angefertigtes Android Betriebssystem zu erstellen und auf dem Wandboard betriebsbereit zu machen.

Die wichtigste Weiterentwicklung in Bezug auf diese Bachelorarbeit ist die Änderung der ersten explorativen Analyse zu einer vollwertigen automatischen Schadprogramm Analyse. Mittels Verhaltenserkennung, Musteranalyse und Auswertung der Protokolldateien können die Ergebnisse an eine Weboberfläche für einen Benutzer präsentiert werden.

Index

Glossar

Anwenderprogramm	Programm, das dem Benutzer ermöglicht spezielle Aufgaben durchzuführen
API	Schnittstelle die dem Entwickler erlaubt Anwendungen zu schreiben und dabei bereits vorhandene, standardisierte Bibliotheksroutinen zu nutzen
App	Kurzform von Applikation: Anwendung (Anwenderprogramm)
Assembler	Programm, das eine maschinenorientierte Programmiersprache in die spezielle Maschinensprache umsetzt
booten	Neustarten eines Betriebssystems, wobei alle gespeicherten Anwenderprogramme neu geladen werden
Botnetz	Gruppe von automatisierten Computerprogrammen
Broadcast	Nachricht, die vom System aus an alle Teilnehmer des Nachrichtennetzes übertragen wird
builden	Prozess in dem Quellcode zu Objektcode konvertiert wird
Bytecode	Generierter Code der zwischen höheren Ausgangssprache und der Maschinennahen Zielsprache liegt
Cloud	IT-Infrastrukturen über ein Netz zur Verfügung stellen
Code	Maschinencode
Command-and-Control-Server	Botnetz-Operator in einem Botnetz
Compiler	Programm, das eine Programmiersprache in die Programmiersprache eines bestimmten Computers übersetzt
Cross-Compiler	Compiler, der Kompilate für andere Systeme erzeugt
Debian	Betriebssystem
Debugger	Programm, das Fehler in der Programmierung sucht und ausschaltet
debuggen	Einen Programmfehler in einem Softwareprogramm beseitigen

Distribution	Zusammenstellung von Software, die als Komplettpaket weitergegeben wird
Domain	Zusammenhängender Teilbereich des hierarchischen Domain Name System
Embedded	Ausdruck für eingebettetes System
Emulator	System, das ein anderes System nachbildet
Entschlüsseln	Wiederherstellung des Klartextes mit Hilfe des verwendeten Schlüssels
Exploit	Ausnutzung einer Schwachstelle, die bei der Entwicklung eines Programms nicht berücksichtigt wurde
Flash-Speicher	Nichtflüchtiger digitaler Speicherbaustein
flashen	ROM-gespeicherte Software überschreiben
Frontend	Benutzeroberfläche
Hybrid	Gebilde aus zwei oder mehreren Komponenten
Hype	Euphorische Begeisterung für ein Produkt
implementieren	In bestehendes Computersystem einsetzen, einbauen und so ein funktionsfähiges Programm erstellen
Infektion	Durch einen Computervirus verursachter Kontakt
initialisieren	Aktion, durch die Computer, Programme oder Hardware betriebsbereit gemacht wird
Instant Messaging Programm	Nachrichtensofortversand durch ein Kommunikationsprogramm
Interpreter	Computerprogramm, das einen Programm-Quellcode einliest, analysiert und ausführt
Interrupt-Controller	Integrierter Schaltkreis zur Verwaltung mehrerer Interrupts
Interrupt	Vorübergehende Unterbrechung des laufenden Programms
Kompilat	Objektdateien oder ausführbare Programme
kompilieren	Eine höhere Programmiersprache in die Maschinensprache eines bestimmten Computers übersetzen

Layout Editor	Gestaltungsprogramm zum Ändern der Oberflächenelemente
Link	Verknüpfung mit einer anderen Datei oder einer anderen Stelle in der selben Datei
Loader	Dienst eines Betriebssystems, lädt ein ausführbares Programm in den Arbeitsspeicher und führt dieses aus
Malware	Schadprogramm, das entwickelt wurde, um unerwünschte oder schädliche Funktionen auszuführen
Man-in-the-Middle	Mittelsmann-Angriff; Angriffsform in Rechnernetzen
Maschinencode	Interner Befehlsfolge die von der Maschine unmittelbar ausgeführt werden
Metadaten	Daten, die anderen Daten übergeordnet sind
Obfuscating	Absichtliche Veränderung von Programmcode, so dass der Quelltext für Menschen schwer verständlich ist
Open-Source	Software deren Quelltext öffentlich eingesehen werden kann
Patch	Korrekturauslieferung für Software
performant	in der Lage, beträchtliche Leistungen zu erbringen
Play Store	Android Markt von Google
Port	Adress-Komponente in Netzwerkprotokollen
Quellcode	Abfolge von Programmanweisungen in einer Programmiersprache
RAW	Rohdatenformat
rebuilden	Erneutes Erstellen
Recovery	Datenwiederherstellung; Sicherung
Repository	Verwaltetes Verzeichnis zur Speicherung und Beschreibung von digitalen Objekten
root	Benutzerkonto, das bei der Installation eines Betriebssystems angelegt wird und mit den höchsten Zugriffsrechten ausgestattet ist
rooten	Das Einrichten der Nutzung des Superuser-Kontos für Anwendungen auf einem mobilen Gerät

Router	Vermittlungsvorrichtung in einem Kommunikationsverbund, die Daten zwischen räumlich getrennten Netzwerken transportiert
Skript	Programme die für kleinere Anwendungen gedacht sind und von einem Interpreter ausgeführt werden
Shell	Kommandozeileninterpreter
Social Engineering	Zwischenmenschliche Beeinflussungen mit dem Ziel vertrauliche Informationen zu erhalten
Tool	Werkzeug
Touchscreen	Ein- und Ausgabegerät mit Steuerung über Berührung
Treiber	Gerätetreiber; Programm, mit dem ein peripheres Gerät gesteuert wird
Trojaner	Trojanisches Pferd; Computerprogramm, das als nützliche Anwendung getarnt ist und im Hintergrund ohne Wissen des Anwenders andere Funktionen erfüllt
Tutorial	Gebrauchsanleitung in schriftlicher oder filmischer Form
Update	Aktualisierte Version einer Software oder einer Datei
Verschlüsseln	Das unkenntlich machen eines Klartextes, mithilfe eines Schlüssels
Virus	Computervirus; Computerprogramm, das vorhandene Software manipulieren oder zerstören kann
WiFi	Synonym für WLAN
Zertifikat	Verifikation einer Verschlüsselung
ZIP	Format für verlustfrei komprimierte Dateien

Quellenverzeichnis

Verzeichnis der verwendeten Literatur

[L-AsBa02] Asteroth, Alexander; Baier, Christel: Theoretische Informatik
Eine Einführung in Berechenbarkeit, Komplexität und formale Sprachen mit 101 Beispielen,
1. Auflage, München, Pearson Studium, 2002

[L-Ayco06] Aycock, John: Computer Viruses and Malware
1. Auflage, University of Calgary, AB, Canada, Springer, 2006

[L-BePa10] Becker, Arno; Pant, Marcus: Android
Grundlagen und Programmierung,
1. Auflage, Heidelberg, dpunkt.verlag GmbH, 2009

[L-Ecke13] Eckert, Claudia: IT-Sicherheit
Konzepte – Verfahren – Protokolle,
8. Auflage, München, Oldenbourg Verlag, 2013

[L-Hero99] Herold, Helmut,: Linux-Unix-Shells
Bourne-Shell, C-Shell, bash, tcsh
3. Auflage, München, Addison Wesley Longman Verlag, 1999

[L-Hoog11] Hoog, Andrew: Android Forensics:
Investigation, Analysis and Mobile Security for Google Android,
1. Auflage, Massachusetts USA, Syngress, 2011

[L-Park11] Steve Parker: Shell Scripting
Expert Recipes for Linux Bash and More
1. Auflage, Indianapolis USA, John Wiley & Sons, 2011

[L-PlWe11] Plötner, Johannes; Wendzel, Steffen: Linux
Das umfassende Handbuch
4.Auflage, Galileo Press, 2011

[L-Yagh13] Yaghmour, Karim: Embedded Android
1.Auflage, Sebastopol, O'Reilly Media, 2013

Verzeichnis der verwendeten Internetquellen

[I-AnDeA16] Android Developers: Building and Running Overview
URL: https://developer.android.com/tools/building/index.html,
zuletzt besucht am 26. April 2016

[I-AnDeB16] Android Developers: Build System Overview
URL: https://developer.android.com/sdk/installing/studio-build.html,
zuletzt besucht am 26. April 2016

[I-AnDeC16] Android Developers: System Permissions
URL: https://developer.android.com/guide/topics/security/permissions.html,
zuletzt besucht am 26. April 2016

[I-AnDeD16] Android Developers: Android Studio Overview
URL: https://developer.android.com/tools/studio/index.html,
zuletzt besucht am 27. April 2016

[I-AnDeE16] Android Developers: Android NDK
URL: https://developer.android.com/tools/sdk/ndk/index.html,
zuletzt besucht am 27. April 2016

[I-AnDeF16] Android Developers: Android Debug Bridge
URL: https://developer.android.com/tools/help/adb.html,
zuletzt besucht am 27. April 2016

[I-AnDeG16] Android Developers: Install Android Studio
URL: https://developer.android.com/sdk/installing/index.html,
zuletzt besucht am 02. Mai 2016

[I-AnDeH16] Android Developers: ADB Shell Commands
URL: https://developer.android.com/studio/command-line/shell.html,
zuletzt besucht am 15. Mai 2016

[I-AnDeI16] Android Developers: Manifest.permission
URL: https://developer.android.com/reference/android/Manifest.permission.html,
zuletzt besucht am 28. Mai 2016

[I-AnDeJ16] Android Developers: <permission>
URL: https://developer.android.com/guide/topics/manifest/permission-element.html,
zuletzt besucht am 26. April 2015

[I-AnDeZ15] Android Developers: Dashboards
URL: https://developer.android.com/about/dashboards/index.html#Platform,
zuletzt besucht am 31. Dezember 2015

[I-AnSo16] Android Source: Licenses Android Open Source Project
URL: https://source.android.com/source/licenses.html,
zuletzt besucht am 24. April 2016

[I-Busy16] BusyBox: BusyBox: The Swiss Army Knife of Embedded Linux
URL: https://busybox.net/about.html,
zuletzt besucht am 13 Mai 2016

[I-Digi16] Digital News Asia: The Evolution Of Mobile Malware: 2004-2014
URL: https://www.digitalnewsasia.com/sites/default/files/images/digital%20economy/Fortinet%20The%20Evolution%20of%20Mobile%20Malware%202004-2014.jpg,
zuletzt besucht am 28. April 2016

[I-Drop16] Dropbear: Dropbear SSH
URL: https://matt.ucc.asn.au/dropbear/dropbear.html,
zuletzt besucht am 20. Mai 2016

[I-DrWi16] Droid Wiki: Partitionen
URL: https://www.droidwiki.de/Partitionen,
erstellt am 26. Dezember 2015,
zuletzt besucht am 25.April 2016

[I-Dude16] Duden: Smartphone, Smart Phone, das.
URL: http://www.duden.de/rechtschreibung/Smartphone,
zuletzt besucht am 21. April 2016

[I-ExTe16] EXP Tech: BeagleBone Black - Rev C
URL: http://www.exp-tech.de/beaglebone-black-rev-c,
zuletzt besucht am 02. Mai 2016

[I-Gabl16] Gabler: Gabler Wirtschaftslexikon „Smartphone“
URL: http://wirtschaftslexikon.gabler.de/Definition/smartphone.html#definition,
zuletzt besucht am 21. April 2016

[I-Gart15] Gartner: Gartner Says Smartphone Sales Surpassed One Billion Units in 2014
URL: https://www.gartner.com/newsroom/id/2996817,
erstellt am März 2015,
zuletzt besucht am 20. April 2016

[I-Geny16] Genymotion: Homepage
URL: https://www.genymotion.com/, zuletzt besucht am 10. Mai 2016

[I-HaKe16] Odroid: ODROID Platforms
URL: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825,
zuletzt besucht am 02. Mai 2016

[I-LoLe16] Lowlevel: Kernel
URL: https://www.lowlevel.eu/wiki/Kernel,
erstellt am 11. Juni 2011,
zuletzt besucht am 24. April 2016

[I-MicrA16] Microsoft: So wird's gemacht: Neuinstallation von Windows
URL: http://windows.microsoft.com/de-de/windows-8/clean-install,
zuletzt besucht am 02. Mai 2016

[I-MicrB16] Microsoft: Installieren von Programmen
URL: http://windows.microsoft.com/de-de/windows/install-program#1TC=windows-7,
zuletzt besucht am 02. Mai 2016

[I-MicrC16] Microsoft: Zum ersten Mal online
URL: http://windows.microsoft.com/de-de/windows-8/get-online-tutorial,
zuletzt besucht am 02. Mai 2016

[I-Nxp16] NXP: IMXANDROID: Android OS for i.MX Applications Processors
URL: http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/i.mx-applications-processors/i.mx53-processors/android-os-for-i.mx-applications-processors:IMXANDROID, zuletzt besucht am 18. März 2016

[I-Orac16] Oracle: Oracle VM VirtualBox User Manual
URL: http://download.virtualbox.org/virtualbox/5.0.14/UserManual.pdf, zuletzt besucht am 02. Mai 2016

[I-Rasp16] Raspberrypi: Raspberry Pi 3 Model B
URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, zuletzt besucht am 02. Mai 2016

[I-Sdc16] SD Association Whitepapers: Video Speed Class: The new capture protocol of SD 5.0
URL: https://www.sdcard.org/downloads/pls/latest_whitepapers/Video_Speed_Class-The_new_capture_protocol_of_SD_5.0.pdf, zuletzt besucht am 22. April 2016

[I-Sec16] Securlist: Mobile cyber-threats: a joint study by Kaspersky Lab and INTERPOL
URL: https://securelist.com/analysis/publications/66978/mobile-cyber-threats-a-joint-study-by-kaspersky-lab-and-interpol/, erstellt am 6. Oktober 2014, zuletzt besucht am 28. April 2016

[I-Tcp16] Tcpiputils.com: IPv4 Address Space report
URL: http://www.tcpiputils.com/browse/ip-address, zuletzt besucht am 22. Mai 2016

[I-Udoo16] Udoo: UDOO QUAD
URL: http://shop.udoo.org/usa/home/udoo-quad.html, zuletzt besucht am 02. Mai 2016

[I-ViToA16] VirusTotal: MD5: eed5195197143f98acea71a514e1539e
URL: https://www.virustotal.com/de/file/88de76be3de5b25b5b2cd9562549cdfd682f541d77ce0f8cf50cd97f46cfb248/analysis/1463945945, zuletzt besucht am 22. Mai 2016

[I-ViToB16] VirusTotal: MD5: 10205ec68e10b7c066405b0367e87169
URL: https://www.virustotal.com/de/file/1a05b6352d4305d7d1ded0e052b039c179866350085715b6881d7b6484cc116f/analysis/1463953164,
zuletzt besucht am 22. Mai 2016

[I-WanA16] Wandboard: Frequent Asked Questions (FAQ)
URL: http://www.wandboard.org/index.php/faq,
zuletzt besucht am 21. April 2016

[I-WanB16] Wandboard: Wandboard i.MX6 Specifications
URL: http://www.wandboard.org/index.php/details,
zuletzt besucht am 22. April 2016

[I-WanC16] Wandboard: WANDBOARD USER GUIDE Revision B1
URL: http://www.wandboard.org/images/downloads/wbquad-revb1-userguide.pdf,
erstellt am 20. Juni 2013,
zuletzt besucht am 22. April 2016

[I-WanD16] Wandboard: Download
URL: http://www.wandboard.org/index.php/downloads,
zuletzt besucht am 22. April 2016

[I-Wire16] Wireshark.org: Wireshark User's Guide
URL: https://www.wireshark.org/download/docs/user-guide-a4.pdf,
zuletzt besucht am 02. Mai 2016

Verzeichnis der verwendeten Texte

[T-Alam16] Alam, S., Riley, R., Sogukpinar, I., & Carkaci, N. DroidClone: Detecting Android Malware Variants by Exposing Code Clones,
URL: http://web.uvic.ca/~salam/papers/DroidClone-detecting-Alam.pdf,
erstellt am 16. Mai 2016,
zuletzt besucht am 25. Mai 2016

[T-Dash16] Dash, S. K., Suarez-Tangil, G., Khan, S., Tam, K., Ahmadi, M., Kinder, J., & Cavallaro, L. DroidScribe: Classifying Android Malware Based on Runtime Behavior
URL: http://www.cs.rhul.ac.uk/home/kinder/papers/most16-droidscribe.pdf,
erstellt am 23. April 2016,
zuletzt besucht am 25. Mai 2016

[T-DuGiVi15] Duc, Nguyen Viet; Giang, Pham Thanh; Vi, Pham Minh:
Permissione Analysis for Android Malware Detection
URL: https://www.researchgate.net/profile/Pham_Giang4/publication/296704790_Permission_Analysis_for_Android_Malware_Detection/links/56d9bce708aee1aa5f8291f4.pdf,
erstellt am November 2015,
zuletzt besucht am 25. Mai 2016

[T-Rast16] Rasthofer, S., Arzt, S., Miltenberger, M., & Bodden, E.
Reverse Engineering Android Apps With CodeInspect.
URL: https://www.semanticscholar.org/paper/Reverse-Engineering-Android-Apps-with-Codeinspect-Rasthofer-Arzt/3b2f3740c3d9c2d05e99eddef2569d5b50e298fb/pdf,
erstellt am 06. April 2016,
zuletzt besucht am 25. Mai 2016

[T-TcSa13] Tchakounté, F., & Dayang, P. :
System calls analysis of malwares on android,
URL: https://www.researchgate.net/profile/Franklin_Tchakounte/publication/257934801_System_Calls_Analysis_of_Malwares_on_Android/links/02e7e52652b89aca95000000.pdf,
International Journal of Science and Tecnology (IJST), 2. Volume, 2013,
zuletzt besucht am 26. Mai 2016

Verzeichnis der verwendeten Software

[S-AiDr16] AirDroid: AirDroid 3
URL: https://srv3.airdroid.com/p14/web/getbinaryredirect?type=exe&channel=,
zuletzt besucht am 13. Mai 2016,
zuletzt heruntergeladen am 13. Mai 2016

[S-AnFiA16] AndroidFilehost: GoogleApps 5.0.x 20150404 Minimal
URL: http://qc1.androidfilehost.com/dl/7UH_kwWHPuHhg3ICo0sW6g/1464663704/9591617793455085O/gapps-5.0.x-20150404-minimal-edition-signed.zip,
zuletzt besucht am 07. Mai 2016,
zuletzt heruntergeladen am 07. Mai 2016

[S-AnFiB16] AndroidFilehost: GoogleApps 4.4.x 20150412 Minimal
URL: http://qc1.androidfilehost.com/dl/S8yAti5eo_hYeRz3zL0ibg/1464663790/9591617793455440 6/gapps-kk-20150412-minimal-edition-signed.zip,
zuletzt besucht am 07. Mai 2016,
zuletzt heruntergeladen am 07. Mai 2016

[S-AnFiC16] AndroidFilehost: GoogleApps 6.0.x 20150514 Minimal
URL: https://qc3.androidfilehost.com/dl/TGhU0py8JJqyt8w0MkRVQw/1465188474/24533103863139560/A-GAPPS+6.0.1_14.05.2016_v5.2.zip
zuletzt besucht am 03. Juni 2016,
zuletzt heruntergeladen 03. Juni 2016

[S-GiHuA16] GitHub: Wandboard – Android 4.4.2 considerations
URL: https://github.com/tkurbad/wandboard/archive/master.zip,
zuletzt besucht am 12. Mai 2016,
zuletzt heruntergeladen am 12. Mai 2016

[S-GiHuB16] GitHub: How to build Android Lollipop 5.1 for BeagleBone Black
URL: https://github.com/csimmonds/bbb-android-device-files/archive/lollipop-5.1.zip,
zuletzt besucht am 25. Mai 2016,
zuletzt heruntergeladen am 25. Mai 2016

[S-MeFiA16] MediaFire: GoogleApps 4.3
URL: http://download2028.mediafire.com/4tu1vh1fyhng/6lzbp10u7kc3ol8/gapps-jb-4.3-20131214-signed.zip,
zuletzt besucht am 07. Mai 2016,
zuletzt heruntergeladen am 07. Mai 2016

[S-MeFiB16] MediaFire: Marshmallow for Raspberry Pi
URL: http://download2139.mediafire.com/sw926fm11fzg/a08655r4nb4jdl2/marshpi2.img.tar.bz2/,
zuletzt besucht am 25. Mai 2016,
zuletzt heruntergeladen am 25. Mai 2016

[S-Kali16] Kali: Kali Linux 32 bit
URL: http://cdimage.kali.org/kali-2016.1/kali-linux-2016.1-i386.iso,
zuletzt besucht am 19. Februar 2016,
zuletzt heruntergeladen am 19. Februar 2016

[S-MyPh16] MyPhoneExplorer: MyPhoneExplorer 1.8.7
URL: http://www.fjsoft.at/download.php?id=1,
zuletzt besucht am 13. Mai 2016,
zuletzt heruntergeladen am 13. Mai 2016

[S-Odro16] Odroid: ODROID-XU3/XU3-Lite Android 4.4.4 OS Alpha 1.5
URL: http://dn.odroid.com/5422/ODROID-XU3/Android/4.4.4_Alpha_1.5_Jan-02-2015/,
zuletzt besucht am 25. Mai 2016,
zuletzt heruntergeladen am 25. Mai 2016

[S-SoFo16] SourceForge: Win32 Disk Imager (2015-09-22)
URL: https://sourceforge.net/projects/win32diskimager/files/latest/download,
zuletzt besucht am 05. November 2015,
zuletzt heruntergeladen am 05. November 2015

[S-Szip16] 7-Zip: Download 7-Zip 16.02 (2016-05-21) for Windows: x64
URL: http://www.7-zip.org/a/7z1602-x64.exe,
zuletzt besucht am 05. November 2015,
zuletzt heruntergeladen am 05. November 2015

[S-TeVi16] Team Viewer: TeamViewer Portable v11.0.59518
URL: http://download.teamviewer.com/download/TeamViewerPortable.zip,
zuletzt besucht am 13. Mai 2016,
zuletzt heruntergeladen am 13. Mai 2016

[S-Udoo16] Udoo: Android 4.4.2
URL: http://download.udoo.org/files/UDOO_Unico/Quad_img/Android_img/udoo_quad_android_v4.4.2-1.0.zip/,
zuletzt besucht am 25. Mai 2016,
zuletzt heruntergeladen am 25. Mai 2016

[S-WanA16] Wandboard: WB All: Android 5.0.2 GA Release (with AP mode)
URL: http://www.wandboard.org/images/downloads/wandboard-lp-5.0.2-ga-20151105-sdcard.zip,
zuletzt besucht am 22. April 2016,
zuletzt heruntergeladen am 05. November 2015

[S-WanB16] Wandboard: WB All: Android 4.4
URL: http://www.wandboard.org/images/downloads/android-4.4.2-wandboard-20150303.zip,
zuletzt besucht am 22. April 2016,
zuletzt heruntergeladen am 09. Februar 2016

[S-WanC16] Wandboard: WB All: Android 4.3
URL: http://www.wandboard.org/images/downloads/android-4.3-wandboard-20150303.zip,
zuletzt besucht am 22. April 2016,
zuletzt heruntergeladen am 09. Februar 2016

[S-WanD16] Wandboard: Android 6.0.1
URL: http://download.wandboard.org/wandboard-imx6/android-6.0/wandboard-all-android-6.0.1-sdcard-20160428.zip,
zuletzt besucht am 03. Juni 2016,
zuletzt heruntergeladen am 03. Juni 2016

[S-XdaF16] XDA Forum: [TOOL]Minimal ADB and Fastboot [4-27-16]
URL: https://www.androidfilehost.com/?fid=24521665358595410,
zuletzt besucht am 27. April 2016,
zuletzt heruntergeladen am 25. Mai 2016

Anlagen

Anlage, Teil A

Der vorliegende Code ist die Ausgabe der Linux-Shell (Terminal), nach Verwendung einer Befehlsfolge. Dies wurde realisiert über eine Doppel Pipeline zur Umsetzung der Aufgabenstellung des Abschnitts 3.2.1.1 Speicherabbild auf die SD-Karte schreiben.

```
Checking that no-one is using this disk right now ... OK

Disk /dev/loop0: 7,6 GiB, 8133484032 bytes, 15885711 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000
```

```
Old situation:

Device       Boot   Start      End  Sectors   Size Id Type
/dev/loop0p1        15438    46313    30876  15,1M 83 Linux
/dev/loop0p2        46314    77189    30876  15,1M 83 Linux
/dev/loop0p3        77190  2300261  2223072   1,1G  5 Extended
/dev/loop0p4      2300262 15885701 13585440   6,5G 83 Linux
/dev/loop0p5        77191  1126973  1049783 512,6M 83 Linux
/dev/loop0p6      1126975  2176757  1049783 512,6M 83 Linux
/dev/loop0p7      2176759  2207633    30875  15,1M 83 Linux
/dev/loop0p8      2207635  2223071    15437   7,6M 83 Linux
/dev/loop0p9      2223073  2238509    15437   7,6M 83 Linux

Partition table entries are not in disk order.

>>> Script header accepted.
>>> Script header accepted.
>>> Script header accepted.
>>> Script header accepted.
>>> Created a new DOS disklabel with disk identifier 0x00000000.
Created a new partition 1 of type 'Linux' and of size 15,1 MiB.
/dev/loop0p2: Created a new partition 2 of type 'Linux' and of size 15,1 MiB.
/dev/loop0p3: Created a new partition 3 of type 'Extended' and of size 1,1 GiB.
/dev/loop0p4: Created a new partition 4 of type 'Linux' and of size 5 GiB.
/dev/loop0p5: Created a new partition 5 of type 'Linux' and of size 512,6 MiB.
/dev/loop0p6: Created a new partition 6 of type 'Linux' and of size 512,6 MiB.
/dev/loop0p7: Created a new partition 7 of type 'Linux' and of size 15,1 MiB.
/dev/loop0p8: Created a new partition 8 of type 'Linux' and of size 7,6 MiB.
/dev/loop0p9: Created a new partition 9 of type 'Linux' and of size 7,6 MiB.
/dev/loop0p10:
```

```
New situation:

Device       Boot   Start      End  Sectors   Size Id Type
/dev/loop0p1        15438    46313    30876  15,1M 83 Linux
/dev/loop0p2        46314    77189    30876  15,1M 83 Linux
/dev/loop0p3        77190  2300261  2223072   1,1G  5 Extended
/dev/loop0p4      2300262 12786021 10485760     5G 83 Linux
/dev/loop0p5        77191  1126973  1049783 512,6M 83 Linux
/dev/loop0p6      1126975  2176757  1049783 512,6M 83 Linux
/dev/loop0p7      2176759  2207633    30875  15,1M 83 Linux
/dev/loop0p8      2207635  2223071    15437   7,6M 83 Linux
/dev/loop0p9      2223073  2238509    15437   7,6M 83 Linux

Partition table entries are not in disk order.

The partition table has been altered.
Calling ioctl() to re-read partition table.
Re-reading the partition table failed.: Das Argument ist ungültig
The kernel still uses the old table. The new table will be used at the next reboot or after you run
partprobe(8) or kpartx(8).
Syncing disks.
```

Anlagen, Teil B

Die vorliegende Auflistung ist die Dateihierarchie der erstellten Projekt-DVD. Sie beinhaltet die SD-Karten Speicherabbilder, sämtliche Software aller Methoden und aller Abbildungen zu dieser Arbeit.

```
/Abbildungen
                Abbildungen, siehe Abbildungsverzeichnis
/Arbeit
                Bachelorarbeit als Word-Dokument und im PDF-Format
/Methoden/Capture_malware
                findet Verwendung im Abschnitt 3.2.12
        Install_app
                findet Verwendung im Abschnitt 3.2.5
                findet Verwendung im Abschnitt 3.2.6
        Install_busybox
                findet Verwendung im Abschnitt 3.2.9
        Install_googleapps
                findet Verwendung im Abschnitt 3.2.4
        Install_loadedMalware
                findet Verwendung im Abschnitt 3.2.12
        Install_malware
                findet Verwendung in Abschnitt 3.2.11
        Install_root_on_Android4.4
                findet Verwendung im Abschnitt 3.2.7
/SD-Karten_Speicherabbilder
                Abschnitt 2.2.2.2 und folgende
```

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung, der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, den 07. Juni 2016

Danilo Morgado Anglés