HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Frau
Katharina Engelhardt

Analyse von Log-Artefakten bei Android Devices

2018

Fakultät **Angewandte Computer- und Biowissenschaften**

BACHELORARBEIT

Analyse von Log-Artefakten bei Android Devices

Autorin:
Katharina Engelhardt

Studiengang:
Allgemeine und Digitale Forensik

Seminargruppe:
FO15w3-B

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

Zweitprüfer:
M.Sc. Lars Richter

Mittweida, 09 2018

Bibliografische Angaben

Engelhardt, Katharina: Analyse von Log-Artefakten bei Android Devices, 133 Seiten, 31 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Bachelorarbeit, 2018

Referat

Die Analyse von Logdateien bietet in der Mobil-Forensik die Möglichkeit herauszufinden, wann welche Aktionen am Handy stattgefunden haben. Das Thema dieser Bachelorarbeit ist die Analyse bestimmter Protokolldateien unter Android. Der Schwerpunkt liegt in der Untersuchung der Ordner *usagestats*, *recent_images* und *com.whatsapp*, welche auf der Datenpartition liegen. In der Zielfragestellung dieser Studie gilt es festzustellen, ob die Protokolldateien in den jeweiligen Ordnern einen Hinweis auf die letzten Aktivitäten des Nutzers geben. Da sich bisher wenige Studien genauer mit der Protokollierung dieser Logdateien auseinandergesetzt haben, werden zu diesem Zwecke verschiedene Szenarien mit einem Android-Smartphone durchgeführt, um zu verstehen, was in den jeweiligen Dateien protokolliert wird. Dazu werden dieselben Szenarien auf drei unterschiedlichen Betriebssystemversionen getestet, um mögliche Unterschiede festzustellen und wesentliche Rückschlüsse für die Forensik zu folgern. Es ist zu beobachten, dass sich einige der untersuchten Dateien zur Beantwortung der Ausgangsfrage eignen, während andere Protokolldateien eher als kritisch zu betrachten sind.

Englischer Titel

Analysis of log-artifacts in Android devices

Abstract

This bachelor thesis deals with the analysis of log artifacts in Android devices. In the context of this study the three folders *usagestats*, *recent_images* and *com.whatsapp*, which are located in the *data* partition, will be examined in particular. The question to be answered is whether the log files reveal information about the user's latest activities. For this purpose various scenarios are performed with an Android phone to understand the content of these logfiles. Subsequently, the results of this thesis are compared to other publications and so essential findings for the forensics are concluded. It can be noted that some of the examined files provide helpful information regarding the initial question, while other log files are to be considered rather critical.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

ADB Android Debug Bridge

AOSP Android Open Source Project

API Application Programming Interface

App Application Software

CWM ClockWorkMod

DCIM Digital Camera Images

DTD Document Type Declaration

HTML Hypertext Markup Language

IM Instant Messaging

IPL Initial Program Loader

JTAG Join Test Action Group

LIFO Last In First Out

OHA Open Handset Alliance

RAM Random Access Memory

SGML Standard Generalized Markup Language

SPL Secondary Program Loader

SQL Structured Query Language

TWRP Team Win Recovery Project

UID User ID

URI Uniform Resource Identifier

VM virtuelle Maschine

VR Versuchsreihe

1 Einleitung

1.1 Übersicht

1.1.1 Zielstellung

Die vorliegende Bachelorarbeit beschäftigt sich mit dem Thema „Analyse von Log-Artefakten bei Android Devices“. Dabei liegt der Kernpunkt auf der Untersuchung bestimmter Logdateien. Ziel dieser Arbeit ist es herauszufinden, ob anhand dieser Protokolldateien die letzten Aktivitäten des Nutzers am Handy nachvollzogen werden können. Dafür werden verschiedene Szenarien mit einem Testhandy ausprobiert, um nachvollziehen zu können, was genau in den Protokolldateien gespeichert wird.

Eine Teilfrage, welche sich aus der Zielfragestellung ergibt ist, ob die Protokollierung der Logdateien sich bei verschiedenen Betriebssystemversionen ändert. Zu diesem Zwecke werden im Methodenteil dieser Arbeit die Szenarien auf verschiedenen Betriebssystemversionen getestet.

Der Schwerpunkt dieser Studie bezieht sich auf die Logdateien, welche sich auf der Datenpartition in dem Ordner *system/usagestats/* befinden. Auch werden verschiedene WhatsApp-Dateien genauer untersucht. Ein weiterer Ordner, welcher im Rahmen dieser Studie interessant ist, ist der Ordner *recent_images*. Er befindet sich ebenfalls auf der Datenpartition und abhängig von der Betriebssystemversion entweder im Ordner *system_ce* oder im Ordner *system*.

Von Interesse ist zudem die Frage, ob sich aus den Logdateien wesentliche Rückschlüsse für die Forensik ziehen lassen und welche Betriebssystemversionen sich für forensische Untersuchungen am besten eignen.

1.1.2 Motivation

Eine Welt ohne Smartphones ist in der heutigen Gesellschaft unvorstellbar. Fast jeder besitzt ein Handy und die Funktionen eines Smartphones werden immer vielfältiger, besonders was die Entwicklung und Erweiterung von Instant Messaging (IM)-Applikationen betrifft. Gerade Android ist ein beliebtes Betriebssystem für Handynutzer. Gemäß einer Studie besaßen im ersten Quartal des Jahres 2018 76% der Menschen in Deutschland ein Handy mit Android Betriebssystem [URL_11]. Die nahezu ständige Interaktion zwischen Benutzer und Handy bedeutet eine Speicherung vielfältiger Informationen auf dem Gerät. Neben ausgetauschten Chatnachrichten, Bildern, Videos und Audios, speichern Handys auch Geoinformationsdaten und viele weitere Details. Für den Forensiker

bietet ein Smartphone demnach eine gute Grundlage, Informationen über dessen Nutzer zu erlangen. Neben einigen persönlichen Informationen protokollieren verschiedene Logdateien die letzten Aktivitäten am Gerät. Forensiker stehen nicht selten vor der Aufgabe herauszufinden, wann beispielsweise ein Handy zuletzt benutzt wurde oder wann der Bildschirm das letzte Mal entsperrt wurde. Protokolldateien können entscheidende Hinweise hierzu liefern und helfen den Ermittlern, den Zeitraum der letzten Nutzung des Handys einzugrenzen.

1.1.3 Stand der Forschung

Ab Version 5.0 von Android gibt es eine neue API. Mit Hilfe dieser API android.app.usage ist dem Nutzer der Zugriff auf die App-Nutzungshistorie möglich. Dabei wird zwischen Tages-, Wochen-, Monats- und Jahresintervallen unterschieden. Für jede Anwendung wird aufgezeichnet, wann sie das letzte Mal benutzt wurde. Ein Zeitstempel erfasst Daten darüber, wann eine Komponente während des Tages in den Vordergrund oder Hintergrund rückt. Eine Komponente wird entweder durch Paket- oder Aktivitätsnamen identifiziert [URL_1].

Ein anderes Paper, welches sich bisher mit den Dateien unter */data/system/usagestats/* beschäftigte ergab, dass es durch diese Logdateien möglich ist herauszufinden, wann eine App das letzte Mal benutzt wurde [Nind2017].

1.2 Android

Android stellt ein mobiles Betriebssystem dar, welches auf Linux basiert. Ursprünglich wurde das Betriebssystem von dem StartUp-Unternehmen „Android“ entwickelt. Im Jahr 2005 übernahm schließlich Google die Entwicklungsarbeiten und hatte zum Ziel, ein freies und offenes Betriebssystem zu entwickeln.

Nachdem Android von Google gekauft wurde, wurde Android als Open Source Project (AOSP) verkauft. Teil des AOSP sind eine Reihe vorinstallierter Anwendungen, welche Android mit sich bringt. Mit der Gründung der Open Handset Alliance (OHA) wurde Android weiterentwickelt. Die OHA ist eine Sammlung von mehr als 80 Technologieunternehmen [Meier2012, Seite 10]. Unter diesen finden sich Hardwarehersteller, Softwareentwickler, führende Mobilfunkunternehmen und viele mehr [Meier2012, Seite 10].

Um ein freies und offenes Betriebssystem zu ermöglichen, wurde der größte Teil des Codes mittlerweile unter der Open Source Apache Lizenz veröffentlicht. Entwickler und Hersteller haben dadurch die Möglichkeit, verschiedene Arten von Anpassungen und Erweiterungen vorzunehmen, weshalb Android sehr benutzerfreundlich ist [Lee2012, Seite 2]. Seit der ersten Veröffentlichung wurde eine Reihe weiterer Versionen inklusive Co-

denamen entwickelt. Ein Ausschnitt dieser Versionen und Codenamen ist in Tabelle 1.1 dargestellt.

Tabelle 1.1: Ausschnitt Versionen und Codenamen [URL_9]

Codename	**Android-Version**	**API-Level**
KitKat	4.4-4.4.4	19
Lollipop	5.0	21
Marshmallow	6.0	23
Nougat	7.0	24
Oreo	8.0.0	26

1.2.1 Systemarchitektur

Das Android Betriebssystem unterteilt sich in vier Schichten. Die unterste Schicht des Software-Stacks bildet der Kernel. Die Kernelschicht ist als Abstraktionsebene zwischen Hardware und Software zu verstehen [Tamma2015, Seite 13]. Die Version des Kernels ist von der Android-Version abhängig. In dieser Ebene befinden sich verschiedene Treiber, wie beispielsweise der Kameratreiber oder Displaytreiber, welche die zugrunde liegende Hardware steuern [Tamma2015, Seite 13]. Weitere Funktionen des Kernels bestehen im Prozessmanagement, der Speicherverwaltung und dem Netzwerkzugriff [Gargenta2012, Seite 8].

Die über dem Kernel liegende Schicht wird von den Bibliotheken gebildet. In dieser Schicht befinden sich die nativen Bibliotheken wie die libc- und SSL-Bibliotheken aber auch andere Bibliotheken für die Wiedergabe von Audio- und Videomaterial [Meier2012, Seite 15]. Neben den nativen Bibliotheken finden sich in dieser Ebene außerdem die Dalvik Virtual Machine und die Android Runtime. Die Dalvik-VM wird zur Ausführung der in Java programmierten Anwendungen benötigt. Dafür wird der Java Bytecode durch den Dex-Compiler in Dalvik-Byte-Code umgewandelt, sodass die Dalvik-VM in der Lage ist, diesen Code zu lesen und zu verwenden [Tamma2015, Seite 14].

Im Unterschied zum Java-Bytecode ist dieser Bytecode für eine eher speicherarme Umgebung ausgelegt. Dalvik beruht auf Just-in-time Kompilierung beim Start einer Anwendung. Dieses Prinzip wurde mit der Android-Version 5.0 abgeändert, es wurde die Android Runtime eingeführt. Sie verwendet keine Just-in-time Kompilierung, sondern eine Vorabkompilierung während der Installation einer Anwendung [Tamma2015, Seite 14].

Die über den Bibliotheken liegende Schicht wird von dem Application Framework gebildet. Diese Schicht enthält beispielsweise den „Activity Manager“, welcher den Lebenszyklus einer App steuert, auf den im späteren noch genauer eingegangen wird. Das Application Framework ist besonders interessant für Entwickler. Es bietet alle Klassen,

um Android-Anwendungen erstellen zu können. Der Content Provider in dieser Schicht übernimmt die Verwaltung der Anwendungsressourcen.

Die letzte und oberste Schicht wird von den Anwendungen selbst gebildet. Dabei wird zwischen vorinstallierten Systemanwendungen und Anwendungen von Drittanbietern unterschieden.

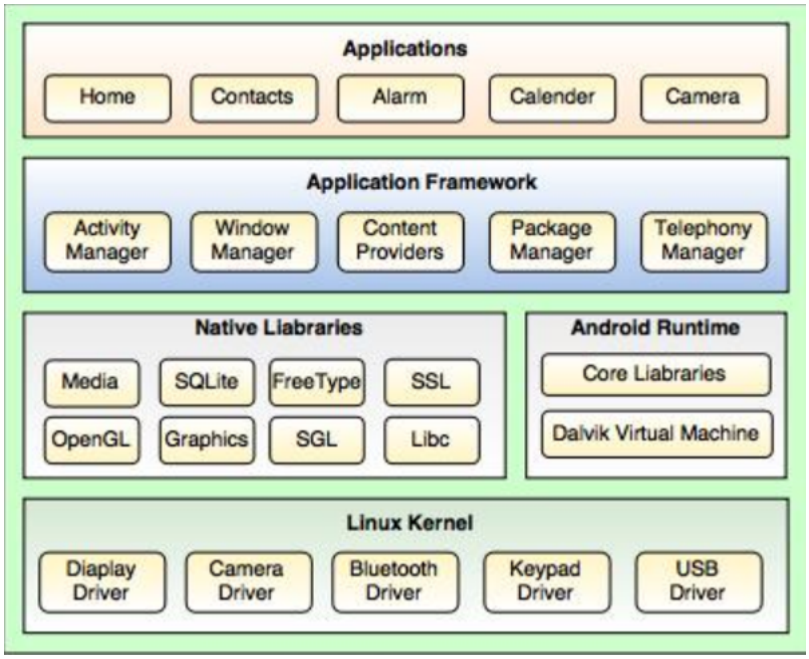

Abbildung 1.1: Android-Architektur [URL_14]

1.2.2 Partitionsschema und wichtige Ordner

Für eine forensische Untersuchung ist es grundlegend zu wissen, wo Daten abgelegt werden. Oft begrenzt sich eine Untersuchung auf bestimmte Bereiche. Bei Android-Geräten gibt es einige Standardpartitionen, die auf jedem Gerät mit beliebiger Version vorzufinden sind. Zusätzlich zu diesen gibt es Partitionen, die sich je nach Hersteller und Version unterscheiden [Tamma2015, Seite 71]. Demnach ist es unwahrscheinlich, dass zwei unterschiedliche Geräte exakt dieselbe Partitionierung vorweisen.

Partitionierung im Allgemeinen beschreibt die logische Aufteilung des Speicherplatzes in Speicherbereiche. Auf die einzelnen Partitionen kann unabhängig voneinander zugegriffen werden [Tamma2015, Seite 71]. Das Partitionslayout wird meist vom Bootloader übernommen [Drake2014, Seite 58].

Jede Partition speichert Daten, die für bestimmte Funktionen spezifisch sind [Di Leom-2015]. Der Befehl *cat /proc/partitions* zeigt auf, welche Partitionen auf dem zu untersuchenden Gerät vorhanden sind [Reibold2016, Seite 100].

Grundlegende Partitionen bei allen Android-Geräten sind */boot, /system, /data, /recovery, /misc* und */cache*. Zudem kann die */sdcard*-Partition wertvolle Informationen für forensische Untersuchungen enthalten. Diese präsentiert die Daten der SD-Karte [Tamma2015, Seite 80]. Sie dient als Speicherort für „persönliche Daten und Dateien“ [Reibold2016, Seite 39]. Neben Multimediadaten wie beispielsweise Audio-, Video- und Bilddateien, können hier ebenfalls Applikationsdaten abgelegt werden.

Eine der bedeutsamsten Partitionen ist die Datenpartition */data*. Diese ist nicht nur im Rahmen dieser Bachelorarbeit interessant, sondern sie ist deshalb so wichtig, weil sie viele Benutzerdaten speichert. Neben Benutzerdaten werden hier ebenfalls große Mengen an Anwendungsdaten abgelegt. Nutzerdaten befinden sich noch auf den Partitionen der SD-Karte und in der Cache-Partition [Di Leom2015].

Die */system*-Partition ermöglicht das Booten in den normalen Modus, was durch ein Löschen dieser Partition nicht mehr möglich wäre. Sie ist auf jedem Android-Gerät vorhanden. In dieser Partition befindet sich das Betriebssystem mit Ausnahme von Kernel und RAM-Disk. Neben Bibliotheken und vorinstallierten Anwendungen enthält diese Partition auch Systembinärdateien [Tamma2015, Seite 81].

Die */cache*-Partition dient als Zwischenspeicher für häufig verwendete Daten und Dateien. Sie unterstützt den beschleunigten Abruf immer wieder genutzter Daten und Dateien. Diese werden hier nur temporär abgelegt. Interessant für forensische Untersuchungen kann diese Partition deshalb sein, weil sich hier möglicherweise Daten befinden, welche nicht mehr in der */data*-Partition vorhanden sind [Mahalik2016, Seite 193].

Die */boot*-Partition beinhaltet alle Dateien und Informationen, welche zum Booten des Telefons notwendig sind [Tamma2015, Seite 72]. Ein Starten des Gerätes ohne diese Partition ist nicht möglich [Muth2013]. Hier befindet sich der Kernel.

Die */recovery*-Partition wird in der deutschen Sprache als Wiederherstellungspartition bezeichnet. Sie ermöglicht das Booten in den Recovery-Modus [Tamma2015, Seite 72]. In diesem Modus können verschiedene Operationen ausgeführt werden, welche später noch näher beschrieben werden.

Die letzte zu erwähnende Standardpartition ist die */misc*-Partition. „In dieser Partition sind grundlegende Einstellung zum jeweiligen Gerät zu finden“ [Reibold2016, Seite 100]. Die hier abgelegten Informationen geben Auskunft über den Zustand des Gerätes wie beispielsweise über Hardware- und USB-Einstellungen [Mahalik2016, Seite 193].

Tabelle 1.2: Partitionen unter Android [Quelle: eigene Tabelle]

Partition	**wesentliche Daten**
/boot	Daten für Bootvorgang inklusive Kernel [Tamma2015, Seite 72-79]
/recovery	„Daten für alternativen Bootvorgang“ [Muth2013]
/misc	verschiedene Hardware-Einstellungen [Tamma2015, Seite 79]
/data	Benutzer- und Anwendungsdaten [Tamma2015, Seite 72]
/system	Betriebssystem ohne Kernel [Tamma2015, Seite 72]
/cache	Zwischenspeicherung von Daten [Tamma2015, Seite 72]

Neben dem Partitionierungsschema stellen auch die Verzeichnisse/Ordner eine wichtige Untersuchungsquelle dar. Die Android-Hierarchie bildet eine angepasste Version an die Linux-Hierarchie [Tamma2015, Seite 74]. Sie ist ebenfalls als baumförmige Struktur dargestellt.

Den obersten Knoten des Verzeichnisbaumes bildet das Root-Verzeichnis [Reibold2016, Seite 101]. Um die komplette Android-Verzeichnisstruktur einsehen zu können, benötigt man Root-Rechte.

Im Folgenden werden forensisch interessante Ordner beschrieben.

Der Ordner *lost+found* speichert Daten, die im Falle einer Beschädigung des Dateisystems wiederhergestellt wurden. Er befindet sich auf der *cache*-Partition.

In */data/data* findet man die privaten Daten aller Anwendungen [Tamma2015, Seite 75]. Dieser Ordner ist forensisch von Bedeutung, weil hier die meisten Benutzerdaten gespeichert sind [Tamma2015, Seite 75]. Auf der Datenpartition */data* befindet sich außerdem oft der Ordner *dalvik-cache*. Auch dieser enthält mehrere Protokolle, die unter Umständen interessante Informationen enthalten.

Die Datei *build.prop* enthält Details über das zu untersuchende Gerät und verschafft einen Überblick über das Gerätemodell. Sie liegt auf der *system*-Partition. Informationen wie Android-Version, Produkt-Modell und CPU-Details können hier in Erfahrung gebracht werden [Tamma2015, Seite 81].

Im Ordner *app*, ebenfalls auf der *system*-Partition, sind Systemanwendungen und vorinstallierte Anwendungen enthalten. Der Ordner *framework* dieser Partition enthält die Quellen für das Android-Framework [Tamma2015, Seite 82].

Die */sdcard*-Partition enthält einige forensisch wichtige Ordner, *media*, *android_secure* oder *DCIM*. In dem zuletzt genannten Ordner befinden sich Fotos. Die durch die Handykamera aufgenommenen Fotos werden dabei unter */DCIM/Camera* gespeichert.

1.2.3 Speicherung von Anwendungsdaten

Grundsätzlich kann man bei Android zwischen Systemanwendungen und vom Benutzer installierten Anwendungen unterscheiden. Systemanwendungen stellen Anwendungen dar, welche mit dem Telefon ausgeliefert werden. Benutzeranwendungen werden vom Benutzer selbst durch einen App-Store, wie beispielsweise den Google Play Store, heruntergeladen [Tamma2015, Seite 85].

Bei Anwendungen kann noch eine weitere Unterteilung stattfinden, nämlich die zwischen den von Android mitgelieferten Apps, den vom Hersteller installierten Apps und den vom Mobilfunkanbieter installierten Anwendungen [Tamma2015, Seite 85]. Alle diese Applikationen, egal welcher Kategorie sie angehören, speichern eine Menge wertvoller Daten, die forensisch von Bedeutung sein können. Neben GPS-Daten können Chat-Nachrichten, Kontakte, Bilder oder die Browser-Historie von Anwendungen gespeichert sein [Tamma2015, Seite 85].

Anwendungsdaten können sowohl auf internen als auch auf externen Speichern liegen.

Daten einer Anwendung, die auf dem internen Speicher abgelegt werden, werden im Unterverzeichnis */data/data* gespeichert [Tamma2015, Seite 85]. Die Anwendungen werden in diesem Verzeichnis nach ihrem Paketnamen benannt. Die Applikation WhatsApp liegt beispielsweise unter */data/data/com.whatsapp*. Das heißt, dass der Ort des internen Speichers vordefiniert ist.

Beim externen Speicher ist dies anders. Dort können die Daten einer Applikation an einem beliebigen Ort gespeichert werden.

Die im internen Speicher gesicherten Daten sind privat und für andere nicht zugänglich [Tamma2015, Seite 87]. Das heißt, dass die in diesem Verzeichnis abgelegten Daten nur mit Root-Berechtigung eingesehen werden können. Welche Daten dort abgelegt werden hat viel mit der App selbst zu tun. Es kommt unter anderem darauf an, ob die App „selbst Daten generiert“ [Reibold2016, Seite 41] und welche Daten die App nutzt. Für die meisten Anwendungen im internen Speicher werden jedoch die gleichen Ordner erstellt, in denen die Daten einer Anwendung gespeichert werden [Tamma2015, Seite 87].

Der *lib*-Ordner beispielsweise speichert Programmbibliotheken, welche von der jeweiligen Anwendung benötigt werden [Tamma2015, Seite 87]. Der Ordner *files*, gespeichert unter */data/data/<Paketname>/*, enthält vom Entwickler gesicherte Daten [Tamma2015, Seite 87]. Der Inhalt dieses Ordners kann von Anwendung zu Anwendung variieren, es handelt sich jedoch meist „um programminterne Dateien“ [Reibold2016, Seite 43].

Shared_pref, *databases* und *cache* sind weitere Ordner, in denen Daten von Anwendungen abgelegt werden.

Der Ordner *databases* enthält Daten, die in SQLite Datenbanken oder in Journal-Dateien abgelegt sind. Das SQLite-Datenbankformat findet Anwendung in vielen mobilen Geräten und ist eine weit verbreitete Speicherart [Tamma2015, Seite 90]. Zweck dieses Datenbankformates ist die strukturierte Speicherung von Daten [Tamma2015, Seite 90]. Datenbankdateien enden auf die Endung *.db* [Reibold2016, Seite 43]. Der Datenbankordner kann forensisch wichtige Informationen enthalten.

Der *cache*-Ordner beschleunigt Anwendungen den Zugriff auf bestimmte Daten.

Der Ordner *shared_pref* unter */data/data/<packagename>/* legt die Daten im XML-Format ab, was eine Speicherung von Schlüssel-Wert-Paaren ermöglicht [Tamma2015, Seite 86]. Alle anderen Ordner, die sich in dem jeweiligen Verzeichnis der Anwendung befinden, wurden vom Anwendungsentwickler erstellt [Tamma2015, Seite 88-89]. Der folgende Screenshot (Abbildung 1.2) zeigt den internen Speicher der WhatsApp-Anwendung.

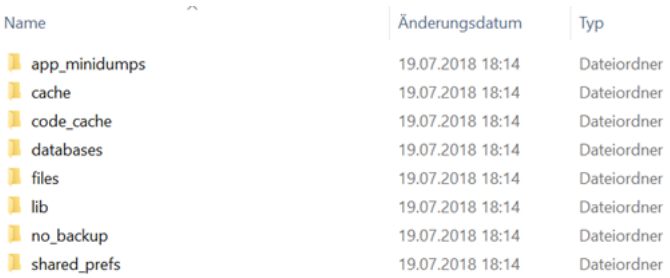

Abbildung 1.2: Interner Speicher WhatsApp-Anwendung [Quelle: eigene Abbildung]

Neben dem internen Speicher besteht die Möglichkeit, Daten extern zu speichern, beispielsweise mithilfe einer SD-Karte. Externe Speicher unterliegen keinen speziellen Sicherheitsvorschriften. Die dort gespeicherten Daten sind frei zugänglich und können mit entsprechender Berechtigung von anderen Anwendungen abgerufen werden [Tamma2015, Seite 90].

Eine weitere Möglichkeit der Datenspeicherung ist die Speicherung auf Netzwerklaufwerken. Daten können auf eigenen webbasierten Diensten wie beispielsweise der Dropbox abgelegt werden [Tamma2015, Seite 91]. Für diese Dienste gibt es Apps, die auf dem Smartphone installiert werden müssen [Reibold2015, Seite 333]. Nachdem die entsprechende App installiert wurde, fungiert das Smartphone als Netzwerk-Client [Reibold2016, Seite 50].

Neben den von den Anwendungen abgelegten Daten, erzeugen Android-Geräte viele Logdateien. Diese Protokolldateien verteilen sich über das ganze Betriebssystem. Sie können nützliche Informationen, z.B. über Anwendungszustände enthalten [Reibold2016,

Seite 50]. Häufig liegen diese Logdateien im XML-Format vor, weshalb der Aufbau von XML-Dateien im nächsten Unterpunkt genauer beschrieben wird.

1.2.4 Aufbau einer XML-Datei

Im späteren Methodenteil (Kapitel 2) dieser Bachelorarbeit werden verschiedene Dateien untersucht, welche unter anderem im XML-Format vorliegen. Um den Inhalt der zu untersuchenden Dateien besser verstehen zu können, ist ein grundlegendes Verständnis zum Aufbau einer XML-Datei notwendig.

Die Abkürzung XML steht für „extensible Markup Language“ [Hitzler2008, Seite 24]. XML versteht sich dabei nicht als Markup-Sprache (zu Deutsch: Auszeichnungssprache), sondern als eine Sprache, welche grundlegende Regeln festlegt, mit denen eine Auszeichnungssprache erstellt werden kann [Westermann2002, Seite 4]. Somit lässt sich XML auch als eine Metabeschreibungssprache beschreiben.

In der Literatur taucht XML oftmals im Zusammenhang mit SGML auf. Dabei versteht sich XML als eine „abgespeckte“ Version von SGML [Hitzler2008, Seite 19]. Ein Beispiel für eine bekannte Auszeichnungssprache ist HTML. Der grundlegende Unterschied zwischen HTML und XML besteht darin, dass HTML einen vordefinierten Satz an Elementen hat und XML diesbezüglich keine Einschränkungen vorweist [Westermann2002, Seite 1].

XML stellt einen Standard dar, welcher vom World Wide Web Consortium herausgegeben wurde [Wegner2014, Seite 17].

In Abbildung 1.3 ist die grundlegende Struktur eines XML-Dokumentes dargestellt.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE telegramm SYSTEM "/xml-resourcen/dtds/telegramm.dtd">
<telegramm pri="wichtig">
  <an>Sarah Bellum</an>
  <von>Oberst Zeitsprung</von>
  <betrifft>Roboter-Sitting-Anweisungen</betrifft>
  <grafik datei-ref="abbildungen/ich.eps"/>
  <nachricht>Vielen Dank, dass Sie sich in meiner Abwesenheit
    um meinen Roboterkumpel <name>Zonky</name> kümmern.
    Er muss <betonung>zweimal täglich</betonung>
    aufgeladen werden. Falls er anfängt, launisch zu werden,
    geben Sie ihm einen Viertelliter Öl. Ich komme sofort zurück,
    wenn ich das bösartige Meisterhirn <bösewicht>
    Dr. Himmelblau Reisweg</bösewicht> aufgespürt habe.
  </nachricht>
</telegramm>
```

Abbildung 1.3: Beispiel XML-Dokument [Ray2002]

Eingeleitet wird ein XML-Dokument meist mit einer XML-Deklaration, wie in Abbildung 1.3 veranschaulicht. Neben der Version der XML-Spezifikation, informiert die XML-Deklaration darüber, dass es sich um ein XML-Dokument handelt [Westermann200 2, Seite 12]. Gleichermaßen beschreibt die Deklaration den in dieser Datei verwendeten Zeichensatz. Der Standardzeichensatz bei XML entspricht meistens Unicode [Ray2001, Seite 1]. Nach einleitender Deklaration kann die „Document Type Declaration“ (DTD) erfolgen. Diese beschreibt die Struktur des Dokumentes in Bezug auf ihre Elemente [Westermann2002, Seite 12].

Der eigentliche Inhalt des Dokumentes wird mit dem Wurzelelement eingeleitet, welches gleichzeitig das erste Element eines XML-Dokumentes bildet. Dabei besitzt jede XML-Datei ausschließlich ein Wurzelelement. Im Beispiel von Abbildung 1.3 ist <*telegramm*> das Wurzelelement, welches alle weiteren Elemente umschließt. Der weitere Inhalt eines solchen Dokumentes kann sich in weitere Sub-Elemente gliedern, welche in der Literatur häufig auch als Knoten bezeichnet werden [Hitzler2008, Seite 21]. Die Sub-Elemente ihrerseits können sich in tiefste Ebenen verschachteln, weshalb das XML-Dokument als hierarchische Struktur angesehen wird.

Bei den Elementen selbst, wird zwischen verschiedenen Elementtypen unterschieden. Es gibt Elemente, die nur Daten, aber keine weiteren Sub-Elemente enthalten. Ferner gibt es Elemente, welche andere Elemente, dafür aber keine Daten enthalten. Ein weiterer Elementtyp wird von den Elementen gebildet, die sowohl Daten als auch Sub-Elemente enthalten. Leere Elemente bilden den letztmöglichen Elementtyp.

Elemente eines XML-Dokumentes sind innerhalb eines sogenannten Start- und End-Tags definiert. Zwischen Start- und End-Tag kann Text stehen, welcher den Inhalt eines Elementes bildet [Wegner2014, Seite 34]. Um die Elemente gezielt ansprechen zu können, besitzen diese einen sogenannten XML-Namen. Im Beispiel von Abbildung 1.3 besitzt das erste Element nach dem Wurzelelement den XML-Namen „an“ und trägt den Inhalt „Sarah Bellum“. Dieses Element ist innerhalb des Start-Tags <*an*> und des End-Tags </*an*> definiert. Grundsätzlich bestehen die meisten Elemente aus einem Start- und End-Tag. Die leeren Elemente bilden jedoch eine Ausnahme. Die Besonderheit dieser Elemente besteht darin, dass diese nicht unbedingt ein End-Tag besitzen müssen [Westermann2002, Seite 69-72]. Ein Beispiel für ein leeres Element ist das Element <*grafik* ../>, in Abbildung 1.3 [Ray2002, Seite 80].

Neben den Elementen findet man in einer XML-Struktur häufig Attribute. Attribute können ein Element weitgehend beschreiben. Sie übermitteln Informationen in Form eines Name-Wert-Paares [Westermann2002, Seite 77]. In Abbildung 1.3 wird dem Element „telegramm“ das Attribut „pri“ zugeordnet, welches den Wert „wichtig“ besitzt.

Bei der Namensvergebung von Attributen und Elementen sind bestimmte Regeln zu beachten. Bei Elementen muss der Name mindestens ein Zeichen lang sein. Beginnen sol-

len diese Namen entweder mit einem Buchstaben oder einem Unterstrich. Im Falle eines Buchstaben können sowohl Klein- als auch Großbuchstaben verwendet werden. Weitere Beschränkungen bezüglich der Länge des Namens gibt es nicht. Darüber hinaus muss der XML-Name des Start-Tags mit dem XML-Namen des dazugehörigen End-Tags übereinstimmen. Entscheidend ist hierbei, dass auch die Groß- und Kleinschreibung identisch ist. Bestimmte Zeichen, wie beispielsweise das Und-Zeichen und das Komma-Zeichen, dürfen nicht verwendet werden [Westermann2002, Seite 72].

Auch die Attribute unterliegen bestimmten Regeln. Die Namensgebung erfolgt analog zu der der Elemente [Westermann2002, Seite 78]. Weiterhin erscheint ein Attribut immer im Start-Tag eines Elementes. Der Wert eines Attributes wird in Anführungszeichen angegeben. Die Verbindung zwischen Attribut-Namen und Attribut-Wert erfolgt mittels eines Gleichheitszeichens, welches unter keinen Umständen fehlen darf. Attribute sind nicht in der Lage weitere Elemente oder Attribute zu beinhalten. Ein Attribut darf innerhalb eines Elementes nur einmal auftreten.

1.2.5 Team Win Recovery Project

TWRP steht für „Team Win Recovery Project“ und wurde von der Team Win Gruppe entwickelt [URL_12]. Es stellt eine Alternative zum ClockWorkMod (CWM) dar und gilt derzeit als eines der bekanntesten Recovery-Tools [Immler2016, Seite 34]. TWRP ist ein Custom-Recovery, welches das Hersteller-Recovery (Stock-Recovery) ersetzen kann. Das Projekt entstand 2011 und wurde als Open Source Projekt entwickelt [Morgillo2016, Seite 175]. TWRP basiert auf einer touchbasierten Benutzeroberfläche, welche es ermöglicht, direkt über den Bildschirm mit dem Wiederherstellungsmodus zu kommunizieren [Morgillo2016, Seite 176]. Es ist außerdem möglich, mit der benutzerdefinierten Wiederherstellung über die Android Debug Bridge (ADB) zu kommunizieren. Erkennt der Computer das entsprechende Gerät über ADB, kann mit dem Kommando *adb shell* eine Verbindung zur Wiederherstellungs-Shell erfolgen [Morgillo2016, Seite 176]. Dabei lohnt es sich auf einige Eigenschaften genauer einzugehen.

Die Backup-Funktion der touchbasierten Benutzeroberfläche ermöglicht es, verschiedene Partitionen zu sichern. Man spricht bei diesen Sicherungsvorgängen auch von einem Nandroid-Backup. Es stellt eine 1:1 Kopie dar und wird aufgrund des NAND-Flash-Speichers so genannt [Porteck2018, Seite 63].

Die Daten eines Backups werden „auf einer Speicherkarte gesichert und können auch nach einem Hard-Reset oder der Installation eines anderen Custom-ROM über Restore wieder zurück gesichert werden“ [Immler2016, Seite 35]. Ein Custom-ROM stellt ein von Drittanbietern entwickeltes Android-Betriebssystem dar.

Eine weitere, von TWRP bereitgestellte Funktion ist das Löschen bestimmter Daten. Dies

kann besonders dann von Bedeutung sein, wenn man ein neues ROM installieren möchte. Die Wipe-Funktion ermöglicht eine Auswahl an zu löschenden Daten und es besteht so die Möglichkeit, einen Factory-Reset durchzuführen und somit das Smartphone wieder auf die Werkseinstellungen zurückzusetzen.

Generell werden alle Funktionen bei TWRP erst dann ausgeführt, wenn man mit dem Schieberegister der Benutzeroberfläche von links nach rechts fährt. Eine der zentralen Funktionen von TWRP ist das Flashen verschiedener *.zip-Dateien. Über die Schaltfläche „Install“ können so beispielsweise Custom-ROMs oder andere Softwares installiert werden.

1.2.6 LineageOS

Nachdem sich Android auf dem Markt durchgesetzt hat, besteht eine neue Herausforderung darin, Custom-ROMs entgegen zu wirken [Barton2016, Seite 84]. Neben dem Stock-ROM gibt es von Drittanbietern entwickelte Custom-ROMs. Diese stellen eine veränderte Form der Android-Firmware dar. Das „custom“ in Custom-ROM leitet sich von dem englischen Wort „customized“ ab und bedeutet so viel wie: „an den Kunden angepasst“ [Rehberg2014, Seite 248]. Das Android-ROM oder Stock-ROM bezeichnet das vom Gerätehersteller vergebene Betriebssystem [Saurabh2016].

Ein Custom-ROM ist nicht das Stock-ROM, es stellt meist eine modifizierte oder komplett andere Form des Stock-ROMs dar [Saurabh2016]. Der offen gelegte Quellcode von Android macht dies möglich.

Es gibt verschiedene Gründe für die Verwendung von Custom-ROMs. Ein Grund dafür ist das Entfernen der Bloatware. Unter Bloatware versteht man die vorinstallierte Software, welche sich auf dem Handy befindet und nicht gebraucht wird.

Custom-ROMs liefern keine Google Apps mit. Diese können jedoch durch das Flashen einer bestimmten *.zip-Datei ebenfalls auf das Handy gebracht werden. Bei der Verwendung von Custom-ROMs entfällt das Warten auf die nächste Betriebssystemversion. Für viele alte Smartphones werden keine Updates mehr bereitgestellt, Besitzer solcher Handys können dann auf Custom-ROMs ausweichen. Mit dem Flashen eines Custom-ROMs ist es möglich, immer die neuste Android-Version auf dem Handy zu haben, sofern das Gerät vom Drittanbieter unterstützt wird [Saurabh2016].

Ist der Bootloader auf dem Smartphone gesperrt, gilt es diesen vor der Installation eines Custom-ROMs zu entsperren. Nicht alle Bootloader sind standardmäßig gesperrt. Nach dem Entsperren folgt die Installation eines Custom-Recovery und anschließend das Flashen eines Custom-ROMs [Saurabh2016].

Einer der weit verbreitetsten Custom-ROMs war jahrelang CyanogenMod. Deren Dienste mussten jedoch nach einigen Turbulenzen eingestellt werden und LineageOS wurde ihr Nachfolger [Wölbert2017]. LineageOS unterstützt derzeit viele verschiedene Geräte [Ye2017, Seite 19]. Es existieren verschiedene Versionen von LineageOS. Bei der Installation eines Custom-ROMs muss darauf geachtet werden, dass die LineageOS-Version der richtigen Android-Version entspricht. Dabei steht die Version 14.1 von LineageOS für die Android-Version 7.1.2.

1.3 Aufbau einer Android-Applikation

Für die in dieser Bachelorarbeit zu untersuchenden Logdateien ist ein besseres Verständnis des Aufbaus einer App notwendig. Neben dem Aufbau werden auch die verschiedenen Lebensphasen einer Aktivität beleuchtet, da auch dies notwendig ist, um die Protokollierung der späteren Dateien nachzuvollziehen.

1.3.1 Android Sandboxing

Bei Android laufen alle Anwendungen in einer eigenen Sandbox, die durch bestimmte Mechanismen geschützt ist [URL_4]. Die Sandbox gehört zu einer von vielen Sicherheitsmaßnahmen, die das Android Betriebssystem ausmachen [Reibold2016, Seite 54].

Die Anwendungen erhalten vom System eine Benutzer-ID (UID). Die UID wird direkt nach erfolgreicher Installation der Anwendung vergeben. Jede Anwendung wird als eigener Prozess ausgeführt und jeder Prozess wiederum hat seine eigene virtuelle Maschine [URL_4]. So läuft jede Anwendung isoliert von anderen Anwendungen und es ist den Applikationen untereinander nicht möglich zu interagieren oder die Daten einer anderen Anwendung zu lesen geschweige denn darauf zuzugreifen [Reibold2016, Seite 56]. Sobald eine Komponente einer App ausgeführt wird, startet das System für diese Anwendung einen Prozess [URL_4].

Ein weiterer Sicherheitsmechanismus ist durch das Prinzip geringster Privilegien gegeben [URL_4]. Dabei hat eine Anwendung nur Zugriff auf Ressourcen, welche sie auch wirklich benötigt. Dieses Prinzip verhindert, dass eine Anwendung auf alle Teile des Systems zugreifen kann [URL_4].

Trotz der Tatsache, dass jede Anwendung in ihrem eigenen Prozess läuft, haben Anwendungen die Möglichkeit, Daten mit anderen Anwendungen auszutauschen. Für die gemeinsame Datennutzung gibt es drei Möglichkeiten.

Geht es darum, dass die Daten nur mit bestimmten Anwendungen geteilt werden, kann dies über eine SharedUserID ermöglicht werden. Alle Anwendungen, die die gleiche

SharedUserID besitzen, können Daten untereinander austauschen und darauf zugreifen [URL_10]. Um zusätzlich Systemressourcen zu sparen ist es möglich, dass Anwendungen mit derselben SharedUserID im gleichen Linux-Prozess ausgeführt werden [URL_4].

Die zweite Möglichkeit, wie Anwendungen gemeinsam auf Daten zugreifen können, wird über Content Provider realisiert. Content Provider stellen Schnittstellen dar, welche den Austausch von Daten zwischen verschiedenen Applikationen ermöglichen [Gargenta2012, Seite 33].

Bei der dritten Möglichkeit speichert die Anwendung ihre Daten so, dass andere Anwendungen auf diese lesend oder sogar „lesend und schreibend zugreifen“ können [URL_10].

1.3.2 Bestandteile einer Anwendung

Grundsätzlich gibt es vier wesentliche Bausteine einer App. Diese sind Activities, Services, Broadcast Receivers und Content Provider. Im Folgenden werden diese Bausteine genauer definiert.

Die Activity (zu Deutsch Aktivität) stellt ein Bild auf der Benutzeroberfläche dar und gilt für den Nutzer als sichtbarer Bestandteil einer App [Baltes-Götz2018, Seite 87]. Grundsätzlich bestehen die meisten Apps aus mehreren solcher Aktivitäten. Um zwischen den einzelnen Oberflächen einer Anwendung zu wechseln, werden neue Aktivitäten gestartet oder man kehrt zu alten Aktivitäten zurück. Bei den meisten Anwendungen wird eine Aktivität als „Main“ angegeben [Liu2011]. Diese Main-Aktivität stellt meist den Startbildschirm einer Anwendung dar [Liu2011]. Die Startaktivität ist häufig mit verschiedenen Bedienelementen ausgestattet, um den Übergang zu anderen Aktivitäten zu ermöglichen [Baltes-Götz2018, Seite 242]. Die Aktivitäten einer Anwendung sind unabhängig voneinander, so wird ermöglicht, dass auch andere Applikationen diese starten können, sofern sie die entsprechende Erlaubnis dafür mit sich bringen.

Der Service zeigt im Gegensatz zu einer Aktivität keine grafische Oberfläche an und agiert demnach unsichtbar [Lee2012, Seite 429]. Diese Komponente läuft im Hintergrund und hat die Aufgabe, langlaufende Prozesse auszuführen. Services werden eingesetzt, wenn bestimmte Funktionen oder Informationen permanent und kontinuierlich verfügbar sein sollen und zwar unabhängig davon, welche Applikation oder Aktivität sich momentan im Vordergrund befindet [Baltes-Götz2018, Seite 87]. Ein Beispiel für die Verwendung eines Dienstes ist das Abspielen von Musik im Hintergrund, während man sich in einer anderen Applikation befindet [Lee2012, Seite 429]. Die Kontrolle über Services inklusive des Startens und Stoppens eines Service, erfolgt durch andere Komponenten.

Der Content Provider ist eine Schnittstelle zur Veröffentlichung und Nutzung von Da-

ten. Sofern Daten von mehreren Anwendungen genutzt werden sollen, bieten Content Provider eine optimale Lösung. Mit deren Hilfe ist es Anwendungen möglich, Daten gezielt abzufragen [Becker2015, Seite 283-290]. Neben der Datenabfrage ist er ebenfalls für die Datenspeicherung verantwortlich [Lee2012, Seite 293]. Content Provider können so konfiguriert werden, dass sie entweder den Zugriff aus anderen Anwendungen erlauben oder nicht [Meier2012, Seite 252]. Die nativen Content Provider unter Android stellen nützliche Datenbanken wie den Medienspeicher oder Kontaktdatenbanken bereit [Meier2012, Seite 252]. Sie sind außerdem in der Lage, Datensätze zu aktualisieren und zu löschen.

Rundrufnachrichten stellen Nachrichten dar, welche sowohl vom System als auch von Anwendungen versendet werden können. Durch den Broadcast Receiver (zu Deutsch: Rundrufempfänger) einer Anwendung ist es Applikationen möglich, auf solche Nachrichten zu reagieren [Baltes-Götz2018, Seite 87]. Die Nachricht über einen niedrigen Akkustand ist ein Beispiel für eine Rundrufnachricht. Diese werden generell gesendet, wenn ein interessantes Ereignis eintritt und sie können zwischen Applikationen ausgetauscht werden [URL_6]. Rundrufnachrichten stellen eine Art Nachrichtensystem zwischen Anwendungen oder zwischen Anwendung und System dar, welches auch außerhalb des normalen Benutzerflusses verwendet werden kann [URL_6]. Das bedeutet, dass durch Broadcast Receiver das System beispielsweise Meldungen an Applikationen ausgeben kann, die gerade nicht in Benutzung sind [URL_4]. Genau wie Services zeigen auch Broadcast Receiver keine eigene Benutzeroberfläche an.

Das Manifest einer Anwendung gibt einen Überblick darüber, welche Komponenten diese beinhaltet. Alle verwendeten Komponenten müssen definiert sein. Möchte man eine Komponente ausführen, welche nicht im Manifest definiert ist, kommt es zu einer Fehlermeldung [Meier2012, Seite 61]. Neben den verwendeten Komponenten definiert das Manifest die notwendige Android-Version, die erforderlich ist, um die Anwendung nutzen zu können [Reibold2015, Seite 412; Staudemeyer2013, Seite 58] sowie die Berechtigungen einer Anwendung. Zusammenfassend kann man sagen, dass das Manifest, welches im XML-Format vorliegt, einen Überblick über die Bestandteile der Anwendung und dessen wesentliche Informationen gibt.

1.3.3 Activity-Stack

Der Activity-Stack (zu Deutsch Aktivitäten-Stapel) wird verwendet, um die Priorität einer Anwendung zu bestimmen. Dabei kann sich eine Aktivität in verschiedenen Zuständen befinden. Sie kann erzeugt, gestoppt, wiederaufgenommen oder zerstört werden. Wann sich eine Aktivität in welchem Zustand befindet, ist für das spätere Verständnis der protokollierten Informationen in der Logdatei wichtig.

Grundsätzlich ist der Status einer Aktivität durch die momentane Position des Aktivitäten-

Stapels gekennzeichnet. Auf diesem Stapel befinden sich alle Aktivitäten, die derzeit laufen. Der Stapel funktioniert nach dem LIFO-Prinzip (last in first out). Wird eine neue Aktivität gestartet, befindet sie sich oben auf dem Stapel. Navigiert der Nutzer zurück oder verlässt anderweitig die Vordergrundaktivität, wird die nächste Aktivität auf dem Stapel nach oben verschoben und ist jetzt aktiv [Meier2012, Seite 88]. Es ist auch möglich, dass aus der „ersten Aktivität heraus, eine weitere gestartet“ [Baltes-Götz2018, Seite 91] wird, welche dann auf dem obersten Punkt des Stapels liegt und die Möglichkeit hat, direkt mit dem Nutzer zu interagieren [Baltes-Götz2018, Seite 91]. Die genaue Funktionsweise eines solchen Activity-Stacks ist in Abbildung 1.4 zu sehen.

Innerhalb des Stapels können Aktivitäten vier Zustände durchlaufen. Während der Erstellung und der Zerstörung einer Aktivität bewegen sie sich in den Stapel hinein und aus dem Stapel heraus. Innerhalb der gesamten Lebensdauer einer Aktivität, also zwischen Ein- und Austreten aus dem Stapel, durchlaufen diese mehrere Zustände der aktiven und sichtbaren Lebenszeiten.

Der erste Zustand einer Aktivität ist der aktive Zustand. Steht eine Aktivität am oberen Rand eines Aktivitäten-Stapels, spricht man von einer aktiven, vordergründigen Aktivität. In diesem Zustand interagiert der Nutzer mit der App und der Zustand bleibt solange erhalten, bis die Aktivität aus dem Fokus tritt, beispielsweise durch Navigation zu einer anderen Aktivität.

Durch solch ein Unterbrechungsereignis verfällt die Aktivität in den pausierten Zustand. Die Aktivität ist zwar noch sichtbar aber nicht mehr unbedingt fokussiert. Dadurch wird die aktive Lebensdauer beendet.

Im gestoppten, angehaltenen Zustand ist die Aktivität für den Nutzer nicht mehr sichtbar. Die sichtbare Lebenszeit einer Aktivität endet hier. Da Aktivitäten öfter zwischen Vorder- und Hintergrund tauschen, ist es nicht unüblich, dass sie mehrere Phasen der sichtbaren Lebenszeiten innerhalb ihres Zyklus durchlaufen. In diesem Zustand wird das Activity-Objekt im Speicher gehalten [Meier2012, Seite 89]. Demnach kann die Aktivität entweder zurückkehren, um mit dem Nutzer zu interagieren, oder sie wird beendet [URL_7].

Sobald eine Aktivität beendet wird, ist sie in einem inaktiven Zustand, dies ist der Fall bevor sie gestartet wird und nach dem sie beendet wird. Eine inaktive Aktivität wird aus dem Stapel entfernt. Um erneut Teil des Stapels zu werden, müsste sie noch einmal gestartet werden [Meier2012, Seite 89].

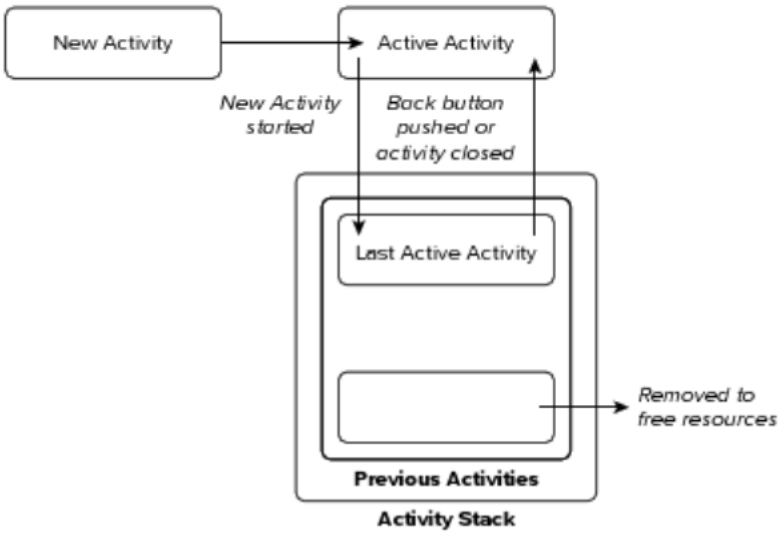

Abbildung 1.4: Activity-Stack [Meier2012]

1.4 Modi von Android

1.4.1 Recovery-Modus

In Kapitel 1.2.2 dieser Bachelorarbeit wird bereits die *recovery*-Partition erwähnt. Diese steht in enger Verbindung mit dem Recovery-Modus von Android. Neben dem Bootloader-Modus ist es unter Android möglich, in den Recovery-Modus zu booten. Dieser Modus stellt einen weiteren Betriebsmodus für Android dar [Hoog2011, Seite 27]. Unter dem Recovery-Modus versteht man den Wiederherstellungsmodus des Gerätes. Das Starten in den Wiederherstellungsmodus ist an den Bootloader gebunden. Der Bootloader eines Smartphones befindet sich in der ersten Partition. Dieser wird beim Einschalten des Smartphones ausgeführt und lädt standardmäßig das Android-ROM. Der Bootloader startet so entweder in den Recovery-Modus oder den Kernel. Der Kernel wiederum würde dann das normale Android-Betriebssystem starten [Morgillo2016, Seite 147].

In den Recovery-Modus kann man mithilfe einer für das Gerät spezifischen Tastenkombination oder mittels der Android Debug Bridge booten. Der ADB-Befehl lautet *adb reboot recovery* [Tamma2015, Seite 62].

Das Recovery-Image ist auf der Recovery-Partition abgelegt, welche auf dem internen Speicher liegt. Es besteht aus einem minimalen Linux-Image mit einfacher Benutzeroberfläche [Drake2014, Seite 63]. Startet man in diesen Modus, kann man mithilfe von Hardwarebuttons die Benutzeroberfläche dieses Mini-Betriebssystems steuern. Das Recovery-System ist als ein komplett eigenständiges System anzusehen. Es ist immer in der Lage, ein funktionierendes System wiederherzustellen, ungeachtet der Funktionsfähigkeit des Haupt-Android-Systems [Morgillo2016, Seite 170-176].

Das vom Hersteller installierte Stock-Recovery verfügt über eingeschränkte Funktionalität und ist auf den meisten Geräten sehr einfach gehalten. Ein Nachteil ist beispielsweise, dass keine Root-Privilegien in der Shell geboten werden [Hoog2011, Seite 272].

Allgemein kann man sagen, dass das Recovery einen Standardmechanismus in Android darstellt, mit welchem Softwareupdates erfolgen können oder auch die gesamte auf dem Gerät installierte Software ersetzt werden kann [Drake2014, Seite 63]. Neben Softwareupdates kann dieser Modus zudem dazu genutzt werden, das Gerät zu formatieren [Hoog2011, Seite 272] oder Benutzerpartitionen zu löschen [Elenkov2015, Seite 357].

Im Gegensatz zum Stock-Recovery bietet ein Custom-Recovery weitaus mehr Funktionalität. Es ersetzt das vom Hersteller installierte Stock-Recovery. Unter Custom-Recovery versteht man eine durch Drittanbieter entwickelte Wiederherstellungsmethode, welche nicht mit den Schlüsseln vom Hersteller signiert ist [Tamma2015, Seite 62]. Aus diesem Grund muss der Bootloader entsperrt vorliegen, um die Installation eines Recovery-Images zu ermöglichen [Tamma2015, Seite 62-65].

Die Entwicklung von Custom-Recoverys durch Drittanbieter ist nur möglich, da der Quellcode von Android offen und frei ist, so auch der Recovery-Quellcode [Morgillo2016, Seite 172]. Die meisten Custom-Recoverys bieten Funktionen wie das Erstellen eines vollen Backups, das Erlauben unsignierter Update-Pakete und das Erlauben signierter Update-Pakete sowie vollen ADB-Zugriff, wobei der ADB-Daemon mit Root-Berechtigungen läuft [Tamma2015, Seite 62-63].

1.4.2 Fastboot-Modus

Der Fastboot-Modus ist neben dem Recovery-Modus ein weiterer Modus unter Android. Fastboot ist ein in Adroid SDK integriertes Protokolldienstprogramm, welches für die Interaktion mit dem Bootloader verwendet werden kann [Hoog2011, Seite 100-102]. Das Fastboot-Protokoll wird für diese Interaktion vom Host gesteuert [Elenkov2015, Seite 351-356]. Die Verbindung geht vom Hostsystem aus und über die Kommandozeile können Befehle und Daten an den Bootloader gesendet werden [Elenkov2015, Seite 351-356]. In der Literatur wird anstelle des Fastboot-Modus auch oft vom Bootloader-Modus gesprochen.

Der Bootloader-Modus stellt einen bestimmten Zustand des Gerätes dar, der es ermöglicht, mit Hilfe des Fastboot-Tools beispielsweise ein Systemabbild auf das Gerät zu übertragen [Morgillo2016, Seite 112]. Jedes Gerät verfügt über solch einen Modus, ist aber nicht immer in der Lage, diesen Modus zu betreten, da bei vielen Geräten der Bootloader gesperrt ist und dieser zunächst freigeschaltet werden muss. Ist der Bootloader freigeschaltet, verfügt man über volle Kontrolle des NAND-Speichers [Morgillo2016, Seite 115].

Das Fastboot-Protokoll kann verwendet werden, um beispielsweise Partitionen auf einem Gerät neu zu flashen. Neben der Ausführung von Installationen und Updates, kann mittels des Fastboot-Protokolls auch der Bootloader entsperrt werden. Der Fastboot-Modus stellt eine Alternative zum Recovery-Modus dar und erlaubt eine weitere Möglichkeit Recovery-Images zu installieren [Tamma2015, Seite 140].

Um in diesem Modus zu starten ist das Drücken einer spezifischen Tastenkombination notwendig, welche je nach Hersteller und Gerät variieren kann. Es besteht außerdem die Möglichkeit, das Gerät über eine USB-Verbindung mit dem Computer zu verbinden und über die Konsole in den Fastboot-Modus zu booten [Dembowski2017, Seite 47-50]. Auf dem PC, der über USB mit dem Handy verbunden ist, ist die Installation der Fastboot-Tools über Android SDK notwendig. Weiterhin benötigt man verschiedene USB-Treiber für Windows-Computer [Jadhav2015, Seite 104].

Neben dem Fastboot- bzw. Bootloader-Modus wird in der Literatur häufig der Download-Modus erwähnt. Er ist die Bezeichnung eines Modus für Samsung-Geräte der Galaxy-Serie [Immler2016, Seite 40-44]. Dieser Modus ist dem Fastboot-Modus sehr ähnlich. Er kann bei Samsung-Geräten mit einer spezifischen Tastenkombination gestartet werden. Ist dies der Fall, wird der normale Android-Kernel-Bootprozess nicht gestartet [Drake2014, Seite 333-334].

1.4.3 Bootprozess

Neben dem Recovery-Modus und dem Fastboot-Modus besteht die dritte Möglichkeit darin, das Handy ganz normal zu booten, wie in Abbildung 1.5 verdeutlicht. Der Bootprozess kann in sieben Schritte unterteilt werden. Der erste Schritt wird durch das Drücken des Start-Knopfes erreicht. Die CPU ist nun mit Strom versorgt und der Boot-ROM-Code wird ausgeführt. Danach wird die Hardware initialisiert und der Boot-ROM-Code sucht das Bootmedium [Hoog2011, Seite 50-52]. Daraufhin wird der primäre oder auch initiale Bootloader in das interne RAM kopiert [Tamma2015, Seite 29]. Ist dieser Schritt erfolgreich abgeschlossen, werden die nächsten Aufgaben vom Bootloader übernommen [Hoog2011, Seite 50-52]. Er ist jetzt für das weitere Starten des Smartphones verantwortlich und bildet den zweiten Schritt des Bootprozesses. Kernel und Bootloader liegen getrennt vor. [Reibold2016, Seite 25].

Bei Android kann man den Bootloader in zwei Phasen unterteilen. Die erste Phase ist die initiale Programmladung (IPL) und die zweite Phase ist die zweite Programmladung (SPL) [Tamma2015, Seite 29]. Dieser zweite Schritt des Bootprozesses lässt sich wiederum in drei Schritte unterteilen. In der ersten Phase hat der initiale Bootloader die Aufgabe, das externe RAM zu erkennen und einzurichten [Tamma2015, Seite 29]. Sofern der erste Teil des Bootloaders diese Aufgabe erfüllt hat, wird der zweite Teil (SPL) in das

RAM kopiert. Die SPL wiederum übernimmt die Initialisierung verschiedener Hardwarekomponenten und hat die Aufgabe, das Betriebssystem zu laden [Tamma2015, Seite 29-30]. Als letzte Phase wird das Laden des Kernels in den Arbeitsspeicher verstanden. Nachdem die noch notwendigen Bootparameter geladen werden, wird die Ausführung an den Kernel übergeben und es folgt der dritte Schritt des Bootprozesses. Der Kernel steuert nun das Gerät, er ist für die Speicherverwaltung und das Prozessmanagement eines Systems verantwortlich [Tamma2015, Seite 30-31].

Als ersten Schritt liest der Kernel das Root-Dateisystem, er hat damit Zugriff auf die Benutzer- und Systempartition [Hoog2011, Seite 50-52]. Durch die Initialisierung von Speicher und Cache ist es möglich, User-Space-Prozesse zu starten [Tamma2015, Seite 30-31].

In dem Root-Dateisystem wird anschließend nach dem Init-Prozess gesucht. Sofern die Suche erfolgreich läuft, wird dieser gestartet. Der Init-Prozess ist immer der erste Prozess, welcher gestartet wird und bietet gleichzeitig die Grundlage dafür, dass andere Prozesse folgen können. Jeder andere „Prozess im System wird aus diesem Prozess oder aus einem seiner Abkömmlinge heraus gestartet“ [Reibold2016, Seite 26].

Der Init-Prozess, welcher die vierte Phase des Bootprozesses bildet, sucht nach dem *init.rc*-Skript, welches sich gewöhnlich in dem Root-Dateisystem befindet [Hoog2011, Seite 51-54]. Dieses Skript liefert dem Kernel Details über das Starten von Core Services. Es wird vom Init-Prozess eingelesen und die Systemdienstprozesse werden daraufhin ausgeführt [Hoog2011, Seite 51-54]. Neben Systemdiensten beschreibt das Skript *init.rc* auch noch das Dateisystem und weitere Parameter, die eingerichtet werden müssen [Tamma2015, Seite 30-32].

Im fünften und sechsten Schritt des Bootprozesses kommen die Dalvik VM und die Zygote zum Einsatz. Die Zygote ist eine der ersten Init-Prozesse, welche erstellt werden. Ihre Aufgabe besteht darin, die Dalvik Virtual Machine zu starten [Tamma2015, Seite 32-34]. Neben der Initialisierung der VM, werden mehrere Instanzen erstellt, um die verschiedenen Android-Prozesse zu unterstützen [Tamma2015, Seite 32-34].

Als siebter und letzter Schritt des Bootvorgangs wird der Systemserver gestartet. Dieser ist für das Starten von Android-Diensten verantwortlich [Reibold2016, Seite 24]. Der Systemserver übernimmt beispielsweise das Starten des Power-Managers oder des Activity-Managers [Tamma2015, Seite 32-37].

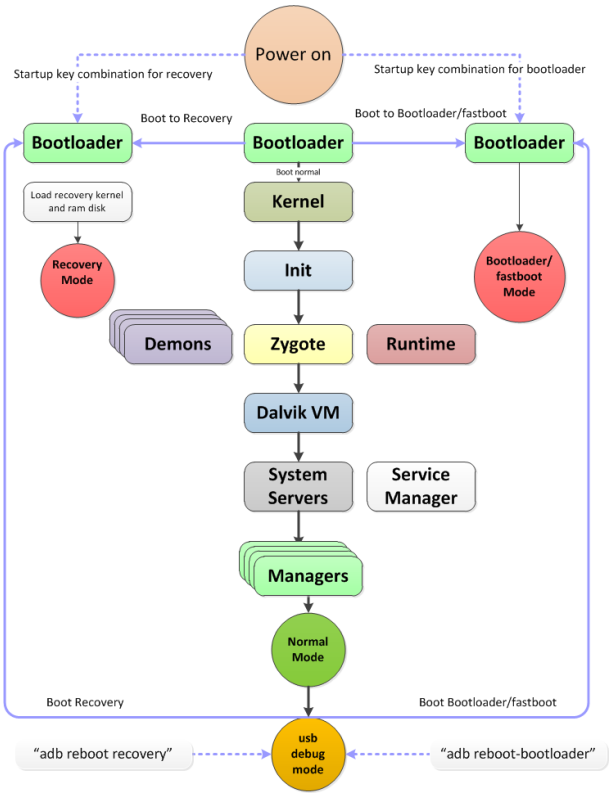

Abbildung 1.5: Bootprozess [URL_13]

1.5 Sicherungsmethoden

1.5.1 Logische Sicherung

Der Begriff der logischen Extraktion wird verwendet, wenn es um eine Extraktionsmethode geht, die keine vollständige, bitweise Kopie des Beweismaterials erbringt [Tamma2015, Seite 100]. Das bedeutet gleichzeitig auch, dass es mit dieser Sicherung nicht möglich ist, gelöschte Daten wiederherzustellen. Unter logischer Extraktion versteht man demnach das Extrahieren der auf dem Gerät vorhandenen Daten über den Zugriff auf das Dateisystem [Bommisetty2014, Seite 347]. Dabei werden alle „zugeordneten“, das heißt nicht gelöschten Daten, extrahiert [Hoog2011, Seite 218-266].

Es gibt jedoch auch Ausnahmen, bei denen auch gelöschte Daten in den extrahierten Daten miteingeschlossen sein können. Ein Beispiel können SQLite-Datenbanken sein, welche zugeordnet sind aber noch gelöschte Datensätze enthalten, die für den Nutzer auch bei einer logischen Extraktion zugänglich sind [Hoog2011, Seite196-197].

Logische Techniken sind in der Regel schneller durchzuführen als physikalische Sicherungsmethoden. Ferner sind diese einfacher zu bedienen und ermöglichen ebenfalls die Extraktion vieler wertvoller Daten. In der Regel benötigt man für die logische Extraktion

keinen Root-Zugriff. Verfügt man jedoch über Root-Berechtigungen, kann man gleichzeitig auch Zugriff auf anwendungsspezifische Daten erlangen, welche unter */data/data* liegen. Der Root-Zugriff beeinflusst hier die mögliche Menge der zu extrahierenden Daten [Bommisetty2014, Seite 347].

Die Durchführung logischer Extraktionen ist mittels *adb pull* oder Content Providern möglich. Die Definition der logischen Extraktion in der Mobilforensik unterscheidet sich von der, welche in der Computerforensik verwendet wird. Dort versteht man unter logischer Extraktion, dass nur Dateien und Verzeichnisse kopiert werden, auf die direkt zugegriffen werden kann [Tamma2015, Seite 99-102]. Das bedeutet gleichzeitig auch, dass beim Kopieren eines Ordners auf die dort gelöschten oder versteckten Dateien nicht zugegriffen werden kann [Tamma2015, Seite 100]. In der Mobil-Forensik ist die Grenze zwischen logischer und physikalischer Sicherung jedoch nicht so deutlich, da auch gelöschte Datensätze unter Umständen zugänglich sein können.

1.5.1.1 ADB-Pull

ADB steht für Android Debug Bridge und ist unter den Platform-Tools von SDK vorhanden [Tamma2015, Seite 48-49]. Man versteht darunter eine Kommandozeilenschnittstelle, welche in der Lage ist, direkt mit dem Android-Gerät zu kommunizieren [Annuzzi2015, Seite 469]. Die Android Debug Bridge ist ein Werkzeug, welches sowohl für den normalen Benutzer als auch für den Entwickler hilfreich sein kann [Rehberg2014, Seite 213]. Dieses Tool bietet verschiedene Möglichkeiten, um unterschiedliche Aktionen auf dem Smartphone ausführen zu können. Neben dem Installieren von Software auf dem Handy, können auch Daten auf die Workstation heruntergezogen werden.

Um mit diesem Werkzeug arbeiten zu können, müssen auf dem Handy zunächst die Entwickleroptionen eingeschaltet werden. Sobald diese aktiviert sind, wird der *adb*-Daemon (adbd) im Hintergrund ausgeführt und wartet auf USB-Verbindungen [Tamma2015, Seite 50]. Generell sind bei ADB drei Hauptkomponenten beteiligt. Der ADB-Daemon auf dem Gerät, der ADB-Server auf dem Rechner und das ADB-Client-Programm, welches ebenfalls auf der Workstation läuft. Das ADB-Client-Programm erlaubt das Senden von Befehlen an das Android-Gerät [Elenkov2015, Seite 277].

Die Kommunikation zwischen *adbd* und dem ADB-Server der Workstation erfolgt über ein virtuelles Netzwerk, welches über eine USB-Verbindung läuft [Hoog2011, Seite 100-102]. Auf gerooteten Handys läuft der *adbd* unter Root-Account, ansonsten nicht. Der ADB-Daemon und der ADB-Server kommunizieren bei der Verbindung über die Ports 5555 bis 5585 [Hoog2011, Seite 100-102]. Sobald ein neues Gerät mit der Workstation verbunden wird, werden zwei aufeinanderfolgende Port-Verbindungen erstellt [Hoog2011, Seite 100-102]. Dabei kommuniziert der gerade Port der beiden Portverbindungen mit der Gerätekonsole, während der ungerade Port für die ADB-Verbindung vorgesehen ist

[Tamma2015, Seite 50]. Das *adb*-Clientprogramm kommuniziert mit dem lokalen Server über Port 5037 [Hoog2011, Seite 100-102].

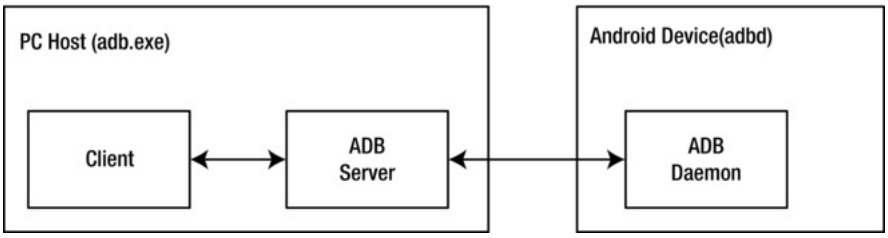

Abbildung 1.6: ADB Kommunikation [Regupathy2014]

Die ADB-Pull-Methode zählt zu den logischen Sicherungsmethoden und wird meist im Rahmen einer gezielten Untersuchung verwendet [Tamma2015, Seite 101]. Mit diesem Befehl können bestimmte Dateien oder Verzeichnisse vom Handy auf die Workstation kopiert werden. Bei dem Kopiervorgang ist zu beachten, dass die Verzeichnisstruktur erhalten bleibt. Datum und Uhrzeit des Inhaltes werden jedoch geändert, deshalb wird empfohlen, dass der Prüfer eine Kopie der Originalausgabe für einen späteren Datums- und Zeitvergleich aufbewahrt [Bommisetty2014, Seite 240-253]. Das *adb pull*-Kommando hat die folgende Syntax: *adb pull* <*remote*> <*local*>. <*Remote*> steht für den Pfad auf dem Android-Gerät und <*local*> für den der Workstation, auf welcher die zu sichernden Daten gespeichert werden sollen. Da *adbd* normalerweise nicht unter einem privilegierten Konto läuft, können mittels dieser Methode nicht alle Dateien auf die Workstation übertragen werden [Tamma2015, Seite 103-108]. Der Root-Zugriff ist die Voraussetzung, dass alle Nutzerdaten kopiert werden können.

1.5.1.2 ADB-Pull über Recovery-Modus

Wie in der letzten Methode beschrieben, kann die ADB-Pull Methode sehr effektiv sein. Um jedoch forensisch sauber arbeiten zu können und so wenige Daten wie möglich zu verändern, bietet sich die Nutzung eines Custom-Recoverys an. Wird die ADB-Pull-Methode während dem laufenden Betrieb eines Smartphones verwendet, können Daten dadurch verändert werden, dass beispielsweise Anwendungen im Hintergrund ausgeführt oder aktualisiert werden [Tamma2015, Seite 107].

Im Allgemeinen kann ADB im Stock-Recovery nicht verwendet werden, weshalb ein Custom-Recovery notwendig ist [Tamma2015, Seite 107; Hoog2011, Seite 273]. Ist bereits ein Custom-Recovery vorhanden, kann mit den richtigen Treibern, die auf der Workstation installiert sein müssen, auf das Gerät zugegriffen werden. Eine Überprüfung dessen, ob das Gerät erkannt wird, kann mittels des Befehls *adb devices* erfolgen. Auch diese Methode erfordert Root-Zugriff, wenn auf die Benutzerdaten unter */data/data* zugegriffen werden soll. Liegen die Daten auf dem zu untersuchenden Smartphone in verschlüsselter Form vor, ist diese Methode nutzlos, da kein Datenzugriff möglich ist [Tam-

ma2015, Seite 107-109]. Sie ist aber hilfreich, wenn man beispielsweise den Sperrbildschirm umgehen muss.

Verfügt das zu untersuchende Smartphone über das Stock-Recovery und nicht über ein Custom-Recovery, findet das Tool Fastboot Anwendung. Der Bootloader muss entschlüsselt vorliegen, ansonsten ist das Flashen eines neuen Recovery-Images nicht möglich [Tamma2015, Seite 64-66]. Ist die Installation eines modifizierten Recovery-Modes erfolgreich beendet, können die Daten mit *adb pull* gesichert werden.

1.5.2 Physikalische Sicherung

Bei den physikalischen Sicherungsmethoden wird zwischen hardwarebasierten und softwarebasierten Methoden unterschieden. Unter softwarebasierten Methoden versteht man Techniken, welche als Software auf dem Gerät mit Root-Zugriff laufen und ein vollständiges physikalisches Abbild der Datenpartition ermöglichen [Hoog2011, Seite 196-197]. Unter hardwarebasierten Techniken versteht man beispielsweise das Extrahieren von Gerätekomponenten [Hoog2011, Seite 268-270]. Hardwarebasierte Methoden kommen dann zum Einsatz, wenn auf dem Gerät kein Root-Zugriff möglich ist. Vertreter der hardwarebasierten Methoden sind die Chip-off-Methode und die JTAG-Methode.

Generell ist unter einer physikalischen Sicherung ein exaktes Abbild des Gerätes zu verstehen [Tamma2015, Seite 145-147]. In der Regel brauchen physikalische Erfassungsmethoden länger als logische Erfassungsmethoden. Erstere unterscheiden sich in dem Sinne von den Logischen, da sie eine komplette Kopie des Speichers darstellt, inklusive nicht zugeordnetem Speicherplatz und Dateischlupf. Da diese Methoden auf das physische Speichermedium abzielen, sind sie beim Datenzugriff nicht auf das Dateisystem angewiesen [Hoog2011, Seite 267-270].

Physikalische Techniken ermöglichen den Zugriff auf gelöschte, versteckte und veraltete Daten im System, denn viele Daten werden nur als gelöscht markiert, sind aber noch vorhanden. Man kann hier also auf weitaus mehr Daten zugreifen als bei den logischen Extraktionsmethoden. Zudem ist es durch einige physikalische Methoden möglich, den Schutz des Passcodes zu umgehen [Hoog2011, Seite 267-270]. Ein Nachteil der physikalischen Extraktionsmethoden ist, dass Fehler in den Techniken das Gerät unbrauchbar machen können.

2 Methodenteil

2.1 Versuchsreihe 1

Für die folgenden Untersuchungen wird ein Samsung Galaxy S4 mini mit der Android-Version 4.4.4 verwendet. Die Modellnummer des Gerätes lautet GT I9195I. Zum Zeitpunkt der Untersuchung ist bereits das Custom-Recovery „Team Win Recovery Project“ auf dem Smartphone installiert. Es ist zu beachten, dass sich die gemessenen Zeiten in den Szenarien auf die Handy-Uhr beziehen. Die Sicherung der zu untersuchenden Ordner erfolgt mittels *adb pull*. Bevor das erste Szenario beginnt, wird ein Factory Reset durchgeführt, um eine forensische Ausgangsbasis zu schaffen. Der erste zu sichernde Ordner befindet sich im Ordner *system* auf der Datenpartition und heißt *usagestats*. Ein weiterer Ordner nennt sich *com.whatsapp* und befindet sich ebenfalls auf der Datenpartition in dem Ordner *data*. Der dritte Ordner *recent_images*, den es in dieser Bachelorarbeit zu untersuchen gilt, ist unter dieser Betriebssystemversion nicht vorhanden.

2.1.1 Szenario 1

Das erste Szenario beginnt am 01.07.2018. Zwei Tage zuvor wird ein Factory Reset durchgeführt. Nachdem das Handy am 01.07.2018 gestartet wird, werden einige Standardanwendungen geöffnet. Dabei fällt auf, dass ein Datum vom 03.01.2014 angezeigt wird und ebenso eine nicht passende Uhrzeit. Deshalb wird das Handy zunächst einmal mit dem WLAN verbunden und die Uhrzeit des Handys mit dem Netzwerk synchronisiert. Daraufhin zeigt das Handy ein Datum vom 01.07.2018 und die tatsächliche Uhrzeit an. Das Gerät wird anschließend heruntergefahren und in den TWRP-Modus gebootet.

2.1.1.1 Protokolldatei in *usagestats*

Aufbau

In dem Ordner *usagestats* existiert eine XML-Datei, welche den Namen *usage-history* trägt. Der Aufbau dieser Datei lässt sich folgendermaßen beschreiben. Die *XML*-Datei beginnt mit einem Wurzelelement *usage-history*, welchem keine weiteren Attribute anhängen. Ein Kind-Knoten dieses Wurzelelementes bildet das Element *pkg*. Dieses wiederum besitzt ein Attribut *name*, dem ein Paket- bzw. Anwendungsname übergeben wird. Ein *pkg*-Element hat seinerseits einen Kind-Knoten, der den Namen *comp* besitzt. Dem *comp*-Element werden die Attribute *name* und *lrt* übergeben. Des Weiteren nimmt das Attribut *lrt* eine 13-stellige Zahl an.

Neben dieser *XML*-Datei befinden sich noch weitere Dateien in diesem Ordner. Alle Namen dieser Dateien weisen dieselbe Syntax auf: *usage-YYYYMMDD*.

Analyse

Nachdem die Daten in dem Ordner *usagestats* gesichert werden, können neben der *usage-history.xml* noch weitere Dateien gefunden werden. Eine Datei trägt einen Zeitstempel vom 01.01.2014, eine vom 03.01.2014 und eine andere vom 01.07.2018. In der Datei vom 01.01.2014 und 03.01.2014 sind alle Ausführungen des Nutzers aufgelistet, welche betätigt wurden, bevor die Umstellung des Datums und der Zeit stattgefunden hat. In der Datei vom 01.07.2018 wiederum sind alle Ausführungen des Nutzers aufgelistet, nachdem die Umstellung der Zeit vollzogen wurde. Eine genauere Analyse der Dateien findet in den folgenden Szenarien statt.

Ergebnis

Die Zeitstempel, welche in der *usage-history*-Datei protokolliert sind, beziehen sich auf die eingestellte Handy-Zeit. Der Zeitstempel 01.01.2014 zeigt in dieser Betriebssystemversion das Werkseinstellungs-Datum des Gerätes an. Erst nach dem Synchronisieren der entsprechenden Zeitzone erfolgt die Angabe des tatsächlichen Datums/Uhrzeit. Die Dateien vom 01.01.2014 und 03.01.2014 bedeuten nicht, dass alte Daten auf dem Smartphone vorhanden sind. Das erste Datum vom 01.01.2014 zeigt lediglich die erste Nutzung nach dem Factory Reset an während das Datum vom 03.01.2014 das erneute Nutzen des Handys zwei Tage später wiedergibt. Es handelt sich um die Daten, welche nach dem Factory-Reset, jedoch vor der Synchronisation entstanden sind.

2.1.1.2 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

In dem nächsten Szenario wird WhatsApp heruntergeladen und genauer untersucht.

2.1.2 Szenario 2

Vom 02.07.2018 bis zum 04.07.2018 werden verschiedene Versuche mit dem Handy durchgeführt. Die entsprechenden Dateien werden anschließend gesichert und ausgewertet. Am 02.07.2018 wird WhatsApp mit der Version 2.18.191 heruntergeladen. Am selben Tag wird ein Kontakt bei WhatsApp hinzugefügt und der Reiter „Einstellungen“ von WhatsApp betätigt. Am 03.07.2018 wird die Rechner-Anwendung auf dem Handy gestartet. Dies geschieht um 13:45 Uhr. Anschließend wird ein Bild mit der Kamera aufgenommen und um 13:46 Uhr wird die PlayStore-Anwendung geöffnet und anschließend wieder geschlossen. Am 04.07.2018 werden mehrere Aktivitäten seitens des Nutzers am Handy unternommen, um die Daten in der Protokolldatei nachvollziehen zu können. Bei

der Beschreibung und Analyse wird im Folgenden des öfteren das Wort „Aktivitäten“ verwendet. Damit werden die Tätigkeiten beschrieben, die durch den Nutzer am Handy ausgeführt werden. Zunächst erfolgt um 13:45 Uhr der Eintritt in das WhatsApp-Hauptmenü. Anschließend wird um 13:47 Uhr der Reiter „Einstellung“ unter WhatsApp angeklickt. Eine Minute später wird das Profil von WhatsApp angesehen. Um 13:50 Uhr wird die Einstellungs-Anwendung geöffnet und dort der Bluetooth-Reiter betreten. Danach wird das Smartphone ausgeschaltet. Im Folgenden wird die Protokolldatei vom 04.07.2018 genauer betrachtet und analysiert.

Untersucht wurden folgende Dateien (**Details siehe Anlage Versuchsreihe 1, Szenario 2**)

2.1.2.1 Protokolldatei in *usagestats*

Ergebnis

In diesem Szenario wird deutlich, was in der Protokolldatei *usage-history* protokolliert wird. Zum einen kann festgehalten werden, dass die Zeit, welche in der Protokolldatei gespeichert wird, in Millisekunden gespeichert ist. Diese Zeit bezieht sich auf die Unix-Epoch-Zeit. Des Weiteren gibt das Attribut *name* des Elementes *pkg* den Namen einer Anwendung an, welche auch unter */data/data/* zu finden ist. Darüber hinaus kann festgestellt werden, dass das Attribut *name* des *comp*-Elementes die einzelnen Ansichten der jeweiligen Anwendung speichert. Der Zeitstempel, der zu diesen Ansichten gespeichert wird, gibt nach dieser Analyse an, wann diese Ansicht zuletzt benutzt wurde. Alle anderen Dateien im Ordner *usagestats*, die einen Namen der Form *usage-YYYYMMDD* tragen, speichern die Aktivitäten, welche an dem besagten Tag betätigt wurden, jedoch ohne Zeitstempel.

2.1.2.2 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Ergebnis

In der Protokolldatei werden in diesem Szenario zwei interessante Ereignisse entdeckt. Das erste scheint nach bisherigen Erkenntnissen den Batteriestatus über einen Tag hinweg zu protokollieren, während das zweite Ereignis nach ersten Beobachtungen sehr wahrscheinlich darüber Auskunft gibt, wann der Bildschirm zuletzt entsperrt wurde. Genaueres wird im nächsten Szenario geklärt.

2.1.3 Szenario 3

Ziel dieses Szenarios ist es herauszufinden, wie lange die Dateien im Ordner *usagestats* gespeichert werden. Außerdem soll festgestellt werden, was genau die BatteryState- und die ScreenLockReceiver-Ereignisse protokollieren.

Dafür wird am 05.07.2018 das Handy angeschaltet. Um 09:17 Uhr wird es über ein USB-Kabel mit dem Computer verbunden und der Bildschirm wird in derselben Minute entsperrt. Zwei Minuten später erfolgt eine erneute Entsperrung des Bildschirms und die USB-Verbindung wird entfernt. Gegen 09:21 Uhr wird das Handy mit einem Netzteil-Ladegerät verbunden. Eine Minute später wird das Ladegerät wieder entfernt. Um 09:23 Uhr wird der Bildschirm erneut entsperrt und es wird eine Nachricht an einen Kontakt über WhatsApp versendet. Das Handy bleibt vom 05.07 bis zum 08.07 in einem angeschalteten Zustand.

Untersucht wurden folgende Dateien **(Details siehe Anlage Versuchsreihe 1, Szenario 3)**

2.1.3.1 Protokolldatei in *usagestats*

Ergebnis

Es kann in diesem Szenario bestätigt werden, dass die Zeitstempel der *usage-history*-Datei immer den letzten Zeitpunkt speichern, an dem eine bestimmte Komponente der Anwendung verwendet wird. Außerdem werden keine der *usage-YYYYMMDD*-Dateien länger als sieben Tage in dem Ordner aufbewahrt. Darüber hinaus wird nur an den Tagen eine *usage-YYYYMMDD*-Datei angelegt, an denen das Handy in Benutzung ist.

2.1.3.2 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Ergebnis

In diesem Szenario wird deutlich, welche Werte das Attribut *plugged* des Batteriestatus annehmen kann und welche Bedeutung diese Werte haben. Der Wert „0“ des Attributes *plugged* bedeutet, dass das Handy mit keinem Ladegerät oder dergleichen verbunden ist. Ein Wert von „1“ nimmt *plugged* an, wenn das Handy mit einem Netzteil-Ladegerät verbunden ist. Der letzte Wert, den *plugged* annehmen kann, ist „2“. Dieser wird angenommen, wenn das Handy mit einem USB-Kabel verbunden ist.

Neben der Auswertung der BatteryState-Ereignisse konnten ebenfalls Erkenntnisse zu den ScreenLockReceiver-Ereignissen gewonnen werden. Dabei wird das Attribut *tag* genauer betrachtet. Dieses kann die Werte „on“, „off“ und „unlock“ annehmen. Der Wert

„unlock“ wird gemäß dieses Szenarios immer dann angenommen, wenn der Bildschirm des Handys entsperrt wird. Außerdem wird der Wert „on“ angenommen, wenn der Bildschirm aufleuchtet, aber nicht entsperrt wird. „Off“ gibt an, dass der Bildschirm gesperrt ist.

Dieses Szenario zeigt außerdem, dass die WhatsApp-Protokolldatei im Laufe der Zeit beginnt, alte Ereignisse zu löschen, um Platz für neue Ereignisse zu schaffen. Ein weiteres Ergebnis dieser Versuchsreihe ist, dass die BatteryState-Ereignisse in gewissen Zeitabständen angelegt werden. Dabei spielt es keine Rolle, ob der Nutzer aktiv am Handy ist oder nicht. Diese Ereignisse werden auch nachts angelegt, wenn keine Interaktion mit dem Benutzer stattfindet. Es scheint nach bisherigen Erkenntnissen so zu sein, dass diese Ereignisse stündlich protokolliert werden.

2.2 Versuchsreihe 2

Zur Untersuchung liegt ein Samsung Galaxy S4 mini mit der Modellnummer GT-I9195I vor. Am 19.07.2018 wird über TWRP ein neues Custom-ROM auf das Handy geflasht. Vor der Installation dieses Custom-ROMs wird ein vollständiger Wipe (Factory Reset) durchgeführt. Version 14.1 von LineageOS wird installiert, welche der Android-Version 7.1.2 entspricht. Neben der neuen Betriebssystemversion werden ebenfalls auch Google-App-Pakete heruntergeladen, um später den Google-PlayStore nutzen zu können. Bei dem Download der Google-Pakete muss darauf geachtet werden, dass der richtige Prozessortyp ausgewählt wird. Im Falle dieses Handys ist es der Prozessortyp „ARM“. Über den „Install“-Reiter von TWRP werden die Dateien der beiden zip-Ordner installiert. Am 19.07.2018 wird Szenario 1 durchgeführt. Es ist anzumerken, dass die gemessenen Zeiten, die im folgenden Szenario beschrieben sind, zum einen mit der Uhr des Handys und zum anderen mit der Epoch-Zeit[1] gemessen werden. Die Daten werden über *adb pull* gesichert und anschließend analysiert. Die erste Protokolldatei liegt auf der Datenpartition unter dem Ordner *system* und heißt *usagestats*. Der komplette *com.whatsapp*-Ordner wird vollständig gesichert, dieser liegt unter */data/data*. Zu guter Letzt wird ein Ordner auf der Datenpartition gesichert, welcher einen Unterordner von dem Ordner *system_ce* bildet und *recent_images* heißt.

Ein Unterschied zu Versuchsreihe 1 liegt darin, dass der Nutzer vor dem vollständigen Laden des Betriebssystems bereits nach der Netzwerkverbindung gefragt wird. Aus diesem Grund ist nach vollständigem Start die richtige Uhrzeit eingestellt. Dies ist bei Version 4.4.4 nicht so, hier entfällt die Abfrage der Netzwerkverbindung, folglich weist die Uhrzeit das Werkseinstellungsdatum auf.

[1] Seit 1.Januar 1970

2.2.1 Szenario 1

Die folgende Beschreibung der Uhrzeiten erfolgt mittels der Handy-Uhr. Im weiteren Verlauf dieses Szenarios werden diese Zeiten mit den gemessenen Zeiten über die Epoch-Zeit verglichen. Am 19.07.2018 um 17:46 Uhr wird WhatsApp mit der Version 2.18.217 auf dem Handy installiert. Um 17:51 Uhr wird die Chat-Ansicht bei WhatsApp geöffnet. Eine Minute später wird ein Profilfoto bei WhatsApp geöffnet. Um 17:53 Uhr wird die Ansicht des Profilfotos verlassen und zur Konversation zurückgekehrt. Um 17:53 Uhr wird ein Kontakt bei WhatsApp hinzugefügt und eine Minute später wird auf den Reiter „Status“ gewechselt. Um 17:55 Uhr werden die Einstellungen bei WhatsApp geöffnet und in derselben Minute wieder verlassen. Danach wird das Handy heruntergefahren und in den Recovery-Mode gestartet.

2.2.1.1 Protokolldateien in */data/system/usagestats/0/*

In diesem Ordner auf der Datenpartition befinden sich vier Unterordner und eine Datei. Die Unterordner heißen „daily“, „weekly“, „monthly“ und „yearly“. In dem Ordner „daily“ wiederum befinden sich zwei *XML*-Dateien, die jeweils eine 13-stellige Zahl als Dateinamen tragen.

In dem Ordner „weekly“ befinden sich zwei *XML*-Dateien, die ebenfalls eine Zahl als Dateinamen aufweisen. Im Ordner „monthly“ und „yearly“ befinden sich auch jeweils zwei solcher *XML*-Dateien.

In den nächsten Tabellen folgt eine Übersicht über die vorhandenen Dateien in den jeweiligen Ordnern.

***XML*-Dateien des Ordners „daily“:**

Tabelle 2.1: Versuchsreihe 2, Szenario 1 XML-Dateien in „daily“ [Quelle: eigene Tabelle]

Dateiname	**Zeitstempel**
1532013966190	19. Juli 2018 17:26:06.190
1532014304413	19. Juli 2018 17:31:44.413

***XML*-Dateien des Ordners „weekly“:**

Tabelle 2.2: Versuchsreihe 2, Szenario 1 XML-Dateien in „weekly“ [Quelle: eigene Tabelle]

Dateiname	**Zeitstempel**
1532013966190	19. Juli 2018 17:26:06.190
1532014304413	19. Juli 2018 17:31:44.413

***XML*-Dateien des Ordners „monthly“:**

Tabelle 2.3: Versuchsreihe 2, Szenario 1 XML-Dateien in „monthly“ [Quelle: eigene Tabelle]

Dateiname	**Zeitstempel**
1532013966190	19. Juli 2018 17:26:06.190
1532014304413	19. Juli 2018 17:31:44.413

***XML*-Dateien des Ordners „yearly“:**

Tabelle 2.4: Versuchsreihe 2, Szenario 1 XML-Dateien in „yearly“ [Quelle: eigene Tabelle]

Dateiname	**Zeitstempel**
1532013966190	19. Juli 2018 17:26:06.190
1532014304413	19. Juli 2018 17:31:44.413

2.2.1.2 Protokolldateien in */data/system/usagestats/0/daily/*

Aufbau

Diese Datei besteht aus einem Wurzelelement, welches den XML-Namen *usagestats* trägt. Weiter werden diesem Wurzelelement die Attribute *endTime* und *version* übergeben. Dem Wurzelelement/Wurzelknoten sind drei Kind-Knoten untergeordnet. Die Kind-Knoten heißen *packages*, *configurations* und *event-log*.

Das Element *packages* hat weitere Kind-Knoten, die den XML-Namen *package* tragen. Dem Kind-Knoten *package* werden die Attribute *lastEvent*, *timeActive*, *package* und *lastTimeActive* zugeordnet. Das Element *configurations* hat ebenfalls Kind-Knoten mit dem Namen *config*. Dem *config*-Element werden die Attribute *timeActive*, *lastTimeActive*, *density*, *sw*, *height*, *width*, *ui*, *scrLay*, *ori*, *navHid*, *nav*, *hardKeyHid*, *keyHid*, *key*, *touch*, *locales*, *mnc*, *mcc*,*fs*,*active* und *count* zugeordnet.

Dem Knoten *event-log* ist wiederum ein Kind-Knoten mit dem Namen *event* untergeordnet. Diesem Kind-Knoten werden die Attribute *package*, *type*, *time* und *class* zugeordnet. Der Aufbau der anderen *XML*-Dateien in diesem Ordner ist analog. In Abbildung 2.1 ist der Aufbau dieser XML-Dateien noch einmal aufgeführt.

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<usagestats endTime="1438817" version="1">
  - <packages>
      <package lastEvent="2" timeActive="1961" package="com.google.android.googlequicksearchbox" lastTimeActive="368618"/>
      <package lastEvent="0" timeActive="0" package="com.android.providers.calendar" lastTimeActive="-1532014304413"/>
      <package lastEvent="0" timeActive="0" package="com.android.providers.media" lastTimeActive="-1532014304413"/>
      <package lastEvent="2" timeActive="469144" package="com.whatsapp" lastTimeActive="1427032"/>
      <package lastEvent="0" timeActive="0" package="com.android.providers.downloads" lastTimeActive="-1532014304413"/>
      <package lastEvent="0" timeActive="0" package="com.google.android.apps.messaging" lastTimeActive="999864"/>
      <package lastEvent="0" timeActive="0" package="com.android.defcontainer" lastTimeActive="-1532014304413"/>
      <package lastEvent="2" timeActive="19126" package="com.cyanogenmod.setupwizard" lastTimeActive="393281"/>
      <package lastEvent="2" timeActive="152396" package="com.android.vending" lastTimeActive="903079"/>
      <package lastEvent="2" timeActive="11388" package="android" lastTimeActive="1429521"/>
      <package lastEvent="0" timeActive="0" package="com.google.android.gm" lastTimeActive="1050214"/>
      <package lastEvent="2" timeActive="60504" package="com.google.android.setupwizard" lastTimeActive="393578"/>
      <package lastEvent="0" timeActive="0" package="com.google.android.music" lastTimeActive="-1532014304413"/>
      <package lastEvent="0" timeActive="0" package="com.android.printspooler" lastTimeActive="-1532014304413"/>
      <package lastEvent="2" timeActive="21606" package="com.google.android.contacts" lastTimeActive="1345276"/>
      <package lastEvent="0" timeActive="0" package="com.google.android.syncadapters.contacts" lastTimeActive="-1532014304413"/>
      <package lastEvent="2" timeActive="308504" package="com.google.android.gms" lastTimeActive="751514"/>
      <package lastEvent="0" timeActive="0" package="com.google.android.gsf" lastTimeActive="-1532014304413"/>
      <package lastEvent="0" timeActive="0" package="com.google.android.tts" lastTimeActive="-1532014304413"/>
      <package lastEvent="2" timeActive="12171" package="com.android.packageinstaller" lastTimeActive="1029450"/>
      <package lastEvent="0" timeActive="0" package="com.google.android.videos" lastTimeActive="-1532014304413"/>
      <package lastEvent="2" timeActive="136643" package="com.google.android.apps.nexuslauncher" lastTimeActive="1438817"/>
      <package lastEvent="0" timeActive="0" package="com.google.android.backuptransport" lastTimeActive="-1532014304413"/>
      <package lastEvent="2" timeActive="12310" package="com.android.settings" lastTimeActive="20048"/>
      <package lastEvent="0" timeActive="0" package="com.android.providers.blockednumber" lastTimeActive="-1532014304413"/>
      <package lastEvent="0" timeActive="0" package="com.android.providers.userdictionary" lastTimeActive="-1532014304413"/>
      <package lastEvent="0" timeActive="0" package="com.google.android.play.games" lastTimeActive="-1532014304413"/>
      <package lastEvent="0" timeActive="0" package="com.android.providers.contacts" lastTimeActive="-1532014304413"/>
      <package lastEvent="0" timeActive="0" package="com.google.android.inputmethod.latin" lastTimeActive="-1532014304413"/>
    </packages>
  - <configurations>
      <config timeActive="786591" lastTimeActive="1023538" density="240" sw="360" height="616" width="360" ui="17" scrLay="268435810" ori="1" navHid="2"
          locales="de-DE" mnc="2" mcc="262" fs="1065353216" active="true" count="2"/>
      <config timeActive="21663" lastTimeActive="1023537" density="240" sw="360" height="336" width="640" ui="17" scrLay="268435810" ori="2" navHid="2" r
          DE" mnc="2" mcc="262" fs="1065353216" count="2"/>
    </configurations>
  - <event-log>
      <event package="com.google.android.setupwizard" type="2" class="com.google.android.setupwizard.network.NetworkActivity" time="7707"/>
      <event package="com.android.settings" type="1" class="com.android.settings.wifi.WifiDialogActivity" time="7738"/>
      <event package="com.android.settings" type="2" class="com.android.settings.wifi.WifiDialogActivity" time="20048"/>
      <event package="com.google.android.setupwizard" type="1" class="com.google.android.setupwizard.network.NetworkActivity" time="20189"/>
      <event package="com.google.android.setupwizard" type="2" class="com.google.android.setupwizard.network.NetworkActivity" time="26555"/>
      <event package="com.google.android.setupwizard" type="1" class="com.google.android.setupwizard.WizardManagerActivity" time="26645"/>
```


Abbildung 2.1: Versuchsreihe 2, Szenario 1 Ausschnitt *XML*-Datei [Quelle: eigene Abbildung]

Analyse

Im Folgenden werden die Auffälligkeiten anhand der Datei *1532014304413.xml* dokumentiert.

Es ist zu erkennen, dass das Attribut *package* des Knoten *package* nur Paketnamen annimmt, welche sich auch auf der Datenpartition im Ordner *data* befinden. Weiterhin enthält das Attribut *lastTimeActive* des Knotens *package* bei manchen Einträgen negative Werte. Diese negativen Werte haben denselben Zahlenwert, den auch der Dateiname der *XML*-Datei trägt.

Es wird nun getestet, in welchem Zusammenhang die Zeitwerte, die in der Logdatei protokolliert sind, mit dem Namen der *XML*-Datei stehen. Dafür wird das Attribut *lastTimeActive*, welches den *package*-Elementen zugeordnet ist, auf den Dateinamen der besagten *XML*-Datei addiert. Der Zeitstempel, der sich daraus ergibt, wird mit den Zeiten im oben ausgeführtem Szenario abgeglichen. Die Ergebnisse sind in Tabelle 2.5 aufgeführt.

Tabelle 2.5: Versuchsreihe 2, Szenario 1 Vergleich Zeitstempel [Quelle: eigene Tabelle]

Attribut *package*	**Attribut *lastTimeActive***	**Dateiname**	**Dateiname + *lastTimeActive***	**Handy-Zeit**	**Epoch-Zeit**
com.whatsapp	1427032	1532014304413	17:55:31	17:55	17:55:31
com.google.android.contacts	1345276	1532014304413	17:54:09	17:53	17:53:59
Android	1429521	1532014304413	17:55:33	17:55	17:55:33

Gemäß dieser Tabelle hat sich der Eintrag in WhatsApp das letzte Mal aktualisiert, als nach oben beschriebenem Szenario der Reiter „Einstellungen“ benutzt wurde. Weiter wurde der Zeitstempel des Paketes „com.google.android.contacts“ das letzte Mal aktualisiert als ungefähr zeitgleich im Szenario der Kontakt bei WhatsApp hinzugefügt wurde. Ferner ergab die Berechnung des Zeitstempels von dem *package*-Element mit dem Attributwert „android“ des Attributes *package* eine Zeit, die nach Ablauf des Szenarios der entspricht, als das Handy heruntergefahren wurde. Des Weiteren ist auffällig, dass alle *package*-Elemente, die einen negativen Wert bei *lastTimeActive* haben, auch eine „0“ bei dem Attribut *lastEvent* haben.

Rechnet man den Wert des Attributes *endTime*, welches ein Attribut des Wurzelelementes darstellt, auf den Dateinamen hinzu, ergibt sich eine Zeit von 17:55:43 Uhr des 19. Juli 2018. Der Wert, welcher in *endTime* gespeichert wird, findet sich noch an zwei weiteren Stellen in der Logdatei. Das *package*-Element, welches den Paketnamen „com.google.android.apps.nexuslauncher“ trägt, zeigt ebenfalls diesen Wert bei dem Attribut *lastTimeActive*. Außerdem taucht dieser ebenfalls bei einem *event*-Element auf, welches auch den Paketnamen „com.google.android.apps.nexuslauncher“ trägt und gleichzeitig den letzten *event*-Eintrag in dieser Datei darstellt.

Neben dem Attribut *package* enthalten die *event*-Elemente noch das Attribut *class*. Dieses beginnt immer mit dem Wert des Attributes *package* des Elements. Außerdem ist für dieses Element mit dem Attribut *time* eine Zeit angegeben. Dieses *time*-Attribut nimmt hier keine negativen, sondern ausschließlich positive Werte an. In der folgenden Tabelle werden die Zeiten der Protokolldatei, die das *event*-Element abspeichert, mit den Zeiten des Szenarios verglichen.

Tabelle 2.6: Versuchsreihe 2, Szenario 1 Gegenüberstellung Zeitwerte [Quelle: eigene Tabelle]

Attribut ***class***	**Attribut** ***time***	**Attribut** ***type***	**Dateiname** ***+ time***	**Handy-Zeit**	**gemessene Epoch-Zeit**	**Szenario**
com.whatsapp. Conversation	1159674	1	17:51:04	17:51	17:52:03	Konversation in Whats-App betreten
com.whatsapp .Conversation	1227452	2	17:52:11	17:52	17:52:11	Konversation verlassen, um Kontaktinfo anzusehen
com.whatsapp .ContactInfo	1227499	1	17:52:11	17:52	17:52:11/ 17:52:12	Ansehen der Kontaktinfo
com.whatsapp .ContactInfo	1284407	2	17:53:08	17:53	17:53:08	Kontaktinfo verlassen
com.whatsapp .Conversation	1284441	1	17:53:08	17:53		nach Kontaktansicht zurück auf Chat
com.whatsapp .Conversation	1323271	2	17:53:47	17:53	17:53:44	Chat-Ansicht verlassen
com.whatsapp .Settings	1399802	1	17:55:04	17:55	17:55:03	Reiter „Einstellungen“ geöffnet
com.whatsapp .Settings	1427032	2	17:55:31	17:55	17:55:31	Reiter „Einstellungen “ verlassen

Für den Vergleich in der obigen Tabelle wurden nur Zeiten verwendet, die möglichst genau und gezielt gemessen werden konnten. Deshalb wurde alles, was vor der ersten Chat-Konversation von WhatsApp protokolliert wurde, nicht beachtet, da es hierzu keine gemessenen Zeiten gab. Man sieht, dass die gemessene Epoch-Zeit exakt oder höchstens mit einer Sekunde Unterschied auf die protokollierte Zeit in der Logdatei passt. Es wurde gemäß oben ausgeführtem Szenario erstmals ein *event*-Element mit dem Wert „com.whatsapp.Conversation“ des Attributes *class* angelegt, als nach Ablauf des Szenarios die Chatansicht eines Kontaktes besucht wurde. Dieser *event*-Knoten ist in oben aufgeführter Abbildung (Abbildung 2.1) nicht zu erkennen, da dort nur ein Ausschnitt der *XML*-Datei gezeigt wird. Nach diesem ersten Kontakt hat das *event-log* mehrere solcher Ereignisse dokumentiert, da im weiteren Verlauf immer wieder diese Ansicht besucht

wurde.

Eine Auffälligkeit ergibt sich durch den Vergleich der *package*- und *event*-Elemente. Das *package*-Element, welches den Attributwert „com.whatsapp“ bei dem Attribut *package* aufweist, hat einen Attributwert „1427032“ bei dem Attribut *lastTimeActive*. Derselbe Wert findet sich bei einem *event*-Element wieder. Dieses *event*-Element stellt den letzten Eintrag zum *com.whatsapp*-Paket in dieser Logdatei dar. Weiterhin wird diesem Element durch das Attribut *class* der Name „com.whatsapp.Settings“ gegeben. Gleicht man dies mit dem oben beschriebenen Szenario ab, stellt man fest, dass WhatsApp das letzte Mal benutzt wurde, als der Reiter „Einstellung/Settings“ bei WhatsApp besucht wurde. Generell ist zu sagen, dass nur die Pakete in den *event*-Elementen auftreten, die unter den *package*-Elementen bei dem Attribut *lastTimeActive* einen positiven Wert haben, also aktiv waren.

Den *event*-Elementen wird durch das Attribut *type* ein Wert von „2“, „1“ und „7“ zugeordnet werden kann. Dabei besitzen alle *event*-Elemente, denen beim Attribut *type* ein Wert von „7“ zugeordnet wird, kein *class*-Attribut.

Die zu untersuchende *XML*-Datei weist zwei *config*-Elemente auf. Die Werte ihrer Attribute stimmen teilweise überein, unterscheiden sich aber auch in einigen Werten, wie in Abbildung 2.2 zu sehen ist.

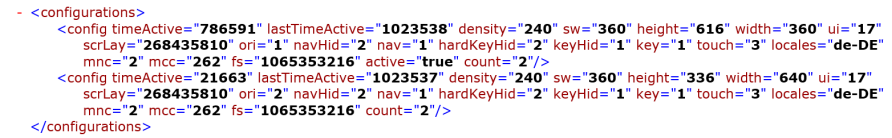

Abbildung 2.2: Versuchsreihe 2, Szenario 1 *config*-Elemente [Quelle: eigene Abbildung]

Neben der *1532014304413.xml* liegt noch die *1532013966190.xml*-Datei im „daily“-Ordner. Es kann festgestellt werden, dass die *1532013966190.xml*-Datei das letzte Ereignis um 17:31:44 Uhr protokolliert, während die *1532014304413.xml* gemäß ihrem Dateinamen um 17:31:44 Uhr anfängt zu protokollieren.

Ergebnis

Die Analyse der Logdatei zeigt, dass zwischen dem Ablauf des Szenarios und den Daten in der Logdatei ein Zusammenhang besteht. Bisherige Untersuchungen haben ergeben, dass sich die Zeitstempel in der Logdatei immer auf den Namen der Logdatei beziehen. Das heißt die Zeiten werden in der Logdatei immer auf den Zeitstempel, der durch den Namen der Logdatei gegeben ist, addiert, um auf die entsprechenden Zeiten im Szenario zu kommen.

Weiterhin kann man davon ausgehen, dass das Attribut *lastTimeActive* der *package*-Elemente die Zeit notiert, an dem eine Applikation das letzte Mal genutzt wurde. Die *event*-Elemente scheinen eine genauere Protokollierung dessen vorzunehmen, was in der Applikation ausgeführt wurde. Unter dem Attribut *class* des *event*-Elementes scheinen die verschiedenen Komponenten einer Anwendung gespeichert zu sein (siehe Kapitel 1.3).

Weiterhin geht aus dieser Untersuchung hervor, welche Bedeutung die Werte „1“ und „2“ haben, die das Attribut *type* der *event*-Elemente annehmen kann. Der Wert „2“ tritt auf, sobald eine bestimmte Ansicht der jeweiligen App verlassen wurde, und der Wert „1“, wenn eine Ansicht der Anwendung betreten wird. Was im *config*-Element protokolliert wird, ergab sich aus diesem Versuch noch nicht. Es sieht aber bisher danach aus, als würden hier Konfigurationsdaten gespeichert werden.

2.2.1.3 Protokolldateien in */data/system/usagestats/0/weekly/*

In diesem Ordner liegen zwei XML-Dateien, welche den Namen *1532013966190.xml* und *1532014304413.xml* tragen.

Aufbau

Beide *XML*-Dateien haben denselben Aufbau. Das Wurzelelement heißt *usagestats* und diesem werden die Attribute *endTime* und *version* zugeordnet. Das Wurzelelement hat drei Kind-Knoten, die *packages*, *configurations* und *event-log* heißen. *Event-log* stellt in dieser *XML*-Datei ein leeres Element dar. Das Element *packages* enthält Unterelemente mit dem *XML*-Namen *package*. Das *configurations*-Element enthält Kind-Knoten mit dem *XML*-Namen *config*. *Package*-Elemente haben die Attribute *lastEvent*, *timeActive*, *package* und *lastTimeActive*. Dem *config*-Element werden die Attribute *timeActive*, *lastTimeActive*, *density*, *sw*, *height*, *width*, *ui*, *scrLay*, *ori*, *navHid*, *nav*, *hardKeyHid*, *keyHid*, *key*, *touch*, *locales*, *mnc*, *mcc*, *fs*, *active* und *count* zugeordnet.

Analyse

Im Folgenden wird die Datei *1532014304413.xml* näher betrachtet. Auffällig ist, dass diese Datei den gleichen Namen wie eine Datei trägt, welche im Ordner „daily“ abgespeichert ist. Mit Hilfe des Tools *WinMerge* werden die beiden *XML*-Dateien verglichen. Es stellt sich heraus, dass die *package*-Elemente und die *config*-Elemente identisch sind. Der einzige Unterschied liegt in den *event-log*-Elementen. Diese werden im Ordner „weekly“ als leeres Element dargestellt, das heißt sie besitzen keinen Inhalt. Dasselbe gilt für die Datei *1532013966190.xml*. Auch diese, verglichen mit der *1532013966190.xml*-Datei aus dem „daily“-Ordner, unterscheidet sich lediglich in den *event*-Elementen.

Ergebnis

Es ist nach oben ausgeführten Szenario anzunehmen, dass in diesem Ordner *XML*-Dateien angelegt werden, die teilweise mit den XML-Dateien im Ordner „daily“ übereinstimmen. Beide Dateien, die sich in dem Ordner „daily“ befinden, sind auch im Ordner „weekly“ vorhanden. Die Dateien im Ordner „weekly“ besitzen ein leeres *event-log*-Element, während das *event-log*-Element der „daily“-Dateien über weitere Kind-Knoten verfügt. Daraus ergibt sich, dass in den „daily“-Dateien eine genauere Protokollierung davon stattfindet, was in den Applikationen vorgenommen wird. Diese Einträge sind in den „weekly“-Dateien nicht vorhanden. Dort wird lediglich dokumentiert, wann eine Applikation das letzte Mal benutzt wurde.

2.2.1.4 Protokolldateien in */data/system/usagestats/0/monthly/*

In diesem Ordner liegen zwei *XML*-Dateien, welche den Namen *1532013966190.xml* und *1532014304413.xml* tragen.

Aufbau

Diese *XML*-Dateien entsprechen dem Aufbau der *XML*-Dateien im Ordner „weekly“.

Analyse

Im Folgenden wird die Datei *1532014304413.xml* näher betrachtet. Auffällig ist, dass diese Datei genauso heißt, wie eine Datei, die im Ordner “weekly“ abgelegt ist. Vergleicht man die beiden Dateien mit Hilfe des Tools *WinMerge*, kommt man zu dem Ergebnis, dass beide Dateien identisch sind. Analog gilt dies für die *1532013966190.xml*-Datei, wenn man diese mit der *1532013966190.xml*-Datei in „weekly vergleicht.

Dass es sich dabei um die exakt gleichen Dateien handelt, belegt auch ein Hashabgleich mittels des Programmes *HashMyFiles*, wie in Tabelle 2.7 zu sehen ist.

Tabelle 2.7: Versuchsreihe 2, Szenario 1 Hashvergleich zwischen „weekly“ und „monthly“ [Quelle: eigene Tabelle]

MD5	**SHA1**	**Pfad**
6665cb089fc76bdf707e05402 ffbf51e	b9de29b35fdd620e81dcf243c 05fa43663c68bf0	C:\android-sdk\platform-tools\0\weekly\ 1532013966190.xml

6665cb089fc76bdf707e05402ffbf51e	b9de29b35fdd620e81dcf243c05fa43663c68bf0	C:\android-sdk\platform-tools\0\monthly\1532013966190.xml
ec06866b90803355d7ad8d5c2a7ea326	27ec6ce95b8ceeb7eb8b82d0bdcbca381ab49276	C:\android-sdk\platform-tools\0\monthly\1532014304413.xml
ec06866b90803355d7ad8d5c2a7ea326	27ec6ce95b8ceeb7eb8b82d0bdcbca381ab49276	C:\android-sdk\platform-tools\0\weekly\1532014304413.xml

Ergebnis

Die „monthly“-Dateien stimmen exakt mit den „weekly“-Dateien überein. Es ist also davon auszugehen, dass diese Logdateien täglich, wöchentlich, monatlich und jährlich abgelegt werden. Für nachfolgende Versuche gilt es herauszufinden, wie lange eine Datei in den jeweiligen Ordnern gespeichert wird. Ziel ist herauszufinden, ob die Dateien im Ordner „monthly“ länger aufbewahrt werden als im Ordner „weekly“.

2.2.1.5 Protokolldateien in */data/system/usagestats/0/yearly/*

Aufbau

Der Aufbau dieser *XML*-Dateien entspricht dem Aufbau der *XML*-Dateien im Unterordner „monthly“ und „weekly“. Es befinden sich zwei Dateien in diesem Ordner, welche die Namen *1532013966190.xml* und *1532014304413.xml* tragen.

Analyse

Erneut wird ein Vergleich mit den *XML*-Dateien in dem Ordner „weekly“ und „monthly“ durchgeführt. Wie sich herausstellt sind beide Dateien, welche sich in dem Ordner „monthly“ befinden, identisch zu denen, die sich im Ordner „yearly“ und „weekly“ befinden. Das Ergebnis ist in Tabelle 2.8 zu sehen.

Tabelle 2.8: Versuchsreihe 2, Szenario 1 Hashvergleich zwischen „weekly“, „monthly“ und „yearly“ [Quelle: eigene Tabelle]

MD5	**SHA1**	**Pfad**
6665cb089fc76bdf707e05402 ffbf51e	b9de29b35fdd620e81dcf243c 05fa43663c68bf0	C:\android-sdk\platform-tools\0\weekly\ 1532013966190.xml
6665cb089fc76bdf707e05402 ffbf51e	b9de29b35fdd620e81dcf243c 05fa43663c68bf0	C:\android-sdk\platform-tools\0\monthly\ 1532013966190.xml
6665cb089fc76bdf707e05402 ffbf51e	b9de29b35fdd620e81dcf243c 05fa43663c68bf0	C:\android-sdk\platform-tools\0\yearly\ 1532013966190.xml
ec06866b90803355d7ad8d5c 2a7ea326	27ec6ce95b8ceeb7eb8b82d0 bdcbca381ab49276	C:\android-sdk\platform-tools\0\weekly\ 1532014304413.xml
ec06866b90803355d7ad8d5c 2a7ea326	27ec6ce95b8ceeb7eb8b82d0 bdcbca381ab49276	C:\android-sdk\platform-tools\0\monthly\ 1532014304413.xml
ec06866b90803355d7ad8d5c 2a7ea326	27ec6ce95b8ceeb7eb8b82d0 bdcbca381ab49276	C:\android-sdk\platform-tools\0\yearly\ 1532014304413.xml

Ergebnis

Auch hier gilt es noch zu klären, wie lange die Dateien im Ordner „yearly“ aufbewahrt werden. Da sich im „weekly“-, „monthly“- und „yearly“-Ordner dieselben Dateien befinden, ist anzunehmen, dass der Unterschied zwischen den Ordnern „weekly“, „monthly“ und „yearly“ in der „Aufbewahrungszeit“ dieser Dateien liegt.

2.2.1.6 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Aufbau

Der Aufbau der WhatsApp-Protokolldatei hat sich im Vergleich zu Versuchsreihe 1 nicht verändert.

Abbildung 2.3: Versuchsreihe 2, Szenario 1 Aufbau WhatsApp-Protokolldatei [Quelle: eigene Abbildung]

Analyse

Was an dieser Textdatei auffällt, ist, dass der Batterie-Status des Handys mit aufgezeichnet wird. Folgende Einträge zum Batterie-Status wurden gefunden:

- newEvent=BatteryStatehealth=good, level=92, plugged=0, scale=100, percent=92.02018-07-19 17:46:48.476
- BatteryStatehealth=good, level=92, plugged=0, scale=100, percent=92.02018-07-19 17:47:14.616
- newEvent=BatteryStatehealth=good, level=91, plugged=0, scale=100, percent=91.02018-07-19 17:48:03.183
- newEvent=BatteryStatehealth=good, level=90, plugged=0, scale=100, percent=90.02018-07-19 17:52:11.819

Ergebnis

Auch in diesem Szenario werden in der WhatsApp-Protokolldatei BatteryState-Ereignisse gefunden. Es gilt in den nachfolgenden Szenarien zu überprüfen, ob die Analyse dieser Ereignisse dieselben Ergebnisse wie Versuchsreihe 1 hervorbringt.

2.2.2 Szenario 2

Die in diesem Szenario beschriebenen Zeiten beziehen sich alle auf die Zeiten, welche mit dem Handy gemessen werden. Das Handy wird nach dem letzten Versuch ausgeschaltet. Am 20.07.2018 gegen 10:17 Uhr wird das Handy wieder angeschaltet. Um 10:19 Uhr wird die „Rechner“-App gestartet. Eine Minute später wird diese beendet. Gegen 10:20 Uhr wird das WhatsApp-Hauptmenü gestartet und um 10:21 Uhr ein Profilfoto auf WhatsApp angeschaut. Um 10:22 Uhr wird zurück auf das WhatsApp-Hauptmenü geschaltet. Eine Minute später wird WhatsApp komplett geschlossen. Um 10:23 Uhr

wird der PlayStore geöffnet und um 10:24 Uhr wieder geschlossen. Um 10:25 Uhr wird das Handy mit einem Netzteil-Ladegerät verbunden und eine Minute später mit einem USB-Anschluss. Anschließend wird der Taskmanager geöffnet. Danach wird es in den Recovery-Modus gebootet.

Untersucht wurden folgende Dateien **(Details siehe Anlage Versuchsreihe 2, Szenario 2)**

2.2.2.1 Protokolldateien in */data/system/usagestats/0/daily/*

Ergebnis

Nach bisherigen Erkenntnissen wird keine *XML*-Datei angelegt, die einen Zeitstempel vom 20.07.2018 trägt. Die Ereignisse dieses Tages werden aber in die Datei vom Vortag mit hineingeschrieben. Aufgrund dessen ist es über die *package*-Elemente nicht mehr möglich nachzuvollziehen, was am Vortag passiert ist, da sich die Zeiten dieser Elemente aktualisiert haben. Es ist jedoch immer noch möglich, anhand der *event*-Elemente zu rekonstruieren, was am Vortag gemacht wurde, weil diese Elemente nicht aktualisiert wurden. Es wurden nur neue *event*-Elemente angelegt. Die neu hinzugekommenen Elemente machen eine Aussage darüber, was am 20.07.2018 passiert ist. Die gemessenen Zeiten bestätigen, welche Bedeutung das Attribut *type* hat. Nach bisherigen Erkenntnissen nimmt es den Wert „1“ an, sobald eine Ansicht im Vordergrund ist und den Wert „2“, sobald eine Ansicht verlassen wird. Außerdem erscheint der Wert „5“, sofern das dazugehörige *event*-Element Attribute der *config*-Elemente übernimmt. Dies wiederum könnte bedeuten, dass der Wert „5“ für Änderungen der Konfigurationsdaten steht. Wann ein *event*-Element den Wert „7“ annimmt, ist aus den bisherigen Untersuchungen noch nicht hervorgegangen.

Die *event*-Elemente geben eine genauere Beschreibung dessen ab, was in der App passiert, sprich welche Ansicht beispielsweise gerade angeschaut wird. Außerdem wird deutlich, dass die *config*-Elemente größtenteils während des Startens eines Smartphones angelegt werden, was die Vermutung unterstützt, dass es sich hierbei lediglich um Konfigurationsdaten handelt. Diese Vermutung gilt es jedoch in den weiteren Szenarien zu bestätigen.

2.2.2.2 Protokolldateien in */data/system/usagestats/0/weekly/*

Ergebnis

Der Ordner „weekly“ scheint eine Protokollierung dessen vorzunehmen, was während der Woche passiert ist. Dabei speichert der Ordner keine *event*-Elemente und somit kann aus diesen Dateien lediglich herausgelesen werden, wann welche App das letzte Mal

benutzt wurde. Dieser Ordner speichert demnach auch die Daten der Vortage.

2.2.2.3 Protokolldateien in */data/system/usagestats/0/monthly/*

Ergebnis

Es bestätigt sich die Vermutung, dass der Ordner „monthly“ dieselben Dateien speichert, die auch schon im Ordner „weekly“ abgespeichert sind. Es ist zu vermuten, dass dieser die Dateien jedoch länger aufbewahrt, als dies im Ordner „weekly“ der Fall ist. Dafür gibt es jedoch nach bisherigen Versuchen noch keinen Beweis und dies gilt es in den nächsten Szenarien herauszufinden.

2.2.2.4 Protokolldateien in */data/system/usagestats/0/yearly/*

Ergebnis

Auch hier wird die Vermutung bestätigt, dass die Dateien in dem Ordner „yearly“ zu denen in den Ordnern „weekly“ und „monthly“ identisch sind. Es gilt demnach herauszufinden, wie lange die jeweiligen Dateien in den Ordnern aufbewahrt werden.

2.2.2.5 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Ergebnis

Nach diesem Szenario kann davon ausgegangen werden, dass das Attribut *plugged* die Werte „0“, „1“ und „2“ annehmen kann. Der Wert „0“ würde nach bisherigen Ergebnissen für ein nicht angeschlossenes Handy sprechen. Der Wert „1“ gibt einen Hinweis darauf, dass ein Handy mit einem Netzteil-Ladegerät aufgeladen wurde, während der Wert „2“ verrät, dass das Smartphone mit einem USB-Stecker verbunden war. Weiterhin hat sich in den Versuchen gezeigt, dass der erste Eintrag „BatteryState“ eine Zeit von 10:17 Uhr aufweist. Zu dieser Zeit wurde nach Ablauf des Szenarios das Handy hochgefahren. Ferner werden mehrere Ereignisse von „BatteryState“ in kurzen Zeitabständen angelegt. Es konnte bisher jedoch nicht festgelegt werden, wie viele von diesen Ereignissen über einen gesamten Tag angelegt werden.

2.2.3 Szenario 3

Dieses Szenario sollte hauptsächlich dazu dienen, herauszufinden, wie lange die Dateien in den jeweiligen Ordnern gespeichert werden. Leider war dies nicht möglich. Das Handy konnte nicht angeschaltet werden. Daraufhin wurde der Akku entnommen und

nach der Sicherung der Daten wurde festgestellt, dass alle Daten aus den ersten beiden Szenarien gelöscht waren. Am 25.07.2018 werden dennoch Versuche durchgeführt. Um ca. 07:46 Uhr wird das Gerät mit dem WLAN verbunden. Gegen 08:05 Uhr wird das WhatsApp-Hauptmenü geöffnet. Eine Minute danach wird die Ansicht des Hauptmenüs einmal gedreht. Um 08:07 Uhr wird die Ansicht wieder in die Anfangsposition gedreht und eine Minute später wird WhatsApp verlassen. Gegen 08:09 Uhr wird der Rechner auf dem Handy geöffnet und eine Minute später wieder geschlossen. Um 08:11 Uhr wird die Einstellungs-App geöffnet und es wird neben der deutschen Sprache unter „Spracheinstellungen“ auch noch die englische Sprache hinzugefügt. Um 08:12 Uhr findet ein App-Wechsel statt und WhatsApp wird erneut gestartet. Um dieselbe Uhrzeit wird das Handy schließlich heruntergefahren.

Untersucht wurden folgende Dateien **(Details siehe Anlage Versuchsreihe 2, Szenario 3)**

2.2.3.1 Protokolldateien in */data/system/usagestats/0/daily/*

Ergebnis

Nach diesem Szenario wird deutlich, dass das Attribut *lastEvent* immer einen Wert von „0“ annimmt, wenn die Anwendung nicht benutzt wurde. Dabei kann das Attribut *lastEvent* einen Wert von „0“ haben, während das Attribut *lastTimeActive* einen positiven Wert annimmt. Dies ist der Fall, wenn die App nicht aktiv vom Nutzer benutzt wurde, aber trotzdem im Hintergrund aktiv war. Ein Beispiel hierfür stellt in diesem Szenario eine eingegangene SMS dar.

Weiterhin konnte die Funktion des Attributes *timeActive* festgestellt werden. Dieses Attribut speichert nach bisherigen Erkenntnissen die Zeit, wie lange eine App aktiv war. *Config*-Elemente speichern Konfigurationseinstellungen eines Handys. *Config*-Elemente werden beim Start eines Smartphones angelegt. Sie werden nach bisherigen Erkenntnissen dann angelegt, wenn Konfigurationsdaten, wie beispielsweise die Änderung der Spracheinstellungen oder die Bildansicht, geändert werden.

In diesem Szenario konnte man ebenfalls erkennen, welche Werte das *type*-Attribut der *event*-Elemente annehmen kann und welche Bedeutung diese haben. Es hat sich bestätigt, dass der Wert „1“angenommen wird, wenn eine Komponente der App in den Vordergrund tritt und der Wert „2“ angenommen wird, wenn diese Komponente wieder in den Hintergrund tritt. Ferner hat sich ergeben, dass der Wert „7“ angenommen wird, wenn der Nutzer die App nicht aktiv genutzt hat, die App jedoch trotzdem im Hintergrund aktiv war, wie beispielsweise beim Empfangen von Nachrichten. Der Wert „5“ wird vom *type*-Attribut immer dann angenommen, wenn Konfigurationsänderungen stattgefunden haben.

2.2.3.2 Protokolldateien in */data/system/usagestats/0/weekly/*

Ergebnis

Es ist weiterhin davon auszugehen, dass dieser Ordner wöchentlich aggregiert, was an welchem Tag gemacht wurde. Es wird dabei jedoch nicht auf die genauen Ereignisse (*event*-Elemente) eingegangen, sondern lediglich notiert, wann eine App das letzte Mal genutzt wurde und wie lange eine Anwendung aktiv war.

2.2.3.3 Protokolldateien in */data/system/usagestats/0/monthly/*

Ergebnis

Es wird weiterhin vermutet, dass die Daten in dem Ordner „monthly“ länger gespeichert werden als in dem Ordner „weekly“.

2.2.3.4 Protokolldateien in */data/system/usagestats/0/yearly/*

Ergebnis

Es wird vermutet, dass der Ordner „yearly“ die Daten über ein ganzes Jahr speichert.

2.2.3.5 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Ergebnis

Nach dieser Analyse wird in regelmäßigen Abständen der Batteriestatus notiert und ebenso oft ein Status darüber angegeben, ob der Bildschirm entsperrt ist oder nicht. Des Weiteren konnten sich die Ergebnisse bezüglich des Batteriestatus von Versuchsreihe 1 bestätigen.

2.2.4 Szenario 4

Ziel des letzten Szenarios dieser Versuchsreihe ist es festzustellen, wie lange die jeweiligen Dateien in den verschiedenen Unterordnern des Ordners *usagestats/0/* aufbewahrt werden. Weiterhin geht es darum nachzuvollziehen, was der Ordner *recent_images*, welcher sich auf der Datenpartition in dem Ordner *system_ce/0/* befindet, speichert. Für dieses Szenario wird das Handy eine Woche lang, abgesehen von einem Tag, angeschaltet und es werden verschiedene Aktionen am Handy durchgeführt.

In diesem Szenario wird es nicht darum gehen, zu analysieren, was genau in den Logdateien unter *usagestats/* protokolliert wird. Das war bereits Teil der ersten drei Szenarien.

Ein weiterer Bestandteil dieses Szenarios ist es herauszufinden, was das Ereignis ScreenLockReceiver der WhatsApp-Logdatei protokolliert. Dafür werden über die Woche hinweg verschiedene Zeitmessungen durchgeführt. Am 01.08.2018 werden die Daten dann über *adb pull* gesichert.

Untersucht wurden folgende Dateien **(Details siehe Anlage Versuchsreihe 2, Szenario 4)**

2.2.4.1 Protokolldateien in */data/system/usagestats/0/daily/*

Ergebnis

Es konnte festgestellt werden, dass die Dateien im Ordner „daily“ sieben Tage gespeichert werden. Man kann demnach nicht mehr die genauen Aktivitäten nachvollziehen, die länger als eine Woche zurückliegen.

2.2.4.2 Protokolldateien in */data/system/usagestats/0/weekly/*

Ergebnis

Die „weekly“-Datei gibt über das Attribut *timeActive* den Wert an, wie lange eine bestimmte Anwendung über eine Woche hinweg im Vordergrund stand. Weiterhin wird in dieser „weekly“-Datei ebenfalls festgehalten, wann die Anwendung das letzte Mal benutzt wurde.

2.2.4.3 Protokolldateien in */data/system/usagestats/0/monthly/*

Ergebnis

Gemäß dieser Analyse aggregiert die Datei, welche sich im „monthly“-Ordner befindet, die Nutzungsstatistik über einen Monat hinweg. Dabei kann man durch das Attribut *timeActive* nachvollziehen, wie lange die App insgesamt über den gesamten Monat benutzt wurde. Außerdem wird auch hier der Zeitpunkt gespeichert, an dem eine App das letzte Mal benutzt wurde.

2.2.4.4 Protokolldateien in */data/system/usagestats/0/yearly/*

Ergebnis

Der Inhalt der *XML*-Datei dieses Ordners stimmt mit dem des Ordners „monthly“ überein. Es ist jedoch anzunehmen, dass die Daten in diesem Ordner über ein komplettes Jahr aggregiert werden, während die Daten im Ordner „monthly“ nur monatsweise aggregiert werden.

2.2.4.5 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Ergebnis

Das Ereignis ScreenLockReceiver protokolliert gemäß dieser Analyse, wann der Bildschirm sich in einem entsperrten oder gesperrten Zustand befindet. Das Attribut *tag* nimmt dabei den Wert „unlock“ an, wenn der Bildschirm entsperrt wird. Weiterhin wird der Wert „on“ angenommen, wenn der Bildschirm aufleuchtet. Dies ist beispielsweise der Fall bei einer eingehenden Nachricht. In dieser Logdatei konnten ebenfalls Einträge gefunden werden, bei denen dieses Attribut den Wert „off“ annimmt. Dieser Wert wird angenommen, wenn der Bildschirm gesperrt ist. Das Attribut *oldLocked* gibt den Zeitpunkt an, an welchem das jeweilige ScreenLockReceiver-Ereignis eingetreten ist.

2.2.4.6 Ordner *recent_tasks* und *recent_images*

Ergebnis

Nach bisheriger Erkenntnis übernimmt der Ordner *recent_tasks* die Protokollierung der Zeitstempelinformationen und anderer Informationen (siehe Analyse) über die *.png*-Dateien, welche sich in dem Ordner *recent_images* befinden. Die *.png*-Dateien stellen dabei Screenshots benutzter Anwendungen dar.

2.3 Versuchsreihe 3

Zur Untersuchung liegt ein Samsung Galaxy S4 mini mit der Modellnummer GT-I9195I vor. Am 05.08.2018 wird über TWRP ein neues Custom-ROM auf das Handy geflasht. Vor der Installation dieses Custom-ROMs wird ein vollständiger Wipe (Factory Reset) durchgeführt. Es wird die Version 13.0 von LineageOS installiert, welche der Android-Version 6.0.1 entspricht. Neben der neuen Betriebssystemversion werden ebenfalls Google-App-Pakete heruntergeladen, um den Google-PlayStore später nutzen zu können. Bei dem Download der Google-Pakete muss darauf geachtet werden, dass der richtige Prozessortyp ausgewählt wird. Im Falle dieses Handys ist es der Prozessortyp „ARM“. Über

den „Install“-Reiter von TWRP werden die Dateien der beiden *.zip*-Ordner installiert. Am 06.08.2018 wird Szenario 1 durchgeführt. Es ist anzumerken, dass die Zeiten, die im folgenden Szenario beschrieben sind, sowohl mit der Uhr des Handys als auch mit Epoch-Zeit gemessen werden. Die Daten werden über *adb pull* gesichert und anschließend analysiert. Die ersten Protokolldateien, die untersucht werden, liegen auf der Datenpartition in dem Ordner */system/usagestats/*. Weiterhin wird der komplette *com.whatsapp*-Ordner gesichert, welcher sich unter */data/data* befindet. Zu guter Letzt wird noch der Ordner *recent_images* auf der Datenpartition gesichert, welcher ebenfalls im Ordner *system* liegt.

2.3.1 Szenario 1

Nachdem am 05.08 das neue Betriebssystem auf dem Handy installiert wird, wird am selben Tag ebenfalls WhatsApp mit der Version 2.18.230 heruntergeladen. Am 06.08.2018 werden verschiedene Versuche durchgeführt. Zunächst wird gegen 08:30 Uhr das Handy gestartet und um 8:31 Uhr wird das WhatsApp-Hauptmenü geöffnet. Eine Minute später wird auf einen WhatsApp-Chat geklickt und danach wird ein Profilfoto angeschaut. Um 8:33 Uhr erfolgt die Rückkehr zur Konversation und um ca. 8:35 Uhr wird ein Kontakt hinzugefügt. In derselben Minute wird der Status von WhatsApp angeschaut und eine Minute später der Reiter „Einstellungen“ erkundet. Um 08:36 Uhr wird WhatsApp wieder verlassen und gegen 08:37 Uhr das Handy heruntergefahren.

2.3.1.1 Ordner in */data/system/usagestats/0/*

In diesem Ordner befinden sich, wie in Versuchsreihe 2, vier Unterordner mit den Namen „daily“, „weekly“, „monthly“ und „yearly“. Es existiert ebenfalls eine Datei „version“, wie in Versuchsreihe 2. Ein Unterschied zu der vorhergehenden Versuchsreihe besteht darin, dass eine Datei mit dem Namen „screen_on_time“ vorzufinden ist.

2.3.1.2 Protokolldateien in */data/system/usagestats/0/daily/*

In diesem Ordner befinden sich zwei Dateien. Diese Dateien heißen *1533480274635.xml* und *1533513600000.xml*.

Aufbau

Der Aufbau der in diesem Ordner liegenden *XML*-Dateien unterscheidet sich geringfügig von denen in Versuchsreihe 2. Die *XML*-Datei besteht wieder aus einem Wurzelelement *usagestats*, welches die Sub-Elemente *packages*, *configurations* und *event-log* besitzt. Dem Wurzelelement sind die Attribute *endTime* und *version* zugeordnet. Das Element

packages besitzt Kind-Knoten mit dem *XML*-Namen *package*. Diesem werden die Attribute *beginIdleTime*, *lastEvent*, *timeActive*, *package*, *lastTimeActiveSystem* und *lastTimeActive* zugeordnet.

Auch das *configurations*-Element besitzt Kind-Knoten, welche den Namen *config* haben. Diese werden durch die Attribute *timeActive*, *lastTimeActive*, *density*, *sw*, *height*, *width*, *ui*, *scrLay*, *ori*, *navHid*, *keyHid*, *key*, *touch*, *locale*, *fs*, *count*, *mnc*, *mcc*, *hardKeyHid* und *fs* definiert. Die *event-log*-Elemente besitzen wiederum auch Kind-Knoten mit dem *XML*-Namen *event*. Die *event*-Elemente haben die Attribute *package*, *type*, *time* und *class*.

Analyse

Zunächst werden die *package*-Elemente betrachtet. In Abbildung 2.4 ist ein Ausschnitt der *package*-Elemente gezeigt.

```
<package beginIdleTime="714508" lastEvent="2" timeActive="282043" package="com.whatsapp"
    lastTimeActiveSystem="23805482" lastTimeActive="23805482"/>
<package beginIdleTime="428183" lastEvent="0" timeActive="0" package="com.android.providers.downloads"
    lastTimeActiveSystem="23519124" lastTimeActive="-1533513600000"/>
<package beginIdleTime="385985" lastEvent="0" timeActive="0" package="com.google.android.apps.messaging"
    lastTimeActiveSystem="23476633" lastTimeActive="23476633"/>
```

Abbildung 2.4: Versuchsreihe 3, Szenario 1 Ausschnitt *package*-Elemente [Quelle: eigene Abbildung]

Es ist auffällig, dass die Attribute *lastTimeActiveSystem* und *lastTimeActive* immer dann übereinstimmen, wenn *lastTimeActive* einen positiven Wert hat. Sofern *lastTimeActive* einen negativen Wert hat, stimmen die Werte der Attribute nicht mehr überein. *LastTimeActiveSystem* nimmt immer einen positiven Wert an. Folgende Tabelle vergleicht die Logeinträge mit dem Ablauf des Szenarios (siehe Tabelle 2.9).

Tabelle 2.9: Versuchsreihe 3, Szenario 1 Vergleich Szenario und *package*-Elemente [Quelle: eigene Tabelle]

Attribut *package*	**Dateiname + *lastTimeActive***	**Szenario**	**Handy-Zeit**	**Epoch-Zeit**
com.google.android.contacts	08:35:13.227	Kontakt hinzufügen	08:35	
com.whatsapp	08:36:45.482	WhatsApp verlassen	08:36	08:36:47

Die Analyse der *event*-Elemente hat Folgendes ergeben (siehe Tabelle 2.10):

Tabelle 2.10: Versuchsreihe 3, Szenario 1 Analyse *event*-Elemente [Quelle: eigene Tabelle]

Attribut *package*	**Dateiname + *lastTimeActive***	**Attribut *type***	**Szenario**	**Handy-Zeit**	**Epoch-Zeit**
com.whatsapp	06.08.2018, 08:30:54.394	7	eingehende WhatsApp-Nachricht	8:30	
com.whatsapp .Main	06.08.2018, 08:31:38	1	WhatsApp-Main öffnen	08:31	08:31:38
com.whatsapp .Conversation	06.08.2018, 08:32:23	1	Konversation auf WhatsApp öffnen	08:32	08:32:23
com.whatsapp .ViewProfilePhoto	06.08.2018, 08:33:14	1	Profilfoto ansehen	08:33	08:33:14
com.whatsapp .ViewProfilePhoto	06.08.2018, 08:33:57	2	Profilansicht verlassen	08:33	08:33:57
com.whatsapp .Settings	06.08.2018, 08:36:11	1	Einstellungen von WhatsApp	08:36	08:36:11
com.whatsapp .Settings	06.08.2018, 08:36:45	2	verlassen von WhatsApp	08:36	08:36:47

Die Analyse der *config*-Elemente führt zu folgendem Ergebnis:

Tabelle 2.11: Versuchsreihe 3, Szenario 1 Analyse *config*-Elemente [Quelle: eigene Tabelle]

Attribut *lastTimeActive*	**Dateiname + *lastTimeActive***
23441039	08:30:41
23425250	08:30:25
23422194	08:30:22
23422167	08:30:22
23422663	08:30:22
23441040	08:30:22

Ergebnis

Aus der Analyse geht hervor, dass die *package*-Elemente zwei neue Attribute im Gegensatz zu Versuchsreihe 2 haben. Die neu hinzugekommenen Attribute tragen die Namen *lastTimeActiveSystem* und *beginIdleTime*. Weiter wird deutlich, dass das Attribut *lastTimeActive* angibt, wann die App das letzte Mal genutzt wurde. Auch das Attribut *lastEvent*

führt zum gleichen Ergebnis wie in Versuchsreihe 2 und gibt an, ob eine App aktiv genutzt wurde oder nicht. Die *config*-Elemente werden ebenfalls wieder angelegt, sobald das Handy gestartet ist. Zu den *event*-Elementen ist zu sagen, dass sie zu den gleichen Ergebnissen wie in Versuchsreihe 2 führen und sich bei diesen auch keine Veränderung in der Struktur ergibt.

2.3.1.3 Protokolldateien in */data/system/usagestats/0/weekly/*

Aufbau

Diese XML-Datei besteht aus einem Wurzelelement *usagestats*, dem die Attribute *endTime* und *version* übergeben werden. Das Wurzelelement hat weiterhin drei Kind-Knoten, die *packages*, *configurations* und *event-log* heißen. Dabei stellt *event-log* ein leeres Element dar, wohingegen das *packages*-Element Kind-Knoten besitzt, die den *XML*-Namen *package* tragen. Diesen werden die Attribute *beginIdleTime*, *lastEvent*, *timeActive*, *package*, *lastTimeActiveSystem* und *lastTimeActive* zugeordnet. Das *configurations*-Element besitzt Kind-Knoten mit dem Namen *config*. Diese werden durch die Attribute *timeActive*, *lastTimeActive*, *density*, *sw*, *height*, *width*, *ui*, *scrLay*, *ori*, *navHid*, *keyHid*, *key*, *touch*, *locale*, *fs*, *count*, *mnc*, *mcc*, *hardKeyHid* und *fs* definiert.

Analyse

Vergleicht man diese Datei mit den Dateien im Ordner „daily“, unterscheiden sich diese in einigen Punkten. Die folgenden Tabellen dienen zur Verdeutlichung.

Tabelle 2.12: VR 3, S 1 Vergleich „daily“- und „weekly“-Dateien [Quelle: eigene Tabelle]

	Dateiname	**Attribut *package***	**Attribut *timeActive***	**Dateiname + *lastTimeActive***
Daily-Datei	1533513600000	com.whatsapp	282043	1533537405482
Daily-Datei	1533480274635	com.whatsapp	91358	1533498738666
Weekly-Datei	1533480274635	com.whatsapp	373401	1533537405482

Tabelle 2.13: VR 3, S 1 Vergleich der Dateien [Quelle: eigene Tabelle]

	Dateiname	**Attribut *package***	**Attribut *timeActive***	**Dateiname + *lastTimeActive***
Daily-Datei	1533513600000	Android	9661	1533537407901
Daily-Datei	1533480274635	Android	79452	1533498741073
Weekly-Datei	1533480274635	Android	89113	1533537407901

Wie den beiden Tabellen zu entnehmen ist, addieren sich die *timeActive*-Werte der „daily“-Dateien zu dem *timeActive*-Wert der „weekly“-Datei. Weiterhin ist zu erkennen, dass das Attribut *lastTimeActive* der „weekly“-Datei den Wert speichert, an dem die entsprechende Anwendung zuletzt genutzt wurde. Wurde die Anwendung beispielsweise zuletzt am 06.08.2018 genutzt, muss man den *lastTimeActive*-Wert der Anwendung auf den Dateinamen der „daily“-Datei vom 06.08.2018 addieren. Dies führt zum gleichen Ergebnis wie die Addition des *lastTimeActive*-Wertes der „weekly“-Datei mit ihrem Dateinamen.

Ergebnis

Nach der Analyse dieser *XML*-Datei wird deutlich, dass die „weekly“-Datei eine Aggregation der *timeActive* Werte der jeweiligen „daily“-Dateien vornimmt. Weiter ist anzumerken, dass der Dateiname der „weekly“-Datei, summiert mit dem Attribut *lastTimeActive*, einen Zeitstempel ergibt, der angibt, wann die App zuletzt genutzt wurde. Dieser Zeitstempel ist identisch zu dem Eintrag der jeweiligen „daily“-Datei, an dem die besagte App das letzte Mal genutzt wurde.

2.3.1.4 Protokolldateien in */data/system/usagestats/0/monthly/*

Aufbau

Der Aufbau dieser Dateien entspricht dem Aufbau der Dateien, welche sich im Ordner „weekly“ befinden.

Analyse

In diesem Ordner befinden sich drei Dateien. Betrachtet man die Letzte dieser Dateien genauer, ergeben sich folgende Erkenntnisse:

Tabelle 2.14: VR 3, S 1 Vergleich „monthly“-Dateien [Quelle: eigene Tabelle]

Attribut *package*	**Dateiname + *lastTimeActive***	**Attribut *timeActive***
com.whatsapp	1533537407901	373401
Android	1533537405482	89113

Diese Werte stimmen exakt mit den Werten überein, die sich bei der Berechnung der „weekly“-Datei ergeben haben (siehe Tabelle 2.12, 2.13).

Ergebnis

Es ist festzuhalten, dass in der „monthly“-Datei, nachdem das Betriebssystem seit zwei Tagen installiert ist, dieselben Werte gespeichert werden wie in der „weekly“-Datei. Dabei werden die *timeActive*-Werte der jeweiligen Anwendungen über die „daily“-Dateien addiert und der *lastTimeActive*-Wert bezieht sich auf den Wert, an dem die App das letzte Mal benutzt wurde.

2.3.1.5 Protokolldatei in */data/system/usagestats/0/yearly/*

Aufbau

In diesem Ordner befindet sich eine *XML*-Datei, die dem Aufbau der Dateien aus den Ordnern „weekly“ und „monthly“ entspricht.

Analyse

Betrachtet man diese *XML*-Datei, ergeben sich folgende Zusammenhänge:

Tabelle 2.15: VR 3, S 1 „yearly“-Dateien [Quelle: eigene Tabelle]

Attribut *package*	**Attribut *timeActive***	**Dateiname + *lastTimeActive***
com.whatsapp	373401	1533537405482
android	89113	1533537407901

Es wird deutlich, dass der Dateiname addiert mit dem Attribut *lastTimeActive* der jeweiligen App auf dasselbe Ergebnis kommt, welches auch die Auswertung der „weekly“-Dateien erbracht hat (siehe Tabelle 2.12, 2.13).

Ergebnis

Auch diese Analyse bestätigt, dass in der „yearly“-Datei dieselben Werte wie in der „weekly“-Datei gespeichert werden. Weiterhin wird deutlich, dass diese Werte ebenfalls mit den Aktivitäten, welche im Szenario durchgeführt wurden, übereinstimmen.

2.3.1.6 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Aufbau

Der Aufbau der WhatsApp-Protokolldatei entspricht dem Aufbau der Logdatei von Versuchsreihe 1 und 2.

Analyse

In dieser Protokolldatei konnten sowohl ScreenLockReceiver- und BatteryState-Ereignisse gefunden werden. Es wird deutlich, dass alle BatteryState-Ereignisse bei dem Attribut *plugged* einen Wert von „0“ annehmen. Weiterhin kann beobachtet werden, dass sowohl BatteryState-Ereignisse vom 05.08.2018 als auch vom 06.08.2018 existieren.

• BatteryStatehealth=good, level=94, plugged=0, scale=100, percent=94.02018-08-05 21:50:36.116
• BatteryStatehealth=good, level=92, plugged=0, scale=100, percent=92.02018-08-06 08:30:40.749

Neben den BatteryState-Ereignissen werden auch ScreenLockReceiver-Ereignisse in der Protokolldatei von WhatsApp hinterlegt. Es ist zu beachten, dass solche Ereignisse nur zum 06.08.2018 angelegt werden und nicht bereits am Vortag, den 05.08.2018. Weiterhin wird deutlich, dass um 08:31 Uhr des 06.08.2018 das Attribut *tag* den Wert „unlock“ angenommen hat, was im Szenario der Zeit entspricht, als der Bildschirm entsperrt wird und das WhatsApp Hauptmenü geöffnet wird.

•ScreenLockReceiver; tag=unlock; locked=false; oldLocked=true2018-08-06 08:31:26.744

Ergebnis

Gemäß der Analyse wird deutlich, dass ScreenLockReceiver-Ereignisse protokolliert werden, sobald das Handy gestartet wird. Dabei haben sich dieselben Erkenntnisse der Protokollierung ergeben wie in Versuchsreihe 1 und 2. Auch der Batteriestatus wird regelmäßig von WhatsApp notiert, sofern sich das Smartphone in einem angeschalteten Zustand befindet. Es ist jedoch unklar, wie oft und in welchen Zeitabständen diese beiden Er-

eignisse abgelegt werden. Fakt ist, dass ein ScreenLockReceiver-Ereignis immer dann angelegt wird, wenn der Bildschirm entsperrt wird.

2.3.1.7 Ordner *recent_tasks* und *recent_images*

Aufbau

Diese Ordner finden sich bei dieser Betriebssystemversion auf der Datenpartition im Ordner *system*. In dem Ordner *recent_images* befinden sich, wie auch in Versuchsreihe 2, verschiedene Screenshots. Die Benennung der *recent_images* erfolgt nach folgendem Schema: <*Nummer*>*_task_thumbnail*.

In dem Ordner *system* der Datenpartition befindet sich ein weiterer Ordner mit dem Namen *recent_tasks*. Dieser Ordner speichert, wie in Versuchsreihe 2, ebenfalls *XML*-Dateien, welche nach einer „Task“ benannt sind, die sich im Ordner *recent_images* befindet. Jedoch existiert ein Screenshot in *recent_images*, zu dem es keinen Eintrag in *recent_tasks* gibt. Folgende zwei Abbildungen zeigen den Inhalt der beiden Ordner.

```
drwx------    2 system   system        4096 Aug  6 06:36 .
drwxrwxr-x   19 system   system        4096 Aug  6 06:37 ..
-rw-------    1 system   system       18391 Aug  6 06:36 215_task_thumbnail.png
-rw-------    1 system   system       15978 Aug  6 06:36 216_task_thumbnail.png
-rw-------    1 system   system        7168 Aug  5 15:03 29_task_thumbnail.png
-rw-------    1 system   system        4216 Aug  5 15:04 34_task_thumbnail.png
-rw-------    1 system   system       22138 Aug  5 15:07 36_task_thumbnail.png
-rw-------    1 system   system       52955 Aug  5 19:46 71_task_thumbnail.png
-rw-------    1 system   system        3674 Aug  5 19:42 72_task_thumbnail.png
```

Abbildung 2.5: VR 3, S 1 Inhalt *recent_images* [Quelle: eigene Abbildung]

```
drwx------    2 system   system        4096 Aug  6 06:36 .
drwxrwxr-x   19 system   system        4096 Aug  6 06:37 ..
-rw-------    1 system   system         789 Aug  6 06:36 215_task.xml
-rw-------    1 system   system        1016 Aug  5 15:03 29_task.xml
-rw-------    1 system   system        1035 Aug  5 15:04 34_task.xml
-rw-------    1 system   system         891 Aug  5 15:07 36_task.xml
-rw-------    1 system   system        1014 Aug  5 19:46 71_task.xml
-rw-------    1 system   system         952 Aug  5 19:42 72_task.xml
```

Abbildung 2.6: VR 3, S 1 Inhalt *recent_tasks* [Quelle: eigene Abbildung]

Der Aufbau der *XML*-Dateien in *recent_tasks* entspricht dem Aufbau der XML-Dateien von *recent_tasks* aus Versuchsreihe 2.

Analyse

Betrachtet man die Screenshots, welche sich im Ordner *recent_images* befinden, fällt auf, dass lediglich die Screenshots in diesem Ordner abgelegt werden und keine Zeitwerte, die darüber informieren, wann die Screenshots entstanden sind. Es wird jedoch deutlich, dass alle Tasks in *recent_tasks* eine Nummer tragen, die auch bei der Benennung der *recent_images* vorkommt. Es soll nun festgestellt werden, ob eine Verbindung

zwischen den beiden Ordnern besteht.

Die *215_task.xml* wird in den nächsten Schritten näher betrachtet, da diese *XML*-Datei einen Zeitstempel vom 06.08.2018 vorweist und dazu gemessene Ergebnisse aus dem Szenario vorliegen. Alle anderen Tasks in *recent_tasks* weisen einen Zeitwert vom 05.08.2018 auf, zu denen keine gemessenen Zeiten vorliegen. Diese Zeitwerte werden in der entsprechenden *XML*-Datei in *recent_tasks* abgelegt. Der *215_task_thumbnail.png*-Screenshot zeigt ein Bild von den WhatsApp-Einstellungen. In der nächsten Tabelle erfolgt ein Vergleich zwischen den protokollierten Zeitwerten in der *215_task*-Datei und den gemessenen Zeiten des Szenarios.

Tabelle 2.16: VR 3, S 1 Auswertung *recent_tasks*-Datei [Quelle: eigene Tabelle]

Task	**Attribut *last_active_time***	**Szenario**	**gemessene Epoch-Zeit**	**gemessene Handy-Zeit**
215	1533537405530, 06.08.2018, 08:36:45	verlassen von WhatsApp-Einstellungen	08:36:47	08:36

Der in dieser XML-Datei des Ordners *recent_tasks* gespeicherte Wert stimmt mit dem Wert überein, als WhatsApp das letzte Mal an diesem Tag geschlossen wurde. Bevor WhatsApp geschlossen wurde, waren die Einstellungen von WhatsApp zu sehen, was auch der Screenshot in *recent_images* zeigt. Eine weitere Erkenntnis ist, dass derselbe Wert, der in dem Attribut *last_active_time* gespeichert ist, auch in der „daily“-Datei des Ordners *usagestats* vom 06.08.2018 auftaucht.

Ergebnis

Nach bisherigen Erkenntnissen scheint der Ordner *recent_images* einen Screenshot aufzunehmen, kurz bevor eine App geschlossen wird. Die zu diesem Screenshot dazugehörigen Informationen sind im Ordner *recent_tasks* in der entsprechenden *XML*-Datei hinterlegt.

2.3.2 Szenario 2

Dieses Szenario wird am 08.08.2018 durchgeführt. Gegen 11:47 Uhr wird das Handy hochgefahren und mit einem USB-Stecker mit dem Computer verbunden. Danach wird die Rechner-App auf dem Handy gestartet und eine Minute später wieder beendet. Gegen 11:48 Uhr wird das Hauptmenü von WhatsApp gestartet und anschließend ein Profilfoto angeschaut. Gegen 11:50 Uhr erfolgt die Rückkehr zum WhatsApp-Hauptmenü und anschließend wird die App verlassen. Um 11:51 Uhr wird die PlayStore-App geöffnet und

in derselben Minute wieder verlassen. Schließlich wird das Handy mit einem Netzteil-Ladegerät verbunden und aufgeladen. Anschließend wird das Smartphone heruntergefahren.

Untersucht wurden folgende Dateien **(Details siehe Anlage Versuchsreihe 3, Szenario 2)**

2.3.2.1 Protokolldateien in */data/system/usagestats/0/daily/*

Ergebnis

Nach diesem Versuch bestätigt sich, dass bei den *package*-Elementen das Attribut *lastTimeActive* auf den Dateinamen addiert werden muss, um festzustellen, wann die entsprechende App das letzte Mal benutzt wurde. Weiterhin ist deutlich geworden, dass das Attribut *lastTimeActiveSystem* denselben Wert wie das Attribut *lastTimeActive* angibt, sofern die App vom Benutzer aktiv genutzt wurde. Wenn jedoch *lastTimeActive* einen negativen Wert annimmt, also die App nicht aktiv genutzt wurde, und *lastTimeActiveSystem* einen positiven Wert, dann deutet der Wert nach Erkenntnissen dieses Szenarios auf die Zeit hin, als das Handy das letzte Mal hochgefahren wurde. Die *event*-Elemente erbringen dieselben Ergebnisse wie die aus Versuchsreihe 2.

2.3.2.2 Protokolldateien in */data/system/usagestats/0/weekly/*

Ergebnis

Es kann bestätigt werden, dass das Attribut *timeActive* der „weekly“-Datei eine Protokollierung der Zeit vornimmt, wie lange die jeweilige Anwendung über die Woche hinweg benutzt wurde. Dies ergibt sich durch die Addition aller *timeActive*-Werte der entsprechenden App der „daily“-Dateien. Des Weiteren wird deutlich, dass sich das Attribut *lastTimeActive* in der „weekly“-Datei auf die jeweilige letzte Nutzung der entsprechenden App bezieht.

2.3.2.3 Protokolldateien in */data/system/usagestats/0/monthly/*

Ergebnis

Die „monthly“-Datei gibt nach bisherigen Erkenntnissen über den *lastTimeActive*-Wert an, wann eine Anwendung das letzte Mal benutzt wurde. Außerdem kann über den *timeActive*-Wert festgestellt werden, wie lange diese in dem entsprechenden Intervall genutzt wurde.

2.3.2.4 Protokolldateien in */data/system/usagestats/0/yearly/*

Ergebnis

In dieser Datei wird einerseits angegeben, wann eine App zuletzt genutzt wurde, und andererseits für wie lange die Anwendung genutzt wurde.

2.3.2.5 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Ergebnis

Dieses Szenario konnte die Ergebnisse von Versuchsreihe 2 bestätigen.

2.3.2.6 Ordner *recent_tasks* und *recent_images*

Ergebnis

Auch in diesem Szenario bestätigt sich die Vermutung, dass die *XML*-Dateien im Ordner *recent_tasks* mit den Screenshots in *recent_images* zusammenhängen. Weiter kann gezeigt werden, dass sich die Zeit, welche im Attribut *last_active_time* der *XML*-Dateien gespeichert wird, auf die Zeit im Szenario bezieht, als die Anwendung verlassen wurde. Zu allen Anwendungen, die zuletzt am Handy benutzt werden, wird nach diesem Szenario ein Screenshot in *recent_images* abgelegt. Falls an den Tagen zuvor dieselbe Anwendung verwendet wurde, wird der Screenshot des Vortages zu der Anwendung gelöscht. Zu allen Anwendungen, welche nicht am letzten Tag, jedoch zuvor genutzt wurden, bleibt der Screenshot in *recent_images* und der entsprechende Eintrag in *recent_tasks* erhalten.

2.3.3 Szenario 3

Dieses Szenario wird am 10.08.2018 durchgeführt. Gegen 08:41 Uhr wird das Handy gestartet. Danach wird das WhatsApp-Hauptmenü geöffnet. Um 08:42 Uhr wird die Ansicht vom WhatsApp-Hauptmenü gedreht. Eine Minute später wird sie wieder zurück gedreht und WhatsApp verlassen. Um 08:44 Uhr wird die Rechner-App geöffnet und in derselben Minute wieder verlassen. Daraufhin werden die Spracheinstellungen zunächst auf „Englisch“ und eine Minute später wieder zurück auf Deutsch gestellt. In diesem Szenario soll vor allem festgestellt werden, welche Bedeutung die *config*-Elemente haben und ob das Ergebnis mit Versuchsreihe 2 übereinstimmt.

Untersucht wurden folgende Dateien **(Details siehe Anlage Versuchsreihe 3, Szenario 3)**

2.3.3.1 Protokolldateien in */data/system/usagestats/0/daily/*

Ergebnis

Auch in dieser Versuchsreihe wird bestätigt, dass die *config*-Elemente Konfigurationseinstellungen und Konfigurationsänderungen speichern. Des Weiteren wird deutlich, dass diese Änderungen auch in den *event*-Elementen festgehalten werden.

2.3.3.2 Protokolldateien in */data/system/usagestats/0/weekly/*

Ergebnis

Es befinden sich zwei „weekly“-Dateien in dem Ordner. Nach der Analyse aggregiert die erste Datei die Zeiten über die ersten drei Tage der Woche, während die zweite Datei (*1533772800000.xml*) die Werte der nachfolgenden Tage protokolliert.

2.3.3.3 Protokolldateien in */data/system/usagestats/0/monthly/*

Ergebnis

Es ist dasselbe Verhalten wie in Szenario 2 dieser Versuchsreihe zu erkennen. Diese Dateien scheinen die Ereignisse über einen Monat hinweg zu speichern.

2.3.3.4 Protokolldateien in */data/system/usagestats/0/yearly/*

Ergebnis

Es ist dasselbe Verhalten wie in Szenario 2 dieser Versuchsreihe zu erkennen. Diese Dateien scheinen die Ereignisse über ein Jahr hinweg zu speichern.

2.3.3.5 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Ergebnis

Es ist zu sehen, dass gemäß dieser Analyse die ScreenLockReceiver-Ereignisse dieselben Ergebnisse liefern wie in Versuchsreihe 1 und 2. Dabei weist das Attribut *tag* den Wert „unlock“ auf, wenn der Bildschirm entsperrt ist. Der Wert „on“ wird dem Attribut zugeordnet, wenn der Bildschirm aufleuchtet, aber nicht entsperrt ist. Den dritten Wert, den das Attribut annehmen kann, ist „off“. Dieser wird angenommen, sobald der Bildschirm gesperrt wird.

2.3.3.6 Ordner *recent_tasks* und *recent_images*

Ergebnis

In diesem Szenario bestätigt sich die Annahme, dass die Eigenschaften der Screenshots in *recent_images* in den *XML*-Dateien von *recent_tasks* gespeichert werden. Weiterhin haben die Werte der *XML*-Dateien in *recent_tasks* im Vergleich mit dem Ablauf des Szenarios ergeben, dass die Screenshots immer ungefähr dann angelegt werden, wenn die App geschlossen wird.

2.3.4 Szenario 4

In diesem Szenario geht es primär darum, herauszufinden, wie lange die jeweiligen Dateien in den Ordnern gespeichert werden. Außerdem soll festgestellt werden, ob die Protokolldateien auch Ereignisse protokollieren, wenn das Handy angeschaltet ist, jedoch keine Interaktion mit dem Benutzer stattfindet. In der folgenden Analyse wird des öfteren von Aktivitäten gesprochen. Damit sind die Tätigkeiten gemeint, die am Handy ausgeführt werden.

Untersucht wurden folgende Dateien **(Details siehe Anlage Versuchsreihe 3, Szenario 4)**

2.3.4.1 Protokolldateien in */data/system/usagestats/0/daily/*

Ergebnis

Wie zu erwarten wurden die Dateien vom 05.08.2018 und 06.08.2018 gelöscht. Damit bestätigen sich die Ergebnisse von Versuchsreihe 2, dass die Dateien nur sieben Tage in dem „daily“-Ordner aufbewahrt werden. Weiterhin findet in diesem Szenario eine Aktualisierung der Zeitstempel der *XML*-Dateien statt, wohingegen der Inhalt der Dateien gleichbleibt. Wenn man die Zeiten der Aktivitäten der vorherigen Tage berechnet, führt dies zu einem Zeitunterschied von zwei Sekunden bei jeder Aktivität.

2.3.4.2 Protokolldateien in */data/system/usagestats/0/weekly/*

Ergebnis

In diesem Ordner befinden sich zwei Dateien. Dabei protokolliert die *1533480276943*-Datei Aktivitäten zwischen dem 05.08.2018 und dem 08.08.2018. Die *1533772802308*-Datei hingegen protokolliert Aktivitäten ab dem 09.08.2018. Auch die Namen dieser beiden Dateien wurden im Vergleich zu Szenario 3 um zwei Sekunden aktualisiert. Da

sich bei der zweiten Datei in diesem Ordner (*1533772802308.xml*) auch der Inhalt der Logdatei aktualisiert, ist es möglich festzustellen, wann das Handy zuletzt benutzt wurde. Bei der ersten Datei in diesem Ordner wird jedoch nur der Dateiname aktualisiert, wohingegen der Inhalt gleichbleibt. Dies wiederum würde bei der Berechnung der zuletzt ausgeführten Aktivitäten in dieser Datei zu einem Unterschied von zwei Sekunden führen.

2.3.4.3 Protokolldateien in */data/system/usagestats/0/monthly/*

Ergebnis

Der Zeitstempel der „monthly"-Dateien wurde ebenfalls um zwei Sekunden aktualisiert.

2.3.4.4 Protokolldateien in */data/system/usagestats/0/yearly/*

Ergebnis

Eine neue Datei wurde auch nach acht Tagen wie zu erwarten nicht angelegt. Der Dateiname der alten Datei hat sich jedoch um zwei Sekunden aktualisiert.

2.3.4.5 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Ergebnis

Obwohl das Gerät am 13.08.2018 nicht benutzt wurde (aber angeschaltet war), wurden in der Protokolldatei sowohl ScreenLockReceiver-Ereignisse als auch der Batteriestatus notiert.

2.3.4.6 Ordner *recent_tasks* und *recent_images*

Ergebnis

Es kann festgehalten werden, dass gemäß dieses Szenarios in dem Ordner *recent_tasks* nur *XML*-Dateien angelegt werden, wenn auch tatsächlich eine Aktivität am Handy stattgefunden hat. Ist das Handy angeschaltet ohne benutzt zu werden, scheinen auch keine *XML*-Dateien angelegt zu werden. Weiterhin ändern bzw. aktualisieren sich diese Dateien nicht. Alle Dateien, welche sowohl in Szenario 3 dieser Versuchsreihe als auch in diesem Szenario in dem Ordner *recent_tasks* vorhanden sind, sind identisch.

3 Ergebnisteil

Ziel dieser Bachelorarbeit ist die Untersuchung von drei Ordnern, welche auf der Datenpartition liegen. Zu diesem Zweck wird die Untersuchung an einem Gerätemodell auf drei verschiedenen Betriebssystemversionen getestet. Was genau diese Protokolldateien in den Ordnern protokollieren, gilt es im Methodenteil (Kapitel 2) herauszufinden. Im Folgenden werden die Ergebnisse zusammengefasst.

Zunächst werden die Protokolldateien unter *usagestats* betrachtet. Sowohl bei der Betriebssystemversion 4.4.4 also auch bei den Versionen 6 und 7 werden diese Protokolldateien auf der Datenpartition in dem Ordner */system/usagestats/* gefunden. Der Aufbau der XML-Dateien und die Ordnerstruktur unterscheiden sich in den einzelnen Betriebssystemversionen.

Eine Zusammenfassung der Ergebnisse von Betriebssystemversion 6 und 7 erfolgt zuerst, da diese dieselbe Ordnerstruktur der *usagestats*-Dateien aufweisen und auch der Aufbau der *XML*-Dateien sehr ähnlich ist. Der Ordner *usagestats* hat in diesen Betriebssystemversionen vier Unterordner, welche die Namen „daily“, „weekly“, „monthly“ und „yearly“ tragen. In jedem dieser Ordner befinden sich *XML*-Dateien, welche Daten zu den entsprechenden Intervallen speichern. Allgemein ist festzustellen, dass sich alle Zeitstempel in den Logdateien auf den Dateinamen der Protokolldatei beziehen. Diese Logdateien unterteilen sich in drei Kind-Knoten, welche verschiedene Attribute speichern.

Der erste Knoten wird von den *package*-Elementen gebildet. Das Attribut *lastTimeActive* speichert einen Wert, der angibt, wann die Anwendung das letzte Mal benutzt wurde. Das Attribut *package* speichert den dazugehörigen Paketnamen bzw. Anwendungsnamen. In beiden Betriebssystemversionen nehmen diese Knoten außerdem die Attribute *lastEvent* und *timeActive* an. *TimeActive* gibt die Zeit an, zu der die besagte Anwendung in dem entsprechenden Intervall im Vordergrund steht. Nach Erkenntnissen dieser Bachelorarbeit nimmt das Attribut *lastEvent* einen Wert von „0“ an, wenn die Anwendung vom Nutzer nicht aktiv benutzt wurde und einen Wert von „2“, wenn die Anwendung in den Hintergrund rückt.

Neben den *package*-Elementen werden in diesen Protokolldateien auch *event*-Elemente gespeichert. Dies sind Kind-Knoten der *event-log*-Elemente. Sie existieren nur in den „daily“-Dateien. Bei den „weekly“-, „monthly“- und „yearly“-Dateien bildet das *event-log*-Element ein leeres Element. Die Untersuchungen dieser Bachelorarbeit haben ergeben, dass die *event*-Elemente der „daily“-Dateien eine genauere Protokollierung dessen vornehmen, was in der Anwendung passiert. Diese besitzen neben dem Attribut *package* zudem das Attribut *class*, welches die Aktivität bzw. eine Komponente der Anwendung angibt. Das Attribut *type* bestimmt den Ereignistyp. Das bedeutet, dass dieses Attribut

angibt, ob sich eine Komponente beispielsweise gerade im Vorder- oder Hintergrund befindet. Es stellte sich heraus, dass ein Wert von „1“ angenommen wird, wenn sich die Komponente in den Vordergrund bewegt und ein Wert von „2“, wenn sie sich in den Hintergrund bewegt. Ein Wert von „5“ wird angenommen, wenn es zu Konfigurationsänderungen kommt und ein Wert von „7“, wenn die Anwendung im Hintergrund aktiv war, beispielsweise bei einem Nachrichteneingang.

Neben den beiden erwähnten Elementen werden noch *config*-Elemente angelegt. Diese speichern Konfigurationseinstellungen und Konfigurationsänderungen und werden größtenteils beim Starten des Handys angelegt. Dabei kann festgestellt werden, dass das Attribut *ori* einen Wert von „1“ und „2“ annehmen kann. Der Wert von „1“ steht für eine vertikale Bildschirmausrichtung und der Wert von „2“ für eine horizontale Bildschirmausrichtung. Das Attribut *locales* gibt nach Erkenntnissen dieser Arbeit an, welche Sprache in dem zu untersuchenden Handy eingestellt ist.

Ferner wurde festgestellt, dass die „daily“-Dateien die Ereignisse über einen Tag hinweg protokollieren, während „weekly“-Dateien die Ereignisse über mehrere Tage protokollieren. Eine „weekly“-Datei speichert Daten maximal über sieben Tage bis eine neue „weekly“-Datei angelegt wird. Für die „monthly“- und „yearly“-Dateien konnte aufgrund der begrenzten Zeit nicht genau festgestellt werden, über welchen Zeitraum die Daten abgelegt werden. Es ist jedoch davon auszugehen, dass sie in den „monthly“-Dateien über maximal einen Monat aggregiert werden, während dies in den „yearly“-Dateien über maximal ein Jahr der Fall ist.

Die XML-Datei unter *usagestats* der Betriebssystemversion 4.4.4 hat zwar einen anderen Aufbau als in Version 6 und 7, speichert jedoch ähnliche Ereignisse. Die *XML*-Datei enthält einen Knoten *pkg*, der über das Attribut *name* den Paketnamen bzw. den Namen der Anwendung angibt. Das *pkg*-Element besitzt Kind-Knoten des *XML*-Namens *comp*. Diesen Elementen wird über das Attribut *name* eine Komponente der Anwendung übergeben. In dem Attribut *lrt* wird ein Zeitstempel in Millisekunden in dem Epoch-Zeit-Format gespeichert. Diese *XML*-Datei aktualisiert ihre Zeitstempel nach jeder Benutzerinteraktion mit dem Handy und setzt die entsprechenden Anwendungskomponenten immer auf den aktuellsten Wert. Es wird demnach angegeben, wann eine bestimmte Komponente der Anwendung das letzte Mal benutzt wurde. Weiterhin werden für jeden Tag die Ereignisse in einer Datei des Formates *usage-YYYYMMDD* gespeichert. Zu diesen Ereignissen existieren jedoch im Gegensatz zu der *usage-history.xml* keine Zeitstempel. Außerdem kann durch die Untersuchung festgestellt werden, dass diese Dateien in dem Ordner nicht länger als sieben Tage aufbewahrt werden.

Eine weitere Datei, die im Rahmen dieser Bachelorarbeit untersucht wird, ist die Protokolldatei von WhatsApp. Sie wird in jeder Betriebssystemversion am gleichen Ort abgespeichert und weist immer denselben Aufbau auf. Die Datei liegt auf der Datenpartition in dem Ordner */data/com.whatsapp/files/Logs/*. Zwei Ereignisse dieser Protokolldatei

werden hier genauer betrachtet, nämlich der Batteriestatus und das ScreenLockReceiver-Ereignis. Die Untersuchung hat ergeben, dass das ScreenLockReceiver-Ereignis speichert, zu welchem Zeitpunkt der Bildschirm entsperrt wird. Dabei kann das Attribut *tag* dieses Ereignisses die Werte „on“, „off“ und „unlock“ annehmen. Der Wert „on“ steht für das Aufleuchten des Bildschirms. Dies geschieht beispielsweise bei einer eingehenden Nachricht oder bei einer Trennung der Steckerverbindung zum Ladegerät. Der Wert „off“ bedeutet, dass der Bildschirm gesperrt ist. Wenn der Bildschirm durch den Nutzer entsperrt wird, nimmt *tag* den Wert „unlock“ an. Bei der Protokollierung des Batteriestatus werden vor allem die Werte des Attributes *plugged* genauer untersucht. Das Attribut kann die Werte „0“, „1“ und „2“ annehmen. Der Wert „0“ zeigt an, dass das Handy nicht mit einem Ladegerät verbunden ist. Wert „1“ wird angenommen, wenn das Handy mit einem Netzteil-Ladegerät verbunden ist und ein Wert „2“, wenn es mit einem USB-Kabel verbunden ist.

Die letzten Ordner, die es im Rahmen dieser Bachelorarbeit zu untersuchen gilt, sind die Ordner *recent_images* und *recent_tasks*. Diese Ordner sind unter der Betriebssystemversion 4.4.4 noch nicht vorhanden, jedoch in Version 6 und 7. Bei Betriebssystem Version 6.0.1 liegen sie auf der Datenpartition in dem Ordner *system*, wohingegen sie bei Betriebssystemversion 7.1.2 auf der Datenpartition in dem Ordner *system_ce* zu finden sind.

Die Untersuchungen ergaben, dass in dem Ordner *recent_tasks XML*-Dateien angelegt werden, die in beiden Betriebssystemversionen den gleichen Aufbau vorweisen. In dem Ordner *recent_images* werden Screenshots von Anwendungen gespeichert, die nach einer bestimmten Tasknummer benannt sind. Diese Tasknummern treten bei beiden Betriebssystemversionen auch in dem Ordner *recent_tasks* auf. Im Ordner *recent_images* werden Screenshots der zuletzt verwendeten Anwendungen abgelegt. Wird dieselbe Anwendung am nächsten Tag erneut verwendet, wird der alte Screenshot gelöscht und ein Neuer wird angelegt. Dieser neue Screenshot besitzt dementsprechend auch eine neue Tasknummer.

Betrachtet man abhängig von den Screenshots die Einträge in *recent_tasks*, fällt auf, dass die *XML*-Dateien in ihrem Namen dieselbe Tasknummer aufweisen wie die Screenshots in *recent_images*. Die Untersuchung hat ergeben, dass diese XML-Dateien die Zeitstempel und weitere Informationen zu den Screenshots speichern, beispielsweise die Paketnamen. Das Attribut *last_active_time* speichert den Zeitpunkt des Verlassens der Anwendung. Unter *recent_images* wird zu diesem Vorgang ein Screenshot erstellt.

Screenshots in *recent_images* und die XML-Dateien in *recent_tasks* werden solange gespeichert, bis die Anwendung erneut benutzt wird. Es ist jedoch zu beachten, dass sich die Speicherfristen auf Untersuchungen beziehen, welche maximal eineinhalb Wochen andauerten.

4 Diskussion

In diesem Kapitel geht es um die wissenschaftliche Einordnung dieser Studie. Dafür werden das Vorgehen und die Ergebnisse der Bachelorarbeit mit anderen Publikationen verglichen.

Das Thema Log-Artefakte bei Android-Devices ist ein breit gefächertes Thema, zu dem es eine Vielzahl von Veröffentlichungen gibt. Die meisten Paper zu diesem Thema beschäftigen sich mit hinterlassenen Artefakten durch Instant Messenger Anwendungen. Vor allem WhatsApp und Viber werden dabei genauer untersucht [Anglano2014, Mahajan2013]. Aber auch Anwendungen wie der Telegram- oder der Line-Messenger werden durchforscht [Anglano2017, Chang2017]. Alyahya et al. beispielsweise untersuchen hinterlassene Artefakte von Snapchat auf Android-Geräten [Alyahya2017]. Weitere Paper beinhalten die Analyse von hinterlassenen Artefakten bei Social-Network-Anwendungen [Al Mutawa2012].

Ein Augenmerk gilt vor allem den Publikationen, die eine Analyse der Logdateien vornehmen, welche auch in dieser Bachelorarbeit untersucht werden. Zunächst werden die *Usagestats*-Dateien behandelt, gefolgt von der WhatsApp-Protokolldatei. Schließlich werden die Untersuchungen der Dateien unter *recent_images* und *recent_tasks* mit bestehender Literatur verglichen und es wird dargelegt, welche Erkenntnisse für die Forensik gefolgert werden können sowie welche Probleme bei der Auswertung dieser Dateien auftreten können.

4.1 Dateien im Ordner *usagestats*

4.1.1 Vergleichbare Studien

Allgemein ist zu sagen, dass es nicht viele vergleichbare Studien gibt, die sich mit den Protokolldateien unter *usagestats* ausführlich auseinandersetzen. Dennoch haben einige Artikel diese Dateien erwähnt und in einem kurzen Abschnitt analysiert. Es gibt außerdem Publikationen, welche sich mit den Nutzungsstatistiken von Anwendungen auseinandersetzen [Singh2017, Dutta2016]. Dafür wurde in den jeweiligen Veröffentlichungen eine Anwendung programmiert, welche die Klassen Usagestats und UsagestatsManager implementiert, um eine Aussage darüber treffen zu können, wie lange eine Anwendung im Vordergrund steht [Singh2017]. Neben diesen Studien existiert eine Dokumentation der zugehörigen API, die im späteren Verlauf mit den Ergebnissen dieser Bachelorarbeit verglichen wird. Zunächst werden jedoch die Publikationen diskutiert, welche die Dateien unter *usagestats* genauer analysiert haben.

4.1.1.1 Digital Forensic Analysis Service Report (Chris Cole, Andrew Nind)

Dieses Paper wurde 2017 veröffentlicht und beschäftigt sich mit der Auswertung eines Smartphones. Ziel von Amnesty International und Secure Works ist es herauszufinden, ob eine bestimmte Anwendung auf dem Handy installiert war oder nicht. Dafür werden verschiedene Ordner und Dateien des Smartphones näher betrachtet.

Mittels der Software UFED4PC von Cellebrite wird ein physikalisches Abbild des Smartphones erzeugt. Bei dem zu untersuchenden Gerät handelt es sich um ein GT-N7100 Galaxy Note 2 [Nind2017]. Weiterhin verwendet das Paper ein Testhandy, auf dem die zu untersuchende Anwendung installiert ist, um zu zeigen, in welchen Ordnern die Anwendung Daten hinterlässt [Nind2017].

Bei der Untersuchung konzentriert man sich vor allem auf die Datenpartition und dem auf dieser Partition befindlichen *system*-Ordner. Dabei wird der Ordner *usagestats* genauer untersucht und auch die Dateien *packages.list*, und *batterystats.bin* werden betrachtet. Die auf der Datenpartition unter dem Ordner *data* gespeicherten Anwendungen sind ebenfalls Bestandteil der Untersuchung [Nind2017]. Das zu untersuchende Handy zeigt im Gegensatz zum Testhandy dieses Papers keinen Eintrag zu der besagten Anwendung.

Nachfolgend werden die Daten im Ordner *usagestats* untersucht. Die Analyse dieser Dateien ergibt, dass Nutzungsstatistiken über Anwendungen in diesen Ordnern gespeichert werden [Nind2017]. Auch wird protokolliert, wie lange die Anwendung aktiv war. Auf dem Testhandy können Spuren zu der zu untersuchenden Anwendung durch die *usagestats*-Dateien nachgewiesen werden, während auf dem Untersuchungshandy Einträge zu dieser App in den Dateien fehlen [Nind2017].

Die *batterystats.bin*-Datei, welche Statistiken über den Batterieverbrauch einzelner Anwendungen führt, wird ebenfalls untersucht. Gleichermaßen werden die Dateien *packages.list* und *packages.xml* auf der Datenpartition im Ordner *system* analysiert, welche ebenfalls Details zu installierten Anwendungen enthalten. Nachdem die verschiedenen Ordner untersucht wurden, konnte in diesem Fall kein Nachweis über die Installation der besagten Anwendung erbracht werden.

Es ist zu beachten, dass das Paper keineswegs eine so gründliche Untersuchung der Dateien unter *usagestats* vorgenommen hat, wie es in dieser Bachelorarbeit der Fall ist. Es wurden weder die einzelnen Attribute der Elemente untersucht noch erfolgten weitere Untersuchungen zu diesen Dateien. Festzuhalten ist jedoch, dass das Paper genau wie diese Bachelorarbeit zu dem Ergebnis gekommen ist, dass in *usagestats* einerseits Nutzungsstatistiken der Anwendungen und andererseits die Zeit, wie lange eine Anwendung aktiv war, gespeichert werden. Ferner zeigt diese Studie, dass auch der Nachweis einer möglicherweise zuvor gelöschten Anwendung durch diese Dateien erbracht werden

kann.

4.1.1.2 Google Glass Forensics (Julie Desautels)

Ein weiteres Paper, welches die Dateien unter *usagestat*s aufgreift, beschäftigt sich mit forensischen Artefakten auf Google Glass [Desautels2014]. Google Glass stellt eine Brille mit integrierter digitaler Gerätekomponente dar [Desautels2014]. Über diese Brille ist es dem Nutzer möglich, Fotos und Videos aufzunehmen, Google-Glass-Anwendungen herunterzuladen sowie Google-Suchanfragen durchzuführen [Desautels2014]. Das Paper erklärt, dass es sich bei Google Glass um ein Android-Gerät der Version 4.0.3 handelt.

Um auf alle Daten zugreifen zu können, muss auch Google Glass gerootet sein. Nachdem das Rooten von Google Glass erfolgreich abgeschlossen ist, wird eine Analyse der dort befindlichen Daten durchgeführt. Neben einer Reihe untersuchter Ordner und Dateien, beleuchtet man den Ordner *usagestats*, welcher sich in dem Ordner *system* befindet. Auch hier werden die Daten in einer *XML*-Datei gespeichert. In diesem Paper wird erklärt, dass diese *usagestats*-Dateien Informationen darüber enthalten, welche Tätigkeiten der Nutzer zuletzt auf Google Glass durchgeführt hat [Desautels2014]. Neben dieser Erkenntnis kann festgestellt werden, dass der Zeitstempel in Millisekunden angegeben wird. In dem Ordner *usagestats* befindet sich gemäß des Papers eine *XML*-Datei, welche den Namen *usage-history.xml* trägt. Einigen Abbildungen dieser Veröffentlichung ist zu entnehmen, dass die *usage-history*-Datei aus einem Wurzelelement *usage-history* besteht. Weitere Knoten dieser *XML*-Datei sind *pkg* und *comp* [Desautels2014].

Das Paper gibt, wie in Abbildung 4.1 zu sehen ist, folgendes Beispiel für den Inhalt der Logdatei an.

```
</pkg>
<pkg name="com.google.glass.maps">
<comp name="com.google.glass.maps.NavigationLauncherActivity" lrt="1394993390744" />
<comp name="com.google.glass.maps.DisclaimerActivity" lrt="1394989769257" />
<comp name="com.google.glass.maps.NavigationActivity" lrt="1394993996418" />
</pkg>
```

Abbildung 4.1: Google Glass Inhalt Protokolldatei [Desautels2014]

Zu den abgebildeten Zeilen der Logdatei wird erklärt, dass die „com.google.glass.maps.NavigationLauncherActivity“ mit dem Wert, welcher in *lrt* gespeichert ist, den Zeitpunkt angibt, wann der Nutzer das letzte Mal eine GPS-Wegbeschreibung genutzt hat und wann er an besagtem Ort angekommen ist [Desautels2014]. Gleichermaßen werden die anderen Einträge in dieser Logdatei beschrieben und ausgewertet.

Die Erkenntnisse dieses Papers und der vorliegenden Bachelorarbeit lassen sich gut vergleichen. Beide Studien haben mit der Android-Version 4 gearbeitet. Der Aufbau der *usage-history*-Datei in dieser Studie stimmt mit dem Aufbau der XML-Datei aus Versuchsreihe 1 dieser Bachelorarbeit überein. Es besteht sowohl eine Übereinstimmung der

Elemente als auch der übergebenen Attribute. Beide Studien haben gezeigt, dass die Daten, welche in der *usage-history*-Datei unter *usagestats* gespeichert sind, protokollieren, wann eine Anwendung das letzte Mal benutzt wurde. Weiterhin kamen beide zu dem Ergebnis, dass der Zeitstempel dabei in Millisekunden angegeben wird.

4.1.1.3 Smartphone data evaluation model: Identifying authentic smartphone data (Heloise Pieterse, Martin Olivier, Renier van Heerden)

Die nächste Publikation, die sich mit den Dateien unter *usagestats* auseinandersetzt, trägt den Titel „Smartphone data evaluation model: Identifying authentic smartphone data“. Dieses Paper legt den Schwerpunkt auf Daten, welche von Anwendungen erstellt werden, und auf Techniken, die die Authentizität der Daten feststellen [Pieterse2018]. Dafür werden verschiedene Smartphones mit unterschiedlichen Betriebssystemversionen verwendet. Bei der späteren Analyse der Handys werden sowohl Originaldaten als auch manipulierte Daten ausgewertet. Bei einem der Experimente wird ein Samsung Galaxy S5 mini benutzt, auf welchem das Betriebssystem Android 6.0.1 läuft. Während der Untersuchung befindet sich das Gerät in einem gerooteten Zustand und dieses Experiment fokussiert sich auf die Auswertung und Feststellung manipulierter Daten.

In der Analyse wird der Schwerpunkt vor allem auf die Standard-Messaging-Anwendung unter Android gelegt [Pieterse2018]. Durch die Analyse der Dateien unter */data/system/usagestats/0/* ergaben sich gemäß dieser Publikation die nachfolgenden Erkenntnisse. Zum einen wird in dieser Datei angezeigt, wann eine Anwendung das letzte Mal verwendet wurde. Außerdem werden die Zeitstempel für verschiedene Intervalle erfasst, nämlich für Tages-, Wochen-, Monats- und Jahresintervalle [Pieterse2018]. Ein weiteres Resultat beschreibt, dass der jeweilige Name der Protokolldatei angibt, wann die Datenerfassung begonnen hat [Pieterse2018]. Aus diesem Grund wurde gefolgert, dass man das Attribut *lastTimeActiveSystem* auf den Dateinamen addieren muss, um zu bestimmen, wann die Anwendung zuletzt verwendet wurde [Pieterse2018]. Ferner kann auch die Addition von Dateiname und Ereigniswert zum gleichen Ergebnis führen, wie in Abbildung 4.2 des Papers zu sehen ist [Pieterse2018].

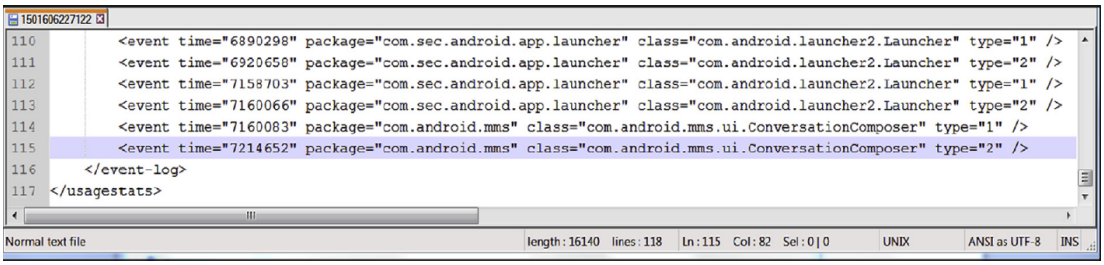

Abbildung 4.2: Ausschnitt Protokolldatei [Pieterse2018]

Sowohl in dieser Studie als auch in der vorliegenden Bachelorarbeit wird bei je einem Experiment die Betriebssystemversion 6.0.1 verwendet. Es ist jedoch zu erwähnen, dass

die Gerätemodelle der Bachelorarbeit und dieser Studie nicht übereinstimmen. Der Ausschnitt in Abbildung 4.2 zeigt den gleichen Aufbau der Logdatei, welcher auch in dieser Bachelorarbeit unter Verwendung desselben Betriebssystems festgestellt werden konnte.

Ein Unterschied zwischen beiden Arbeiten besteht in der Interpretation der Ergebnisse. Beide Studien haben herausgefunden, dass die Zeit, wann eine Anwendung das letzte Mal genutzt wurde, durch die Addition von Dateinamen und Ereigniswert der *event*-Elemente bestimmt werden kann. Was die Berechnung bei den *package*-Elementen angeht, ist man jedoch auf unterschiedliche Ergebnisse gekommen.

Diese unterschiedlichen Ergebnisse werden anhand der SuperSu-Anwendung, welche in der Veröffentlichung von Pieterse et al. untersucht wird, erklärt. Ein Ziel der Studie von Pieterse et al. besteht darin zu klären, wann die SuperSu-Anwendung das letzte Mal benutzt wurde. In dem Paper wurde, wie in Abbildung 4.3 dargestellt, folgender Eintrag in der Logdatei zu dieser Anwendung gefunden:

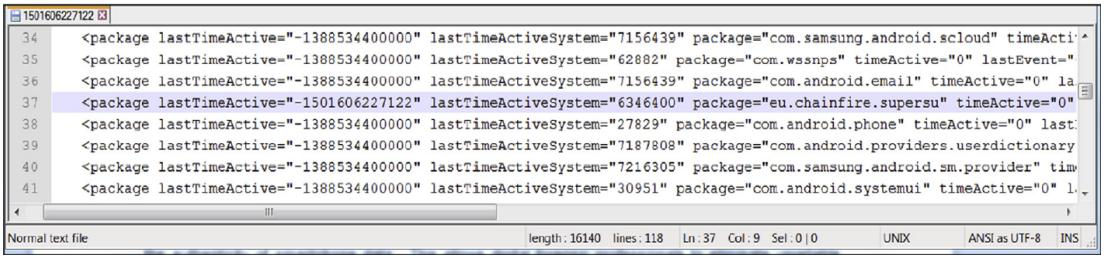

Abbildung 4.3: SuperSU-Anwendung [Pieterse2018]

Der Dateiname zu dieser Protokolldatei ist in dem Paper nicht mit angegeben. Das Paper berichtet jedoch, dass man durch die Addition des Dateinamens und des Wertes *lastTimeActiveSystem* auf einen Wert kommt, der aussagt, wann die Anwendung das letzte Mal benutzt wurde. Dabei wird in dieser Studie weder auf den dazugehörigen *timeActive*- noch auf den *lastEvent*-Wert eingegangen.

Die Ergebnisse dieser Bachelorarbeit können nicht bestätigen, dass man durch die Addition des *lastTimeActiveSystem*-Wertes mit dem Dateinamen auf einen Wert kommt, der dem entspricht, wann die Anwendung das letzte Mal genutzt wurde. Ein Gegenbeispiel dafür ist der Abbildung 4.4 zu entnehmen.

```
<package beginIdleTime="627197" lastEvent="0" timeActive="0" package="com.android.providers.calendar" lastTimeActiveSystem="23718172"
    lastTimeActive="-1533513600000"/>
<package beginIdleTime="428183" lastEvent="0" timeActive="0" package="com.android.providers.media" lastTimeActiveSystem="23519124"
    lastTimeActive="-1533513600000"/>
<package beginIdleTime="714508" lastEvent="2" timeActive="282043" package="com.whatsapp" lastTimeActiveSystem="23805482"
    lastTimeActive="23805482"/>
```

Abbildung 4.4: VR 3, S 1 *package*-Elemente [Quelle: eigene Abbildung]

An dem besagten Tag wurde die Kalender-App nicht benutzt. Dennoch weist der *last-*

TimeActiveSystem-Wert einen positiven Wert auf, wohingegen der *lastTimeActive*-Wert negativ ist. Dieselbe Eigenschaft weist auch der Eintrag zu der Anwendung SuperSu in Abbildung 4.3 auf. WhatsApp hingegen wurde beispielsweise an diesem Tag benutzt und weist den gleichen Wert bei den Attributen *lastTimeActive* und *lastTimeActiveSystem* auf.

Die Bachelorarbeit kam zu dem Resultat, dass die Werte von *lastTimeActiveSystem* und *lastTimeActive* immer dann gleich sind, wenn der *lastTimeActive-Wert* positiv ist. Aus einem positiven *lastTimeActive*-Wert kann nach Erkenntnissen dieser Arbeit geschlossen werden, dass die besagte Anwendung in irgendeiner Weise aktiv war. Es muss jedoch noch der *lastEvent*-Wert betrachtet werden, um aussagen zu können, ob der Nutzer die App aktiv genutzt hat oder nicht. Ist der *lastTimeActive*-Wert negativ, weist der *lastTimeActiveSystem*-Wert eine Zeit auf, die der entspricht als das Smartphone gestartet wurde. Es kann demnach nicht allein an dem *lastTimeActiveSystem*-Wert festgemacht werden, wann die Anwendung zuletzt benutzt wurde. Dies haben zumindest die Untersuchungen dieser Bachelorarbeit ergeben.

4.1.1.4 Vergleich mit der API

Wie in dieser Studie deutlich geworden ist, hat sich der Aufbau und die Ordnerstruktur der *usagestats*-Dateien von Betriebssystemversion 4.4.4 zu 6.0.1 verändert. Der Android-Dokumentation zufolge gibt es seit Betriebssystemversion 5.0 eine neue API, die *android.app.usage* API [URL_1]. Diese API hat die Aufgabe, Nutzungsdaten für Anwendungen zu sammeln. Diese Nutzungsdaten werden in Intervallen gespeichert. Es wird zwischen Tages-, Wochen-, Monats- und Jahresintervallen unterschieden [URL_1]. Gemäß der Dokumentation werden die Daten in den jeweiligen Intervallen für einen bestimmten Zeitraum abgelegt. Tägliche Daten werden dabei maximal sieben Tage aufbewahrt, wöchentliche Daten maximal vier Wochen, monatliche Daten maximal sechs Monate und jährliche Daten maximal zwei Jahre [URL_1]. Je nach Intervall erfolgt eine unterschiedliche Protokollierung der Daten. Zum einen wird in diesen Intervallen gespeichert, wann eine Anwendung das letzte Mal benutzt wurde, und zum anderen die Gesamtzeit der Vordergrundaktivität einer Anwendung [URL_1]. Darüber hinaus werden Änderungen der Gerätekonfiguration sowie das Wechseln einer Komponente zwischen Vorder- und Hintergrund protokolliert. Eine Komponente wird dabei entweder durch Paket- oder Aktivitätsnamen identifiziert [URL_1]. Die API besteht aus verschiedenen Klassen. Neben der Klasse ConfigurationStats gibt es außerdem die Klassen UsageStats, UsageEvents.Event und UsageStatsManager und viele weitere [URL_2].

Die Klasse UsageEvents.Event gibt einen Überblick über die möglichen Ereignistypen, die ein Event annehmen kann. Der Klassendokumentation zufolge wird ein Ereignis angelegt, wenn sich der Zustand einer Komponente ändert [URL_8]. Dabei wird der Zeitpunkt des Eintretens eines Ereignisses in Millisekunden angegeben. Je nach API-Level sind unterschiedliche Ereignistypen möglich. Tabelle 4.1 gibt einen Überblick über eini-

ge solcher Ereignistypen.

Tabelle 4.1: Ereignistypen nach API [Quelle: eigene Tabelle]

Ereignistyp	**Wert**	**API-Level**
Änderung der Gerätekonfiguration [URL_8]	5 [URL_8]	21 [URL_8]
Komponente, die sich in den Hintergrund verschoben hat [URL_8]	2 [URL_8]	21 [URL_8]
Komponente, die sich in den Vordergrund verschoben hat [URL_8]	1 [URL_8]	21 [URL_8]
Bildschirm ist in interaktiven Zustand übergegangen [URL_8]	15 [URL_8]	28 [URL_8]
Paket wird in irgendeiner Weise vom Nutzer angesprochen [URL_8]	7 [URL_8]	23 [URL_8]

Neben den in dieser Tabelle aufgeführten Ereignisse existieren noch weitere.

Eine weitere Klasse, welche von der android.app.usage API verwendet wird, ist die Klasse „Configuration“. Nach der Klassendokumentation beschreibt die Klasse neben Gerätekonfigurationen auch benutzerspezifische Konfigurationen [URL_5]. Auch diese Klasse bringt verschiedene Methoden und Variablen mit sich. In Tabelle 4.2 ist ein Überblick über einige dieser Variablen gegeben.

Tabelle 4.2: „Configuration“-Klasse API [Quelle: eigene Tabelle]

Feld	**Bedeutung**	**API-Level**
Mcc [URL_5]	Mobile Country Code [URL_5]	1 [URL_5]
Mnc [URL_5]	Mobile Network Code [URL_5]	1 [URL_5]
Orientation [URL_5]	Gesamtausrichtung des Bildschirms [URL_5]	1 [URL_5]
screenLayout [URL_5]	Bitmaske des Gesamtlayouts des Bildschirms [URL_5]	4 [URL_5]

Dem Feld „orientation“ können verschiedene Werte zugeordnet werden. Die Werte dieser Variable ändern sich der Dokumentation zufolge, wenn sich der Bildschirm dreht [URL_3]. Dabei kann ein Wert von „1“ oder „2“ angenommen werden. Der Wert „1“ bezieht sich auf eine vertikale Bildschirmausrichtung, der Wert „2“ auf eine horizontale Bildschirmausrichtung [URL_5].

Betrachtet man die Ereignistypen der Dokumentation genauer, ist festzustellen, dass die verschiedenen Typen auch im Zuge dieser Bachelorarbeit in den Dateien von *usagestats* auftreten. In den Betriebssystemversionen sechs und sieben waren es die Ereignistypen

mit den Werten „1“, „2“, „5“ und „7“. Ereignistyp „2“ erscheint, wenn die Anwendung in den Hintergrund tritt. Dies entspricht auch der Dokumentation von Android. Wert „5“ wird angenommen, wenn es zu Änderungen in den Konfigurationseinstellungen kommt. Ein Wert von „1“ tritt auf, wenn die Komponente in den Vordergrund rückt. Der Wert „7“ kann im Rahmen dieser Bachelorarbeit immer dann beobachtet werden, wenn beispielsweise eine SMS oder eine WhatsApp-Nachricht eingeht, der Benutzer aber nicht aktiv am Handy tätig ist. Außerdem kommt auch diese Bachelorarbeit zu dem Ergebnis, dass die Komponenten einer Anwendung entweder durch ihren Aktivitäts- oder Paketnamen identifiziert werden.

Auch die Klasse „Configuration“ speichert Parameter, die in den Dateien von *usagestats* wiedergefunden werden. In den Untersuchungen hat sich beispielsweise der Wert von „orientation“ geändert, wenn die Ansicht einer Anwendung gedreht wurde. Der Wert von „2“ wird angenommen, wenn der Bildschirm von einer vertikalen Ausgangslage in eine horizontale Ausgangslage gedreht wird. Genau diese Beobachtung wird auch in der Dokumentation von Android notiert. Neben dem Feld „orientation“ sind noch viele weitere in der Dokumentation beschrieben.

4.1.2 Folgerungen für die Forensik und Probleme

Bezüglich der zu untersuchenden Protokolldateien ergaben sich vor allem bei Betriebssystemversion 6 und 7 Unklarheiten. Bei beiden Betriebssystemen werden nach einer gewissen Zeit die Zeitstempel der Dateinamen von den Dateien in den Ordnern „daily“, „weekly“, „monthly“ und „yearly“ aktualisiert.

Bei den Dateien, die gleichzeitig ihren Inhalt aktualisiert haben, erwies sich diese Angelegenheit als nicht besonders problematisch, weil immer noch festgestellt werden kann, wann das Handy zuletzt benutzt wurde. Von der Aktualisierung des Dateinamens waren jedoch auch Dateien betroffen, deren Inhalt sich nicht geändert hat.

Da sich die Zeitstempel innerhalb der Protokolldatei auf den Dateinamen beziehen kann somit nicht mehr festgestellt werden, wann genau die Ereignisse stattgefunden haben. Es ist jedoch zu erwähnen, dass es sich bei der Aktualisierung „nur“ um eine Zeitdifferenz von wenigen Sekunden handelt. Bei Betriebssystemversion 7.1.2 wurde diese Verschiebung nur bei einer Datei beobachtet, während bei Betriebssystemversion 6 alle Dateien in den jeweiligen Ordnern betroffen waren. Auch im Internet herrscht großer Diskussionsbedarf darüber, ob den Ergebnissen dieser Dateien zu trauen ist.

Weiterhin scheint die Benennung der Dateinamen in den Betriebssystemversionen unterschiedlich. In Betriebssystemversion 6.0.1 weisen die Dateinamen meist einen Zeitstempel von 02:00 Uhr nachts auf. Unter Betriebssystemversion 7.1.2 nehmen die Dateinamen eine Zeit zwischen 7:00 und 9:00 Uhr morgens an. Zudem fällt die Anzahl der Dateien,

die nach der ersten Benutzung des Handys in den jeweiligen Ordnern angelegt werden, unterschiedlich aus. Manchmal wird in den „weekly“-, „monthly“- und „yearly“- Ordnern nur eine Datei angelegt, gelegentlich kommt es aber auch vor, dass sich nach dem ersten Versuch zwei oder drei Dateien in diesen Ordnern befinden.

Bei Betriebssystemversion 4.4.4 traten diese Probleme nicht auf. Hier haben sich die Zeitstempel der Protokolldatei immer dann aktualisiert, wenn die besagte Anwendung benutzt wurde.

Im Rahmen dieser Bachelorarbeit wurden die Protokolldateien unter *usagestats* an nur einem Gerätemodell untersucht. Es stellt sich die Frage, ob sich der Aufbau der Protokolldatei bei anderen Gerätemodellen unterscheiden könnte. Nach dem Vergleich mit anderen Publikationen kann jedoch festgestellt werden, dass sowohl bei Betriebssystemversion 4 als auch bei Betriebssystemversion 6.0.1 das Gerätemodell die Struktur der Protokolldateien nicht beeinflusst [Desautels2014, Pieterse2018].

Für die Forensik sind die Resultate dieser Dateien durchaus interessant. Die *usagestats*-Dateien ermöglichen festzustellen, ob eine Anwendung auf dem Gerät installiert war und ob diese benutzt wurde. Dies gilt für alle untersuchten Betriebssysteme. Außerdem kann man bei allen Versionen nachvollziehen, welche Komponenten einer Anwendung aktiv waren. Des Weiteren kann durch die Dateien in Betriebssystemversion 4 genau gesagt werden, wann die Komponenten einer Anwendung das letzte Mal benutzt wurden.

Wie in der Veröffentlichung „Google Glass Forensics“ von Julie Desautels [Desautels2014] dargelegt, kann unter der Betriebssystemversion 4 von Android mithilfe der Maps-Aktivität bestimmt werden, wann diese zuletzt benutzt wurde und wann der Nutzer an einem bestimmten Ziel angekommen ist. Die Lokation selbst lässt sich jedoch aus dieser Protokolldatei nicht erschließen.

4.2 WhatsApp

4.2.1 Vergleichbare Studien

Neben dieser Studie gibt es eine Vielzahl von Veröffentlichungen zum Thema Log-Artefakte bei Android Devices, insbesondere solche, die von Instant Messaging-Anwendungen erzeugt werden. Vor allem WhatsApp und Viber wurden in der Vergangenheit gründlich untersucht. Was WhatsApp angeht, konzentrieren sich die meisten Paper auf die in den Datenbanken *msgstore.db* und *wa.db* abgelegten Daten. Oft wird die eigentliche Protokolldatei, welche in */data/data/com.whatsapp/files/Logs/* liegt, nur geringfügig besprochen. Zwei bis drei Publikationen haben jedoch eine gründlichere Untersuchung dessen vorgenommen, was in der Protokolldatei abgelegt wird.

4.2.1.1 Analysis of WhatsApp log file for information retrieval (Harjinder Singh, Mr.Ashwani Kumar)

Ein Paper, welches sich genauer mit der WhatsApp-Protokolldatei auseinandersetzt, heißt „Analysis of Whatsapp log file for information retrieval“ und wurde im April 2018 veröffentlicht [Singh2018]. Ziel dieser Publikation ist die Analyse forensischer Artefakte der WhatsApp-Protokolldatei. Dabei soll eine Verknüpfung der vorgefundenen Beweise mit realen Ereignissen stattfinden [Singh2018].

Das Vorgehen dieser Untersuchung ähnelt dem Vorgehen der Untersuchung dieser Bachelorarbeit. Es wird ein Testhandy der Marke Samsung (GT-L19001T) mit der Betriebssystemversion 4.1.2 verwendet. Bevor die Experimente mit dem Testhandy durchgeführt werden, wird es über die eingebaute Funktionalität auf die Werkseinstellung zurückgesetzt. Nach dem abgeschlossenen Factory Reset wird WhatsApp installiert und es werden verschiedene Szenarien erprobt.

Um Zugriff auf die zu sichernde Protokolldatei zu bekommen, wird das Smartphone mittels Dr.Phone gerootet [Singh2018]. Die Sicherung der Protokolldatei erfolgt über *adb pull*. Unter */data/data/com.whatsapp/files/Logs* werden drei Dateien gefunden. Neben der Protokolldatei „whatsapp“ befinden sich außerdem noch zwei gezippte Dateien in diesem Ordner [Singh2018].

Das Paper und die Bachelorarbeit sichern die Daten über *adb pull*. Auch in der Durchführung der verschiedenen Szenarien gibt es Gemeinsamkeiten. In beiden Arbeiten werden typische Nutzeraktionen wie das Senden von Textnachrichten oder das Hinzufügen eines Kontaktes betätigt.

Die zu vergleichende Studie tauscht neben einfachen Textnachrichten auch Standorte und Bilder aus [Singh2018]. Singh et al. veranschaulichen in einer Tabelle, wann welche Aktivitäten ausgeführt werden. Die Zeitangabe in dieser Tabelle ist nur minutengenau gemessen und es geht aus dem Paper nicht hervor, womit die Zeit gemessen wurde [Singh2018].

Unterschiede der Arbeiten liegen darin, dass die Protokolldatei von WhatsApp auf andere Ereignisse untersucht wird. Trotz der unterschiedlichen Schwerpunkte können gemeinsame Erkenntnisse festgehalten werden.

Zum einen ergibt sich sowohl in dieser Bachelorarbeit als auch in dem zu vergleichenden Paper die Erkenntnis, dass die WhatsApp-Protokolldatei die Ereignisse in Unix-Epochzeitformat ablegt. Das zu vergleichende Paper stellt außerdem fest, dass die Datei eine Protokollierung aller Ereignisse, welche in WhatsApp stattfinden, notiert [Singh2018]. Singh et al. notieren, dass ein- und ausgehende Nachrichten sowie Sprachanrufe und deren Dauer protokolliert werden [Singh2018]. Auch der übertragene Medientyp

wird durch die Protokolldatei festgehalten. Es ist gemäß den Ergebnissen des Papers also möglich nachzuvollziehen, ob ein Bild, eine Lokation oder eine Textnachricht gesendet wird [Singh2018] und ob eine Nachricht gelöscht wurde. Der Inhalt der Nachricht wird allerdings nicht aufgezeigt, sondern nur die ID der Nachricht [Singh2018].

Zudem kommen beide Studien zu dem Resultat, dass die Einträge der Logdatei mit realen Ereignissen zusammenhängen und die Logdatei von der Anwendung automatisch aktualisiert wird [Singh2018].

Abschließend ist anzumerken, dass Singh et al. mehr Ereignisse der Logdatei untersucht haben als es im Rahmen dieser Bachelorarbeit möglich war. Beide Arbeiten zeigen aber, dass die Protokolldatei forensisch interessante Informationen beinhalten kann.

4.2.1.2 Forensic Analysis of WhatsApp Messenger on Android Smartphones (Cosimo Anglano)

Zwei weitere Paper beschäftigen sich mit der Analyse von WhatsApp-Artefakten und sind in ihren Ergebnissen sehr ähnlich. Aus diesem Grund wird nur eine der beiden Studien näher beleuchtet, nämlich die „Forensic Analysis of WhatsApp Messenger on Android Smartphones“ von Cosimo Anglano von 2014 [Anglano2014]. Die andere Publikation heißt „Forensic Analysis of WhatsApp on Android Smartphones“ [Singh2017].

Im Folgenden wird die Publikation von Cosimo Anglano genauer beschrieben. Ziel dieser Arbeit ist es, die von WhatsApp hinterlassenen Artefakte genauer zu untersuchen [Anglano2014].

Bevor eine Analyse der Testgeräte erfolgt, werden verschiedene Szenarien durchgeführt. Die Versuche werden jedoch nicht an physischen Handys durchgeführt, sondern mit softwareemulierten Geräten [Anglano2014]. Dabei dient YouWave als Virtualisierungsplattform, welche in der Lage ist, das Verhalten eines Android-Gerätes nachzubilden [Anglano2014]. Außerdem werden mehrere YouWave-Maschinen benötigt, um die verschiedenen in dem Szenario beteiligten Geräte darzustellen. Jedes Gerät übernimmt dabei entweder die Rolle des Senders oder die des Empfängers.

Nach jedem Experiment werden alle beteiligten Geräte untersucht. Bei der Analyse wird neben den Datenbanken *msgstore.db* und *wa.db* auch die Protokolldatei von WhatsApp näher betrachtet. Ein Ziel dieser Studie besteht darin, die Artefakte in der Protokolldatei mit den Daten der Datenbanken in Verbindung zu bringen. Es kann festgestellt werden, dass mehrere Ereignisse protokolliert werden, wenn ein Kontakt zur Datenbank hinzugefügt wird.

Cosimo Anglano belegt in seiner Studie, dass es möglich ist, blockierte Kontakte zu iden-

tifizieren, da beim Blockieren eine WhatsApp-ID protokolliert wird. Beim Entsperrungsvorgang ist dies anders, hier wird keine ID notiert. Somit kann bei mehreren blockierten Kontakten aus dem Entsperrungsereignis nicht hervorgehen, welcher Kontakt entsperrt wurde [Anglano2014]. Durch die Protokolldatei von WhatsApp ist man folglich in der Lage festzustellen, ob und wann ein bestimmter Benutzer blockiert wurde. Ob dieser zu einem bestimmten Zeitpunkt noch gesperrt ist, kann nur unter bestimmten Umständen herausgefunden werden.

Zum einen ist dies der Fall, wenn die Protokolldatei nach der Sperrung keine weiteren Entsperrungsereignisse notiert [Anglano2014]. Die zweite Möglichkeit besteht darin, dass nur ein Nutzer blockiert wurde und nach der Blockade nur ein Entsperrungsereignis aufgetreten ist. In diesem Falle ist genau bestimmbar, wann der entsprechende Nutzer entsperrt wurde [Anglano2014]. Allgemein gilt, dass solche Aussagen nur getroffen werden können, wenn die Protokolldatei noch über diese Einträge verfügt.

Zudem stellt der Verfasser des Papers fest, dass sich unter Umständen auch gelöschte Kontakte ermitteln lassen. Zu diesem Zweck rekonstruiert man die in der Vergangenheit hinzugefügten Kontakte und vergleicht sie mit der *wa.db* Datenbank [Anglano2014].

In beiden Arbeiten werden die gleichen Protokolldateien ausgewertet, sie unterscheiden sich aber im Vorgehen und der Ergebnisfindung. In dieser Bachelorarbeit wird nur ein Gerät untersucht, während in der Studie alle am Szenario beteiligten Geräte ausgewertet werden [Anglano2014]. Die Studie ermittelt das Verhalten der Artefakte anhand einer Betriebssystemversion, nämlich der Version Android 4, die Bachelorarbeit stützt sich auf mehrere Betriebssystemversionen.

Vergleicht man die Ergebnisse, ergibt sich, dass die Studie weder auf die Batterieereignisse noch auf die ScreenLockReceiver-Ereignisse eingeht. Das Paper untersucht andererseits WhatsApp-Artefakte intensiver, neben der Protokolldatei werden beispielsweise auch die Datenbanken beleuchtet.

Beide Arbeiten stellen fest, dass die Ereignisse in der Protokolldatei nicht dauerhaft gespeichert werden, sondern nach einer bestimmten Zeit gelöscht werden. Auch steht der hohe Stellenwert in forensischer Hinsicht für beide außer Frage, was die Untersuchung von WhatsApp Protokolldateien bei der Auswertung eines Smartphones betrifft.

4.2.1.3 Forensic Analysis of Instant Messenger Applications on Android Devices (Aditya Mahajan, M. S. Dahiya, H. P. Sanghvi)

Diese Studie wurde 2013 durchgeführt. Ziel des Papers besteht darin, herauszufinden, welche Daten von den Instant Messenger Applikationen WhatsApp und Viber abgelegt werden und ob diese im internen Speicher vorzufinden sind oder nicht [Mahajan2013].

Für die Untersuchungen werden fünf Handys mit drei Betriebssystemversionen getestet. Die Betriebssystemversionen sind Froyo, GingerBread und IceCreamSandwich [Mahajan2013]. Für die Analyse wird UFED verwendet.

Bevor die Smartphones untersucht werden, wird mit jedem Gerät eine Reihe verschiedener Aktivitäten durchgeführt, unter anderem das Austauschen von Chat-, Audio- und Videonachrichten. In dieser Studie werden sowohl die Anwendung WhatsApp als auch Viber kontinuierlich für zwei Wochen genutzt. Dabei sind die Anwendungen entweder bereits auf dem Handy installiert oder sie werden im Google-PlayStore heruntergeladen [Mahajan2013]. Falls sie bereits installiert sind, bedeutet dies, dass sie durch den Nutzer bereits verwendet wurden. In diesem Fall werden die Nutzer anschließend nach den Aktivitäten gefragt, die sie bis dahin ausgeführt haben. In drei dieser fünf Testhandys waren die Anwendungen bereits installiert und wurden von Nutzern schon drei Monate lang verwendet. Bei den übrigen zwei Handys wird nur Viber aus dem PlayStore heruntergeladen, denn WhatsApp war ebenfalls schon installiert. Einige der Handys befinden sich im gerooteten Zustand, andere nicht [Mahajan2013].

Die Analyse von WhatsApp konzentriert sich neben den angelegten Datenbankdateien auch auf die Protokolldatei von WhatsApp. Dadurch, dass dieselben Benutzeraktivitäten an den verschiedenen Handys durchgeführt werden, soll festgestellt werden, ob sich die Verzeichnisstruktur unterscheidet [Mahajan2013]. Man findet sowohl heraus, was in den Datenbanken von WhatsApp gespeichert wird, als auch, dass die Protokolldatei von WhatsApp Echtzeitereignisse protokolliert [Mahajan2013]. Auch Viber speichert Nachrichten und Anruf-Verläufe [Mahajan2013].

Beide Arbeiten weisen Gemeinsamkeiten und Unterschiede auf. Die Hardwareausstattung der Studie bezieht sich auf fünf verschiedene Handys, auf denen drei verschiedene Betriebssystemversionen verteilt sind. Die Sicherung der Daten erfolgt - im Gegensatz zu dieser Bachelorarbeit - über UFED und nicht über ADB-Pull.

Eine Gemeinsamkeit besteht darin, dass vor der Analyse der Testhandys bei beiden Studien bestimmte Szenarien durchgeführt werden. Außerdem befinden sich die Handys in dem zu vergleichenden Paper nur teilweise im gerooteten Zustand, während das Testhandy in dieser Bachelorarbeit immer gerootet ist.

Die Studie befasst sich weniger gründlich mit der Auswertung der Protokolldatei. Hier

werden keine genaueren Untersuchungen der Ereignisse, die in der Protokolldatei protokolliert sind, unternommen. Beide Arbeiten stellen fest, dass Echtzeitereignisse protokolliert werden, die vorliegende Bachelorarbeit geht hier im Vergleich genauer auf einige Ereignisse, die in der Protokolldatei notiert wurden, ein.

4.2.2 Folgerungen für die Forensik und Probleme

Aus der Analyse der WhatsApp-Protokolldatei können für die Forensik mehrere Schlüsse gezogen werden. Zum einen ist diese Datei auf allen Betriebssystemversionen gleich aufgebaut. In dieser Studie wird auf den protokollierten Batteriestatus und auf die ScreenLockReceiver-Ereignisse ein besonderes Augenmerk gelegt. Besonders das ScreenLockReceiver-Ereignis kann wertvoll sein, denn die WhatsApp-Protokolldatei protokolliert mit diesem Ereignis, wann der Bildschirm das letzte Mal entsperrt wurde. Ein entsperrtes Smartphone wiederum kann als Indiz dafür angesehen werden, wann das Handy zum letztem Mal benutzt wurde.

Auch die Protokollierung des Batteriestatus ist für die Forensik von Interesse. Da dieser regelmäßig protokolliert wird, ist es möglich nachzuvollziehen, wann ein Gerät wegen eines zu schwachen Akkus ausgegangen ist.

4.3 Die Ordner *recent_images* und *recent_tasks*

4.3.1 Vergleichbare Studien

Zu den Ordnern *recent_images* und *recent_tasks* gibt es keinen Forschungsstand, auf dem man aufbauen könnte. Lediglich in einer Publikation wird der Ordner *recent_tasks* näher besprochen [Parikh2015]. Der Ordner *recent_images* wird in keiner anderen Literatur genauer untersucht.

4.3.1.1 Analysis of Android Smart Watch Artifacts (Shreyas Parikh, Dhaval Chavda, Shourjo Chakraborty, Dr. Parag H. Rughani, Dr. M. S. Dahiya)

Dieses Paper beschäftigt sich mit der Auswertung von Android-Smartphone-Watches. Eine Smart-Watch ist eine Uhr, die dauerhafte Verbindung zum Android-Smartphone hält [Parikh2015]. Dadurch kann die Uhr wertvolle Informationen über den Nutzer enthalten. Ziel dieser Studie ist es, interessante Artefakte von Smart-Watches herauszufinden.

Für die Untersuchung wird eine Sony Smartwatch 3 mit der Android-Version 5.0.2 verwendet. Die zu untersuchende Uhr ist gerootet. Zuerst wird ein 1:1 Abbild erstellt. Bei der Analyse wird auch der Ordner *recent_tasks* untersucht, der auf der Datenpartition in

dem Ordner *system* zu finden ist.

Das Paper kommt zu der Erkenntnis, dass der Ordner eine Liste von verschiedenen Aufgaben im *XML*-Format enthält [Parikh2015]. Des Weiteren wird festgestellt, dass für jede Aufgabe eine separate *XML*-Datei angelegt wird, die sowohl den Paketnamen als auch die erste und letzte aktive Zeit der Anwendung enthält [Parikh2015]. In Abbildung 4.5 ist ein Ausschnitt dieser XML-Datei zu sehen.

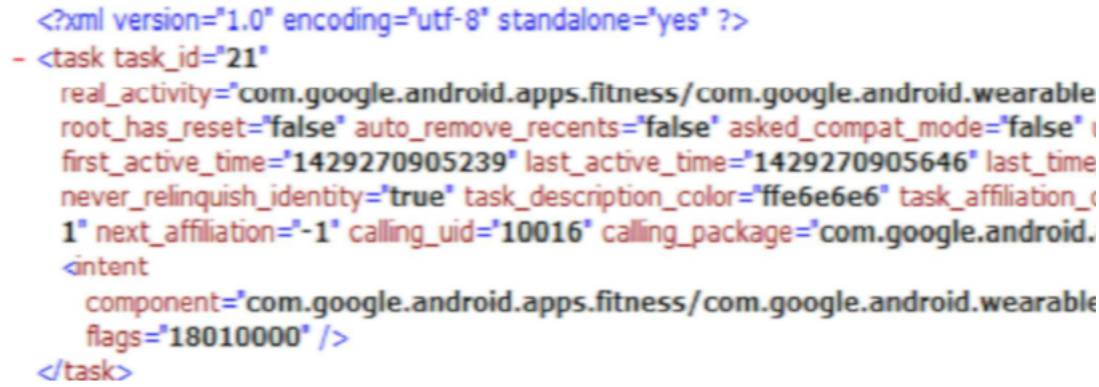

Abbildung 4.5: Ausschnitt *recent_tasks* Smart-Watch [Parikh2015]

Im Unterschied zu diesem Paper wurde der Ordner *recent_tasks* in dieser Bachelorarbeit unter Betriebssystemversion 6.0.1 und 7.1.2 untersucht. Wie der Abbildung 4.5 zu entnehmen ist, werden die Zeitwerte in Millisekunden nach Epoch-Zeit abgespeichert. Auch der Aufbau der XML-Datei aus dem Paper entspricht dem Aufbau, welcher in dieser Bachelorarbeit festgestellt wurde. Die Erkenntnis, dass zu jeder verwendeten Anwendung eine separate *XML*-Datei angelegt wird, kann durch die Untersuchungen in dieser Bachelorarbeit bestätigt werden. Ein Unterschied zwischen beiden Arbeiten ergibt sich aus der Verbindung zwischen den Ordnern *recent_tasks* und *recent_images*. Diese wird in dem Paper nicht hergestellt, der Ordner *recent_images* wird nicht einmal erwähnt.

4.3.2 Folgerungen für die Forensik und Probleme

Die Zeitwerte in den *recent_tasks*-Dateien geben durch das Attribut *last_active_time* den Wert an, an dem die Anwendung das letzte Mal benutzt wurde. Somit ist es möglich nachzuvollziehen, welche Aktivitäten der Nutzer zuletzt am Handy durchgeführt hat. Zusätzlich werden in dem Ordner *recent_images* Screenshots zu diesen Anwendungen angelegt. Dadurch ist es unter Umständen möglich herauszufinden, ob eine Anwendung auf dem Handy installiert war oder nicht, denn nach Erkenntnissen dieser Bachelorarbeit wird zu jeder benutzten Anwendung ein Screenshot angelegt.

4.4 Fazit und Ausblick

Zusammenfassend kann man sagen, dass die Beantwortung der Zielfragestellung über die letzten Aktivitäten des Nutzers am Handy geklärt werden konnte.

Die Durchführung verschiedener Szenarien auf unterschiedlichen Betriebssystemversionen von Android erzielt besonders bei den Dateien, welche in den Ordnern *recent_images* und *recent_tasks* liegen, vertrauenswürdige Ergebnisse. Allerdings können diese beiden Ordner nur in zwei von drei Betriebssystemversionen (Version 6 und 7) gefunden werden.

Auch die Dateien unter *usagestats* protokollieren, wann welche Anwendung zuletzt genutzt wurde. Jedoch ist hier eine forensische Verwertung fragwürdig, da sich bei zwei der drei Betriebssystemversionen die Dateinamen, auf die sich die Daten innerhalb der Protokolldateien beziehen, nach einer gewissen Zeit aktualisieren, was zu einer Verfälschung der Zeitstempel führt. Unter Betriebssystemversion 4 von Android haben sich die Protokolldateien unter *usagestats* sowohl in bestehender Literatur als auch durch die Ergebnisfindung dieser Bachelorarbeit als nützlich erwiesen.

Auch die WhatsApp-Protokolldatei macht es möglich, Aussagen darüber zu treffen, wann das Handy zuletzt benutzt wurde. Besonders die in dieser Datei protokollierten Screen-LockReceiver-Ereignisse bringen Ergebnisse darüber, wann der Bildschirm zuletzt entsperrt wurde. Diese Bachelorarbeit stützt ihre Untersuchungen nur auf ein Gerätemodell. Durch den Vergleich mit bestehender Literatur wird deutlich, dass die Struktur der zu untersuchenden *XML*-Dateien nicht von dem Gerät, sondern von der Betriebssystemversion abhängig ist.

Die Analyse von *usagestats*-Dateien aktuellerer Betriebssystemversionen könnte durchaus interessant sein, da sich sowohl die Ordnerstruktur als auch der Aufbau dieser XML-Dateien in den verschiedenen Betriebssystemversionen ändern. Dies haben die Ergebnisse dieser Arbeit verdeutlicht.

Darüber hinaus könnte sich eine mögliche Studie damit befassen herauszufinden, wodurch die Aktualisierung der Zeitstempel der Dateinamen bei den Versionen 6 und 7 verursacht wird, indem man den Quellcode von Android genauer untersucht. Für weitere Forschungen wäre es außerdem vorstellbar, zu prüfen, ob es andere Protokolldateien unter Android gibt, welche ebenfalls die letzten Aktivitäten eines Nutzers speichern und zu ähnlichen Ergebnissen führen.

Anhang A: Versuchsreihe 1

A.1 Szenario 2

A.1.1 Protokolldatei unter *usagestats*

Aufbau

Der Aufbau der *XML*-Datei ist identisch zu Szenario 1. Es sind jedoch im Vergleich zu Szenario 1 noch weitere Dateien in dem Ordner angelegt worden. Hinzugekommen sind die Dateien *usage-20180702*, *usage-20180703* und *usage-20180704*. Diese hinzugekommenen Dateien enthalten keine Zeitstempel. Sie enthalten nur die Information, was an dem besagten Tag (siehe Dateiname) mit dem Handy gemacht wurde. In Abbildung A.2 ist ein Auszug einer solchen Datei zu sehen.

```
</pkg>
- <pkg name="com.whatsapp">
    <comp name="com.whatsapp.registration.RegisterPhone" lrt="1530520602879"/>
    <comp name="com.whatsapp.HomeActivity" lrt="1530704824527"/>
    <comp name="com.whatsapp.ContactInfo" lrt="1530520739416"/>
    <comp name="com.whatsapp.Settings" lrt="1530704884885"/>
    <comp name="com.whatsapp.Main" lrt="1530704709212"/>
    <comp name="com.whatsapp.gdrive.GoogleDriveActivity" lrt="1530520646198"/>
    <comp name="com.whatsapp.ProfileInfoActivity" lrt="1530704944207"/>
    <comp name="com.whatsapp.Conversation" lrt="1530520772823"/>
    <comp name="com.whatsapp.ContactPicker" lrt="1530520673010"/>
    <comp name="com.whatsapp.registration.RegisterName" lrt="1530520651612"/>
    <comp name="com.whatsapp.registration.VerifySms" lrt="1530520635263"/>
    <comp name="com.whatsapp.registration.EULA" lrt="1530520213995"/>
</pkg>
```

Abbildung A.1: Versuchsreihe 1, Szenario 2 Ausschnitt *usage-history* [Quelle: eigene Abbildung]

Abbildung A.2: Versuchsreihe 1, Szenario 2 Ausschnitt *usage-20180704* [Quelle: eigene Abbildung]

Analyse

In der folgenden Tabelle wird der Ablauf der Szenarien mit den Einträgen der *usage-history*-Datei vom 04.07.2018 verglichen.

Tabelle A.1: Versuchsreihe 1, Szenario 2 Vergleich *usage-history* und Szenario [Quelle: eigene Tabelle]

pkg *name*	**comp *name***	**Attribut *lrt***	**gemessene Handy-Zeit**	**Ablauf Szenario**
com.sec.android .app.popupcalculator	com.sec.android .app.popupcalculator.Calculator	03.07.2018, 13:45:39.085	03.07.2018, 13:45	Verlassen der Rechner-Anwendung
com.whatsapp	com.whatsapp.Settings	04.07.2018, 13:48:04.885	04.07.2018, 13:48	Einstellung von WhatsApp verlassen
com.whatsapp	com.whatsapp.Main	04.07.2018 13:45:09.212	04.07.2018, 13:45	WhatsApp-Hauptmenü gestartet und anschließend wieder verlassen
com.whatsapp	com.whatsapp .ProfileInfoActivity	04.07.2018, 13:49:04.207	04.07.2018, 13:49	Profilansicht von WhatsApp verlassen
com.whatsapp	com.whatsapp .Conversation	02.07.2018, 10:39:32.823	02.07.2018, 10:39	Kontakt bei WhatsApp schreiben
com.whatsapp	com.whatsapp .ContactPicker	02.07.2018, 10:37:53.010	02.07.2018, 10:37	Hinzufügen eines Kontaktes bei WhatsApp
com.android.vending	com.google.android .finsky.activities .MainActivity	03.07.2018, 13:46:09.293	03.07.2018, 13:46	Verlassen der PlayStore-App
com.sec.android .app.camera	com.sec.android .app.camera.Camera	03.07.2018, 13:45:52.643	03.07.2018, 13:45	Bild wurde mit Kamera aufgenommen
com.android.settings	com.android.settings.SubSettings	04.07.2018 ,13:50:02.718	04.07.2018, 13:50	Verlassen der Einstellungs-Anwendung

Es ist der Tabelle zu entnehmen, dass die Daten der Protokolldatei mit dem Ablauf des Szenarios übereinstimmen. In der *usage-history*-Datei, welche am 04.07.2018 gesichert

wird, sind sowohl Daten vom 04.07.2018 als auch Daten der vorherigen Tage zu sehen. Durch die obige Tabelle geht hervor, dass die Protokolldatei immer den letzten Zeitstempel protokolliert, an dem die entsprechende Komponente der Anwendung benutzt wurde. Diese Feststellung kann durch folgende zwei Abbildungen belegt werden. Dafür wird der Eintrag von *com.whatsapp.Settings* genauer betrachtet.

```
  </pkg>
- <pkg name="com.whatsapp">
    <comp name="com.whatsapp.registration.RegisterPhone" lrt="1530520602879"/>
    <comp name="com.whatsapp.HomeActivity" lrt="1530520773292"/>
    <comp name="com.whatsapp.ContactInfo" lrt="1530520739416"/>
    <comp name="com.whatsapp.Settings" lrt="1530520807642"/>
    <comp name="com.whatsapp.Main" lrt="1530520662722"/>
    <comp name="com.whatsapp.gdrive.GoogleDriveActivity" lrt="1530520646198"/>
    <comp name="com.whatsapp.ProfileInfoActivity" lrt="1530520793463"/>
    <comp name="com.whatsapp.Conversation" lrt="1530520772823"/>
    <comp name="com.whatsapp.ContactPicker" lrt="1530520673010"/>
    <comp name="com.whatsapp.registration.RegisterName" lrt="1530520651612"/>
    <comp name="com.whatsapp.registration.VerifySms" lrt="1530520635263"/>
    <comp name="com.whatsapp.registration.EULA" lrt="1530520213995"/>
  </pkg>
```

Abbildung A.3: Versuchsreihe 1, Szenario 2 Eintrag „com.whatsapp.settings“ vom 02.07.2018 [Quelle: eigene Abbildung]

```
  </pkg>
- <pkg name="com.whatsapp">
    <comp name="com.whatsapp.registration.RegisterPhone" lrt="1530520602879"/>
    <comp name="com.whatsapp.HomeActivity" lrt="1530704824527"/>
    <comp name="com.whatsapp.ContactInfo" lrt="1530520739416"/>
    <comp name="com.whatsapp.Settings" lrt="1530704884885"/>
    <comp name="com.whatsapp.Main" lrt="1530704709212"/>
    <comp name="com.whatsapp.gdrive.GoogleDriveActivity" lrt="1530520646198"/>
    <comp name="com.whatsapp.ProfileInfoActivity" lrt="1530704944207"/>
    <comp name="com.whatsapp.Conversation" lrt="1530520772823"/>
    <comp name="com.whatsapp.ContactPicker" lrt="1530520673010"/>
    <comp name="com.whatsapp.registration.RegisterName" lrt="1530520651612"/>
    <comp name="com.whatsapp.registration.VerifySms" lrt="1530520635263"/>
    <comp name="com.whatsapp.registration.EULA" lrt="1530520213995"/>
  </pkg>
```

Abbildung A.4: Versuchsreihe 1, Szenario 2 Eintrag „com.whatsapp.settings“ vom 04.07.2018 [Quelle: eigene Abbildung]

Die folgende Tabelle dient noch einmal zur Verdeutlichung dieser Zeitstempelaktualisierung:

Tabelle A.2: Versuchsreihe 1, Szenario 2 Aktualisierung der Zeitstempel [Quelle: eigene Tabelle]

Sicherung der *usage-history.xml*	**comp *name***	**Attribut *lrt***	**gemessene Handy-Zeit**	**Ablauf Szenario**
02.07.2018	com.whatsapp.Settings	02.07.2018, 10:40:07	02.07.2018, 10:40	WhatsApp Einstellungen geschlossen
04.07.2018	com.whatsapp.Settings	04.07.2018, 13:48:04	04.07.2018, 13:48	WhatsApp Einstellungen geschlossen

A.1.2 Protokolldatei */data/data/com.whatsapp/files/Logs/*

Aufbau

Die WhatsApp-Protokolldatei befindet sich auf der Datenpartition. Sie ist unter */data/data/com.whatsapp/files/Logs/* gespeichert und hat den Namen *whatsapp*.

Abbildung A.5: Versuchsreihe 1, Szenario 2 Ausschnitt WhatsApp-Protokolldatei [Quelle: eigene Abbildung]

Der Aufbau dieser Protokolldatei wirkt auf den ersten Blick etwas „chaotisch“. Betrachtet man diesen jedoch genauer, kann man eine chronologische Aneinanderreihung von verschiedenen Ereignissen sehen. Diese Ereignisse sind immer mit einem Zeitstempel versehen, der bis auf die Millisekunde angegeben ist.

Analyse

Es folgt eine Analyse der WhatsApp-Protokolldatei, welche am 04.07.2018 gesichert wird. Diese Datei protokolliert seit der Installation von WhatsApp für jeden Tag Battery-State-Ereignisse. Ebenso werden seit dem 02.07.2018 ScreenLockReceiver-Ereignisse in dieser Datei mehrfach protokolliert. Im Folgenden wird jeweils ein Beispiel für diese Ereignisse gezeigt.

• BatteryStatehealth=good, level=83, plugged=0, scale=100, percent=83.02018-07-02 10:30:28.932
• ScreenLockReceiver; tag=off; locked=true; oldLocked=false2018-07-02 10:31:14.573

A.2 Szenario 3

A.2.1 Protokolldatei unter *usagestats*

Aufbau

Der Aufbau ist identisch zu Szenario 2 dieser Versuchsreihe.

Analyse

Vergleicht man die *usage-history*-Datei, welche am 04.07.2018 gesichert wird, mit der Datei, welche am 08.07.2018 gesichert wird, ist zu erkennen, dass sich einige Einträge in der Protokolldatei aktualisiert haben. Die aktualisierten Ereignisse sind in Tabelle A.3 dargestellt.

Tabelle A.3: Versuchsreihe 1, Szenario 3 Aktualisierung Zeitstempel [Quelle: eigene Tabelle]

pkg *name*	**comp *name***	**Attribut *lrt***	**gemessene Handy-Zeit**	**Szenario**
com.whatsapp	com.whatsapp .Conversation	05.07.2018, 09:23:13.481	05.07.2018, 09:23	WhatsApp Kontakt schreiben
com.android .incallui	com.android .incallui.InCall-Activity	07.07.2018, 09:51:26.572	07.07.2018, 09:51:26	eingehender Anruf

Da die Ereignisse der *usage-history*-Datei aktualisiert werden, gilt es nun zu prüfen, ob der Inhalt der anderen Dateien, die in dem Ordner *usagestats* gespeichert sind, gleich

bleibt. Dafür werden die Dateien, welche am 04.07.2018 gesichert werden, mit denen verglichen, die am 08.07.2018 gesichert werden.

Tabelle A.4: Versuchsreihe 1, Szenario 3 Hashvergleich [Quelle: eigene Tabelle]

Dateiname	**Sicherung**	**MD5**	**SHA1**
usage-20180703	04.07.2018	cbb4f45c32360ca0df1735d503e40dd4	c1f3809ce9bc16da775647fa59aac6cc60d10125
usage-20180703	08.07.2018	cbb4f45c32360ca0df1735d503e40dd4	c1f3809ce9bc16da775647fa59aac6cc60d10125
usage-20180704	04.07.2018	a1003c6d674abda2d07b87775dd3ea6e	b307d3d50d089646727a7c638202a9e89bb405b1
usage-20180704	08.07.2018	a1003c6d674abda2d07b87775dd3ea6e	b307d3d50d089646727a7c638202a9e89bb405b1

Wie in Tabelle A.4 zu sehen ist, werden die Dateien zu den jeweiligen Tagen nicht verändert. Als Nächstes gilt es festzustellen, wie lange die Dateien in dem Ordner *usagestats* aufbewahrt werden. Durch die Sicherungen an unterschiedlichen Tagen kann festgestellt werden, dass die *usage-history.xml* immer vorhanden ist und nicht gelöscht wird. Alle anderen Dateien in diesem Ordner werden nach einer gewissen Zeit gelöscht. Es kann anhand der verschiedenen Sicherungen jedoch nicht genau festgestellt werden, wie lange die Dateien in dem Ordner aufbewahrt sind. Die Datei *usage-20180701* beispielsweise wird für fünf Tage in dem Ordner gespeichert, während die *usage-20180703*-Datei sechs Tage gespeichert wird. Weiterhin kann durch die verschiedenen Sicherungen beobachtet werden, dass der Ordner *usagestats* maximal sechs Dateien enthält.

A.2.2 Protokolldatei unter */data/data/com.whatsapp/files/Logs/*

Aufbau

Die Daten werden einmal am 05.07.2018 und einmal am 08.07.2018 gesichert. Der Aufbau der Protokolldatei entspricht in beiden Sicherungen dem Aufbau von Szenario 2. Jedoch enthält die Protokolldatei, welche am 08.07.2018 gesichert wurde, nur Einträge vom 08.07.2018. Alle anderen Einträge in dieser Protokolldatei sind gelöscht.

Analyse

Im Folgenden wird die WhatsApp-Protokolldatei vom 05.07.2018 untersucht. Die Beobachtungen zu den ScreenLockReceiver-Ereignissen und zum Batteriestatus sind in den folgenden Tabellen (Tabelle A.5 und A.6) zu sehen.

Tabelle A.5: Versuchsreihe 1, Szenario 3 BatteryState-Ereignisse [Quelle: eigene Tabelle]

Ereignis	**Attribut *plugged***	**Zeitstempel**	**gemessene Handy-Zeit**	**Szenario**
BatteryState	2	05.07.2018, 09:17:24.341	09:17	USB-Kabel mit Handy verbunden
BatteryState	0	05.07.2018, 09:20:04.894	09:20	Handy ist mit keinem Ladegerät verbunden
BatteryState	1	05.07.2018, 09:21:08.956	09:21	Handy ist mit Netzteil-Ladegerät verbunden

Tabelle A.6: Versuchsreihe 1, Szenario 3 ScreenLockReceiver-Ereignisse [Quelle: eigene Tabelle]

Ereignis	**Attribut *tag***	**Zeitstempel**	**gemessene Handy-Zeit**	**Szenario**
ScreenLock-Receiver	*unlock*	05.07.2018, 09:19:17.415	09:19	Entsperrung des Bildschirms
ScreenLock-Receiver	*off*	05.07.2018, 09:20:23.622	09:20	Bildschirm wird gesperrt
ScreenLock-Receiver	*on*	05.07.2018, 09:21:29.449	09:21	Handy wird mit Ladegerät verbunden und Bildschirm leuchtet auf
ScreenLock-Receiver	*unlock*	05.07.2018, 09:23:04.868	09:23	Entsperren des Bildschirms

Betrachtet man die Protokolldatei, welche am 08.07.2018 gesichert worden ist, genauer, bemerkt man, dass auch BatteryState-Ereignisse angelegt werden, obwohl der Nutzer das Handy nicht benutzt hat. Im Folgenden sind einige Beispiele dieser Ereignisse zu sehen.

- BatteryStatehealth=good, level=75, plugged=0, scale=100, percent=75.02018-07-08 00:17:30.016
- BatteryStatehealth=good, level=75, plugged=0, scale=100, percent=75.02018-07-08 00:23:17.059
- BatteryStatehealth=good, level=75, plugged=0, scale=100, percent=75.02018-07-08

01:23:17.082

Anhang B: Versuchsreihe 2

B.1 Szenario 2

B.1.1 Protokolldateien in */data/system/usagestats/0/daily/*

In diesem Schritt werden die Dateien im Ordner „daily“ genauer untersucht. Außerdem soll festgestellt werden, inwiefern sich diese Dateien von denen am Vortag unterscheiden.

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu Szenario 1 nicht verändert.

Analyse

Es ist aufgefallen, dass zu dem heutigen Tag keine eigene *XML*-Datei angelegt wurde. Ferner sind dieselben *XML*-Dateien im Ordner „daily“, die auch am Vortag im Ordner „daily“ waren. Dabei ist eine dieser Logdateien identisch zu der am Vortag, während sich die andere von denen am Vortag unterscheidet. Tabelle B.1 zeigt einen Hashwertabgleich zwischen den jeweiligen Dateien zum Vortag.

Tabelle B.1: VR 2, S 2 Hashabgleich der „daily“ -Dateien aus Szenario 1 und 2 [Quelle: eigene Tabelle]

MD5	**SHA1**	**Pfad**
b6b4ab792f06b31416d00207ddf96ba9	89ddd8a3a3e21b620154a38fed4c157a47d6cdde	C:\android-sdk\platform-tools\0\daily\1532013966190.xml
b6b4ab792f06b31416d00207ddf96ba9	89ddd8a3a3e21b620154a38fed4c157a47d6cdde	C:\Users\ninag\Desktop\Versuchsreihe 2\Szenario 1\0\daily\1532013966190.xml
3b4df6b28a872541aa417d155b9929cd	a23a9a528439a47f0ec86c2972700b9e05ccea9a	C:\android-sdk\platform-tools\0\daily\1532014304413.xml

d486b5c275181ec60e58106c09e07f4d	b46f05f88e7ec5393d0b552b27722f97b41fba4f	C:\Users\ninag\Desktop\Versuchsreihe 2\Szenario 1\0\daily\1532014304413.xml

Die erste Datei in dieser Tabelle entspricht dabei der *1532013966190.xml*-Datei von diesem Szenario, während die zweite Datei in der Tabelle der *1532013966190.xml*-Datei von Szenario 1 entspricht. Wie zu sehen ist, sind diese identisch. Die letzten beiden Zeilen der Tabelle stellen die *1532014304413.xml*-Datei von Szenario 1 und von diesem Szenario dar. Diese sind jedoch nach der Hashwertberechnung nicht identisch. Es gilt nun herauszufinden, worin die Unterschiede dieser Dateien bestehen.

Dafür werden zunächst die *package*-Elemente in Betracht gezogen. Mittels des Tools *WinMerge* ist es möglich, die beiden Dateien miteinander zu vergleichen. Auffällig ist, dass sich das Attribut *endTime* des Wurzelelementes verändert hat. Bei den *package*-Elementen sind außerdem einige hinzugekommen und andere haben das Attribut *lastTimeActive* aktualisiert. Das Attribut *lastEvent* nimmt außerdem nur den Wert „0“ oder „2“ an. Alle *package*-Elemente, die einen Wert von „2“ haben, haben weiterhin eine *timeActive*, die größer als null ist und außerdem einen positiven *lastTimeActive*-Wert. Ein Ausschnitt davon ist in Abbildung B.1 gezeigt.

```
- <packages>
    <package lastEvent="0" timeActive="0" package="com.google.android.ext.services" lastTimeActive="-1532014304413"/>
    <package lastEvent="0" timeActive="0" package="com.android.providers.telephony" lastTimeActive="-1532014304413"/>
    <package lastEvent="2" timeActive="9008" package="com.google.android.googlequicksearchbox" lastTimeActive="60959228"/>
    <package lastEvent="0" timeActive="0" package="com.android.providers.calendar" lastTimeActive="-1532014304413"/>
    <package lastEvent="0" timeActive="0" package="com.android.providers.media" lastTimeActive="-1532014304413"/>
    <package lastEvent="2" timeActive="616672" package="com.whatsapp" lastTimeActive="60682383"/>
    <package lastEvent="0" timeActive="0" package="com.android.providers.downloads" lastTimeActive="-1532014304413"/>
```

Abbildung B.1: VR 2, S 2 *package*-Element [Quelle: eigene Abbildung]

Vergleicht man einige der veränderten *package*-Elemente mit dem oben durchgeführten Szenario, zeigen sich folgende Zusammenhänge (siehe Tabelle B.2).

Tabelle B.2: VR 2, S 2 Abgleich *package*-Elemente und Szenario [Quelle: eigene Tabelle]

Attribut *package*	**Dateiname**	**Attribut *lastTimeActive***	**Dateiname + *lastTimeActive***	**Szenario**	**Epoch-Zeit**	**Handy-Zeit**
com.whatsapp	1532014304413	60682383	10:23:06	WhatsApp das letzte Mal verlassen	10:23:06	10:23

com.android .vending	1532014304 413	60752297	10:24:16	PlayStore verlassen	10:24:18	10:24
com.google .android.cal- culator	1532014304 413	60497805	10:20:02	Rechner verlassen	10:20:07	10:20
android	1532014304 413	60951489	10:27:35	Handy wurde ausge- schaltet		10:27

Man kann erkennen, dass die Zeiten in der Logdatei im Zusammenhang mit den gemessenen Zeiten des Szenarios stehen. Die gemessene Epoch-Zeit stimmt dabei mit einigen Ereignissen in der Logdatei exakt überein, bei anderen gemessenen Zeiten gibt es eine Abweichung von wenigen Sekunden.

Als Nächstes werden die *event*-Elemente genauer betrachtet. Dabei gibt es zunächst eine große Übereinstimmung zwischen den Elementen der *1532014304413.xml*-Datei des Ordners „daily“ von Szenario 1 und der *1532014304413.xml*-Datei von diesem Szenario. Das bedeutet gleichzeitig auch, dass die Aktivitäten einer Person vom Vortag noch exakt nachvollziehbar sind, da sich die *event*-Elemente nicht aktualisieren, sondern nur neu hinzukommen. Weiterhin zeigt die Logdatei aber auch, dass viele neue *event*-Elemente dazugekommen sind, welche die Ereignisse dieses Szenarios (20.07.2018) protokollieren. Tabelle B.3 gibt einen Überblick über diese *event*-Elemente und vergleicht sie mit den gemessenen Zeiten des Szenarios.

Tabelle B.3: VR 2, S 2 Vergleich Szenario und *event*-Elemente [Quelle: eigene Tabelle]

Attribut *class*	**Attribut *type***	**Attribut *time***	**Dateiname + *time***	**Szenario**	**Epoch-Zeit**	**Handy-Zeit**
com.whatsapp .Main	1	60534749	10:20:39	Haupt- Menü bei Whatsapp	10:20:39	10:20
com.whatsapp .QuickContact- Activity	1	60569523	10:21:13	WhatsApp Profilfoto angeschaut	10:21:13	10:21
com.whatsapp .QuickContact- Activity	2	60620090	10:22:04	Profilansicht verlassen	10:22:03	10:22

com.whatsapp. HomeActivity	1	60620134	10:22:04	durch Verlassen der Profilansicht wieder auf vorherige Ansicht	10:22:03	10:22
com.whatsapp .HomeActivity	2	60682383	10:23:06	WhatsApp verlassen	10:23:06	10:23
com.google.android.finsky.activities.MainActivity	1	60721752	10:23:46	PlayStore öffnen	10:23:46	10:23
com.google.android.finsky.activities.Main Activity	2	60752297	10:24:16	PlayStore verlassen	10:24:18	10:24

Anhand dieser Tabelle kann man sehen, dass vier der sieben gemessenen Zeiten mittels der Epoch-Zeit genau mit den Zeiten in der Logdatei übereinstimmen. Die gemessene Handy-Zeit stimmt dabei immer mit der Zeit der Logdatei überein, ist aber auch nur auf die Minute genau.

Des Weiteren ist aufgefallen, dass das Attribut *type* auch den Wert „5“ annehmen kann. Eine Besonderheit liegt darin, dass jedes Mal, wenn der Wert „5“ bei dem Attribut *type* angenommen wird, das *event*-Element zusätzliche Attribute hat. Diese zusätzlichen Attribute sind immer Bestandteil der Attribute, die normalerweise einem *config*-Element übergeben werden. Ein Beispiel zeigt Abbildung B.2.

```
<event package="android" height="640" keyHid="2" touch="1" fs="1065353216" type="5" time="60335445"/>
<event package="android" height="616" fs="1065353216" type="5" time="60335456"/>
<event package="android" ui="17" fs="1065353216" type="5" time="60335486"/>
<event package="android" touch="3" fs="1065353216" type="5" time="60336180"/>
<event package="android" keyHid="1" fs="1065353216" type="5" time="60337222"/>
```

Abbildung B.2: VR 2, S 2 *type* „5“ der *event*-Elemente [Quelle: eigene Abbildung]

Zusätzlich zu den Werten „1“, „2“ und „5“ hat das *type*-Attribut bisher auch noch den Wert „7“ angenommen.

Neben den *event*-Elementen haben sich auch die *config*-Elemente dieser Datei aktualisiert. Die Datei hat sich von zwei *config*-Elementen auf sechs ausgeweitet. Dabei ist

lediglich eins identisch geblieben, vier sind dazugestoßen und eins hat sich aktualisiert. Berechnet man die Zeiten der neuen und aktualisierten *config*-Elemente, kommt man auf folgende Ergebnisse (siehe Tabelle B.4):

Tabelle B.4: VR 2, S 2 Auswertung *config*-Elemente [Quelle: eigene Tabelle]

Attribut *lastTimeActive*	**Dateiname**	**Dateiname + *lastTimeActive***
60368599	1532014304413	10:17:53
60782254	1532014304413	10:24:46
60337221	1532014304413	10:17:21
60335485	1532014304413	10:17:19
60335455	1532014304413	10:17:19
60336179	1532014304413	10:17:20

Es ist zu beobachten, dass ein Großteil dieser *config*-Elemente gegen 10:17 Uhr angelegt wurde. Dies entspricht nach Ablauf des Szenarios der Zeit, als das Handy gestartet wurde. In der Zeit als verschiedene Apps geöffnet wurden, wurde kein *config*-Element angelegt.

B.1.2 Protokolldateien in*/data/system/usagestats/0/weekly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu Szenario 1 nicht verändert. Auch im Ordner „weekly“ befinden sich zwei *XML*-Dateien, die denselben Namen wie die *XML*-Dateien im Ordner „daily“ tragen.

Analyse

Es ist aufgefallen, dass die Datei *1532013966190.xml* dieses Ordners mit der Datei des Ordners „weekly“ vom Vortag identisch ist. Des Weiteren fällt auf, dass die *package*-Elemente der *1532014304413.xml*-Datei mit denen der *1532014304413.xml* aus dem „daily“-Ordner desselben Tages exakt übereinstimmen. Der einzige Unterschied zwischen diesen beiden Dateien besteht darin, dass in der *1532014304413.xml*-Datei des „daily“-Ordners die *event*-Elemente aufgelistet sind, aber in der *1532014304413.xml*-Datei des „weekly“-Ordners nicht, weil das *event-log*-Element ein leeres Element darstellt.

B.1.3 Protokolldateien in */data/system/usagestats/0/monthly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu Szenario 1 nicht verändert. In diesem Ordner befinden sich zwei *XML*-Dateien mit dem Namen *1532014304413.xml* und *1532013966190.xml*.

Analyse

Es kann festgestellt werden, dass die *1532014304413.xml*-Datei dieses Ordners der im Ordner „weekly“ entspricht. Dies gilt ebenfalls für die *1532013966190.xml*-Datei, welche identisch zu der im „weekly“-Ordner ist. Tabelle B.5 belegt dies an einem Hashwertvergleich.

Tabelle B.5: VR 2, S 2 Hashabgleich von „weekly“ - und „monthly“ -Dateien [Quelle: eigene Tabelle]

MD5	**SHA1**	**Pfad**
6665cb089fc76bdf707e05402ffbf51e	b9de29b35fdd620e81dcf243c05fa43663c68bf0	C:\android-sdk\platform-tools\0\monthly\1532013966190.xml
6665cb089fc76bdf707e05402ffbf51e	b9de29b35fdd620e81dcf243c05fa43663c68bf0	C:\android-sdk\platform-tools\0\weekly\1532013966190.xml
8e78c59f2f8fe77630e3c5d9de849a79	ffcf4df706f3d20d7cf3c7e404b0efd9e1534c75	C:\android-sdk\platform-tools\0\monthly\1532014304413.xml
8e78c59f2f8fe77630e3c5d9de849a79	ffcf4df706f3d20d7cf3c7e404b0efd9e1534c75	C:\android-sdk\platform-tools\0\weekly\1532014304413.xml

B.1.4 Protokolldateien in */data/system/usagestats/0/yearly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu Szenario 1 nicht verändert.In diesem Ordner befinden sich zwei *XML*-Dateien mit dem Namen *1532014304413.xml* und

1532013966190.xml.

Analyse

Wie auch im vorherigen Szenario stimmen die *XML*-Dateien dieses Ordners mit denen des Ordners „monthly“ und „weekly“ überein. In Tabelle B.6 ist dies durch einen Hashabgleich belegt.

Tabelle B.6: VR 2, S 2 Hashabgleich der „weekly“ -, „monthly“ - und „yearly“ -Dateien [Quelle: eigene Tabelle]

MD5	**SHA1**	**Pfad**
6665cb089fc76bdf707e05402ffbf51e	b9de29b35fdd620e81dcf243c05fa43663c68bf0	C:\android-sdk\platform-tools\0\weekly\1532013966190.xml
6665cb089fc76bdf707e05402ffbf51e	b9de29b35fdd620e81dcf243c05fa43663c68bf0	C:\android-sdk\platform-tools\0\monthly\1532013966190.xml
6665cb089fc76bdf707e05402ffbf51e	b9de29b35fdd620e81dcf243c05fa43663c68bf0	C:\android-sdk\platform-tools\0\yearly\1532013966190.xml
8e78c59f2f8fe77630e3c5d9de849a79	ffcf4df706f3d20d7cf3c7e404b0efd9e1534c75	C:\android-sdk\platform-tools\0\weekly\1532014304413.xml
8e78c59f2f8fe77630e3c5d9de849a79	ffcf4df706f3d20d7cf3c7e404b0efd9e1534c75	C:\android-sdk\platform-tools\0\monthly\1532014304413.xml
8e78c59f2f8fe77630e3c5d9de849a79	ffcf4df706f3d20d7cf3c7e404b0efd9e1534c75	C:\android-sdk\platform-tools\0\yearly\1532014304413.xml

B.1.5 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Aufbau

Der Aufbau dieser Protokolldatei hat sich im Vergleich zu Szenario 1 nicht verändert.

Analyse

Es können mehrere Ereignisse zum Batteriestatus und den ScreenLockReceiver-Ereignissen gefunden werden. In der folgenden Tabelle (Tabelle B.7) werden Ereignisse des Batteriestatus mit dem Ablauf des Szenarios verglichen.

Tabelle B.7: VR 2, S 2 Darstellung BatteryState-Ereignisse [Quelle: eigene Tabelle]

Attribut *plugged*	**Zeitstempel in Protokolldatei**	**Ablauf Szenario**	**gemessene Epoch-Zeit**	**gemessene Handy-Zeit**
1	10:25:53	Handy wird mit Netzteil-Ladegerät verbunden	10:25:24	10:25
2	10:26:32	Handy wurde mit USB-Kabel am Computer verbunden	10:26:14	10:26

B.2 Szenario 3

B.2.1 Protokolldateien in */data/system/usagestats/0/daily/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu Szenario 1 nicht verändert.

Analyse

Wie in Abbildung B.3 zu erkennen ist, wurde die Datei um 06:12 Uhr am 25.07.2018 angelegt.

```
-rw-------    1 system   system       13518 Jul 25 06:12 1532497506451
```

Abbildung B.3: VR 2, S 3 Kommando *ls -al* [Quelle: eigene Abbildung]

Die Datei *1532497506451.xml* ergibt umgerechnet einen Zeitstempel von 07:45:06.451 Uhr des 25.07.2018. Die Dateien der vorherigen Szenarien befinden sich nicht mehr in diesem Ordner.

Zunächst werden die *package*-Elemente genauer betrachtet. Dabei fällt auf, dass das Attribut *lastTimeActive* jedes Mal, wenn das Attribut *lastEvent* desselben Elementes den Wert „2“ hat, einen positiven Wert annimmt. Weiterhin nehmen jedoch auch *package*-Elemente einen positiven Wert bei *lastTimeActive* an, wenn der *lastEvent-Wert* „0“ ist. Die folgende Tabelle (Tabelle B.8) vergleicht die Werte der *package*-Elemente mit den im Szenario gemessenen Zeiten.

Tabelle B.8: VR 2, S 3 Vergleich *package*-Elemente und Szenario [Quelle: eigene Tabelle]

Attribut *package*	**Attribut *lastTimeActive***	**Dateiname + *lastTimeActive***	**Attribut *lastEvent***	**gemessene Epoch-Zeit**	**Handy-Zeit**	**Szenario**
com.whatsapp	1645473	08:12:31	2		08:12	WhatsApp beendet
com.google .android.calculator	1504588	08:10:11	2	08:10:11	08:10	Rechner-App beendet
com.android .settings	1621290	08:12:07	2		08:12	Einstellungs-App beendet
com.google .android.apps .messaging	60647	07:46:07	0		07:46	Empfangen von SMS

Aus dieser Tabelle geht hervor, dass die gemessenen Zeiten des Szenarios mit den Werten der Logdatei übereinstimmen. Ein weiteres Ziel dieses Szenarios besteht darin, herauszufinden, was genau das Attribut *timeActive* dokumentiert. Dies wird anhand des Öffnens und Schließens der Rechner-Anwendung belegt, da hier nach Abhandlung des Szenarios genaue Messzeiten vorliegen. In Tabelle B.9 sind die Ergebnisse dieser Messungen zu sehen.

Tabelle B.9: VR 2, S 3 Untersuchung der *timeActive*-Werte [Quelle: eigene Tabelle]

Attribut *package*	**Attribut *lastTimeActive***	**Attribut *timeActive***	***lastTimeActive - timeActive***	**Dateiname + 1440431**	**Szenario**	**Epoch-Zeit**	**Handy-Zeit**
com .google .android .calculator	1504588	64157	1440431	08:09:06	Öffnen der Rechner-App	08:09:06	08:09

Da die Rechner-Anwendung nur einmal geöffnet wurde, kann man den Wert des Attributes *timeActive* von dem Wert des Attributes *lastTimeActive* abziehen. Addiert man das resultierende Ergebnis nun auf den Dateinamen ergibt dies eine Zeit, die im Szenario der Zeit entspricht, als die App gestartet wurde.

Im nächsten Schritt werden die *config*-Elemente näher betrachtet. Insgesamt wurden in dieser Datei acht *config*-Elemente angelegt. Dabei wird sechs dieser Elemente ein *lastTimeActive*-Wert zugewiesen, der eine Zeit zwischen 07:45 und 07:46 aufweist. Dies entspricht ungefähr der Zeit, als das Smartphone gestartet wurde. Tabelle B.10 vergleicht drei *config*-Elemente mit den gemessenen Zeiten des Szenarios. Es ist auffällig, dass das Attribut *ori* nur bei einem Element den Wert „2“ annimmt. Alle anderen *config*-Elemente, auch diese, die nicht in dieser Tabelle gezeigt sind, haben bei diesem Attribut einen Wert von „1“. Nach Ablauf des Szenarios wird der Wert „2“ angenommen, als das Handy gedreht wird und somit die Bildschirmansicht verändert wird.

Das Attribut *lastTimeActive* nimmt dabei eine Zeit an, die der entspricht, als das Handy wieder in die ursprüngliche Ansicht zurückgedreht wird. Zieht man dabei die *timeActive* von der *lastTimeActive* ab, ergibt sich eine Zeit, die nach Ablauf des Szenarios dieser entspricht, als das Handy zum ersten Mal gedreht wurde. Das Ergebnis ist in Tabelle B.11 dargestellt. Weiterhin ist aufgefallen, dass sich das Attribut *locales* geändert hat, als man nach Ablauf des Szenarios die Spracheinstellungen umgestellt hat.

Tabelle B.10: VR 2, S 3 Änderung der Spracheinstellung [Quelle: eigene Tabelle]

Attribut *lastTimeActive*	**Attribut *timeActive***	**Attribut *ori***	***locales***	**Dateiname + *lastTimeActive***	**Szenario**	**Epoch-Zeit**	**Handy-Zeit**
35564	33873	1	de-DE	07:45:42	Hochfahren des Handys		
1343042	53832	2	de-DE	08:07:29	erneutes Drehen von WhatsApp-Ansicht	08:07	08:07:29
1590785	0	1	de-DE, en-DE	08:11:37	Sprach-einstellungen geändert	08:11	

Tabelle B.11: VR 2, S 3 Untersuchung Bildschirmrotation [Quelle: eigene Tabelle]

Dateiname + *lastTimeActive* - *timeActive*	**Szenario**	**gemessene Epoch-Zeit**	**gemessene Handy-Zeit**
08:06:35	Drehen von WhatsApp-Ansicht	08:06:35	08:06

Zu guter Letzt werden in dieser Datei die *event*-Elemente betrachtet. Hier geht es in diesem Szenario vor allem darum, herauszufinden, was die verschiedenen Werte der *type*-Attribute für eine Bedeutung haben. In Tabelle B.12 findet eine Gegenüberstellung der gemessenen Zeiten des Szenarios und der Logdatei statt.

Tabelle B.12: VR 2, S 3 Untersuchung des Attributes *type* [Quelle: eigene Tabelle]

Attribut *package*	**Attribut *type***	**Attribut *class***	**Dateiname + *time***	**Szenario**	**Epoch-Zeit**	**Handy-Zeit**
com.whatsapp	7		07:46:01	Handy angeschaltet und Nachrichten über WhatsApp empfangen		07:46
com.google.android.apps.messaging	7		07:46:07	Handy angeschaltet und SMS empfangen		07:46
com.android.settings	1	com.android.settings.Settings$WifiSettingsActivity	07:46:15	WiFi-Verbindung eingestellt		07:46

com.whatsapp	7		07:47:12	Empfangen weiterer WhatsApp-Nachrichten nach Wifi-Verbindung		07:47
com.whatsapp	7		07:47:16	Empfangen weiterer WhatsApp-Nachrichten nach Wifi-Verbindung		07:47
com.whatsapp	1	com.whatsapp.Main	08:05:52	Betreten WhatsApp-Hauptmenü	08:05:52	08:05
android	5		08:06:35	Drehen der WhatsApp-Ansicht	08:06:35	08:06
android	5		08:07:29	erneutes Drehen der WhatsApp-Ansicht	08:07:29	08:07
com.google.android.calculator	1	com.android.calculator2.CalculatorGoogle	08:09:06	Öffnen der Rechner-App	08:09:06	08:09
com.google.android.calculator	2	com.android.calculator2.CalculatorGoogle	08:10:11	Schließen der Rechner-App	08:10:11	08:10
com.android.settings	1	com.android.settings.Settings	08:11:06	Öffnen der Einstellungs-App	08:11:06	08:11

android	5		08:11:37	Änderung der Spracheinstellung		08:11

B.2.2 Protokolldateien in */data/system/usagestats/0/weekly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu Szenario 1 nicht verändert.

Analyse

Auch in diesem Ordner befindet sich keine Datei der vorherigen Szenarien. Es befindet sich lediglich eine *XML*-Datei in diesem Ordner, welche den Namen *1532497506451.xml* trägt. Es kann festgestellt werden, dass diese Datei bis auf die *event*-Elemente identisch zu der Datei im Ordner „daily“ ist. Das *event-log*-Element stellt hierbei wieder ein leeres Element dar und besitzt deshalb keine weiteren Kind-Knoten.

B.2.3 Protokolldateien in */data/system/usagestats/0/monthly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu Szenario 1 nicht verändert.

Analyse

Es ist aufgefallen, dass die Datei in diesem Ordner identisch zu der Datei im Ordner „weekly“ ist.

B.2.4 Protokolldateien in */data/system/usagestats/0/yearly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu Szenario 1 nicht verändert.

Analyse

Der Hashberechnung zufolge stimmt auch diese *XML*-Datei (*1532497506451.xml*) mit den Dateien des Ordners „monthly“ und „weekly“ überein.

B.2.5 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu Szenario 1 nicht verändert.

Analyse

Es ist auffällig, dass der Batteriestatus erneut dokumentiert wird, jedoch enthält die Datei nur Einträge vom 25.07.2018. Alle anderen Einträge, die in Szenario 1 und 2 vorzufinden waren, sind gelöscht. Neben dem Batteriestatus werden auch ScreenLockReceiver-Ereignisse in dieser Protokolldatei angelegt. Da zu diesen keine gemessenen Zeiten im Szenario vorliegen, werden diese erst im nächsten Szenario genauer untersucht. Die Protokollierung des Batteriestatus erbrachte dieselben Ergebnisse wie in Versuchsreihe 1.

B.3 Szenario 4

B.3.1 Ordner in */data/system/usagestats/0/*

Am 01.08.2018 befinden sich wie zuvor vier Unterordner „daily“, „weekly“, „monthly“ und „yearly“ sowie eine Datei *version* in dem Ordner */usagestats/0/*. Folgende Tabellen geben einen Überblick darüber, welche *XML*-Dateien sich in den jeweiligen Unterordnern befinden.
XML-Dateien des Ordners „daily“:

Tabelle B.13: VR 2, S 4 „daily“-Dateien [Quelle: eigene Tabelle]

Dateiname	**Zeitstempel**
1532585317803	26.07.2018, 06:08:37.803
1532674140619	27.07.2018, 08:49:00.619
1532766942300	28.07.2018, 08:35:42.300
1532931631492	30.07.2018, 08:20:31.492
1533020964854	31.07.2018, 09:09:24.854
1533108636859	01.08.2018, 09:30:36.859

XML-Dateien des Ordners „weekly“:

Tabelle B.14: VR 2, S 4 „weekly“-Dateien [Quelle: eigene Tabelle]

Dateiname	**Zeitstempel**
1533108636859	01.08.2018, 09:30:36.859
1532497511269	25.07.2018, 07:45:11.269

XML-Dateien des Ordners „monthly“:

Tabelle B.15: VR 2, S 4 „monthly“-Dateien [Quelle: eigene Tabelle]

Dateiname	**Zeitstempel**
1532497511269	25.07.2018, 07:45:11.269

XML-Dateien des Ordners „yearly“:

Tabelle B.16: VR 2, S 4 „yearly“-Dateien [Quelle: eigene Tabelle]

Dateiname	**Zeitstempel**
1532497511269	25.07.2018, 07:45:11.269

B.3.2 Protokolldateien in */data/system/usagestats/0/daily/*

Aufbau

Es ist anzumerken, dass bei der Sicherung vom 01.08.2018 eine *XML*-Datei zu dem 25.07.2018 nicht mehr in diesem Ordner existiert. Bei der Sicherung vom 31.07.2018 gab es noch eine Datei zum 25.07.2018. Diese trug jedoch einen anderen Dateinamen als die Datei vom 25.07.2018 von Szenario 3. Die Datei, welche am 31.07.2018 gesichert wird, heißt *1532497511269.xml* und ergibt einen Zeitstempel vom 25. Juli 2018 07:45:11.269, während die Datei, die in Szenario 3 gesichert wird, *1532497506451.xml* heißt und einen Zeitstempel vom 25. Juli 2018 07:45:06.451 ergibt.

Analyse

Wie in Szenario 4 beschrieben, wurden in der vergangenen Woche täglich verschiedene Aktionen am Handy durchgeführt. Das Handy blieb nur an einem Tag, den 29.07.2018, komplett ausgeschaltet, weshalb auch keine Datei für diesen Tag angelegt wurde.

B.3.3 Protokolldateien in */data/system/usagestats/0/weekly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu Szenario 1 nicht verändert.

Analyse

Wie weiter oben dargestellt, befinden sich in dem „weekly“-Ordner zwei Dateien. Eine Datei trägt dabei einen Zeitstempel vom 25.07.2018 und die andere vom 01.08.2018. Es gilt nun festzustellen, ob die Datei des „weekly“-Ordners vom 25.07.2018 die Zeitstempel der Anwendungen, welche am 31.07.2018 benutzt wurden, aktualisiert. Dafür werden die Einträge der letzten „daily“-Datei, die zu dieser Woche zählt (25.07-31.07), mit den Einträgen der „weekly“-Datei vom 25.07.2018 verglichen. Die Ergebnisse sind in den folgenden Tabellen gezeigt.

Tabelle B.17: VR 2, S 4 „daily“-Einträge [Quelle: eigene Tabelle]

Dateiname in daily	**Attribut *package***	**Attribut *lastTimeActive***	**Dateiname + *lastTimeActive***
1533020964854	com.whatsapp	16753601	1533037718455
1533020964854	com.android.settings	83850772	1533104815626
1533020964854	com.google.android.apps .nexuslauncher	83876072	1533104840926

Tabelle B.18: VR 2, S 4 „weekly“-Einträge [Quelle: eigene Tabelle]

Dateiname in weekly	**Attribut *package***	**Attribut *lastTimeActive***	**Dateiname + *lastTimeActive***
1532497511269	com.whatsapp	540207186	1533037718455
1532497511269	com.android.settings	607304357	1533104815626
1532497511269	com.google.android.apps .nexuslauncher	607329657	1533104840926

Durch diese Tabellen wird deutlich, dass in der „weekly“-Datei der Zeitstempel zu jeder Anwendung gespeichert wird, an dem die Anwendung im Laufe der Woche das letzte Mal benutzt wurde.

Betrachtet man das Attribut *timeActive* genauer, ergeben sich folgende Zusammenhänge:

Tabelle B.19: VR 2, S 4 *timeActive*-Werte der „daily“-Dateien [Quelle: eigene Tabelle]

Dateiname in daily	**Attribut *package***	**Attribut *timeActive***
1532585317803 (26.07.2018)	com.whatsapp	394596
1532674140619 (27.07.2018)	com.whatsapp	173153
1532766942300 (28.07.2018)	com.whatsapp	51993
1532931631492 (30.07.2018)	com.whatsapp	353888
1533020964854 (31.07.2018)	com.whatsapp	4081
1532497511269 (25.07.2018)	com.whatsapp	139612

Tabelle B.20: VR 2, S 4 *timeActive*-Werte von „com.android.settings“ [Quelle: eigene Tabelle]

Dateiname in daily	**Attribut *package***	**Attribut *timeActive***
1532585317803 (26.07.2018)	com.android.settings	12130
1532674140619 (27.07.2018)	com.android.settings	16339744
1532766942300 (28.07.2018)	com.android.settings	24919
1532931631492 (30.07.2018)	com.android.settings	874194
1533020964854 (31.07.2018)	com.android.settings	33447
1532497511269 (25.07.2018)	com.android.settings	140316

Tabelle B.21: VR 2, S 4 *timeActive*-Werte der „weekly“-Datei [Quelle: eigene Tabelle]

Dateiname weekly	**Attribut *package***	**Attribut *timeActive***
1532497511269	com.whatsapp	1117323
1532497511269	com.android.settings	17424750

Addiert man alle *timeActive*-Werte der „daily“-Dateien eines bestimmten *package*-Elementes, die zu einer Woche gehören, ergibt sich der Wert, den *timeActive* in der dazugehörigen „weekly“-Datei annimmt.

B.3.4 Protokolldateien in */data/system/usagestats/0/monthly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu den vorherigen Szenarien nicht verändert.

Analyse

In dem Ordner „monthly“ befindet sich genau wie im „weekly“-Ordner eine Datei, die ein Datum vom 25.07.2018 trägt. Der Inhalt dieser Dateien stimmt jedoch nicht überein.

Folgende Schlüsse bezüglich der „monthly“-Datei haben sich ergeben.

Tabelle B.22: VR 2, S 4 *timeActive*-Werte von „com.whatsapp“ [Quelle: eigene Tabelle]

Dateiname daily	**Attribut *package***	**Attribut *timeActive***
1532585317803 (26.07.2018)	com.whatsapp	394596
1532674140619 (27.07.2018)	com.whatsapp	173153
1532766942300 (28.07.2018)	com.whatsapp	51993
1532931631492 (30.07.2018)	com.whatsapp	353888
1533020964854 (31.07.2018)	com.whatsapp	4081
1532497511269 (25.07.2018)	com.whatsapp	139612
1533108636859 (01.08.2018)	com.whatsapp	37033

Tabelle B.23: VR 2, S 4 *timeActive*-Werte aller *package*-Elemente „com.android.settings“ [Quelle: eigene Tabelle]

Dateiname daily	**Attribut *package***	**Attribut *timeActive***
1532585317803 (26.07.2018)	com.android.settings	12130
1532674140619 (27.07.2018)	com.android.settings	16339744
1532766942300 (28.07.2018)	com.android.settings	24919
1532931631492 (30.07.2018)	com.android.settings	874194
1533020964854 (31.07.2018)	com.android.settings	33447
1532497511269 (25.07.2018)	com.android.settings	140316
1533108636859 (01.08.2018)	com.android.settings	65089

Tabelle B.24: VR 2, S 4 *timeActive*-Werte der „monthly“-Datei [Quelle: eigene Tabelle]

Dateiname monthly	**Attribut *package***	**Attribut *timeActive***
1532497511269	com.whatsapp	1154356
1532497511269	com.android.settings	17489839

Addiert man alle *timeActive*-Werte der „daily“-Dateien eines bestimmten *package*-Elementes, die zu einem Monat gehören, ergibt sich der Wert, den *timeActive* in der dazugehörigen „monthly“-Datei annimmt. Auch der *lastTimeActive*-Wert jedes *package*-Elementes entspricht dem der letzten Nutzung der jeweiligen Anwendung. Dies verdeutlichen die Tabellen B.25 und B.26.

Tabelle B.25: VR 2, S 4 *lastTimeActive*-Werte der „monthly“-Datei [Quelle: eigene Tabelle]

monthly Dateiname	**Attribut *package***	**Attribut *lastTimeActive***	**Dateiname + *lastTimeActive***
1532497511269	com.whatsapp	613171858	1533110683127
1532497511269	android	519722509	1533017233778

1532497511269	com.google.android.calculator	613255964	1533110767233
1532497511269	com.android.vending	613031275	1533110542544

Tabelle B.26: VR 2, S 4 *lastTimeActive*-Werte der „daily“-Datei [Quelle: eigene Tabelle]

daily Dateiname	**Attribut *package***	**Attribut *lastTimeActive***	**Dateiname + *lastTimeActive***
1533108636859	com.whatsapp	2046268	1533110683127
1532931631492	android	85602286	1533017233778
1533108636859	com.google.android.calculator	2130374	1533110767233
1533108636859	com.android.vending	1905685	1533110542544

Es wird klar, dass die *package*-Elemente „com.whatsapp“, „com.google.android.calculator“ und „com.android.vending“ zuletzt am 01.08 benutzt wurden. Außerdem stimmt die Zeit der letzten Nutzung mit der protokollierten Zeit in der „monthly“-Datei überein. Weiterhin wurde das *package*-Element „android“ nicht zuletzt am 01.08.2018, sondern am 30.07.2018 benutzt. Auch dieser Wert wird in der „monthly“-Datei notiert.

B.3.5 Protokolldateien in */data/system/usagestats/0/yearly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu den vorherigen Szenarien nicht verändert.

Analyse

Auch in diesem Szenario stimmt die Datei des Ordners „yearly“ mit der Datei des Ordners „monthly“ überein. Dies belegt folgender Hashvergleich (Tabelle B.27):

Tabelle B.27: VR 2, S 4 Hashvergleich zwischen „monthly“- und „yearly“-Dateien [Quelle: eigene Tabelle]

MD5	**SHA1**	**Pfad**
0afe44a1bb0afebb73fffe5403cb5d69	becf95f5956a2193c774eaab83ff88a8bf05fa59	C:\android-sdk\platform-tools\0\yearly\1532497511269.xml
0afe44a1bb0afebb73fffe5403cb5d69	becf95f5956a2193c774eaab83ff88a8bf05fa59	C:\android-sdk\platform-tools\0\monthly\1532497511269.xml

B.3.6 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu den vorherigen Szenarien nicht verändert.

Analyse

In diesem Szenario wird festgestellt, welche Parameter das Ereignis ScreenLockReceiver speichert. Es ist anzumerken, dass nur Ereignisse betrachtet werden, zu denen eine gemessene Zeit im Szenario vorliegt. Auffällig ist, dass an den Tagen, an denen das Handy angeschaltet war (zwischen 25.07-01.08), mehrere ScreenLockReceiver-Ereignisse pro Tag protokolliert werden. Folgende Zusammenhänge haben sich ergeben (Tabelle B.28):

Tabelle B.28: VR 2, S 4 Ereignis ScreenLockReceiver [Quelle: eigene Tabelle]

Ereignis	**Attribut *tag***	**Attribut *Locked***	**Attribut *old-Locked***	**Handy-Zeit**	**Ablauf Szenario**
ScreenLock-Receiver	on	true	true2018-07-25 15:56:46.768	15:56	Aufleuchten von Bildschirm
ScreenLock-Receiver	off	true	true2018-07-25 16:15:56.162	16:15	Entsperren von Bildschirm
ScreenLock-Receiver	on	true	true2018-07-26 08:14:53.165	08:14	Aufleuchten von Bildschirm
ScreenLock-Receiver	on	true	true2018-07-26 08:50:14.785	08:50	Eingehende WhatsApp-Nachricht und Aufleuchten von Bildschirm
ScreenLock-Receiver	on	true	true2018-07-26 09:24:31.332	09:24	Aufleuchten von Bildschirm
ScreenLock-Receiver	on	true	true2018-07-26 10:06:30.714	10:06	Empfangen von WhatsApp-Nachrichten und Aufleuchten von Bildschirm

ScreenLock-Receiver	unlock	false	true2018-07-26 10:08:41.528	10:08	Entsperren von Bildschirm
ScreenLock-Receiver	unlock	false	true2018-07-26 13:59:05.238	13:59	Entsperren von Bildschirm
ScreenLock-Receiver	on	true	true2018-07-30 08:21:11.589	08:21	Eingehende WhatsApp-Nachricht und Aufleuchten von Bildschirm
ScreenLock-Receiver	unlock	false	true2018-07-30 08:22:28.251	08:22	Entsperren von Bildschirm
ScreenLock-Receiver	on	true	true2018-07-30 09:20:30.399	09:20	Aufleuchten von Bildschirm
ScreenLock-Receiver	on	true	true2018-07-31 11:18:46.796	11:18	Aufleuchten von Bildschirm
ScreenLock-Receiver	unlock	false	true2018-07-31 13:48:26.642	13:48	Entsperren von Bildschirm
ScreenLock-Receiver	on	true	true2018-08-01 09:18:46.275	09:18	Aufleuchten von Bildschirm

B.3.7 Ordner *recent_tasks* und *recent_images*

Aufbau

In dem Ordner *recent_images*, welcher auf der Datenpartition in dem Ordner *system_ce/0/* liegt, werden verschiedene *.png*-Dateien abgespeichert. Diese Dateien haben bezüglich des Dateinamens alle dieselbe Syntax. Diese sieht folgendermaßen aus: *<Zahl>_task_thumbnail*. Es werden in diesem Ordner jedoch keine Zeitstempel zu den Screenshots abgelegt. Betrachtet man den Inhalt dieses Ordners, werden einem folgende Dateien mittels des Kommandos *ls -al* angezeigt (Abbildung B.4):

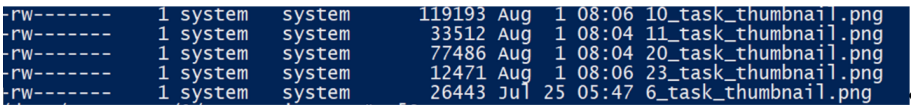

Abbildung B.4: VR 2, S4 Inhalt *recent_images* [Quelle: eigene Abbildung]

Ein Beispiel für eine *.png*-Datei im Ordner *recent_images* ist in Abbildung B.5 zu sehen.

Abbildung B.5: VR 2, S 4 Beispiel *.png*-Datei [Quelle: eigene Abbildung]

Recent_tasks stellt einen weiteren Unterordner von *system_ce/0/* dar. Dieser besteht aus *XML*-Dateien, welche alle nach einer „Task“ benannt sind, die sich auch im Ordner *recent_images* befinden (siehe Abbildung B.6):

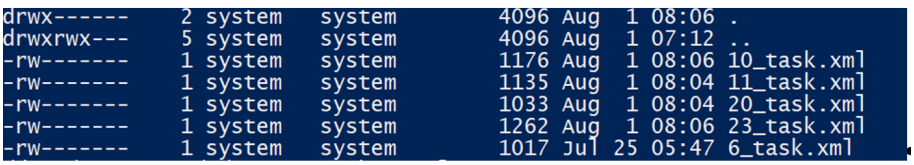

Abbildung B.6: VR 2, S 4 Inhalt *recent_tasks* [Quelle: eigene Abbildung]

Die *XML*-Dateien im Ordner *recent_tasks* weisen alle denselben Aufbau auf. Sie beginnen mit dem Wurzelelement *task*, welches verschiedene Attribute besitzt. Zu diesen Attributen gehören *min_height*, *min_width*, *privileged*, *resize_mode*, *calling_package*, *calling_uid*, *next_affiliation*, *prev_affiliation*, *task_affiliation*, *task_affiliation_color*, *task_thumbnail_info_screen_orientation*, *task_thumbnail_task_height*, *task_thumbnail_task_width*, *task_description_colorBackground*, *task_description_color*, *never_relinquish_identity*, *last_time_moved*, *last_active_time*, *first_active_time*, *task_type*, *effective_uid*, *user_setup_complete*, *user_id*, *asked_compat_mode*, *auto_removed_recents*, *root_has_reset*, *affinity*, *real_activity_suspended*, *real_activity* und *task_id*. Das Wurzelelement wiederum besitzt ein Kind-Knoten *intent*, das die Attribute *flags*, *component* und *action* besitzt. In Abbildung B.7 ist der Aufbau dieser *XML*-Dateien veranschaulicht:

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<task min_height="-1" min_width="-1" privileged="true" resize_mode="2" calling_package="com.google.android.googlequicksearchbox"
calling_uid="10041" next_affiliation="-1" prev_affiliation="-1" task_affiliation="10" task_affiliation_color="-12417548"
task_thumbnailinfo_screen_orientation="1" task_thumbnailinfo_task_height="960" task_thumbnailinfo_task_width="540"
task_description_colorBackground="fffafafa" task_description_color="ff4285f4" never_relinquish_identity="true"
last_time_moved="1533110771564" last_active_time="1533110798482" first_active_time="1532499011159" task_type="0"
effective_uid="10041" user_setup_complete="true" user_id="0" asked_compat_mode="false" auto_remove_recents="false" root_has_reset="false"
affinity="com.google.android.googlequicksearchbox" real_activity_suspended="false"
real_activity="com.google.android.googlequicksearchbox/com.google.android.apps.gsa.searchnow.SearchNowActivity" task_id="10">
    <intent flags="10108000"
        component="com.google.android.googlequicksearchbox/com.google.android.apps.gsa.searchnow.SearchNowActivity"
        action="android.intent.action.ASSIST"/>
</task>
```

Abbildung B.7: VR 2, S4 Aufbau XML-Datei *recent_tasks* [Quelle: eigene Abbildung]

Analyse

In den nächsten Schritten wird festgestellt, ob zwischen den Ordnern *recent_images* und *recent_tasks* ein Zusammenhang besteht. Dafür wird sich vor allem auf die Attribute *first_active_time* und *last_active_time* des Wurzelelementes *task* der *XML*-Dateien in *recent_tasks* konzentriert. In der nächsten Tabelle (Tabelle B.29) werden die berechneten Zeiten dieses Ordners mit dem Ablauf des Szenarios verglichen.

Tabelle B.29: VR 2, S 4 Protokollierung in *recent_tasks* [Quelle: eigene Tabelle]

Task	**Attribut *component***	**Attribut *calling_package***	**Attribut *first_active_time***	**Ablauf Szenario**	**Epoch-Zeit**	**Handy-Zeit**
11	com.android .settings/ .Settings	com.google .android.apps. nexuslauncher	1532499066774 (25.07, 08:11:06)	Einstel-lungs-App geöffnet	25.07, 08:11:06	08:11

23	com.google .android. calculator/ com.android .calculator2. Calculator Google	com.google .android.apps. nexuslauncher	1533110725936 (01.08, 10:05:25)	Auf Rechner-App gewechselt ohne App-Wechsel	01.08, 10:05:25	10:05

Aus dieser Tabelle geht hervor, dass das Attribut *first_active_time* nach Ablauf des Szenarios die Zeit angibt, als die App gestartet wird. In der nächsten Tabelle wird der Wert des Attributes *last_active_time* genauer untersucht.

Tabelle B.30: VR 2, S 4 Attribut *last_active_time* in *recent_tasks* [Quelle: eigene Tabelle]

Task	**Attribut *component***	**Attribut *calling_package***	**Attribut *last_active_time***	**Ablauf Szenario**	**Epoch-Zeit**	**Handy-Zeit**
20	com.whats-app/.Main	com.whatsapp	1533110684101 (01.08, 10:04:44)	nach einem App-Wechsel Whats-App verlassen	10:04:44	10:04
10	com.google .android. googlequick- searchbox/ com.google. android.apps .gsa.search- now.Search- NowActivity	com.google .android. googlequick- searchbox	1533110798482 (01.08, 10:06:38)	Nach einem App-Wechsel Google-quick-search-box verlassen		10:06
11	com.android .settings/ .Settings	com.google .android.apps. nexuslauncher	1533110645669 (01.08, 10:04:05)	Einstel-lungs-App verlassen	10:04:05	10:04

Nach dieser Tabelle scheint das Attribut *last_active_time* zu protokollieren, wann eine

App das letzte Mal geschlossen wurde bzw. wann sie das letzte Mal aktiv war. Alle im Ordner *recent_tasks* aufgeführten Applikationen sind in dem Ordner *recent_images* mit einem Bild belegt.

Anhang C: Versuchsreihe 3

C.1 Szenario 2

C.1.1 Protokolldateien unter */data/system/usagestats/0/daily/*

Aufbau

In diesem Ordner befinden sich vier Dateien. Somit sind im Vergleich zu Szenario 1 dieser Versuchsreihe noch zwei weitere Dateien hinzugekommen, welche einen Zeitstempel vom 07.08.2018 und 08.08.2018 zeigen.

Analyse

In Tabelle C.1 werden die *package*-Elemente genauer betrachtet und mit den gemessenen Zeiten des Szenarios verglichen.

Tabelle C.1: VR 3, S 2 Vergleich *package*-Elemente und Szenario [Quelle: eigene Tabelle]

Attribut *package*	**Attribut *lastTime-Active***	**Dateiname + *lastTime-Active***	**Handy-Zeit**	**Epoch-Zeit**	**Szenario**
com.whats-app	35443045	08.08.2018, 11:50:43.045	11:50	11:50:43	Verlassen von Whats-app
com.google. android.cal-culator	35306751	08.08.2018, 11:48:26.751	11:48	11:48:28	Verlassen der Rechner-App
com.android. vending	35511489	08.08.2018, 11:51:51.489	11:51	11:51:52	Verlassen der Playstore-App

Hier wird deutlich, dass die gemessenen Zeiten des Szenarios mit den Zeiten in der Protokolldatei übereinstimmen.

Als Nächstes wird die Bedeutung des Attributes *lastTimeActiveSystem* verdeutlicht (siehe Tabelle C.2).

In dieser Tabelle wird gegenübergestellt, welche Werte *lastTimeActiveSystem* annimmt, wenn die dazugehörige App im Szenario nicht verwendet wird. Man kann erkennen, dass die Addition mit dem Dateinamen einen Wert ergibt, der im Szenario einer Zeit entspricht, als das Handy hochgefahren wurde. Nimmt das Attribut *lastTimeActive* einen positiven Wert an, stimmt dieser mit dem entsprechenden Wert von *lastTimeActiveSystem* überein.

Tabelle C.2: VR 3, S 2 Attribut *lastTimeActiveSystem* [Quelle: eigene Tabelle]

Attribut *package*	**Attribut *lastTimeActiveSystem***	**Attribut *lastTimeActive***	**Dateiname + *lastTimeActiveSystem***	**Szenario**
com.android.providers.calendar	35277632	(-) 1533686400000	11:47:57.632	Hochfahren des Smartphones
com.android.providers.telephony	35209773	(-) 1533686400000	11:46:49.773	Hochfahren des Smartphones

Als Nächstes werden die *event*-Elemente genauer betrachtet. Dabei werden die Werte der *event*-Elemente mit den gemessenen Werten des Szenarios verglichen (Tabelle C.3):

Tabelle C.3: VR 3, S 2 Vergleich Szenario und *event*-Elemente [Quelle: eigene Tabelle]

Attribut *class*	**Dateiname + *time***	**Szenario**	**Handy-Zeit**	**Epoch-Zeit**
com.whatsapp.Main	08.08.2018, 11:48:50.989	Betreten WhatsApp-Mainmenü	11:48	11:48:52
com.whatsapp.QuickContactActivity	08.08.2018, 11:49:24.774	Anschauen eines WhatsApp-Profilfotos	11:49	11:49:25
com.whatsapp.QuickContactActivity	08.08.2018, 11:50:10.734	Verlassen der Profilfoto-Ansicht	11:50	11:50:10
com.android.calculator2.Calculator-Google	08.08.2018, 11:47:47.936	Betreten der Rechner-App	11:47	11:47:48
com.android.calculator2.Calculator-Google	08.08.2018, 11:48:26.751	Verlassen der Rechner-App	11:48	11:48:28

com.google.android.finsky.activities.MainActivity	08.08.2018, 11:51:18.930	Betreten der PlayStore-App	11:51	11:51:20
com.google.android.finsky.activities.MainActivity	08.08.2018, 11:51:51.489	Verlassen der PlayStore-App	11:51	11:51:52

C.1.2 Protokolldateien in */data/system/usagestats/0/weekly/*

Aufbau

In diesem Ordner befindet sich eine *XML*-Datei, welche denselben Namen wie die Datei aus Szenario 1 des Ordners „weekly“ trägt. Der Aufbau dieser Dateien hat sich im Vergleich zu Szenario 1 nicht verändert.

Analyse

Im Folgenden werden die *package*-Elemente dieser Datei näher betrachtet.

Tabelle C.4: VR 3, S 1 *package*-Elemente der „weekly“-Datei [Quelle: eigene Tabelle]

Attribut *package*	**Attribut *timeActive***	***lastTimeActive***	**Dateiname + *lastTimeActive***
com.whatsapp	511476	241568410	08.08.2018, 11:50:43.045
com.google.android.calculator	44156	241432116	08.08.2018, 11:48:26.751
com.android.vending	133159	241636854	08.08.2018, 11:51:51.489

Die Summe von Dateiname und *lastTimeActive*-Wert ergibt dabei dieselbe Zeit, welche auch die letzte „daily“-Datei des Ordners speichert (siehe C.1).
Die folgenden zwei Tabellen zeigen den Zusammenhang der vier „daily“-Dateien und der „weekly“-Datei.

Tabelle C.5: VR 3, S 2 *timeActive*-Werte der „daily“-Dateien [Quelle: eigene Tabelle]

Dateiname	**Attribut *package***	**Attribut *timeActive***	**Attribut *lastTimeActive***	**Dateiname + *lastTimeActive***
1533480274635 (erste „daily“-Datei)	com.whatsapp	91358	18464031	05.08.2018, 21:52:18.666
1533480274635 (erste „daily“-datei)	com.google.android.calculator	5341	1250333	05.08.2018, 17:05:24.968
1533480274635 (erste „daily“-Datei)	com.android.vending	100600	18354987	05.08.2018, 21:50:29.622
1533513600000 (zweite „daily“-Datei)	com.whatsapp	282043	23805482	06.08.2018, 08:36:45.482
1533600000000 (dritte „daily“-Datei)	com.whatsapp	26119	27050814	07.08.2018, 09:30:50:814
1533686400000 (vierte „daily“-Datei)	com.whatsapp	111956	35443045	08.08.2018, 11:50:43.045
1533686400000 (vierte „daily“-Datei)	com.google.android.calculator	38815	35306751	08.08.2018, 11:48:26.751
1533686400000 (vierte „daily“-Datei)	com.android.vending	32559	35511489	08.08.2018, 11:51:51.489

Tabelle C.6: VR 3, S 2 Summe *timeActive*-Werte der „daily“-Dateien [Quelle: eigene Tabelle]

Attribut *package*	**Summe *timeActive*-Werte**
com.whatsapp	511476
com.google.android.calculator	44156
com.android.vending	133159

Tabelle C.6 zeigt dieselben Werte, welche auch in der „weekly“-Datei gespeichert sind (siehe Tabelle C.4).

C.1.3 Protokolldateien in */data/system/usagestats/0/monthly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu Szenario 1 nicht verändert.

Analyse

Im Folgenden findet eine Auswertung der package-Elemente statt (siehe Tabelle C.7):

Tabelle C.7: VR 3, S1 Auswertung *package*-Elemente der „monthly“-Datei [Quelle: eigene Tabelle]

Attribut *package*	**Attribut *timeActive***	**Attribut *lastTimeActive***	**Dateiname + *lastTimeActive***
com.whatsapp	511476	1806570479	08.08.2018, 11:50:43.045
com.android.vending	128595	1806638923	08.08.2018, 11:51:51.489
com.google.android .calculator	44156	1806434185	08.08.2018, 11:48:26.751

Es wird deutlich, dass hier dieselben Werte wie in der „weekly“-Datei gespeichert werden (siehe Tabelle C.4).

C.1.4 Protokolldateien in */data/system/usagestats/0/yearly/*

Aufbau

Der Aufbau hat sich im Vergleich zu Szenario 1 nicht verändert.

Analyse

Auch hier werden dieselben Ergebnisse wie bei der „monthly“- und „weekly“-Datei erreicht, wenn man die *lastTimeActive*-Werte auf den Dateinamen addiert.

C.1.5 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Aufbau

Der Aufbau dieser Protokolldatei entspricht dem von Versuchsreihe 2. Es ist zu erkennen, dass eine fortlaufende Protokollierung verschiedener Ereignisse stattfindet.

Analyse

Es kann beobachtet werden, dass für alle vergangenen Tage an denen das Handy angeschaltet war, Einträge zum Batteriestatus existieren. Für den 08.08.2018 wurden bezüglich des Batteriestatus folgende Einträge gefunden (siehe Tabelle C.8):

Tabelle C.8: VR 3, S 2 BatteryState-Ereignisse WhatsApp-Protokolldatei [Quelle: eigene Tabelle]

Attribut *plugged*	**Zeitstempel Log-datei**	**Szenario**	**gemessene Handy-Zeit**
0	2018-08-08 11:47:06.105	Handy anschalten	11:47
2	2018-08-08 11:47:10.662	Handy mit USB-Kabel verbinden	11:47
1	2018-08-08 11:52:13.703	Handy mit Netzteil-Ladegerät verbinden	11:52

Auch zum ScreenLockReceiver-Ereignis konnten Einträge zu jedem Tag gefunden werden.

C.1.6 Ordner *recent_tasks* und *recent_images*

Aufbau

Die Syntax der Dateinamen in *recent_images* ist gleich geblieben. Es sind jedoch noch mehr Einträge hinzugekommen, während andere gelöscht wurden.
Folgende Dateien befinden sich in den Ordnern *recent_images* und *recent_tasks*:

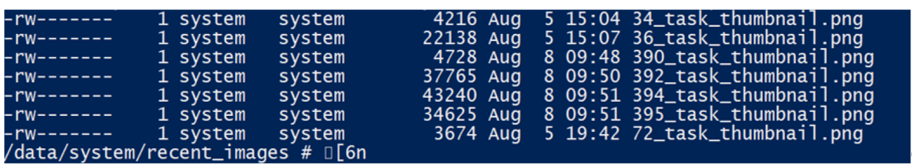

Abbildung C.1: VR 3, S 2 Einträge *recent_images* [Quelle: eigene Abbildung]

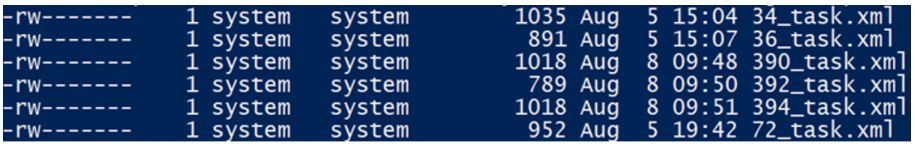

Abbildung C.2: VR 3, S 2 Einträge *recent_tasks* [Quelle: eigene Abbildung]

Der Aufbau der Dateien im Ordner *recent_tasks* entspricht dem Aufbau aus Szenario 1.

Analyse

Eine Auswertung der Zeitstempel, welche in den *XML*-Dateien unter *recent_tasks* gespeichert sind, wird in Tabelle C.9 gezeigt.

Tabelle C.9: VR 3, S 2 Auswertung Zeitstempel in *recent_tasks* [Quelle: eigene Tabelle]

Tasknummer	**Attribut *last_active_time***	**gemessene Epoch-Zeit**	**Szenario**
394 (Screenshot PlayStore)	08.08.2018, 11:51:52.857	11:51:52	Verlassen der Playstore-App
392 (Screenshot WhatsApp)	08.08.2018, 11:50:44.405	11:50:43	Verlassen von WhatsApp
390 (Screenshot Rechner)	08.08.2018, 11:48:28.116	11:48:28	Verlassen der Rechner-App

C.2 Szenario 3

C.2.1 Protokolldateien in */data/system/usagestats/0/daily/*

Aufbau

In diesem Ordner befinden sich fünf *XML*-Dateien. Somit ist im Gegensatz zu Szenario 2 eine Weitere hinzugekommen. Alle anderen vier Dateien stimmen mit denen von Szenario 2 des Ordners „daily“ überein.

Analyse

Auf die *package*-Elemente wird in diesem Szenario nicht weiter eingegangen, weil diese dieselben Ergebnisse wie zuvor zeigen. Bei den *event*-Elementen werden nur diese betrachtet, die Attribute der *config*-Elemente aufweisen. In Tabelle C.10 und C.11 werden

diese Elemente mit dem Ablauf des Szenarios verglichen.

Tabelle C.10: VR 3, S 3 Bildschirmrotation [Quelle: eigene Tabelle]

Attribut *package*	**Attribut *ori***	**Dateiname + *time***	**Attribut *type***	**Szenario**	**Handy-Zeit**
Android	2	10.08.2018, 08:42:45.095	5	WhatsApp-Ansicht drehen	08:42
Android	1	10.08.2018, 08:43:22	5	WhatsApp-Ansicht nochmal drehen	08:43

Tabelle C.11: VR 3, S 3 Änderung Spracheinstellung [Quelle: eigene Tabelle]

Attribut *package*	**Dateiname + *time***	**Attribut *type***	**Attribut *locale***	**Szenario**	**Handy-Zeit**
Android	10.08.2018, 08:45:35	5	en-GB	Spracheinstellung Englisch	08:45
Android	10.08.2018, 08:46:31	5	de-DE	Spracheinstellung Deutsch	08:46

Es ist zu erkennen, dass sowohl die Drehung der WhatsApp-Ansicht als auch die Änderung der Spracheinstellung in den *event*-Elementen gezeigt werden.

Zur Änderung der Spracheinstellung findet sich auch ein Eintrag in den *config*-Elementen. Alle weiteren *config*-Elemente weisen Zeiten auf, die der Zeit des Startens des Handys entsprechen.

Tabelle C.12: VR 3, S 3 Änderung der *config*-Elemente [Quelle: eigene Tabelle]

Attribut *lastTimeActive* + Dateiname	**Attribut *ori***	**Attribut *locale***	**Szenario**	**Handy-Zeit**
10.08.2018, 08:46:31	1	en-GB	Spracheinstellung Englisch	08:45
10.08.2018, 08:46:31	1	de-DE	Spracheinstellung Deutsch	08:46

C.2.2 Protokolldateien in */data/system/usagestats/0/weekly/*

Aufbau

In diesem Szenario befinden sich zwei Dateien in dem Ordner „weekly“. Im Vergleich zu Szenario 2 ist eine weitere Datei hinzugekommen, welche *1533772800000.xml* heißt. Die *1533480274635.xml* aus Szenario 2 und diesem Szenario stimmen überein.

Der Aufbau der Dateien des Ordners „weekly“ dieses Szenarios entsprechen dem Aufbau der Dateien des Ordners „weekly“ der vorherigen Szenarien.

Analyse

Im Folgenden wird die *1533772800000.xml*-Datei genauer untersucht.

Tabelle C.13: VR 3, S 3 Untersuchung „weekly“ -Datei [Quelle: eigene Tabelle]

Attribut *lastTime-Active* + Dateiname	**Attribut *package***	**Attribut *timeActi-ve***	**Attribut *lastE-vent***	**Szenario**	**Handy-Zeit**	**Epoch-Zeit**
10.08.2018, 08:43:49.913	com.whatsapp	92087	2	Verlassen von Whats-App	08:43	08:43:53
10.08.2018, 08:44:46.986	com.google .android.cal-culator	20910	2	Verlassen der Rechner-App	08.44	08:44:48

Hier stimmen die *timeActive*-Werte mit denen der zuletzt angelegten „daily“-Datei überein. Es wird demnach scheinbar für dieses Szenario eine neue „weekly“-Datei angelegt.

C.2.3 Protokolldateien in */data/system/usagestats/0/monthly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu den vorherigen Szenarien nicht verändert.

Analyse

Hier ist das gleiche Verhalten wie in Szenario 2 zu erkennen.

C.2.4 Protokolldateien in */data/system/usagestats/0/yearly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu den vorherigen Szenarien nicht verändert.

Analyse

Auch hier ist das Verhalten der *XML*-Datei identisch zu dem aus Szenario 2 dieser Versuchsreihe. Die *XML*-Datei aktualisiert die Werte der Applikationen, die am 10.08.2018 benutzt wurden. Weiterhin vergrößert sich bei diesen Apps auch die *timeActive*.

C.2.5 Protokolldateien in */data/data/com.whatsapp/files/Logs/*

Aufbau

Der Aufbau hat sich im Vergleich zu den vorherigen Szenarien nicht verändert.

Analyse

Im Folgenden werden die ScreenLockReceiver-Ereignisse betrachtet, zu denen eine gemessene Zeit vorliegt.

Tabelle C.14: VR 3, S 3 ScreenLockReceiver-Ereignisse [Quelle: eigene Tabelle]

Attribut *tag*	**Attribut *old-Locked***	**Szenario**	**Handy-Zeit**
Unlock	true2018-08-10 08:47:07.741	Bildschirm entsperren	08:47
Off	false2018-08-10 08:48:05.325	Bildschirm sperren	08:48
On	true2018-08-10 08:48:10.187	Bildschirm aufleuchten	08:48

C.2.6 Ordner *recent_tasks* und *recent_images*

Aufbau

In diesem Ordner sind im Vergleich zu Szenario 2 neue Screenshots hinzugekommen und andere wurden gelöscht. Die Anwendungen, die seit Szenario 2 nicht mehr benutzt wurden (PlayStore und YouTube), zeigen immer noch dieselben Screenshots und die gleiche Nummerierung. Das bedeutet, dass die Nummern 34, 72 und 394 gleichgeblieben sind. Alle anderen Screenshots wurden gelöscht und durch neue ersetzt.

Analyse

Im Folgenden werden die neu hinzugekommen Screenshots inklusive der neu angelegten *XML*-Dateien in *recent_tasks* in Verbindung gesetzt und mit dem Ablauf des Szenarios verglichen.

Tabelle C.15: VR 3, S 3 *recent_task*-Dateien [Quelle: eigene Tabelle]

Task-nummer	**Attribut *last_active-_time***	**Attribut *component***	**Abbildung *recent_images***	**Szenario**	**Epoch-Zeit**	**Handy-Zeit**
455	10.08.2018, 08:43:51.832	com.whats-app/.Main	WhatsApp-Mainmenü	WhatsApp verlas-sen	08:43:53	08:43
457	10.08.2018, 08:44:48.899	com.google .android.cal-culator/ com.google. android.cal-culator2.Cal-culatorGoogle	Rechner-App	Rechner-App verlas-sen	08:44:48	08:44
459	10.08.2018, 08:46:40.007	com.android. settings/ .Settings	Einstellungs-App	Einstel-lungen verlas-sen		08:46

C.3 Szenario 4

C.3.1 Protokolldateien in */data/system/usagestats/0/daily/*

Dieser Ordner besteht aus sechs Dateien, die folgende Zeitstempel aufweisen:

Tabelle C.16: VR 3, S 4 Aktualisierung des Dateinamens [Quelle: eigene Tabelle]

Dateiname	Zeitstempel
1533600002308	07.08.2018, 02:00:02
1533686402308	08.08.2018, 02:00:02
1533859202308	10.08.2018, 02:00:02
1533945602308	11.08.2018, 02:00:02
1534032000000	12.08.2018, 02:00:00
1534118400000	13.08.2018, 02:00:00

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu den vorherigen Szenarien nicht verändert.

Analyse

Es ist auffällig, dass alle *XML*-Dateien einen Zeitstempel aufweisen, der eine Zeit von 02:00 Uhr hat. Die älteste „daily“-Datei in diesem Ordner verweist auf ein Datum vom 07.08.2018. Vergleicht man die Dateien dieses „daily“-Ordners mit denen aus dem „daily“-Ordner von Szenario 3, fällt auf, dass die Dateinamen aktualisiert werden, der Inhalt jedoch nicht. Die Tabellen C.17 und C.18 zeigen die Unterschiede in den Dateinamen.

Tabelle C.17: VR 3, S 4 Dateinamen Szenario 3 [Quelle: eigene Tabelle]

Dateiname Szenario 3	Zeitstempel der Datei
1533600000000	07.08.2018, 02.00.00
1533686400000	08.08.2018, 02.00.00
1533859200000	10.08.2018, 02:00:00

Tabelle C.18: VR 3, S 4 Dateinamen in Szenario 4 [Quelle: eigene Tabelle]

Dateiname Szenario 4	Zeitstempel der Datei
1533600002308	07.08.2018, 02:00:02
1533686402308	08.08.2018, 02:00:02
1533859202308	10.08.2018, 02:00:02

C.3.2 Protokolldateien in */data/system/usagestats/0/weekly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu den vorherigen Szenarien nicht verändert.

Analyse

Auch hier haben sich die Zeitstempel im Gegensatz zu Szenario 3 um zwei Sekunden aktualisiert.

C.3.3 Protokolldateien in*/data/system/usagestats/0/monthly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu den vorherigen Szenarien nicht verändert.

Analyse

Das Verhalten der Dateien ist analog zu den „weekly“-Dateien. Auch hier hat demnach eine Aktualisierung der Dateinamen um zwei Sekunden stattgefunden.

C.3.4 Protokolldateien in */data/system/usagestats/0/yearly/*

Aufbau

Der Aufbau dieser Dateien hat sich im Vergleich zu den vorherigen Szenarien nicht verändert.

Analyse

Hier ist ebenfalls das Verhalten der „yearly“-Dateien analog zu dem der „monthly“- und „weekly“-Dateien. Die Dateinamen wurden um zwei Sekunden aktualisiert.

C.3.5 Protokolldatei in */data/data/com.whatsapp/files/Logs/*

Aufbau

Der Aufbau dieser Datei hat sich im Vergleich zu den vorherigen Szenarien nicht verändert.

Analyse

Es kann beobachtet werden, dass zu jedem Tag, an dem das Handy angeschaltet war, sowohl ScrenLockReceiver-Ereignisse als auch der Batteriestatus notiert werden. Dies gilt auch für die Tage, an denen das Handy nicht benutzt wurde. Ein Beispiel hierfür sind die Einträge des 13.08.2018.
• ScreenLockReceiver; tag=on; locked=true; oldLocked=true2018-08-13 09:56:51.915
• BatteryState{health=good, level=100, plugged=0, scale=100, percent=100.0}2018-08-13 09:57:39.462

C.3.6 Ordner *recent_tasks* und *recent_images*

Aufbau

Auch der Aufbau der *XML*-Dateien in *recent_tasks* hat sich im Vergleich zu den vorherigen Szenarien nicht verändert.

Analyse

Es ist zu erkennen, dass in *recent_tasks* noch eine *XML*-Datei vorhanden ist, welche eine Zeit speichert, die auf den 05.08.2018 zurückzuführen ist. Weiterhin ist die zuletzt angelegte *XML*-Datei auf den 12.08.2018 zurückzuführen. Für den 13.08.2018 existiert keine *XML*-Datei.

Literaturverzeichnis

[Al Mutawa2012] Al Mutawa, Noora; Baggili, Ibrahim; Marrington, Andrew (2012): Forensic analysis of social networking applications on mobile devices. In: Digital Investigation 9, S24-S33. DOI: 10.1016/j.diin.2012.05.007.

[Alyahya2017] Alyahya, Tadani; Kausar, Firdous (2017): Snapchat Analysis to Discover Digital Forensic Artifacts on Android Smartphone. In: Procedia Computer Science 109, S. 1035–1040. DOI: 10.1016/j.procs.2017.05.421.

[Anglano2014] Anglano, Cosimo (2014): Forensic Analysis of WhatsApp Messenger on Android Smartphones. In: Digital Investigation 11 (3), S. 201–213. DOI: 10.1016/j.diin.2014.04.003.

[Anglano2017] Anglano, Cosimo; Canonico, Massimo; Guazzone, Marco (2017): Forensic analysis of Telegram Messenger on Android smartphones. In: Digital Investigation 23, S. 31–49. DOI: 10.1016/j.diin.2017.09.002.

[Annuzzi2015] Annuzzi, Joseph; Conder, Shane; Darcey, Lauren (2015): Advanced Android application development. 4th ed (Online-Ausg.). Upper Saddle River, NJ: Addison-Wesley.

[Baltes-Götz2018] Baltes-Götz, Bernhard (2018): Einführung in die Entwicklung von Android-Apps. Universität Trier Zentrum für Informations-, Medien- und Kommunikationstechnologie.

[Barton2016] Barton, Thomas; Müller, Christian; Seel, Christian (2016): Mobile Anwendungen in Unternehmen. Wiesbaden: Springer Fachmedien Wiesbaden.

[Becker2015] Becker, Arno; Pant, Marcus (2015): Android 5. Programmieren für Smartphones und Tablets. 4., aktualisierte und erweiterte Auflage. Heidelberg [Germany]: Dpunkt.verlag.

[Bommisetty2014] Bommisetty, Satish; Tamma, Rohit; Mahalik, Heather (2014): Practical mobile forensics. Dive into mobile forensics on iOS, Android, Windows, and BlackBerry devices with this action-packed, practical guide. Birmingham, UK: Packt Pub (Community experience distilled).

[Chang2017] Chang, Ming Sang; Chang, Chih Yen (2017): Forensic Analysis of LINE Messenger on Android. In: Journal of Computers Vol. 29 No. 1, 2018, pp. 11-20 doi:10.3966/199115992018012901002.

[Dembowski2017] Dembowski, Klaus (2017): Smartphone- und Tablet-Hacks. Mess-, Steuer- und Kommunikationsschaltungen selbst gebaut und programmiert. 1. Auflage. Heidelberg: Dpunkt.verlag.

[Desautels2014] Desautels Julie (2014): Google Glass Forensics. Champlain College.

[Di Leom2015] Di Leom, Ming; DOrazio, Christian Javier; Deegan, Gaye; Choo, Kim-Kwang Raymond (2015): Forensic Collection and Analysis of Thumbnails in Android. In: 2015 IEEE Trustcom/BigDataSE/ISPA. 2015 IEEE Trustcom/BigDataSE/ISPA. Helsinki, Finland, 20.08.2015 - 22.08.2015: IEEE, S. 1059–1066.

[Drake2014] Drake, Joshua J. (2014): Android hacker's handbook. Indianapolis, IN: John Wiley & Sons.

[Dutta2016] Dutta, Anandi; Bayoumi, Magdy (2016): Introducing a Novel Smart Design Framework for a Reconfigurable Multi-Processor Systems-on-Chip (MPSoC) Architecture. In: 2016 IEEE International Conference on Smart Computing (SMARTCOMP). St Louis, MO, USA, 18.05.2016 - 20.05.2016: IEEE, S. 1–3.

[Elenkov2015] Elenkov, Nikolay (2015): Android security internals. An in-depth guide to Android's security architecture. Online-Ausg. San Francisco, CA: No Starch Press.

[Gargenta2012] Gargenta, Marko (2012): Einführung in die Android-Entwicklung. 1. Auflage. CA 95472: O'Reilly Media.

[Hitzler2008] Hitzler, Pascal (2008): Semantic web. Grundlagen. 1. Aufl. Berlin: Springer Berlin (EXamen.press).

[Hoog2011] Hoog, Andrew (2011): Android Forensics: Investigation, Analysis and Mobile Security for Google Android: Syngress.

[Immler2016] Immler, Christian (2016): Android Hacking. Behalten Sie Ihre Daten auf dem Handy und nicht beim Geheimdienst: Schwachstellen und Sicherheitslücken finden und beseitigen. 1. Auflage. Haar: Franzis Verlag (Hacking).

[Jadhav2015] Jadhav, Mahesh Sambhaji (2015): Software Testing in Multimedia and Graphics: Easy to understand Quick to Learn: HighTechEasy.

[Lee2012] Lee, Wei-Meng (2012): Beginning Android 4 application development. Hg. v. Chaim Krause. Indianapolis, Ind.: Wrox/John Wiley & Sons.

[Liu2011] Liu, Jianye; Yu, Jiankun (2011): Research on Development of Android Applications. In: 2011 4th International Conference on Intelligent Networks and Intelligent Systems. Kuming, China, 01.11.2011 - 03.11.2011: IEEE, S. 69–72.

[Mahajan2013] Mahajan, Aditya; S. Dahiya, M.; P. Sanghvi, H. (2013): Forensic Analysis of Instant Messenger Applications on Android Devices. In: IJCA 68 (8), S. 38–44. DOI: 10.5120/11602-6965.

[Mahalik2016] Mahalik, Heather; Bommisetty, Satish; Tamma, Rohit (2016): Practical mobile forensics. Practical mobile forensics: a hands-on guide to mastering mobile forensics for the iOS, Android, and Windows Phone platforms. Second edition: Packt Publishing.

[Meier2012] Meier, Reto (2012): Professional Android 4 application development. [3rd ed.]. Indianapolis, IN.: Wiley/[Wrox].

[Morgillo2016] Morgillo, Ivan; Viola, Stefano (2016): Learning embedded Android N programming. Create the perfectly customized system by unleashing the power of Android OS on your embedded device. Birmingham: Packt Publishing.

[Muth2013] Muth, Denise (2013): Leitfaden zur forensischen Untersuchung von Android-Smartphones. In: Voluntaris 1 (1), S. 35–38. DOI: 10.5771/2196-3886-2013-1-35.

[Nind2017] Nind, Andrew (2017): Amnesty International Digital Forensic Analysis Services Report. SecureWorks United Kingdom House.

[Parikh2015] Parikh, Shreyas; Chavda, Dhaval; Chakraborty, Shourjo; Rughani, Dr. Parag H (2015): Analysis of Android Smart Watch Artifacts. In: International Journal of Scientific & Engineering Research, Issue 8, August-2015.

[Pieterse2018] Pieterse, Heloise; Olivier, Martin; van Heerden, Renier (2018): Smartphone data evaluation model: Identifying authentic smartphone data. In: Digital Investigation 24, S. 11–24. DOI: 10.1016/j.diin.2018.01.017.

[Porteck2018] Porteck, Stefan; Spier, Alexander (2018): Lückenlose Backups per Root-Tool (2018): In: c't Android (2018): & Mehr aus Smartphones und Tablets rausholen (German Edition). Online verfügbar unter https:// books.google.de/books?id=zrZKD wAAQ BAJ&pg=PA63&dq=recovery+android&hl=de&sa =X&ved=0ahUKEwjN0erxi57cAhUGLewKHVedB VgQ6AEIKjAA#v=onepage&q=recovery%20android &f=false.

[Ray2002] Ray, Eric T. (2002): Einführung in XML. 1. Aufl., korrigierter Nachdr. Beijing [u.a.]: O'Reilly.

[Ray2001] Ray, Erik T. (2001): Learning XML. Beijing, Cambridge, Mass.: O'Reilly.

[Regupathy2014] Regupathy, Rajaram (2014): Unboxing Android USB. A hands-on approach with real World examples. Berkeley, CA, New York, NY: Apress; Distributed to the Book trade worldwide by Springer.

[Rehberg2014] Rehberg, Andreas Itzchak (2014): Das inoffizielle Android-Handbuch. Einsteiger-Workshop, Apps, Datensicherung, Sicherheit, Privatsphäre, Tuning, Root-Zugang und mehr: Mit Android können Sie mehr als nur telefonieren! 4. Aufl. Haar: Franzis Verlag (Smartphone Programmierung).

[Reibold2015] Reibold, Holger (2015): Galaxy S6 kompakt. Das Anwenderhandbuch. [Saarbrücken]: Brain Media (Mobile. Edition).

[Reibold2016] Reibold, Holger (2016): Android Forensik Quickstart: Der praxisorientierte Einstieg in die Welt der digitalen Forensik von Android-Geräten: BookRix.

[Saurabh2016] Saurabh, Manjrekar; Ramesh, Bhati (2016): Custom ROM – a prominent aspects of Android. In: International Journal of Advanced Research in Computer Engineering & Technology (IJARCET).

[Singh2017] Singh, Aman Kr.; Prajapati, Ashish Kr.; Kumar, Vikash; Mishra, Subhankar (2017): Usage Analysis of Mobile Devices. In: Procedia Computer Science 122, S. 657–662. DOI: 10.1016/j.procs.2017.11.420.

[Singh2018] Singh, Harjinder; Kumar, Mr.Ashwani (2018): Analysis of WhatsApp log file for information retrieval. In: International Journal of Engineering, Science and Mathematics.

[Staudemeyer2013] Staudemeyer, Joerg (2013): Android-Programmierung kurz and gut (2nd edition). [Place of publication not identified]: O'Reilly Verlag.

[Tamma2015] Tamma, Rohit; Tindall, Donnie (2015): Learning Android forensics. A hands-on guide to Android forensics, from setting up the forensic workstation to analyzing key forensic artifacts. Birmingham, England, Mumbai [India]: Packt Publishing (Community experience distilled).

[Wegner2014] Wegner, Lutz Dr. (2014): Einführung in XML, Skriptum zur gleichnamigen Vorlesung an der Universität Kassel im Wintersemester 2014/2015. Universität Kassel.

[Westermann2002] Westermann, Erik (2002): Learn XML in a weekend. Cincinnati, Ohio: Premier Press.

[Wölbert2017] Wölbert, Christian (2017): Flashen für Einsteiger. CyanogenMod und LineageOS installieren. In: c't 2017, Heft 4.

[Ye2017] Ye, Roger (2017): Android System Programming: Packt Publishing.

Internetquellen

[URL_1] Android Developers (2018): Android 5.0 APIs. Online verfügbar unter https://developer.android.com/about/versions/android-5.0#System, zuletzt aktualisiert am 31.07.2018, zuletzt geprüft am 19.08.2018.

[URL_2] Android Developers (2018): android.app.usage. Online verfügbar unter https://developer.android.com/reference/android/app/usage/package-summary, zuletzt aktualisiert am 06.06.2018, zuletzt geprüft am 19.08.2018.

[URL_3] Android Developers (2018): App resources overview. Online verfügbar unter https://developer.android.com/guide/topics/resources/providing-resources#OrientationQualifier, zuletzt aktualisiert am 17.04.2018, zuletzt geprüft am 19.08.2018.

[URL_4] Android Developers (2018): Application Fundamentals. Online verfügbar unter https://developer.android.com/guide/components/fundamentals, zuletzt aktualisiert am 30.04.2018, zuletzt geprüft am 07.08.2018.

[URL_5] Android Developers (2018): Configuration. Online verfügbar unter https://developer.android.com/reference/android/content/res/Configuration, zuletzt aktualisiert am 06.06.2018, zuletzt geprüft am 19.08.2018.

[URL_6] Android Developers (2018): Content provider basics. Online verfügbar unter https://developer.android.com/guide/topics/providers/content-provider-basics, zuletzt aktualisiert am 21.06.2018, zuletzt geprüft am 07.08.2018.

[URL_7] Android Developers (2018): Understand the Activity Lifecycle. Online verfügbar unter https://developer.android.com/guide/components/activities/activity-lifecycle, zuletzt aktualisiert am 02.07.2018, zuletzt geprüft am 07.08.2018.

[URL_8] Android Developers (2018): UsageEvents.Event. Online verfügbar unter https://developer.android.com/reference/android/app/usage/UsageEvents.Event, zuletzt aktualisiert am 06.06.2018, zuletzt geprüft am 19.08.2018.

[URL_9] Android Open Source Project (2018): Codenames, Tags, and Build Numbers. Online verfügbar unter https://source.android.com/setup/start/build-numbers, zuletzt aktualisiert am 07.08.2018, zuletzt geprüft am 23.08.2018.

[URL_10] Heise Medien GmbG & Co. KG: Die Architektur von Android. Online verfügbar unter https://www.heise.de/ct/artikel/Die-Architektur-von-Android-1901647.html, zuletzt geprüft am 20.07.2018.

[URL_11] Statista GmbH: Marktanteile von Android und iOS am Absatz von Smartphones in Deutschland von Januar 2012 bis Juni 2018. Online verfügbar unter https://de.statista.com/statistik/daten/studie/256790/umfrage/marktanteile-von-android-und-ios-am-smartphone-absatz-in-deutschland/, zuletzt geprüft am 19.07.2018.

[URL_12] TeamWin-TWRP: About. Online verfügbar unter https://twrp.me/about/, zuletzt geprüft am 14.07.2018.

[URL_13] Technology Tab: Android bootloader/fastboot mode and recovery mode explained/Android boot process. Online verfügbar unter https://tektab.com/2015/10/31/android-bootloaderfastboot-mode-and-recovery-mode-explained/, zuletzt geprüft am 07.08.2018.

[URL_14] TutorialRide: Architecture of Android (2018). Online verfügbar unter https://www.tutorialride.com/android/architecture-of-android.htm, zuletzt aktualisiert am 28.02.2018, zuletzt geprüft am 07.08.2018.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

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

Mittweida, 24.09.2018