JOHANNES KEPLER
UNIVERSITÄT LINZ | JKU

TNF

Technisch-Naturwissenschaftliche
Fakultät

Nagios für Android
Monitoring von Android-Geräten mittels Nagios

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Masterstudium

Netzwerke und Sicherheit

Eingereicht von:
Christoph Wiesner

Angefertigt am:
Institut für Netzwerke und Sicherheit (INS)

Beurteilung:
Assoz.Prof. Mag. Dipl.-Ing. Dr. Michael Sonntag

Linz, November 2014

Zusammenfassung

Monitoring Systeme wie Nagios ermöglichen die Überwachung von Komponenten (Clients, Server, Router, ...) in einer Netzwerkinfrastruktur und den Anwendungen, die auf diesen Geräten ausgeführt werden. Die Unterstützung der dafür notwendigen Protokolle fehlt allerdings meist in Betriebssystemen für mobile Endgeräte.
In der folgenden Arbeit wird das „Nagios für Android“ System beschrieben. Es ermöglicht, Einstellungen und Werte von Android Geräten mit Hilfe der Nagios Monitoring Anwendung zu überwachen. Das System besteht aus einer Android Anwendung, um die Daten auf einem Gerät zu sammeln, einer Web Anwendung um verbundene Android Geräte zu verwalten, und einer Java Anwendung für die Kommunikation mit Nagios.
Nach der Beschreibung der Problemstellung werden einige Grundlagen des Android Betriebssystems und der Nagios Monitoring Anwendung erläutert. Im folgenden Kapitel werden die einzelnen Teilaufgaben der Komponenten des Systems und die dabei stattfindenden Abläufe beschrieben. Im Anschluss erfolgt eine Beschreibung der Implementierung gefolgt von einer Bedienungsanleitung mit Informationen für die Installation, Konfiguration und Ausführung. Am Ende werden durchgeführte Tests und die Auswirkungen der Android Anwendung hinsichtlich Akku- und Speicherverbrauch beschrieben.

Abstract

Monitoring systems like Nagios enable the monitoring of components(clients, servers, routers, ...) in a network infrastructure and the applications that are executed on these devices. However often the support of the necessary protocols is missing from operating systems for mobile devices.
This thesis describes the „Nagios für Android“ system. It enables the monitoring of values and settings of Android devices with the help of the Nagios monitoring application. The system comprises of an Android application to collect the data on the device, a Web application to administrate the connected Android devices and a Java application to communicate with Nagios.
Following the description of the problem are some basics of the Android operating system and the Nagios monitoring application. The responsibilities of the system components and the involved procedures are described in the next chapter. Afterwards is a description of the component implementation followed by a manual with details for installation, configuration and execution. At the end performed tests and the impact of the Android application regarding battery and memory usage are shown.

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

Tabellenverzeichnis

Nagios für Android
Monitoring von Android-Geräten mittels Nagios

1 Aufgabenstellung

1.1 Problembeschreibung

Die Anzahl mobiler Endgeräte steigt nicht nur im privaten Bereich, sondern auch im Unternehmensumfeld durch Konzepte wie „Bring your own Device“ (BYOD) beständig an. Neben den Vorteilen für Benutzer und Unternehmen entstehen jedoch auch Problemen hinsichtlich der Sicherheit eines Unternehmensnetzwerkes und der Geheimhaltung vertraulicher Informationen. Komponenten der Netzwerkinfrastruktur und etablierte Server- und Client Betriebssysteme unterstützen üblicherweise Protokolle wie das Simple Network Monitoring Protocol (SNMP) oder die Windows Management Instrumentation (WMI). Diese Unterstützung fehlt üblicherweise im Android Betriebssystem und der für die Anwendungsentwicklung zur Verfügung stehenden SDK. Das bedeutet, um im Moment Daten über Parameter wie die Netzwerkeinstellungen, Akkuladezustand oder verfügbarer Speicherplatz zu sammeln und an ein Monitoringsystem wie Nagios zu übertragen, werden Anwendungen und Bibliotheken von Drittherstellern benötigt. Ziel der Arbeit ist es, eine Applikation für das Android Betriebssystem zu entwickeln, mit der Informationen über ausgewählte Parameter, Werte und Einstellungen eines Gerätes gesammelt werden. Diese Informationen sollen von der Applikation an einen Nagios Server, das aufgrund seiner großen Verbreitung und Erweiterbarkeit ausgewählt wurde, übertragen werden. Damit soll eine zentrale Überwachung von Android-Geräten innerhalb eines privaten Heim- oder Unternehmensnetzwerk und eventuell auch in einem öffentlichen Netzwerk, ermöglicht werden.

1.2 Wichtige Punkte

Die folgenden Punkte sollen für die Masterarbeit ausgearbeitet werden.

1. Eine Voraussetzung für das Monitoring ist die Festlegung von Parametern und Werten, die gesammelt und überwacht werden können. Ziel hier ist demnach die Erstellung einer Liste der möglichen Informationen, die gesammelt werden können, wie zum Beispiel der Standort, die Orientierung oder eine Liste der installierten Applikationen.
   Für jeden Wert muss außerdem festgestellt werden, ob er für ein Monitoring sinnvoll und relevant ist. Ein weiteres Augenmerk hier ist die Privatsphäre der Benutzer, da meist sehr viel private Information auf einem mobilen Gerät gespeichert sind und auch während der Benutzung entstehen.

2. Für Nagios existieren zahlreiche Erweiterungen wie der „Nagios Remote Plugin Executor“ (NRPE) die dessen Funktionalität erweitern. Eine Frage hier ist, ob eine der Erweiterungen oder das darunterliegende Prinzip verändert oder unverändert für das Monitoring System verwendet werden kann.

3. Neben der Entwicklung der Android Anwendung selbst, muss auch eine verlässliche und sichere Lösung für die Kommunikation zwischen dem Server und der Anwendung gefunden werden. Eine wesentliche Frage hier ist, ob es bereits Unterstützung dafür von Nagios und Android gibt oder ob man eine vorhandene Bibliothek verwendet.
   Eine weitere Frage hier ist, ob ein Update der Informationen in periodischen Abständen von der Applikation gesendet wird oder ob der Server durch eine Anfrage an die Applikation ein Update anfordert.
   Es muss auch eine Lösung für den Fall gefunden werden, dass das Gerät sich nicht in derselben Netzwerkumgebung, wie zum Beispiel ein Heim- oder Unternehmensnetzwerk, wie der Server befindet, sondern in einem entfernten Netzwerk.

4. Ein wesentlicher Punkt ist die Implementation der Android Applikation selbst. Neben Details wie den unterstützten Android Versionen, einer graphischen Oberfläche für die Konfiguration und Ausgabe von Statusinformation, wären auch Funktionen wie eine Unterstützung für

die Konfiguration der Anwendungen von einer zentralen Stelle aus interessant.

2 Grundlagen und verwendete Technologien

2.1 Android

Android [1] ist ein Open-Source Betriebssystem für mobile Endgeräte wie Smartphones und Tablet-Computer. Entwickelt und veröffentlicht wird es als Teil des Android Open Source Project [2] von der Open Handset Alliance [3].

2.1.1 Architektur

Abbildung 1 zeigt die Architektur des Android Betriebssystems nach [Maker and Chan(2009)].

2.1.1.1 Kernel

Die unterste Schicht bildet der Linux Kernel der für den Einsatz in mobilen Endgeräten modifiziert wurde. Diese Änderungen umfassen zum Beispiel Anpassungen an der Energieverwaltung oder das Android Shared Memory (Ashmem) System um Anwendungen die Verwendung von gemeinsamen Speicher zu ermöglichen und diesen zu verwalten.

[1] http://www.android.com/ (5.11.2014)
[2] http://source.android.com/ (5.11.2014)
[3] http://www.openhandsetalliance.com/ (5.11.2014)

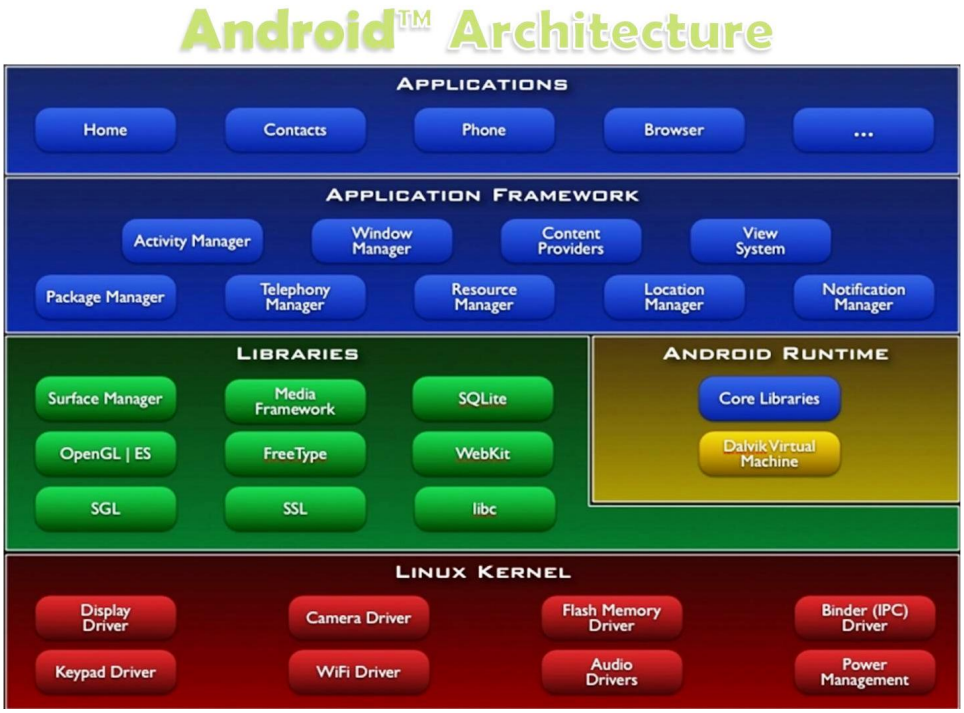

Abbildung 1: Architektur des Android Betriebssystems
[Maker and Chan(2009)]

2.1.1.2 C Bibliothek Bionic

Eine Schicht über dem Kernel befindet sich die für Android entwickelte C Bibliothek namens Bionic. Sie verwendet Teile der BSD Standard C Bibliothek zusammen mit Android eigenem Code. Es gibt mehrere Gründe, die für das Implementieren einer eigenen Bibliothek sprechen anstatt eine bestehende zu verwenden, wie z. B. die weit verbreitete GNU C Bibliothek.
Einer der Gründe ist die Lizenz unter der die Bibliotheken veröffentlicht sind. Die für Android bevorzugte Lizenz ist die Apache License, Version 2.0 obwohl Teile wie zum Beispiel Kernel Patches unter der GPLv2 veröffentlicht werden. Gründe für die Entscheidung gegen die GPL oder LGPL werden unter [Project(2013e)] näher erläutert.
Ein weitere Grund ist der Ressourcenbedarf, da auf mobilen Endgeräten meist sehr viel weniger Haupt- und Sekundärspeicher zur Verfügung steht und daher Bibliotheken, die für den Einsatz auf Desktop und Server Systemen konzipiert wurden, mehr für Laufzeitperformanz und weniger für ge-

ringen Speicherverbrauch optimiert sind.

2.1.1.3 Virtuelle Maschine (VM)

Das Android Betriebssystem verwendet nicht die Java 2 Micro Edition (J2ME)[4] sondern verfügt über eine eigene Virtuelle Maschine namens Dalvik. Die VM wurde, wie die C Bibliothek, mit einem geringen Speicherverbrauch als Ziel entwickelt. Mit Android Version 4.4 wurde eine neue experimentelle Virtuelle Maschine namens ART als Weiterentwicklung von Dalvik vorgestellt.
Die Android VM verwendet einen eigenen Bytecode und speichert diesen in einer einzigen .dex Datei im Gegensatz zu den class Dateien in Java. Für Anwendungsentwickler ergeben sich allerdings keinerlei Unterschiede. Sie schreiben herkömmlichen Java Code der in class Dateien kompiliert wird. Erst im Anschluss übersetzt das dx Programm den Java Bytecode zu einer .dex Datei wie in Abbildung 2 gezeigt wird.

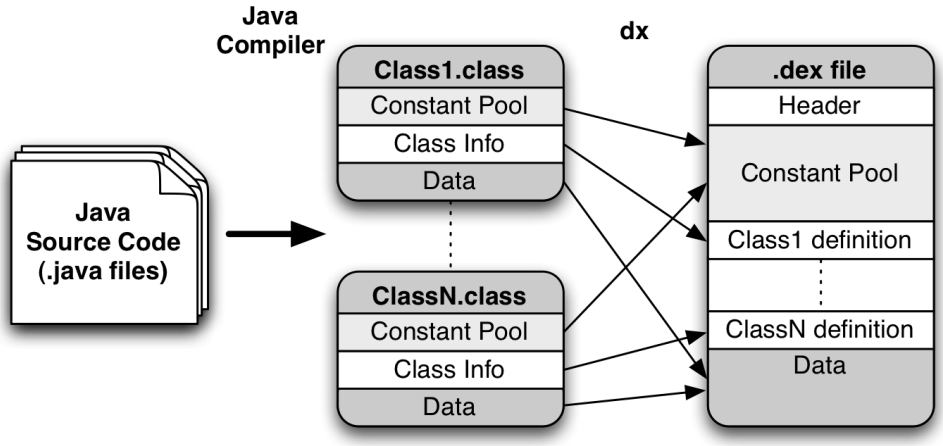

Abbildung 2: Kompilierung von Java Code in .dex Datei
[Enck et al.(2011)Enck, Octeau, McDaniel, and Chaudhuri]

2.1.1.4 Anwendungsbibliotheken

Die Java Bibliothek basiert auf dem Apache Harmony [5] Projekt. Sie wurde allerdings um zahlreiche Pakete erweitert, wie etwa für die Interaktion

[4] http://www.oracle.com/technetwork/java/embedded/javame/index.html (5.11.2014)
[5] http://harmony.apache.org/ (5.11.2014)

mit den Gerätesensoren oder für die Verwendung der SQLite Datenbank. Ursprünglich waren Entwickler auf die Verwendung der Java Version 6 für die Entwicklung ihrer Android Anwendung beschränkt. Ab Version 22.6.0 der Android Development Tools (ADT) und mindestens Android Version 2.2 als Zielumgebung für die Anwendung kann nun auch Java 7 verwendet werden [Project(2014a)]. Android bietet zusätzlich die Möglichkeit, Teile einer Anwendung in C/C++ zu entwickeln [Project(2014c)]. Dadurch können zum Beispiel bestehende Bibliotheken wiederverwendet werden, jedoch wird die Verwendung des Native Development Kit (NDK) nur empfohlen, wenn es absolut nötig für die Entwicklung der Anwendung ist. Für die Verwendung der NDK Bibliotheken ist, abhängig von der CPU Architektur, eine minimale Android Version erforderlich, die unter dem Punkt „Android platform compatibility“ auf [Project(2014c)] beschrieben werden.

2.1.2 Relevante Features

Im folgenden Abschnitt werden einige grundlegende Eigenschaften der Android Plattform erläutert, die vor allem für Anwendungsentwickler relevant sind. Weitere Informationen sind unter [Project(2013d)] zu finden.

2.1.2.1 Signierung

Jede Anwendung muss von ihrem Entwickler mit einem Zertifikat signiert werden. Dies dient vor allem dazu, den Entwickler identifizieren zu können. Hier kann auch ein selbst signiertes Zertifikat verwendet werden. Wird eine neuere Version nicht mit demselben Zertifikat signiert wird sie als komplett neue Anwendung und nicht als Aktualisierung betrachtet. Der Entwickler ist auch für die sichere Aufbewahrung seines Zertifikats verantwortlich. Ein gestohlenes Zertifikat kann zum Beispiel dazu benutzt werden, um Schadcode als Update einer bestehenden oder neue Anwendung eines Entwicklers auszuliefern. Weiter Informationen über das Signieren von Anwendungen und dem sicheren Erstellen und Aufbewahren von Zertifikaten findet sich unter [Project(2013i)].

2.1.2.2 Getrennte Anwendungen

Da Android auf dem Linux Kernel basiert, verwendet es für Dateien und Anwendungen das UNIX Rechtesystem mit Rechten für den Besitzer, die

Gruppe und Dritte. Jeder Anwendung wird nun bei ihrer Installation eine separate Benutzer-ID zugewiesen. Auch das System besitzt eine eigene ID. Dies stellt sicher, dass eine Anwendung keinen Zugriff auf die Dateien des Systems oder einer anderen Anwendung erhält. Jede Anwendung wird in einem eigenen Prozess ausgeführt, der wiederum seine eigene VM besitzt.
Eine Ausnahme dieser Trennung ist die Zuweisung einer gemeinsamen Benutzer-ID (android:sharedUserId) für mehrere Anwendungen in deren Manifest Datei [Project(2013f)]. Voraussetzung dafür ist allerdings, dass die Anwendungen mit demselben Zertifikat signiert wurden.

2.1.2.3 Anwendungsrechte

Anwendungen können nicht direkt auf die Hardware, System- oder Benutzerdaten zugreifen. Das ist nur über die API und spezielle Rechte möglich. Die für die Ausführung benötigten Rechte müssen in der Manifest Datei angegeben werden. Während der Installation einer Anwendung wird diese Liste angezeigt und der Benutzer gefragt, ob er zustimmt oder ablehnt. Eine individuelle Konfiguration einzelner Berechtigungen ist hier nicht möglich. Für eine erfolgreiche Installation müssen alle Rechte als gesamtes akzeptiert werden. Bei einer Ablehnung wird die Installation abgebrochen.

2.1.2.4 Komponenten

Es gibt einige wichtige Komponenten die für die Erstellung einer Android Anwendung verwendet werden können [Project(2013d)]. Das sind

- Activity
- Service
- Broadcast Receivers
- Content Providers

2.1.2.4.1 Activity

Das graphische Benutzerinterface einer Anwendung besteht aus einer oder mehreren Seiten. Während das statische Layout durch XML Dateien definiert wird, steuern die Activity Klassen das Verhalten einer Seite z. B. beim Drücken einer Schaltfläche. Die Activity Klassen arbeiten unabhängig

voneinander und können auch von anderen Anwendungen verwendet werden falls dies erlaubt ist. Eine E-Mail-Anwendung könnte zum Beispiel ihre Activity für das Senden von E-Mails für andere Anwendungen verfügbar machen.

2.1.2.4.2 Service

Ein Service wird durch eine der Komponenten gestartet, läuft im Hintergrund, und besitzt anders als eine Activity kein Benutzerinterface. Komponenten können sich an einen bereits gestarteten Service binden, um mit diesem zu interagieren. Am besten geeignet ist ein Service für längere Operationen wie zum Beispiel dem Download einer Datei oder dem Abspielen von Hintergrundmusik, auch wenn der Benutzer eine andere Anwendung geöffnet hat. Zu beachten ist hier, dass der Service an sich kein eigener Thread ist, sondern im Hauptthread der Anwendung ausgeführt wird.

2.1.2.4.3 Broadcast Receivers

In Android werden Anwendungen über Ereignisse wie zum Beispiel das Ein- oder Ausschalten des Gerätes oder das Ändern des Datums durch Systemweite Broadcasts informiert. Eine Anwendung kann einen oder mehrere Broadcast Receiver registrieren, die beim Auftreten des entsprechenden Ereignisses gestartet werden, um eine Aktion ausführen. Der Broadcast Receiver selbst ist nicht geeignet, um längere Operationen durchzuführen. Er ist nur eine Art Auslöser der zum Beispiel einen Service startet, der die eigentliche Arbeit übernimmt.

2.1.2.4.4 Content Providers

Eine Anwendung, die zum Beispiel eine Liste von Kontakten verwaltet, kann diese für andere Anwendungen zur Verfügung stellen. Der Zugriff und die Abfrage der Daten erfolgt über einen Content Provider. Er stellt somit ein einheitliches Interface für andere Anwendungen zur Verfügung, unabhängig davon ob die Daten als einfache Textdatei oder in einer SQL-Datenbank gespeichert sind. Falls nötig, kann den abfragenden Anwendungen auch die Modifikation der Daten erlaubt werden.

2.1.3 Mögliche Android Informationsquellen

Die folgende Liste an Werten wurde aus den möglichen Berechtigungen die eine Android Anwendung erlangen kann [Project(2013g)] und den Informationen, die auf einem laufenden Android Gerät und für Anwendungsentwickler unter [6] verfügbar sind, zusammengestellt. Sie stellt Werte dar, die für den Zweck des Gerätemonitorings ausgewählt werden könnten.

2.1.3.1 Netzwerkverbindungen

2.1.3.1.1 Wireless LAN

Wi-Fi ist eine der primären Möglichkeiten um mobile Geräte mit einem bestehenden Computernetzwerk zu verbinden. Anders als Kabelnetzwerke handelt es sich bei einer Wi-Fi Verbindung im ein Broadcastnetzwerk, wodurch jeder im Sendebereich die Kommunikation abhören kann. Dadurch stellen unverschlüsselte oder durch schwache Passwörter oder Algorithmen schlecht gesicherte Netzwerke ein Risiko dar. Neben den grundlegenden Verbindungsparametern wie IP-Adresse, Subnetzmaske oder BSSID ist vor allem die verwendete Verschlüsselung der Verbindung interessant um festzustellen, ob sensitive Information ungeschützt übertragen wird.

2.1.3.1.2 Bluetooth

Bluetooth ist eine einfache Möglichkeit, Verbindungen zwischen Geräten, wie zum Beispiel Kopfhörern, Freisprecheinrichtungen oder Laptops, über Entfernungen bis zu einhundert Metern herzustellen. Neben dem Verbindungsstatus des Bluetooth Adapters selbst sind hier vor allem die Anzahl und der Typ der verbundenen Geräte von Interesse.

2.1.3.1.3 Mobile Datenverbindung

Die mobile Datenverbindung ermöglicht es von beinahe überall innerhalb des Funknetzwerks des Netzanbieters Zugriff auf das Internet zu erlangen. Je nach Tarif können dafür hohe Kosten anfallen, falls ein vorgegebenes Limit überschritten oder unbemerkt Daten übertragen werden. Neben der Überwachung des Verbindungsstatus der Datenverbindung sind hier noch

[6] http://developer.android.com/index.html

statistische Werte wie die Menge an gesendeten oder empfangenen Datenvolumen interessant.

2.1.3.1.4 Virtual Private Network (VPN)

Ein VPN ermöglicht es Hosts in verschiedenen Netzwerken und über das Internet zu kommunizieren, als ob sie sich im selben lokalen Netzwerk befinden. Android ermöglicht es, eine VPN Verbindung für das Gerät einzurichten. In Versionen vor Android 4.0 wurde Anwendungen, die den entsprechenden Broadcastreceiver registriert haben, eine Änderung des VPN Status durch einen Broadcast mitgeteilt. Mit Version 4.0 wurde dieser Broadcast entfernt. Eine Anwendung kann für sich selbst eine VPN Verbindung erstellen, über die sie kommuniziert. Allerdings ist immer nur eine einzige VPN Verbindung zur selben Zeit erlaubt [Project(2013j)]. Erstellt eine Anwendung eine Verbindung, wird eine bereits bestehende beendet. Android benachrichtigt in diesem Fall die Anwendung, deren Verbindung beendet wurde. Eine Möglichkeit Verbindungen zu beobachten wäre, in der Android Monitoring Anwendung selbst eine VPN Verbindung zu erstellen und auf den Fall zu warten, dass sie durch eine andere Anwendung beendet wird. Da die Benachrichtigung selbst keine weiteren Informationen erhält, ist der Zeitpunkt an dem sie auftritt inzwischen die einzige Information die gewonnen werden kann.

2.1.3.1.5 Near Field Communication(NFC)

NFC ist eine Technik für die Kommunikation mit anderen NFC fähigen Geräten oder NFC Markierungen über eine Entfernung von wenigen Zentimetern. Anwendungsgebiete dafür sind etwa bargeldlose Bezahlung oder das Starten einer Anwendung oder Webseite, wenn ein NFC-Tag erfasst wird. Eine aktivierte NFC Schnittstelle ist eine mögliche Schwachstelle somit wäre der Status der Schnittstelle und möglicherweise auch der Inhalt oder der Typ (Bezahlung, Tag, ...) empfangener NFC Nachrichten eine relevante Information.

2.1.3.2 Gerät

2.1.3.2.1 Anrufeinstellungen

Die Anrufeinstellungen enthalten Informationen wie Rufnummernbegrenzung, Anrufumleitung, Anrufsperre oder Internettelefonie. Relevant können diese Informationen für den Fall sein, dass Probleme mit dem Tätigen oder Empfangen eines Anrufes bestehen oder es eine Liste mit erlaubten Rufnummern gibt, die als einzige angerufen werden dürfen.

2.1.3.2.2 Audioeinstellungen

Einstellungen wie der verwendete Klingelton oder die Lautstärke stellen weniger relevante Information für das Monitoring dar. Für den Fall, dass der Besitzer nicht auf Anrufe antwortet, wäre interessant, ob das Gerät auf Vibration gestellt ist oder nicht.

2.1.3.2.3 Bildschirmeinstellungen

Hier sind Einstellungen wie das Hintergrundbild, das Design oder die Bildschirmhelligkeit enthalten. Sie bieten keine relevanten Informationen für eine Überwachung.

2.1.3.2.4 Speicher

Android Geräte verfügen üblicherweise über drei verschiedene Arten von Speicher.

- Interner Gerätespeicher
- Externer Speicher/SD-Karte
- Hauptspeicher

Grundlegende Informationen hier wären der verwendete und noch zur Verfügung stehende Speicher, falls ein Zugriff auf die Werte von der API zugelassen wird. Relevanter zum Beispiel könnte das Durchsuchen der SD-Karte nach Windows PE oder Unix ELF Dateien, die Malware enthalten könnten und entsprechend kontrolliert werden müssen, bevor das Gerät sich etwa mit dem Netzwerk des Unternehmens verbindet.

2.1.3.2.5 Batterie

Information in dieser Kategorie sind neben dem Ladezustand des Akkus ob das Gerät gerade geladen wird und auch welche Art von Ladegerät (AC, USB) verwendet wird. Ein Grund um diese Informationen zu sammeln ist einerseits, Geräte zu finden die aufgeladen werden müssen, und andererseits für den Fall, dass ein Gerät keine Informationen mehr sendet, festzustellen ob ein Gerät keinen Strom mehr hat indem man den Ladezustand beim letzten erfolgreichen Update betrachtet.

2.1.3.2.6 Installierte und laufende Anwendungen

Die von Google betriebene Online Plattform Google Play [7], enthält mehr als eine Million Android Anwendungen jeder Art. Obwohl übermittelte Anwendungen überprüft werden, findet sich Spy- oder Malware unter den verfügbaren Anwendungen. Die auf dem Gerät installierten und auch laufenden Anwendungen können mit einer White- und/oder Blacklist verglichen werden. So können neue und unbekannte oder unerwünschte Anwendungen gefunden und gemeldet werden.

2.1.3.2.7 Frame Buffer

Eine Anwendung kann einen Screenshot seiner eigenen graphischen Oberfläche erstellen und in eine Datei speichern. Ein Screenshot einer anderen Anwendung oder von Android selbst benötigt allerdings Root Berechtigungen. Außer bei Problemen mit der Monitoring Anwendung, bei denen ein Screenshot erstellt und zum Beispiel per E-Mail an den Administrator gesendet werden könnte, scheint es deshalb keinen Nutzen dafür zu geben.

2.1.3.2.8 State Dump

Mithilfe der Debug Werkzeuge, die in den Entwicklerwerkzeugen enthalten sind, ist es möglich Statusinformation von Systemdiensten zu erhalten. Normale Anwendungen benötigen allerdings Root Berechtigungen um diese Informationen zu erhalten, daher sind diese hier nicht relevant.

[7] https://play.google.com/store (5.11.2014)

2.1.3.3 Persönliche Informationen

2.1.3.3.1 Kontakte

Anwendungen können Zugriff auf die Liste der Kontakte und Kontaktdaten wie Namen oder Telefonnummer, die auf dem Gerät gespeichert sind, erhalten. Ein Grund diese Liste zu kontrollieren wäre festzustellen, ob alle wichtigen Firmenkontakte im Gerät gespeichert sind.

2.1.3.3.2 Sync

Anwendungen können die von Android zur Verfügung gestellte Sync Komponente verwenden, um Daten mit einem Server zu synchronisieren. Der Zugriff auf einfache Informationen wie den Namen des Sync-Accounts sind möglich, sensible Informationen wie Passwörter sind nicht zugänglich.

2.1.3.3.3 Benutzerprofile

Das Benutzerprofil repräsentiert den Besitzer des Gerätes. Ältere Android Versionen verwenden nur ein einziges Profil. Mit Android 4.2 wurde die Unterstützung für mehrere Profile auf Tablets eingeführt.

2.1.3.3.4 Accounts

Android stellt einen zentralen Account Manager zur Verfügung der es Anwendungen ermöglicht, Login Informationen wie Benutzername und Passwort abzulegen. Eine Anwendung kann zwar die Liste der Accounts abfragen, jedoch nicht sensible Informationen wie Passwörter. Diese Informationen sind nur für die Anwendung verfügbar die den Account erstellt hat.

2.1.3.3.5 Browserchronik und Lesezeichen

Die Chronik der im Browser besuchten Internetseiten und die erstellten Lesezeichen sind vielleicht nützlich um festzustellen ob ein Benutzer eine bestimmte Seite besucht hat, jedoch nicht relevant für das normale Monitoring.

2.1.3.3.6 Feeds

Anwendungen können Zugriff auf Feeds, für die sich der Benutzer angemeldet hat, erlangen und diese Daten verarbeiten. Für das Monitoring sind diese Daten allerdings nicht relevant.

2.1.3.3.7 Social Stream

Daten von sozialen Netzwerken wie Statusupdates oder Ereignisse werden in dem sogenannten Social Stream eingetragen. Dieser kann von Anwendungen verarbeitet werden um zum Beispiel eine Benachrichtigung anzuzeigen. Für das Monitoring allerdings ist der Social Stream nicht relevant.

2.1.3.3.8 Anrufliste

Geführte Telefonate werden in der Anrufliste eingetragen und Anwendungen haben Zugriff auf Informationen wie der Nummer, Zeit, Datum sowie Anrufdauer. Die Anrufliste könnte archiviert werden, damit geführte Telefonate nicht bestritten werden können.

2.1.3.3.9 SMS

Anwendungen können SMS Nachrichten schreiben und senden, allerdings können sie auch gespeicherte und eingehende Nachrichten lesen. Auf Firmeneigenen Geräten könnten so die SMS Nachrichten archiviert werden.

2.1.3.3.10 MMS

Android erlaubt es Anwendungen, eingehende MMS Nachrichten zu verarbeiten. So könnten zumindest die empfangenen Nachrichten archiviert werden.

2.1.3.3.11 Benutzerwörterbuch

Android verfügt über ein Benutzerwörterbuch, in dem Benutzer und Anwendungen Einträge erstellen können. Diese Einträge stellen allerdings keine relevanten Informationen für ein Monitoring dar.

2.1.3.3.12 Kalender

Die Kalender Anwendung ist üblicherweise auf allen Android Geräten installiert. Anwendungen können Einträge im Kalender erstellen und zum Beispiel mit einem Server synchronisieren. Die Daten könnten verwendet werden um festzustellen, ob wichtige Ereignisse, wie etwa ein wichtiges Meeting, in den Kalender eingetragen wurde.

2.1.3.4 System

2.1.3.4.1 Dock

Android Geräte können mit verschiedenen Docking Stationen verbunden werden. Anwendungen werden mittels eines Broadcasts über Änderungen des Docking Status informiert. Zusätzlich zum Status kann auch die Art (Auto, Desktop) der Docking Station ermittelt werden [Project(2013c)]. Ein Anwendungsbeispiel wäre eine Anwendung, die Verkehrsmeldungen von einem Server abruft und die Aktualisierungsrate erhöht, wenn das Gerät mit der Docking Station eines Autos verbunden wird.

2.1.3.4.2 Zeit und Datum

Die Einstellungen für Uhrzeit, Datum und Zeitzone können per Hand oder automatisch über das Netzwerk erfolgen. Ein Grund um diese Werte in das Monitoring aufzunehmen wäre um zu kontrollieren, ob sie in allen Geräten korrekt eingestellt sind.

2.1.3.4.3 Eingabehilfen

Die Eingabehilfe enthält Optionen wie den TalkBack Service, der gesprochenes Feedback auf Aktionen wie dem Starten einer Anwendung liefert oder das Vorlesen von Passwörtern. Diese Einstellungen sollen die Barrierefreiheit von Android fördern und sind wenig relevant für das Monitoring.

2.1.3.4.4 Entwicklereinstellungen

Entwickler können ihre Anwendungen auf einem Emulator oder echten Android Geräten testen. Die Entwicklereinstellungen sollen diesen Vorgang erleichtern. Die meisten der Optionen, wie die Anzeige der Koordinaten an denen der Bildschirm berührt wird oder die CPU-Auslastung, sind für ein Monitoring wenig relevant. Die Entwicklereinstellungen enthalten allerdings auch Optionen wie USB-Debugging, die ein Sicherheitsrisiko darstellen können. Vor Android 4.2.2 konnte man mithilfe der Android Debug Bridge (ADB) [Project(2013b)] und aktivierten USB-Debugging Zugriff auf ein Gerät erlangen ohne den Bildschirm entsperren zu müssen. Mit Android 4.2.2 wurde das Secure USB-Debugging eingeführt [Project(2013h)], welches eine Autorisierung erfordert, wenn sich ein unbekannter Host per ADB mit

dem Gerät verbinden möchte. Der Status des USB-Debuggings ist somit für Geräte mit älteren Versionen eine relevante Information für das Monitoring.

2.1.3.4.5 Geräteinformation und Telefonstatus

Die Geräteinformationen beinhalten Daten wie die Kernel-Version, Android-Version oder die Modellnummer. Der Telefonstatus enthält Informationen wie die IMEI, W-LAN Mac Adresse, Seriennummer oder die Stärke des Mobilfunksignals. Mithilfe der Android-Version kann beispielsweise festgestellt werden, ob verfügbare Aktualisierungen auf allen Geräten installiert wurden. IMEI und Seriennummer können verwendet werden, um das Gerät zu identifizieren.

2.1.3.4.6 Standortdienste

Diese Einstellungen legen fest welche Technik für die Ermittlung des Standortes verwendet wird. Üblicherweise stehen zwei Dienste mit variierender Genauigkeit zur Verfügung. Die Google Standortdienste verwenden W-LAN und Mobilfunknetze um einen ungefähren Standort zu ermitteln. Durch die Verwendung von GPS lässt sich die Genauigkeit auf einige Meter Abweichung verbessern. Falls im Zuge des Monitoring die Standortinformation gesammelt wird, lässt sich mit dem verwendeten Dienst und dem Genauigkeitsgrad abschätzen, wie gut die ermittelte Standortinformation ist.

2.1.3.4.7 Sicherheit

Die Sicherheitseinstellungen beinhalten Optionen wie die Bildschirmsperre, den Zertifikatsspeicher für Benutzerzertifikate oder ob Passwörter bei der Eingabe sichtbar sind. Relevante Information hier ist zum Beispiel ob die Bildschirmsperre und das automatische Sperren aktiviert sind um zu verhindern, dass Unbefugte einfachen Zugriff auf das Gerät erhalten.

2.1.3.4.8 Logdateien

Eine Anwendung kann auf die Log-Dateien die von ihr geschrieben werden zugreifen, wegen des Sandbox Konzeptes allerdings nicht auf Dateien anderer Anwendungen oder System Log-Dateien. Dafür wären Root Privilegien nötig. Ein möglicher Anwendungszweck für gesammelte Log-Dateien der Monitoring Anwendung wäre sie im Falle eines Fehlers zu übermitteln.

2.1.3.4.9 Ein- und Ausschaltereignis

Zu den zahlreichen Broadcasts die von Android gesendet werden gehören auch Benachrichtigungen, wenn das System fertig hochgefahren ist oder ausgeschaltet wird. Diese können zum Beispiel verwendet werden um einen Service nach dem Hochfahren des Gerätes zu starten. Ansonsten sind die Zeitpunkte des Ein- oder Ausschaltens nicht interessant für das Monitoring.

2.1.3.4.10 Rootstatus

Anwendungen laufen in Android üblicherweise mit wenigen Privilegien und Zugriff auf das System und andere Anwendungen ist nur im Rahmen der verfügbaren Berechtigungen möglich. Das System und jede Anwendung werden mit einer anderen Unix UID ausgeführt wodurch Daten und sogar die Prozesse selbst voneinander getrennt sind. Nach dem Rooten eines Gerätes werden Anwendungen mit erhöhten Berechtigungen ausgeführt und können zum Beispiel auf Systemdateien oder Einstellungen zugreifen.

2.1.3.5 Sensoren

2.1.3.5.1 Standort/Orientierung

- Koordinaten
  Einer Android Anwendung stehen grundsätzlich zwei Wege zur Verfügung um die aktuelle Position des Gerätes zu ermitteln. Die erste Methode ist eine grobe Positionsbestimmung mithilfe des Google eigenen Dienstes durch die Verwendung von Wi-Fi und Funkturmsignalen. Die genauere Methode verwendet GPS für die Ermittlung des Standortes. Vorteil der groben Positionsbestimmung ist die Verfügbarkeit innerhalb von Gebäuden solange Signale empfangen werden. Es kann auch einige Zeit dauern bis die Position mittels GPS bestimmt wird und die Daten verfügbar sind.

- Geomagnetisches Feld
  Dieser Sensor misst Änderungen im Erdmagnetfeld. Daraus lassen sich Schlüsse auf die Orientierung ziehen.

- Nähe
  Misst die Entfernung eines Objektes zur Front des Gerät in Zentimetern oder als Binärer (Nah, Fern) Wert. Wird zum Beispiel verwendet

um während eines Telefonats die Entfernung des Kopfes zu messen und Aktionen durch zufällige Berührungen des Bildschirms zu verhindern.

Die Orientierung des Gerätes ist nicht relevant für das Monitoring. Im Gegensatz dazu kann der genaue Standort eines Gerätes zum Beispiel verwendet werden um eine Person im Falle eines Notfalles zu finden. Auch eine einfache Feststellung wie „das Gerät befindet sich im Gebäude“ oder „das Gerät befindet sich nicht im Gebäude“ könnte getroffen werden.

2.1.3.5.2 Umgebung

Android unterstützt die folgenden Sensoren um Informationen über die Umgebung in der sich das Gerät befindet zu sammeln.

- Temperatur
- Helligkeit
- Luftdruck
- Feuchtigkeit

Das Sammeln dieser Umgebungsinformation kann nützlich sein falls das Android Gerät unter raueren Bedingungen, wie zum Beispiel in Industrieumgebungen oder im Außenbereich eingesetzt wird.

2.1.3.5.3 Bewegung

Neben den Umgebungssensoren werden auch die folgenden Bewegungssensoren unterstützt.

- Beschleunigung
  Misst die Beschleunigungskraft entlang der drei Achsen inklusive der Schwerkraft.
- Schwerkraft
  Misst die Gravitationskraft entlang der drei Achsen.
- Gyroskop
  Misst Rotationsgeschwindigkeit entlang der drei Achsen.

- Lineare Beschleunigung
  Misst die Beschleunigungskraft entlang der drei Achsen ohne Schwerkraft.

Über einen längeren Zeitraum gesammelt können diese Daten Aufschluss darüber geben, ob und wie das Gerät gerade bewegt wird. Alleine für sich sind die Daten wenig relevant für das Monitoring.

2.2 Nagios

Nagios[8] ist eine Monitoring Software für IT-Infrastrukturen die für Linux entwickelt wird. Sie ist in einer „Core“ und einer „XI“ Version verfügbar. Core ist die frei verfügbare Open-Source Version, während es sich bei XI um eine erweiterte kommerzielle Version handelt.

2.2.1 Features

Nagios unterstützt gängige Netzwerk Management Protokolle wie SNMP und WMI um Statusinformationen von Hosts im Netzwerk abzufragen oder per SNMP Traps zu erhalten. Zusätzlich gibt es zahlreiche Erweiterungen, um die Funktionalität zu erweitern.

2.2.2 Aktive Checks

Eine Möglichkeit der Erweiterung sind die aktiven Checks durch Nagios Plug-ins. Ein aktiver Check wird durch Nagios ausgelöst indem es ein Plug-in startet, um einen Wert, wie zum Beispiel den verfügbaren Festplattenspeicher, auf einem Host abzufragen und das Ergebnis an Nagios zurückzugeben. Ein Problem hier sind allerdings Statusinformationen, die nicht über das Netzwerk von einem anderen Host abgefragt werden können. Eine Lösung hierfür sind Erweiterungen wie der Nagios Remote Plug-in Executor (NRPE) [Enterprises(2013b)]. NRPE ermöglicht es, Plug-ins nicht nur lokal auf dem Nagios Server Host, sondern auch auf anderen Hosts im Netzwerk auszuführen. Abbildung 3 zeigt den Aufbau einer NRPE Architektur.

[8] http://www.nagios.org/ (5.11.2014)

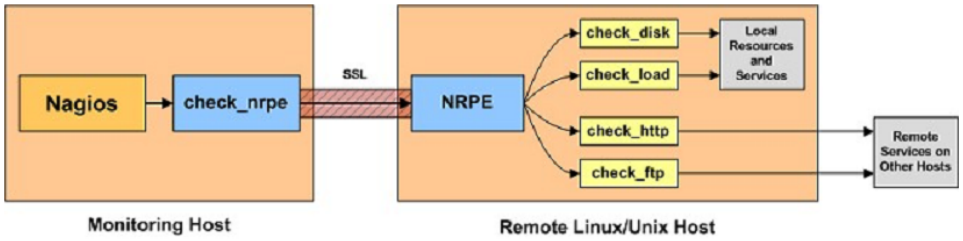

Abbildung 3: NRPE Architektur
[Enterprises(2013b)]

2.2.3 Passive Checks

Es ist nicht immer möglich einen aktiven Check durchzuführen, weil zum Beispiel ein Host oder Service nicht direkt abgefragt werden kann. Deswegen bietet Nagios zusätzlich die Möglichkeit von passiven Checks. Die passiven Check Resultate, die den Status eines Hosts oder Service darstellen, müssen dazu in ein vorgegebenes Format gebracht und an Nagios übermittelt werden. Zu diesem Zweck erstellt Nagios eine Named Pipe in Form einer Datei, der „Externen Kommando“ Datei. Nagios Kommandos und passive Check Resultate werden in diese Datei geschrieben, in einem FIFO Buffer gespeichert und in regelmäßigen Abständen durch Nagios verarbeitet. Der Nagios Remote Data Processor (NRDP) [Enterprises(2013a)] bietet ein Web-Interface, mit dem die Daten auch von anderen Hosts im Netzwerk an Nagios übertragen werden können. Anwendungsfälle sind unter anderem asynchrone Ereignisse und falls ein Host nicht vom Nagios Server Host aus erreichbar ist oder durch einen aktiven Check geprüft werden kann. Abbildung 4 zeigt die NRDP Architektur.

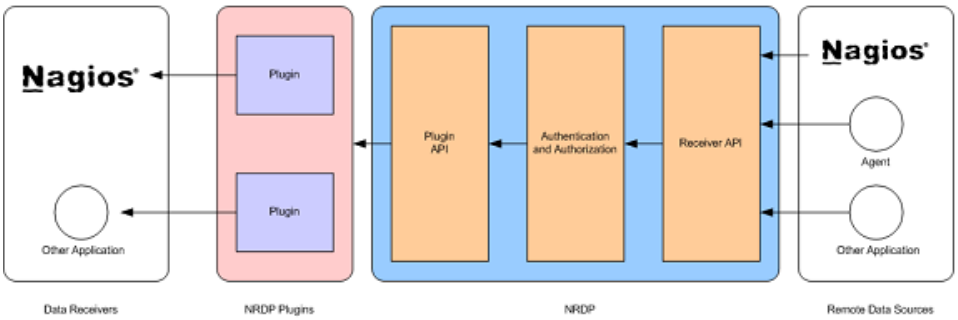

Abbildung 4: NRDP Architektur
[Enterprises(2013a)]

3 Lösungsansatz

3.1 Beschreibung

Für das Monitoring der Android Geräte werden passive Checks verwendet. Die Verwendung von aktiven Checks erfordert die Erreichbarkeit der Android Geräte über das Netzwerk um eine Abfrage durchzuführen. Dies ist bei mobilen Endgeräten nicht immer oder nur durch zusätzlichen Aufwand möglich. Passive Checks erlauben es hingegen, Abfrageergebnisse an eine zentrale Stelle, deren IP-Adresse oder Domäne bekannt ist, zu senden. Dafür muss in Kauf genommen werden, dass eine Abfrage der Daten nur auf den Android Geräten gestartet werden kann.
Um die Aufgabenstellung erfüllen zu können muss das System verschiedene Teilaufgaben erfüllen. Für die Erfüllung dieser Aufgaben ist es erforderlich das System in verschiedene Komponenten aufzuteilen, da sie nicht von einer einzelnen Anwendung alleine erfüllt werden können.

1. Die verschiedenen Gerätedaten werden von den Android Anwendungen auf den Geräten gesammelt und an eine Web Anwendung gesendet.

2. Sind alle erforderlichen Informationen vorhanden, werden sie, mit Hilfe einer Java Anwendung, an den Nagios Server übertragen.

3. Auf diesem Nagios Host werden die Daten von dieser Java Anwendung in die „Externe Kommando“ Datei geschrieben, damit sie von Nagios gelesen und verarbeitet werden kann.

Die Web Anwendung soll neben ihrer Hauptaufgabe zusätzlich noch die folgenden Aufgaben erfüllen.

1. Die Android Anwendungen benötigen einige Parameter um sie zu konfigurieren. Anstatt jede Anwendung manuell auf den Geräten zu konfigurieren, soll die Konfiguration von einer zentralen Stelle aus verteilt werden. Eine neu installierte Android Anwendung benötigt so nur die URL der Web Anwendung um sich vollständig konfigurieren zu können.

2. Die Web Anwendung stellt ein einfaches Verwaltungsinterface zur Verfügung um Gerätelisten zu verwalten. Diese Benutzerverwaltung ermöglicht es neue Geräte zu registrieren und sie einer Black- oder Whi-

telist hinzuzufügen. Damit wird unter anderem festgelegt, ob eine bestimmte Android Anwendung Zugriff auf das System hat oder nicht.

3.2 Android Anwendung

Die Android Anwendung soll neben Installation und einer ersten Konfiguration nur wenig Interaktion durch den Benutzer erfordern. Das Abfragen und Senden der Daten soll nach Möglichkeit automatisch im Hintergrund stattfinden.

3.2.1 Registrierung

Android Geräte müssen im Monitoring System registriert sein um ihre Daten senden und eine Konfiguration herunterladen zu können. Abbildung 5 zeigt die Aktionen, die während einer Registrierung durch die Android Anwendung, die Web Anwendung und den Administrator durchgeführt werden.

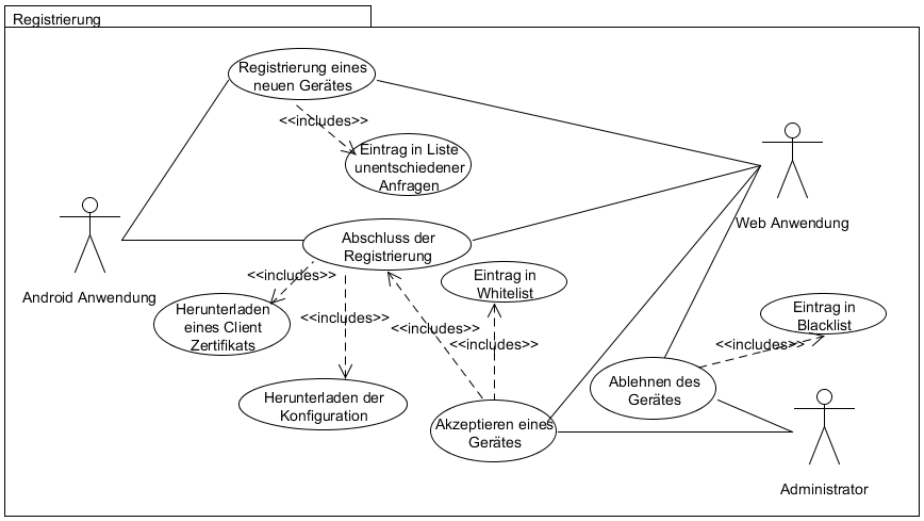

Abbildung 5: Registrierung eines Gerätes

Die folgenden Schritte sind die von der Android Anwendung durchgeführten Aktionen.

1. Die Registrierung wird durch den Benutzer gestartet, indem er die entsprechende Schaltfläche auf der Startseite der Anwendung betätigt. Ist

keine gültige URL für eine Web Anwendung in den Anwendungseinstellungen angegeben wird eine entsprechende Warnung angezeigt.

2. Um ein Gerät zu identifizieren werden eindeutige Informationen benötigt, die als Parameter in der Anfrage enthalten sind. Idealerweise sollten diese Parameter für jedes Gerät unterschiedlich und schwer vorhersagbar sein. Damit die Parameter nicht unverschlüsselt übertragen werden, ist die Verbindung durch HTTPS mit einem Server Zertifikat gesichert. Die Verwendung und Kombination mehrerer Werte soll die Chance, dass zwei Geräte fälschlicherweise als identisch angesehen werden, vermindern.
   Aus den verschiedenen Werten, die auf einem Android Gerät abgefragt werden können, sind die folgenden Werte als Parameter für eine Anfrage ausgewählt worden.

   - ID
     Diese ID wird beim ersten Start der Anwendung zufällig generiert und in den privaten Anwendungsdaten gespeichert. Werden die Daten gelöscht oder die Anwendung entfernt und neu installiert, ist die ID verloren und das Gerät muss erneut registriert werden.
   - IMEI und AndroidID
     Die International Mobile Station Equipment Identity ist eine 15-stellige Seriennummer und soll laut Standard eine eindeutige Identifikation eines Endgerätes ermöglichen[Association(2013)]. Obwohl die IMEI einzigartig sein soll, wird sie mit der ID des Android Betriebssystems kombiniert, um die Chance auf Kollisionen zu verringern.
   - Modellbezeichnung
     Die Modellbezeichnung ist ein Parameter, der aus verschiedenen Einzelwerten zusammengesetzt ist. Enthalten sind die Hersteller- und Modellbezeichnung sowie die installierte Betriebssystemversion.
   - Gerätealias
     Die Android Anwendung erlaubt es in ihren Einstellungen einen Namen für das Gerät zu definieren. Damit soll es einem Administrator ermöglicht werden das Gerät anhand dieses Namens einfacher zu identifizieren.

3. Die Web Anwendung bearbeitet die Anfrage und sendet eine Antwort zurück. Je nach Inhalt der Antwort führt die Android Anwendung verschiedene Schritte aus.

    - Eintrag wurde erstellt oder wird noch bearbeitet
      Es wurde noch keine Entscheidung von einem Administrator hinsichtlich der Registrierungsanfrage getroffen. Ist die Automatik aktiviert, legt die Android Anwendung den Zeitpunkt fest an dem der aktuelle Status das nächste mal geprüft wird. Der Benutzer kann auch manuell durch drücken der entsprechenden Schaltfläche weitere Anfragen senden.
    - Die Anfrage wurde akzeptiert
      Die Seiten der Web Anwendung für das Herunterladen der aktuellen Android Anwendungskonfiguration und das Senden der Gerätedaten werden durch HTTPS mit Server und Client Zertifikat gesichert. Der erste Schritt ist also das Abrufen eines entsprechenden Client Zertifikates von der Web Anwendung. Hat die Android Anwendung erfolgreich ein Zertifikat erhalten, ruft sie anschließend die aktuelle Konfiguration ab und verarbeitet sie an. Der Benutzer kann jetzt in der Android Anwendung das automatische Senden der Gerätedaten aktivieren.
    - Die Anfrage wurde abgelehnt
      Die Android Anwendung kann weiterhin Anfragen an die Web Anwendung senden, diese werden aber automatisch abgelehnt. Zusätzlich wird bei dieser Antwort die Automatik für das Senden von Registrierungsanfragen deaktiviert.

3.2.2 Sammeln der Daten

Es gibt zwei Möglichkeiten, wie das Senden der Gerätedaten durchgeführt werden kann. Der Benutzer kann manuell über die graphische Oberfläche der Anwendung den Sammelzyklus starten oder die Automatik sendet die Anfragen dem Zeitplan in der Konfiguration folgend. Abbildung 6 zeigt die einzelnen Schritte die von der Android- und Web Anwendung während durchgeführt werden.

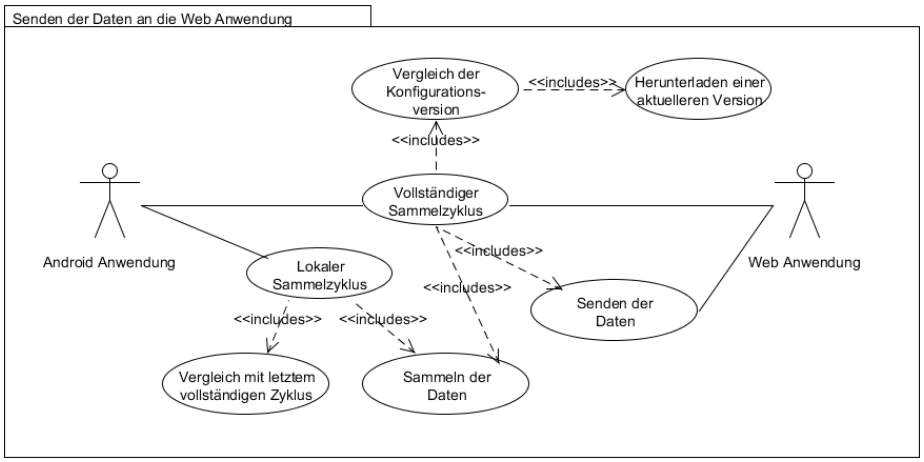

Abbildung 6: Senden der Gerätedaten

Die folgenden Schritte werden von der Android Anwendung durchgeführt.

1. Der erste Schritt in einem Sammelzyklus ist das Sammeln der benötigten Informationen, wie etwa dem Standort oder Temperatur, aus den verschiedenen Quellen.

2. Die Android Anwendung unterstützt zwei verschiedene Sammelzyklen, einen „Vollständigen“ und einen „Lokalen“, in denen die Gerätedaten abgefragt werden. Im vollständigen Zyklus werden die Daten gesammelt und anschließend an die Web Anwendung gesendet. Der lokale Zyklus führt ebenfalls eine Abfrage der Werte durch, allerdings sendet er sie nicht, sondern vergleicht sie mit den Werten des letzten vollständigen Zyklus um Abweichungen zu finden. Der Zeitplan, der die Abstände zwischen den Zyklen und auch den Zeitraum in dem sie ausgeführt werden festlegt, wird in der Konfiguration für die Android Anwendungen festgelegt. In den beiden folgenden Punkten werden die einzelnen Schritte während der Sammelzyklen erklärt.

    - Vollständig
      Ein vollständiger Sammelzyklus beinhaltet das Sammeln aller Daten und senden an die Web Anwendung. Anschließend werden sie noch in den Anwendungsdaten als Momentaufnahme gesichert. Wurden die Daten gesendet, vergleicht die Android Anwendung die Version ihrer lokalen Konfiguration mit der Version

auf dem Server um festzustellen ob eine neuere verfügbar ist. Nach der Prüfung und dem eventuellen Herunterladen der Konfiguration wird mit Hilfe des vorgegebenen Zeitplans der nächste vollständige oder lokale Sammelzyklus geplant.

- Lokal
  Ein vollständiger Sammelzyklus verbraucht mehr Energie, vor allem wenn der Abstand zwischen zwei Zyklen sehr gering ist. Ein lokaler Sammelzyklus sendet die Informationen nicht wie der vollständige Zyklus. Stattdessen werden die gesammelten Daten mit der letzten Momentaufnahme verglichen. Damit sollen bestimmte Ausprägungen eines Wertes oder Abweichungen, die eine in der Konfiguration festgelegte Abweichung überschreiten, festgestellt und aufgezeichnet werden. Im nächsten vollständigen Zyklus werden die aufgezeichneten Meldungen über Abweichungen zu den gesendeten Daten hinzugefügt. Durch diese lokalen Prüfungen können die Werte eines Gerätes öfters abgetastet werden, ohne sie jedes mal an die Web Anwendung senden zu müssen.

3.2.3 Konfiguration der Anwendung

Es wäre möglich, jede installierte Android Anwendung eines Monitoring Systems einzeln zu konfigurieren. Bei einer steigenden Anzahl an Geräten zieht so jedoch jede Änderung eines Konfigurationsparameters einen erheblichen Aufwand nach sich. Aus diesem Grund wird ein Teil der Konfiguration, die als XML Dokument auf dem Web Server gespeichert ist, durch die Web Anwendung als zentrale Stelle verteilt. In der Anwendung selbst können die folgenden Einstellungen festgelegt werden.

- Die URL des Web Servers, auf dem die Web Anwendung ausgeführt wird. Sie muss eingegeben werden, bevor mit dem Registrierungsvorgang begonnen werden kann.
- Der Alias des Gerätes, um den Administrator die Unterscheidung zu erleichtern.
- Falls der Web Server eine selbst-signiertes Zertifikat verwendet kann dieses importiert werden, damit diesem beim Herstellen einer Verbindung vertraut wird.

Für den Zugriff auf die Konfiguration muss die Android Anwendung mit der Web Anwendung registriert sein und über ein gültiges Client Zertifikat verfügen. Nach dem Herunterladen werden die XML Daten verarbeitet und die enthaltenen Konfigurationsparameter in den Anwendungsdaten gesichert, anstatt das XML Dokument als ganzes zu speichern. Die Verwendung der Android API ermöglicht den Zugriff auf einzelne Parameter und erspart das Parsen der kompletten XML Konfiguration.

3.3 Web Anwendung

Die Web Anwendung soll die Verwaltung registrierter und die Registrierung neuer Geräte erleichtern. Zusätzlich empfängt sie die Daten von den Geräten, ermöglicht das Herunterladen der XML Konfigurationsdatei und überprüft ob die Android Anwendungen dem vorgegebenen Zeitplan für die Durchführung der automatischen Sammelzyklen folgen.

3.3.1 Konfiguration

Die Web Anwendung wird über eine XML Datei konfiguriert. Der Pfad der Datei wird im Context der Web Anwendung festgelegt wie in Kapitel 5.2 genauer beschrieben wird.

3.3.2 Registrierung

Empfängt die Web Anwendung eine Anfrage für eine Geräteregistrierung, prüft sie zunächst, ob sich ein Gerät mit identischen Parametern bereits im System befindet. Dazu werden die drei Listen, die von der Anwendung verwaltet werden kontrolliert.

- Die erste Liste ist eine Whitelist mit den Geräten, deren Anfragen akzeptiert wurden und dadurch Zugriff auf Funktionen wie das Herunterladen der Konfiguration der Android Anwendung erhalten.
- Die zweite Liste ist eine Blacklist mit den Geräten deren Anfragen abgelehnt wurden. Weitere Anfragen von in dieser Liste enthaltenen Geräten werden automatisch abgelehnt.
- Die dritte Liste enthält die Geräte, die eine Anfrage für die Registrierung gesendet haben, für die aber noch keine Entscheidung durch den Administrator getroffen wurde.

Enthält keine der Listen einen Eintrag für das Gerät, so wird ein neuer Eintrag in der Liste der unentschiedenen Anfragen erstellt und eine Antwort an die Android Anwendung gesendet, dass das Gerät aufgenommen wurde. Ist schon ein Eintrag in der Liste der ausstehenden Anfragen vorhanden, enthält die Antwort nur die Information, dass die Anfrage noch bearbeitet wird. Ist ein Eintrag in der White- oder Blacklist vorhanden, wird der Android Anwendung mitgeteilt ob das Gerät akzeptiert beziehungsweise abgelehnt wurde.

3.3.3 Empfang der Gerätedaten

Beim Empfang einer Anfrage werden die enthaltenen Parameter mit der Whitelist abgeglichen um zu prüfen ob das Gerät zugelassen ist. Dies geschieht zusätzlich zur Prüfung des Client Zertifikates durch die Web Anwendung. Ist für das Gerät mit den entsprechenden Parametern ein Eintrag auf der Whitelist vorhanden und das Client Zertifikat ist gültig ist der nächste Schritt die Weiterleitung der empfangenen Daten an die Java Anwendung, die sie in die Nagios Kommando Datei schreibt um den Status des entsprechenden Gerätes zu aktualisieren. Die IP-Adresse und Port Nummer der Java Anwendung wird über die Konfiguration der Web Anwendung festgelegt. Für den Fall, dass die Kommunikation zwischen der Web Anwendung und der Java Anwendung über unsichere Kanäle stattfindet, kann sie bei Bedarf, durch den entsprechenden Eintrag in der Konfiguration, mit TLS verschlüsselt werden.

3.3.4 Zeitplanüberprüfung

Die Android Anwendung folgt einem Zeitplan für das Senden der Gerätedaten, der in ihrer XML Konfiguration festgelegt ist. Zu Beginn einer Überprüfung liest die Web Anwendung den Zeitplan aus dieser Konfigurationsdatei. Sendet ein Gerät seine Daten an die Web Anwendung, wird ein Zeitstempel bei deren Empfang gespeichert. Mit Hilfe dieses Zeitstempels und des Zeitplans aus der XML Konfiguration kann nun festgestellt werden, ob ein geplanter Sendezeitpunkt versäumt wurde. Hat ein Gerät eine festgelegte Anzahl an vollständigen Sammelzyklen nicht durchgeführt, sendet die Web Anwendung ein Kommando an die Java Anwendung auf dem Nagios Host um den Status des Gerätes zu ändern. Die maximal erlaubte Anzahl für

versäumte vollständige Sammelzyklen wird über die Konfiguration der Web Anwendung festgelegt.

3.4 Java Anwendung

Die Java Anwendung wird auf demselben Host ausgeführt wie der Nagios Server. Sie führt den letzten Schritt in der Kette für das Aktualisieren der Gerätedaten durch, indem sie empfangene Daten in die „Externe Kommando“ Datei schreibt wie in Abbildung 7 zu sehen ist. Die Anwendung wird über die Konsole gestartet und erhält ihre Konfiguration mit Hilfe von Kommandozeilenparametern.

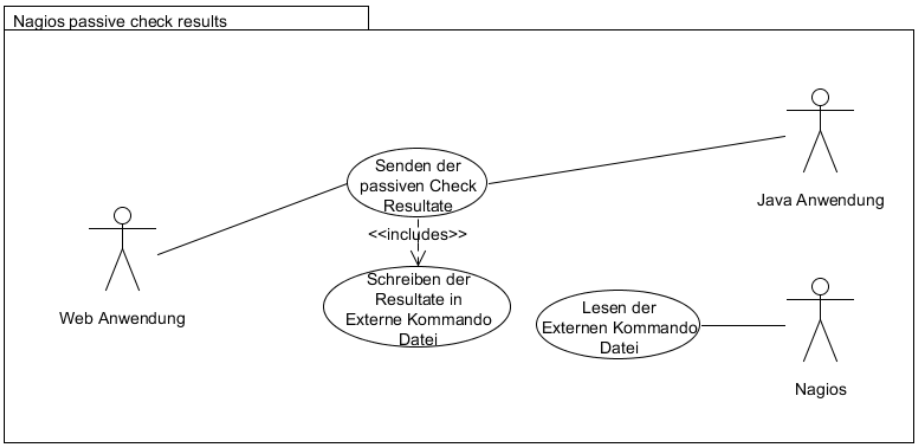

Abbildung 7: Java Anwendung schreibt in Kommando Datei

3.4.1 Schreiben der Resultate

Nach dem Start verarbeitet die Anwendung die Kommandozeilenparameter und versucht den angegebenen Netzwerkport zu reservieren. Treten keine Fehler auf, beginnt die Anwendung auf Verbindungen der Web Anwendung zu warten. Über die Kommandozeile kann angegeben werden ob die Verbindungen verschlüsselt oder unverschlüsselt aufgebaut werden sollen. Wird eine Verbindung aufgebaut, empfängt die Anwendung die gesendeten Daten, ergänzt sie mit einem UNIX Zeitstempel und schreibt sie in die „Externe Kommando“ Datei. Die syntaktische Prüfung wird von Nagios selbst durchgeführt, jedoch gibt es keinerlei Rückmeldung an die Java Anwendung, das heißt sie kann nur den erfolgreichen Empfang bestätigen. Ist die Prüfung

der Daten erfolgreich, aktualisiert Nagios das entsprechende Gerät in seiner Hostliste.

3.5 Ausgewählte Informationsquellen

3.5.1 Network connections

3.5.1.1 Wi-Fi Verbindung

Der Hauptgrund für die Auswahl der Wi-Fi Schnittstelle ist die Frage, ob das Gerät mit einem Netzwerk verbunden ist, und für den Fall, dass eine Verbindung besteht, die Überprüfung der Verbindungsparameter. Die ausgewählten Werte sind hier neben dem Status des Wi-Fi Adapters und der SSID der Verbindung vor allem die Parameter für Authentifizierung und Schlüsselverwaltung (Keine/statisches WEP, WPA_EAP, WPA_PSK, IEEE8021X) und die Verschlüsselung (CCMP, TKIP, WEP104, WEP40). Das Ziel ist die Feststellung, ob eine Verbindung zu einem ungesicherten Netzwerk besteht, über das sensible Daten unverschlüsselt übertragen werden könnten.

3.5.1.2 Bluetooth

Die Überprüfung des Bluetooth Adapters umfasst den Status des Adapters, ob eine Verbindung zu einem anderen Bluetoothgerät hergestellt ist, und falls möglich auch die Art (z. B. AUDIO_VIDEO_HEADPHONES, COMPUTER_LAPTOP, etc.) des verbundenen Gerätes.

3.5.1.3 Mobile Datenverbindung

Die Überwachung der Mobilen Datenverbindungen dient vor allem um festzustellen, ob die mobile Datenverbindung und das Roaming aktiviert/verbunden ist. Dafür werden einfache Labels (z. B. CONNECTED/DISCONNECTED) verwendet um den Status in Nagios darzustellen.

3.5.2 Gerät

3.5.2.1 Akku

Über diesen Wert kann festgestellt werden, ob eine Aufladung nötig ist oder das Gerät bereits aufgeladen wird. Sendet das Gerät keine Daten mehr,

kann geprüft werden ob der Ladelevel in den zuletzt empfangenen Daten so niedrig war, dass es zu einer Abschaltung des Gerätes führte. Dazu werden der Level als Prozentwert und der Ladestatus, ob der Akku gerade geladen oder entladen wird, von der Android Anwendung gesammelt und übertragen. Zusätzlich zu Ladelevel und -status wird noch die Temperatur des Akkus gemessen und gesendet, falls sie verfügbar ist.

3.5.2.2 Anwendungen

Die Liste sämtlicher installierter und laufender Anwendung zu sammeln und zu senden wäre möglich, aber nicht praktikabel. Stattdessen sollen mithilfe einer White- und Blacklist benötigte oder nicht erwünschte Anwendungen überprüft werden. Ist eine Anwendung auf der Whitelist nicht installiert wird sie als Teil der gesendeten Daten gemeldet. Gleiches gilt falls eine Anwendung auf der Blacklist auf dem Gerät installiert ist. Die Listen werden als Teil der Konfiguration für die Android Anwendung an die Geräte verteilt.

3.5.3 System

3.5.3.1 Datum und Zeit

Datum, Zeit und Zeitzone werden überprüft, um die automatische Synchronisierung über ein Netzwerk oder eine manuelle Einstellung zu kontrollieren. Die Formatierung der Daten richtet sich nach dem Gebietsschema, das auf dem Gerät eingestellt ist.

3.5.3.2 Geräteinformationen

Diese Werte werden aus zwei Gründen gesammelt. Ein Grund ist die Feststellung welche Android Version installiert ist. Diese Information wird verwendet um festzustellen ob diese Version aktuell ist oder ob der Hersteller eine neue veröffentlicht hat, die gegebenenfalls auf den Geräten installiert werden soll. Außerdem enthalten die Geräteinformationen Werte die zur Identifizierung und Authentifizierung des Gerätes durch die Web Anwendung verwendet werden. Dazu gehören die IMEI, der Hersteller und die Modellbezeichnung.

3.5.3.3 Entwicklereinstellungen

Aus der Kategorie der Entwicklereinstellungen wird nur die USB-Debugging Option überprüft, da diese Einstellung Auswirkungen auf die Sicherheit des Gerätes haben kann.

3.5.3.4 Rootstatus

Das Rooten eines Gerätes stellt ein Risiko für Sicherheit und Privatsphäre eines Gerätes und des Benutzers dar. Daher wird überprüft ob das System gerootet ist und die Android Anwendung mit Root Berechtigungen ausgeführt wird. Für die Feststellung wird die RootTools [9] Bibliothek verwendet.

3.5.4 Sensors

3.5.4.1 Standort

Die Standortdaten werden zusammen mit dem verwendeten Standortdienst und der Abschätzung der Genauigkeit übermittelt. Allerdings ist die Darstellung in Längen und Breitengraden an sich wenig hilfreich. Die Android Anwendung unterstützt zwei verschiedene Methoden für die Darstellung des Standortes. Die erste Methode stellt die Entfernung von einem festgelegten Ausgangspunkt als Richtungsvektor, mit der Länge in Metern und dem Winkel in Grad von Norden ausgehend dar. Der Ausgangspunkt wird als Längen- und Breitengrad in der Konfiguration angegeben. Die zweite Methode erlaubt die Definition eines rechteckigen Bereichs durch einen Ausgangs- und Endpunkt die, wie in der ersten Methode, als Längen- und Breitengrad angegeben werden. In diesem Modus wird nur angegeben, ob sich das Gerät innerhalb oder außerhalb dieses Bereiches befindet.

3.5.4.2 Temperatur

Der Temperaturwert wird verwendet, um den Zustand des Gerätes und auch der Umgebung, in der es gerade eingesetzt wird, zu überwachen. So kann festgestellt werden, ob die Gerätetemperatur oder Umgebungstemperatur sich außerhalb des empfohlenen Temperaturbereichs befinden. Zusammen mit den Standortdaten und den verwendeten Standortdiensten kann eventuell auch abgeschätzt werden, ob sich das Gerät innerhalb oder außerhalb

[9] https://github.com/Stericson/RootTools/wiki (5.11.2014)

eines Gebäudes befindet. Die Temperatur wird als einfacher numerischer Wert übermittelt.

4 Lösungsbeschreibung

4.1 Android Anwendung

Abbildung 8 gibt einen Überblick über die wichtigsten Komponenten der Implementierung und der Interaktionen zwischen ihnen.

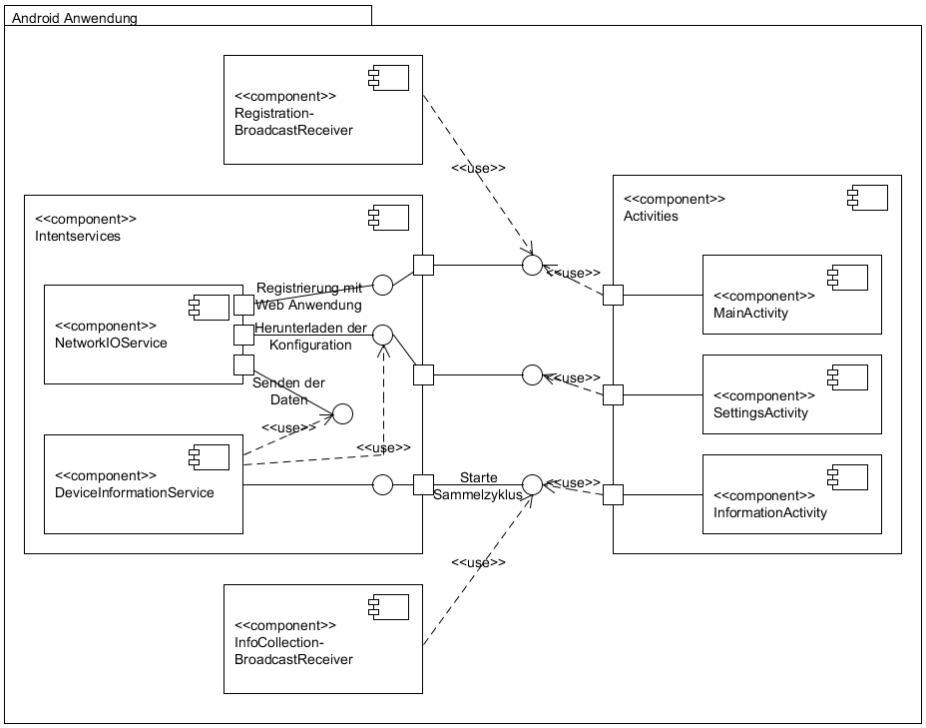

Abbildung 8: Komponenten der Android Anwendung

4.1.1 Hintergrunddienste

Die meisten Aktionen der Android Anwendung werden von Threads im Hintergrund bearbeitet, um das Blockieren der Benutzeroberfläche zu verhindern. Dafür werden die beiden Klassen `DeviceInformationService` und `NetworkIOService`, die in Abbildung 9 dargestellt sind, verwendet. Beide Serviceklassen erweitern die `android.app.IntentService` Klasse. Im Gegensatz zu einem normalen `Service` wird ein `IntentService`

bereits in einem eigenen Thread ausgeführt. Ein `IntentService` wird von einem Aufrufer gestartet und führt nach seiner Erstellung die Anweisungen in der `onHandleIntent(final Intent intent)` Methode aus. Im Anschluss wird der `IntentService` wieder beendet. Eine Rückmeldung an die im Vordergrund laufende Benutzeroberfläche erfolgt durch Anwendungsinterne Broadcasts vom Typ `android.content.BroadcastReceiver`, die beim Öffnen einer Seite der Benutzeroberfläche registriert und beim Verlassen wieder gelöscht werden, um den Empfang zu aktivieren oder zu deaktivieren.

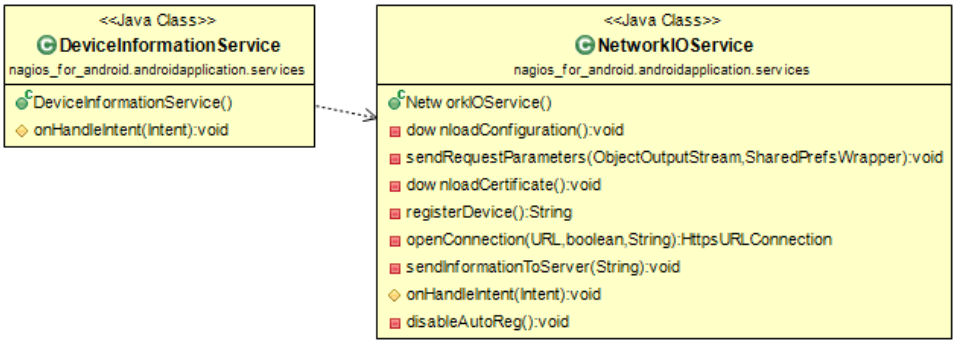

Abbildung 9: Service Klassen der Android Anwendung

4.1.1.1 `DeviceInformationService`

Nicht alle Gerätedaten können unmittelbar abgefragt und gesendet werden. Einige Datenquellen liefern Werte nur über Ereignisse nach dem Beobachter-Entwurfsmuster. Bei der Abfrage der Daten kann daher nur der entsprechende Listener registriert werden, um auf Ereignisse der Quelle zu warten. Für die manuell vom Benutzer gestarteten Abfragen werden die Listener nur für den Zeitraum der Abfrage registriert. Werden die Abfragen automatisch durchgeführt, werden die Listener beim Einschalten der Automatik registriert und beim Ausschalten wieder entfernt. Wenn ein Ereignis auftritt, werden die Daten gespeichert, um sie der nächsten Abfrage zu verwenden.
Da die Android Anwendung zwei Arten von Sammelzyklen unterstützt, entscheidet der `DeviceInformationService`, ob die Daten gesendet oder nur gesammelt und mit den gespeicherten Werten der letzten vollständigen Abfrage verglichen werden. Überschreitet während eines lokalen Sammelzyklus ein Wert eine vorgegebene Schranke oder nimmt eine bestimmte Ausprägung an, speichert die Anwendung diesen Umstand und fügt bei der

nächsten vollständigen Abfrage eine entsprechende Meldung zu den gesendeten Gerätedaten hinzu.

4.1.1.2 `NetworkIOService`

Sämtliche Netzwerkoperationen, wie das Herunterladen der Konfiguration oder das Senden der Daten werden zentral von diesem `IntentService` durchgeführt. Neben dem Aufbau der Verbindung und dem Senden der Parameter werden auch die Antworten der Web Anwendung hier verarbeitet.

4.1.2 Anwendungsdaten

Die Anwendung verwendet die von Android zur Verfügung gestellten Funktionen für den Zugriff und die Bearbeitung von Anwendungsdaten, um verschiedenste Informationen persistent zu speichern. Diese Daten werden privat für die Android Anwendung angelegt und andere Anwendungen können im Normalfall nicht darauf zugreifen. Für die Speicherung von Daten auf dem Gerät stehen grundsätzlich drei Methoden zur Verfügung.

- Datei [10]
  Dateien können entweder im internen Telefonspeicher oder mit den entsprechenden Berechtigungen auf einem externen Speichermedium angelegt werden. Der interne Speicher besteht aus einem Verzeichnis in einem speziellen Bereich des Dateisystems, das einem Anwendungspaket zugewiesen wird. Anwendungen können grundsätzlich auf alle Dateien auf einem externen Speichermedium zugreifen. Trotzdem ist es auch hier möglich Anwendungseigene Dateien zu erstellen, auf die zwar von anderen Anwendungen zugegriffen werden können, die aber bei einer Deinstallation der Anwendung ebenfalls gelöscht werden.

- SQL-Datenbank [11]
  Android verwendet die SQLite [12] Datenbank um Anwendungen die Verwendung einer Datenbank zu ermöglichen. Über das `android.database.sqlite` Java Paket kann eine Anwendung Datenbanken erstellen und bearbeiten.

[10] http://developer.android.com/training/basics/data-storage/files.html (5.11.2014)
[11] http://developer.android.com/training/basics/data-storage/databases.html (5.11.2014)
[12] https://sqlite.org/ (5.11.2014)

- SharedPreferences [13]
  `SharedPreferences` funktionieren wie eine `HashMap`, indem sie einen Datenwert mit einem Schlüssel assoziieren unter dem der Wert adressiert werden kann. Sie verweisen auf eine XML Datei in den Anwendungsdaten, in die abgelegte Schlüssel-Wert Paare geschrieben und von der sie wieder gelesen werden.

In der Android Anwendung wird für die Speicherung von Daten ausschließlich die `SharedPreferences` Funktionalität verwendet. Einzige Ausnahme ist das Client Zertifikat, das in einer Java KeyStore (JKS) Datei abgelegt wird. Die Verwendung einer SQL Datenbank wäre zwar möglich, ist in diesem Fall aber ungeeignet, da es sich um einzelne, sich nicht wiederholende Werte handelt. Die folgenden Informationen werden auf dem Gerät gespeichert:

- Die Parameter der Anwendungskonfiguration anstatt der gesamten XML Konfiguration in einer Datei.
- Die Werte der letzten vollständigen Abfrage der Geräteinformationen, um sie mit den lokalen Zwischenabfragen vergleichen zu können.

4.1.3 Registrierung

Bevor die Android Anwendung die Gerätedaten senden kann, ist eine Registrierung des Gerätes in der Web Anwendung nötig. Abbildung 10 zeigt die Klassen, die für die Durchführung einer Registrierungsanfrage benötigt werden. Die ursprüngliche Anfrage startet der Benutzer indem er auf der Startseite der Anwendung, deren Verhalten in der `MainActivity` Klasse definiert wird, die „Register“ Schaltfläche betätigt. Die `NetworkIOService` Klasse sendet die Netzwerkanfrage mit den notwendigen Parametern wie der IMEI und Android-ID und verarbeitet auch die Antwort der Web Anwendung. Die Web Anwendung sendet mit ihrer Antwort den Status der Registrierung, der in den Anwendungsdaten gespeichert und im Textfeld auf der Startseite angezeigt wird falls die Benutzeroberfläche geöffnet ist. Alle nachfolgenden Anfragen rufen den aktuellen Status der Registrierung von der Web Anwendung ab. Um zu vermeiden, dass der Benutzer ständig die Anwendung öffnen und die Schaltfläche drücken muss, um den aktuellen Status

[13] http://developer.android.com/training/basics/data-storage/shared-preferences.html (5.11.2014)

zu erfahren oder die Registrierung abzuschließen, wird die `Registration BroadcastReceiver` Klasse verwendet. Sie erweitert die Klasse `android.content.BroadcastReceiver` um Broadcasts des Android Betriebssystems empfangen zu können. Um eine Aktion zu einem bestimmten Zeitpunkt durchzuführen werden die Broadcasts des Android `AlarmManager` benötigt. Der `AlarmManager` ermöglicht es Ereignisse zu registrieren die zum einem festgelegten Zeitpunkt ausgelöst werden. In der Registrierung des Ereignisses wird der `BroadcastReceiver` angegeben, der für die Ereignisbehandlung erstellt wird. Der `RegistrationBroadcastReceiver` startet nach seiner Erstellung den `NetworkIOService` um eine Registrierungsanfrage zu senden. Im Anschluss bestimmt er den Zeitpunkt für die nächste automatische Anfrage und registriert ein Ereignis im `AlarmManager`.
Abbildung 10 zeigt auch den `BootCompleteBroadcastReceiver`, der von Android aufgerufen wird wenn das System fertig hochgefahren ist. Seine Aufgabe ist die Aktivierung der Automatik für die Registrierung oder das Senden der Gerätedaten falls diese zum Zeitpunkt des Ausschaltens aktiviert waren.

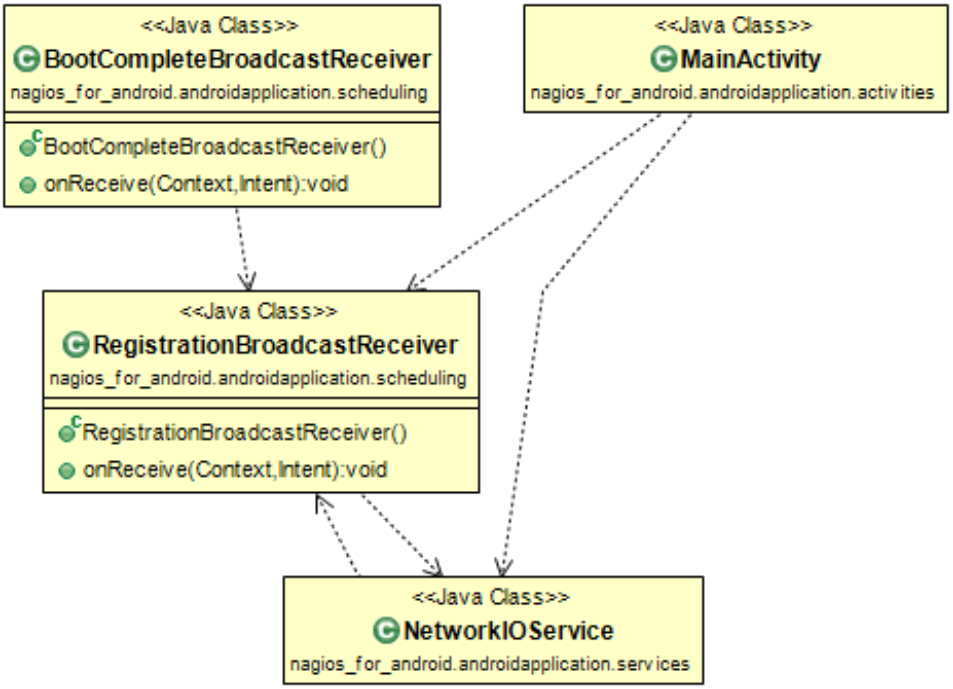

Abbildung 10: Klassen für Registrierung eines Gerätes

4.1.4 Konfiguration

Wird die Registrierungsanfrage eines Gerätes akzeptiert, wird die Konfiguration als Teil des Registrierungsprozesses heruntergeladen. Eine Möglichkeit sie anschließend zu aktualisieren, ist manuell durch Drücken der „Download configuration“ Schaltfläche auf der Einstellungsseite der Benutzeroberfläche. Zusätzlich wird eine Aktualisierungsprüfung am Ende jedes vollständigen Sammelzyklus durchgeführt. Abbildung 11 zeigt die verschiedenen Klassen, die an den manuellen und automatischen Aktualisierungen beteiligt sind. Der `NetworkIOService` sendet dabei wieder die Anfrage und Parameter und lädt anschließend die Konfiguration herunter falls eine neuere Version vorhanden ist. Nach dem Herunterladen der Konfiguration im XML Format, wird sie anschließend verarbeitet und die Parameter in den Anwendungsdaten gesichert.

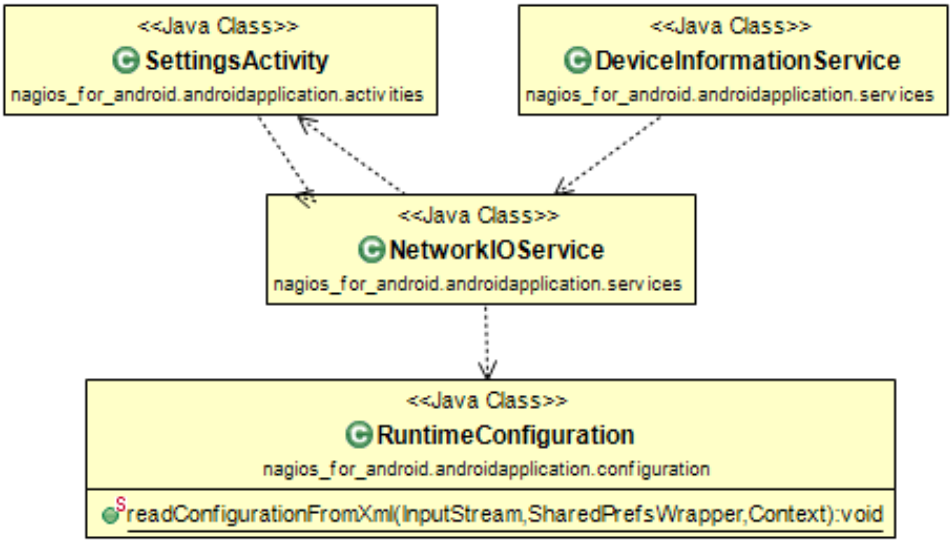

Abbildung 11: Klassen für Herunterladen der Konfiguration

4.1.5 Periodisches Senden der Gerätedaten

Die Anwendung soll die Gerätedaten in regelmäßigen, in der Konfiguration festgelegten Abständen, an die Web Anwendung senden. Die vollständigen und lokalen Sammelzyklen sollen automatisch erfolgen und keine Interaktion durch den Benutzer erforderlich machen. Zusätzlich verfügt die Anwendung über eine Seite in der Benutzeroberfläche, in der die aktuellen

Daten manuell angezeigt und an die Web Anwendung gesendet werden können. Um die automatischen Sammelzyklen zu realisieren wird, wie für die automatischen Registrierungsanfragen, ein `BroadcastReceiver` und der `AlarmManager` verwendet. In Abbildung 12 ist der `InfoCollection BroadcastReceiver` dargestellt, der für das Starten eines vollständigen oder lokalen Sammelzyklus verwendet wird. Aufgrund der Restriktionen für `BroadcastReceiver` wird der `DeviceInformationService` gestartet, der den jeweiligen Sammelzyklus in einem eigenen Thread durchführt. Nach dem Start des `IntentService` bestimmt der Receiver noch der Zeitpunkt des nächsten geplanten Broadcastereignisses und registriert es im `AlarmManager`.

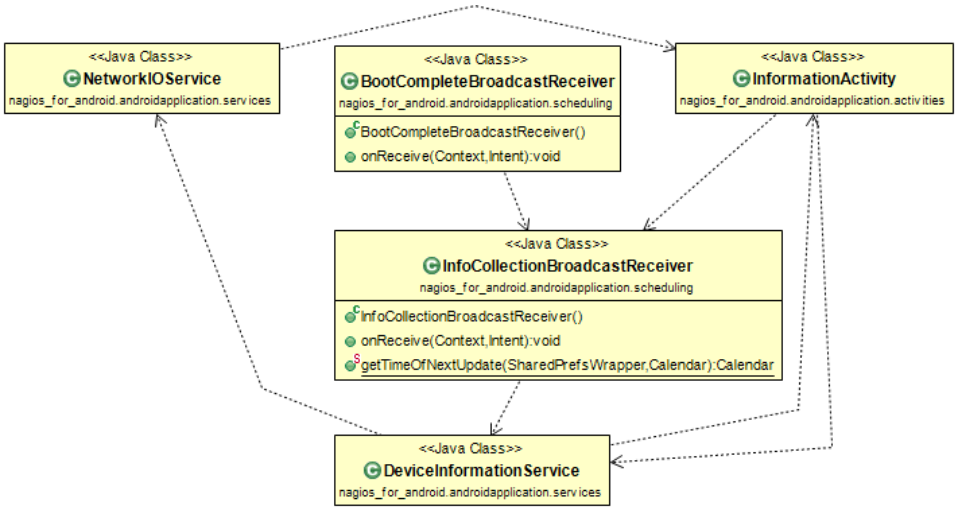

Abbildung 12: Senden der Gerätedaten

4.1.6 Benötigte Berechtigungen

Die Android Anwendung benötigt für ihre Ausführung die folgenden Berechtigungen.

- Genauer- (GPS) und ungefährer (Netzwerkbasierter) Standort
- Bluetooth-Verbindungen herstellen
  Für die Feststellung des Bluetoothstatus.
- Uneingeschränkter Internetzugriff

- Telefonstatus lesen und identifizieren
  Für den Zugriff auf die IMEI des Gerätes
- Netzwerkstatus anzeigen
  Für den Zugriff auf den Status der Mobilen Datenverbindung und des Roamings.
- WLAN-Status anzeigen
- Benachrichtigung wenn fertig hochgefahren
  Um das automatische Senden oder Registrieren nach einem Neustart, falls erforderlich, wieder zu aktivieren.
- SD-Karten-Inhalt lesen
  Um die SD-Karte nach *.cer Dateien zu durchsuchen und als Server Zertifikat importieren zu können.

4.2 Web Anwendung

Neben den Java Klassen einer normalen Anwendung benötigt eine Web Anwendung einige XML Konfigurationsdateien im `WEB-INF/` Verzeichnis des Projekts für die Ausführung auf einem Server. Einstellungen wie die Zuweisung einer HTML Seite zu einer URL werden in der `web.xml` Datei festgelegt. Für die Implementierung der Administrator Webseiten verwendet die Web Anwendung die Apache MyFaces [14] Implementierung des JavaServer Faces Framework und zusätzlich die Apache MyFaces Trinidad [15] Komponentenbibliothek. Die Einstellungen für JavaServer Faces werden in der `faces-config.xml` Datei festgelegt.

4.2.1 web.xml

Ein `javax.servlet.http.HttpServlet` ermöglicht die Behandlung von HTTP Anfragen wie GET und POST. Um ein `HttpServlet` einer URL zuzuweisen ist ein entsprechender Eintrag in `web.xml` nötig, wie in Listing 1 gezeigt wird.

```
<servlet>
  <description></description>
  <display-name>DeviceConfiguration</display-name>
```

[14] http://myfaces.apache.org/ (5.11.2014)
[15] https://myfaces.apache.org/trinidad/ (5.11.2014)

```
  <servlet-name>DeviceConfiguration</servlet-name>
  <servlet-class>nagios_for_android.webapplication.controller.
      DeviceConfigurationServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>DeviceConfiguration</servlet-name>
  <url-pattern>/devices/deviceconfiguration.xhtml</url-pattern>
</servlet-mapping>
```

Listing 1: Definition eines Servlets

Ressourcen wie eine JDBC Datenbankverbindung, die im Context des Server oder der Anwendung definiert werden, müssen referenziert werden bevor sie verwendet werden können. Listing 2 zeigt eine Referenz auf eine JDBC Verbindung. In der Anwendung kann mit dem Namen, der im `<res-ref-name>` Element angegeben ist, eine Instanz von `javax.sql.DataSource` erzeugt und die Datenbank verwendet werden.

```
<resource-ref>
  <description>JDBC Connection</description>
  <res-ref-name>jdbc/nagiosforandroid</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

Listing 2: Referenz auf die im Context definierte JDBC Verbindung

Bevor eine HTTP Anfrage durch ein `HttpServlet` behandelt wird, durchläuft sie eine Reihe von `javax.servlet.Filter` Klassen. Diese Filterkette kann von der Web Anwendung erweitert werden um zum Beispiel bestimmte Anfrageparameter zu prüfen. Eine Anwendung von Filtern ist die Überprüfung ob ein Benutzer, der Anfragen an das Administrator-Interface sendet, angemeldet ist. Ist keine gültige Session für den Benutzer vorhanden, wird er an die `/login.xhtml` Seite umgeleitet. Ein zweiter Filter wird verwendet um die Client Zertifikate zu prüfen, die von der Android Anwendung für den Zugriff auf bestimmte Funktionen gesendet werden müssen. Listing 3 zeigt wie ein `Filter` registriert und einem URL Muster zugewiesen wird.

```
<filter>
  <filter-name>LoginFilter</filter-name>
  <filter-class>nagios_for_android.webapplication.filter.
      LoginFilter</filter-class>
</filter>
<filter-mapping>
```

```
  <filter-name>LoginFilter</filter-name>
  <url-pattern>/secured/*</url-pattern>
</filter-mapping>
```

Listing 3: Login Filter für Admin Interface

Eine komplette Beschreibung für alle Elemente im `web.xml` Deployment Descriptor findet man unter [Corporation(2014b)].

4.2.2 faces-config.xml

JavaServer Faces verwendet Beans um die Web Komponente mit der dahinterliegenden Logik zu verbinden. Es handelt sich dabei um normale Java Klassen, deren Felder und Methoden von einer Java Server Faces Seite aus verwendet werden können. Listing 4 zeigt die Definition eines Beans mit Namen und der Java Klasse. Zusätzlich wird noch der Gültigkeitsbereich des Beans angegeben. `<managed-bean-scope>session</managed-bean-scope>` legt fest, dass eine einzige Instanz von `LoginController` für die gesamte Session verwendet wird und gültig ist. Eine weitere Möglichkeit wäre z. B. `<managed-bean-scope>request</managed-bean-scope>` wo für jede Anfrage eine neue Instanz des Beans erstellt wird.

```
<managed-bean>
  <managed-bean-name>loginController</managed-bean-name>
  <managed-bean-class>nagios_for_android.webapplication.controller.
      LoginController</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

Listing 4: Definition eines Beans in Java Server Faces

JavaServer Faces ermöglicht die Definition von Navigationsregeln die festlegen, unter welchen Umständen ein Wechsel von einer Webseite zu einer anderen erlaubt und durchführbar ist. Listing 5 zeigt einen Wechsel von `/login.xhtml` zu `/secured/pendingrequests.xhtml` wenn ein Benutzer sich erfolgreich anmelden kann. Die Anmeldeanfrage wird in der `authenticateUser()` Methode des `loginController` Beans geprüft. Das Element `<from-action>#loginController.authenticate User</from-action>` legt fest, dass die Navigationsregel nach dem Aufruf dieser Methode geprüft werden soll. `authenticateUser()` hat einen Rückgabewert vom Typ `java.lang.String` deren Wert entweder ”authenticated” oder null sein kann je nachdem ob die Anmeldung erfolgreich

war oder nicht. Entspricht der Rückgabewert dem, im `<from-outcome> authenticated</from-outcome>` Element angegeben Wert, ist die Regel gültig und der Benutzer wird auf die angegebene Seite umgeleitet.

```
<navigation-rule>
  <display-name>login</display-name>
  <from-view-id>/login.xhtml</from-view-id>
  <navigation-case>
    <from-action>#{loginController.authenticateUser}</from-action>
    <from-outcome>authenticated</from-outcome>
    <to-view-id>/secured/pendingrequests.xhtml</to-view-id>
    <redirect />
  </navigation-case>
</navigation-rule>
```

Listing 5: Definition einer Navigationsregel

Weitere Informationen über die Konfiguration von Java Server Faces findet man unter [Corporation(2014a)].

4.2.3 Datenbank

Neben Benutzernamen und Passwörtern für die Anmeldung im Administrationsinterface, werden auch die Einträge auf Whitelist, Blacklist und die noch nicht entschiedenen Anfragen in einer Datenbank gespeichert. Die Whitelist enthält Geräte deren Registrierungsanfrage vom Administrator akzeptiert wurde. Diese Geräte haben Zugriff auf sämtliche Komponenten des Android Anwendungsinterface der Web Anwendung. Geräte auf der Blacklist wurden vom Administrator abgelehnt. Sendet ein solches Gerät eine Anfrage, wird diese nicht von der Web Anwendung bearbeitet. Die dritte Liste enthält Geräte, die eine Registrierungsanfrage gesendet haben, die jedoch noch nicht von einem Administrator auf die White- oder Blacklist verschoben wurden. Abbildung 13 zeigt die `SQLDeviceDataSource` Klasse. Um die Web Anwendung mit der verwendeten Datenbank zu verbinden, wird Java Database Connectivity (JDBC)[16] eingesetzt. Die Tabellen für die Listen enthalten die folgenden Spalten.

1. DeviceAlias, varchar(100)
   Diese Spalte enthält den Namen, der vom Benutzer in den Einstellungen der Android Anwendung für das Gerät vergeben werden kann.

[16] http://www.oracle.com/technetwork/java/javase/jdbc/index.html (5.11.2014)

2. UUID, varchar(50)
   Diese ID wird von der Android Anwendung bei ihrem ersten Start zufällig generiert.

3. IMEI und AndroidID, varchar(50)
   Die International Mobile Station Equipment Identity (IMEI) und die ID des Android Betriebssystems des Gerätes.

4. DeviceModel, varchar(100)
   Dieser Name setzt sich aus verschiedenen Werten wie Hersteller und Modellbezeichnung des Gerätes zusammen.

5. Added, DATETIME
   Die Zeit zu der das Gerät in die Liste der ausstehenden Registrierungsanfragen, also der Zeitpunkt der ersten Registrierungsanfrage, eingetragen wurde.

6. LastConnection, DATETIME
   Der Zeitpunkt an die Android Anwendung zuletzt erfolgreich die Resultate eines vollständigen Sammelzyklus an die Web Anwendung gesendet hat.

7. MissedUpdates, int
   Die Web Anwendung prüft in regelmäßigen Abständen ob alle Geräte der Whitelist die Resultate ihrer vollständigen Sammelzyklen laut Zeitplan senden. Verpasst ein Gerät einen geplanten Sendezeitpunkt wird dieser Zähler inkrementiert. In der Konfiguration der Web Anwendung kann das Limit angegeben werden, bei dessen Erreichen eine Änderung des Gerätestatus in Nagios erforderlich wird. Beim Empfang der Informationen eines Gerätes wird der jeweilige Zähler wieder auf null gesetzt.

8. ClientCert, BIT
   Diese Spalte ist nur in der Whitelist Tabelle vorhanden und gibt an ob an das Gerät mit dem entsprechenden Eintrag ein Client Zertifikat gesendet wurde. Um ein neues Client Zertifikat zu erhalten muss der Wert manuell wieder auf null gesetzt oder das Gerät neu registriert werden.

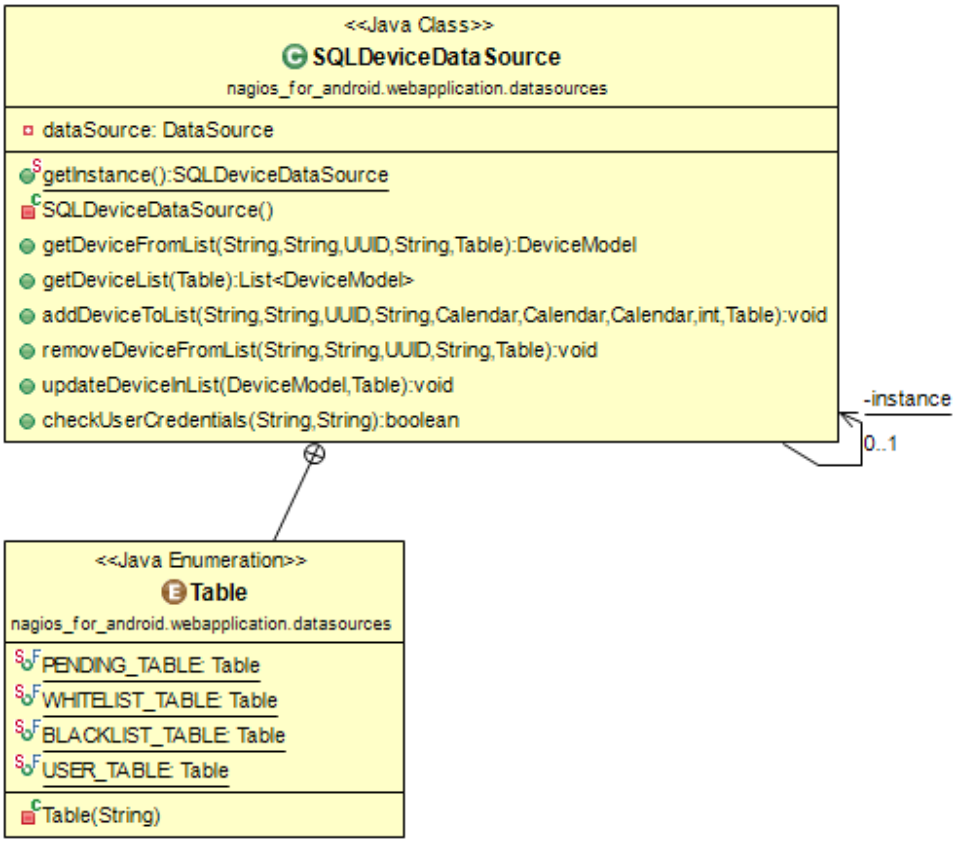

Abbildung 13: SQL Datenquellen

4.2.4 Einhaltung der Abfragezeitpläne

Die Web Anwendung prüft in regelmäßigen Abständen ob Geräte auf der Whitelist es versäumt haben, ihre Gerätedaten laut Zeitplan zu senden. Die Überprüfung wird in der `DeviceUpdateCheck` Klasse durchgeführt und durch den `java.util.concurrent.ScheduledExecutorService` gestartet, die es ermöglicht ein `java.lang.Runnable` in regelmäßigen Abständen in einem eigenen Thread auszuführen. Die Abstände zwischen den Überprüfungen werden in der Konfiguration der Web Anwendung festgelegt. Die Registrierung findet beim Start der Web Anwendung statt. Dazu wird die `WebAppStartup` Klasse, eine Implementation des `javax.servlet.ServletContextListener` Interface, verwendet. Der `ServletContextListener` wird bei der Initialisierung und Beendigung einer Web Anwendung über die `contextDestroyed` und `contextInitialized` Methoden benachrichtigt. Die Registrierung von `WebAppStartup` wird in Lis-

ting 6 gezeigt. In der `contextInitialized` Methode wird nun die regelmäßige Ausführung des `DeviceUpdateCheck` Threads registriert und in `contextDestroyed` wird die Registrierung wieder entfernt. Für die Überprüfung werden die Zeitstempel in der Whitelist, an denen zuletzt erfolgreich die Gerätedaten empfangen wurden, mit Hilfe des Zeitplanes aus der Android Anwendungskonfiguration mit der aktuellen Zeit verglichen. Ist der Unterschied zwischen aktueller Zeit und dem Zeitstempel größer als das Sendeintervall des aktuellen Tages im Zeitplan, wird der Zähler für versäumte Abfragen des Gerätes in der Whitelist auf die Anzahl der insgesamt versäumten vollständigen Sammelzyklen gesetzt. Überschreitet dieser Wert das festgesetzte Limit wird der Status des Gerätes in Nagios durch senden einer Anweisung in Form eines Passiven Check Resultats geändert.

```
<listener>
  <listener-class>nagios_for_android.webapplication.config.
      WebAppStartup</listener-class>
</listener>
```

Listing 6: Registrierung von WebAppStartup

4.2.5 Administrator-Interface

Das Interface für Administratoren besteht aus verschiedenen Internetseiten und ermöglicht die Verwaltung der Einträge von Android Geräten auf den drei Listen. Das Administrationsinterface besteht aus den folgenden Internetseiten für die Darstellung der Listen und den zugehörigen, in Abbildung 14 dargestellten, Controllern für die Manipulation der Daten.

- `index.xhtml`
  Die Index Seite enthält nur eine Navigationsleiste mit Verweisen auf die anderen Internetseiten.

- `login.xhtml`, `LoginController`
  Die Anmeldung in das Administrationsinterface erfolgt über die Login Seite und erfordert einen gültigen Benutzernamen und ein Passwort. Erst wenn der Benutzer in der aktuellen HTTPS Sitzung angemeldet ist, kann er alle Internetseiten öffnen. Zu diesem Zweck erstellt die Web Anwendung ein Managed Bean des `LoginController` für die gesamte Sitzung, das die An- und Abmeldung des Benutzers

durchführt. Jede gesendete Anfrage durchläuft eine Kette von Filtern des Typs `javax.servlet.Filter` in denen verschiedene Aktionen, wie etwa Logging, durchgeführt werden können. Die Web Anwendung erweitert diese Filterkette um den `LoginFilter` um zu verhindern, dass ein Benutzer ohne gültige Anmeldung Zugang zu den Gerätelisten erhält. Wird eine Anfrage an eine der geschützten Seiten(`pendingrequests.xhtml`, `whitelist.xhtml`, `blacklist.xhtml`) gesendet, wird der `LoginFilter` aufgerufen und prüft ob der Benutzer der aktuellen Sitzung angemeldet ist. Ist eine gültige Anmeldung vorhanden, wird der nächste Filter in der Kette aufgerufen, ansonsten erfolgt eine Weiterleitung zur Login Seite.

- `pendingrequests.xhtml`, `PendingRequestsController`
  Diese Seite zeigt alle Geräte in der Liste der noch unentschiedenen Anfragen dar. Für jedes Gerät in der Liste kann nun entschieden werden, ob das Gerät akzeptiert und zur Whitelist oder abgelehnt und zur Blacklist hinzugefügt wird.

- `whitelist.xhtml`, `WhiteListController` und `blacklist.xhtml`, `BlackListController`
  Sie zeigen die Geräte die, in der jeweiligen Liste eingetragen sind und ermöglichen es, Einträge wieder zu entfernen.

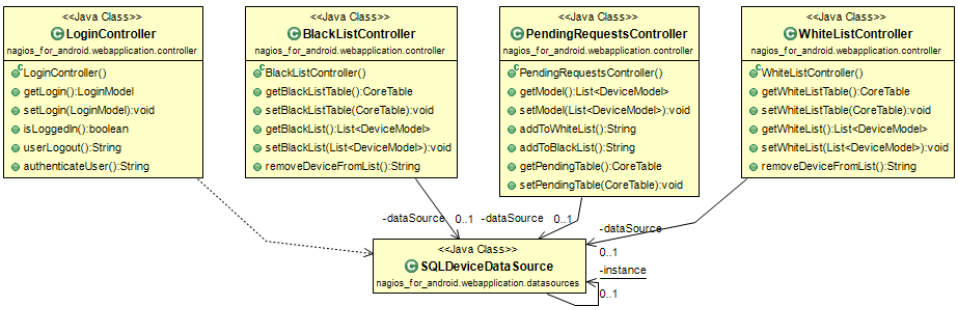

Abbildung 14: Interface für Administration

4.2.6 Interface für Android Anwendungen

Die Android Anwendung kommuniziert mit der Web Anwendung über HTTP Anfragen. Dieses Interface besteht jedoch nicht aus HTML Seiten, sondern nur über verschiedene Servlet Klassen die `javax.servlet.http.`

`HttpServlet` erweitern, um GET und POST Anfragen bearbeiten. Die Seiten sind in zwei Sicherheitsstufen unterteilt, die den Zugriff beschränken.

- Nur Server Zertifikat
  Die Verbindung zu den Seiten `registration.xhtml` und `certificate.xhtml` ist mit HTTPS gesichert und ein Zugriff ist grundsätzlich für jeden möglich. Die Authentifizierung erfolgt über die Parameter, die in den Anfragen enthalten sein müssen. Die Registrierungsseite benötigt die Parameter um festzustellen, ob sich das Gerät bereits in einer der Gerätelisten befindet und den Status der Registrierung mit der Antwort zu senden. Die Zertifikatseite prüft anhand der Parameter ob ein Client berechtigt ist, ein Zertifikat herunterzuladen. Hat ein Client ein Zertifikat erhalten, wird die „ClientCert“ Spalte seines Eintrages in der Whitelist der Datenbank auf eins gesetzt. Dieser Wert teilt der Web Anwendung mit, dass der Client ein Zertifikat hat und kein neues mehr ausgestellt werden darf.

- Server und Client Zertifikat
  Die beiden Seiten `/devices/deviceinformation.xhtml` und `/devices/deviceconfiguration.xhtml` benötigen zusätzlich zu den Parametern ein gültiges Client Zertifikat für die Authentifizierung. Die Web Anwendung speichert und verwaltet Kopien der ausgestellten Client Zertifikate der Geräte auf der Whitelist. Sendet ein Gerät eine Anfrage an eine dieser Seiten, wird geprüft ob ein Client Zertifikat vorhanden ist, es durch das Server Zertifikat der Web Anwendung signiert und mit dem, mit der IMEI+AndroidID Kombination als Alias, gespeicherten Zertifikats übereinstimmt.

4.2.6.1 Registrierung

Im Verlauf einer erfolgreichen Registrierung sendet die Android Anwendung Anfragen an die folgenden Seiten der Web Anwendung, die von den in Abbildung 15 sichtbaren Servlets behandelt werden.

- `registration.xhtml`, `RegistrationServlet`
  Das Servlet liest die Parameter aus der Anfrage und prüft, wenn die benötigten Parameter vorhanden sind, ob ein Eintrag mit identischen Werten in einer der Gerätelisten vorhanden ist. Ist noch kein Eintrag vorhanden, erstellt das Servlet eine Instanz der `SQLDeviceData`

`Source` Klasse um einen Eintrag in der SQL Datenbank zu erstellen. Anschließend wird der Android Anwendung in der Antwort noch mitgeteilt das der Eintrag erstellt wurde. Ist schon ein Eintrag in einer der Listen vorhanden, wird in der Antwort mitgeteilt, dass die Anfrage noch bearbeitet wird oder bereits akzeptiert beziehungsweise abgelehnt wurde.

- `certificate.xhtml`, `CertificateServlet`
  Um ein Zertifikat zu erhalten, muss sich ein Eintrag für das Gerät auf der Whitelist befinden. Das Client Zertifikat wird in der `create ClientCertificate(final PublicKey clientPubKey)` Methode der `ClientCertManager` Klasse erstellt. Dazu sendet die Android Anwendung den öffentlichen Schlüssel eines zuvor generierten Schlüsselpaares. Die Web Anwendung kann dem Gerät hier nur aufgrund der gesendeten Parameter vertrauen. Das bedeutet zu diesem Zeitpunkt könnte ein Angreifer, mit Hilfe einer Man-in-the-Middle Attacke, das generierte Client Zertifikat abfangen und es benutzen, um falsche Gerätedaten zu senden.
  Im Anschluss an den Empfang des Schlüssels und der Generierung des Client Zertifikats wird das Server Zertifikat geladen, um mit dessen privaten Schlüssel das neue Client Zertifikat zu signieren. Die Werte der Parameter für Aussteller, Inhaber und Gültigkeitsdauer werden ebenfalls aus dem Server Zertifikat übernommen. Die Web Anwendung speichert eine Kopie des erstellten Zertifikats in einem eigenen Keystore für die Client Zertifikate mit der IMEI+AndroidID des Client Gerätes als Alias.

- `/devices/deviceconfiguration.xhtml`, `DeviceConfigurationServlet`
  Hat die Android Anwendung erfolgreich ein Zertifikat erhalten, ruft sie zum Abschluss der Registrierung noch die aktuelle Version der Anwendungskonfiguration von der Web Anwendung ab.

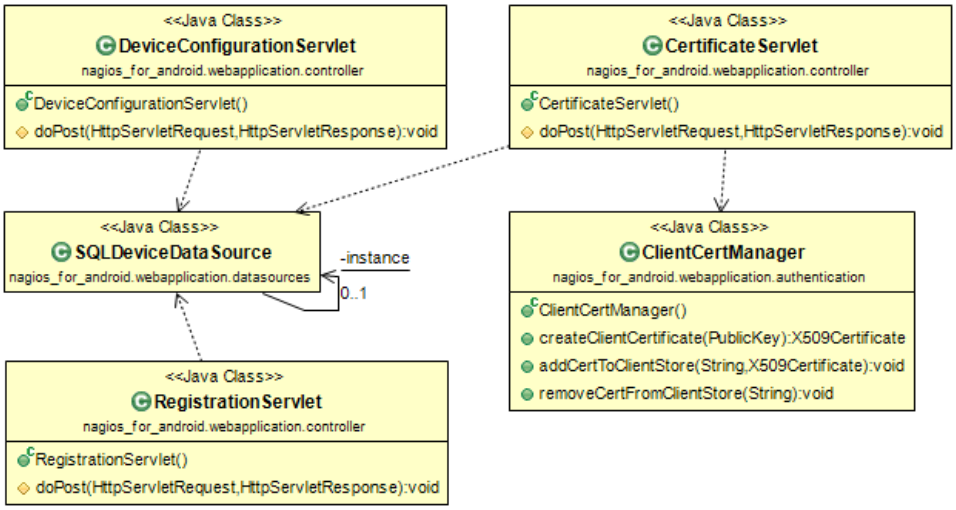

Abbildung 15: Registrierung eines Gerätes

4.2.6.2 Gerätedaten

Das `DeviceInformationServlet` prüft zu Beginn, ob die Anfrage die notwendigen Parameter enthält und ob ein Eintrag in der Whitelist für das entsprechende Gerät existiert. Abbildung 16 zeigt die `CommandFileOutputWriterConnection` Klasse die, für die Übertragung der empfangenen Daten an die Java Anwendung verwendet wird. Um die Netzwerkkommunikation zwischen Android- und Web Anwendung währenddessen nicht zu blockieren, implementiert `CommandFileOutputWriterConnection` das `java.lang.Runnable` Interface, um die Übertragung in einem eigenen Thread auszuführen. Je nachdem, ob die Verbindung zur Java Anwendung verschlüsselt oder unverschlüsselt ist, wird entweder ein `javax.net.ssl.SSLSocket` oder ein `java.net.Socket` von der `CommandFileOutputWriterConnection` Klasse verwendet.

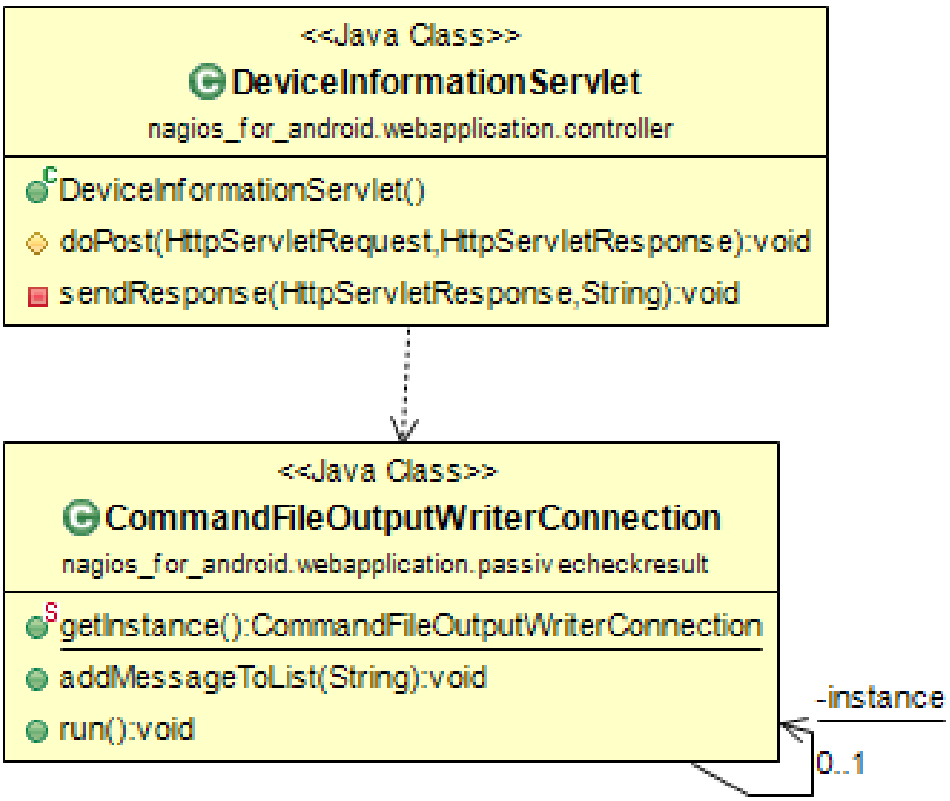

Abbildung 16: Empfang und Weiterleitung von Gerätedaten

4.3 Java Anwendung

Die Java Anwendung empfängt die passiven Check Resultate über das Netzwerk von der Web Anwendung und schreibt sie in die „Externe Kommando“ Datei von Nagios.

4.3.1 Aufbau

Abbildung 17 zeigt die einzelnen Klassen der Java Anwendung. Die Anwendung wird über die Kommandozeile gestartet und mit Hilfe von Kommandozeilenparametern konfiguriert. Die Parameter werden in der `main` Methode der `ApplicationMainClass` Klasse überprüft.

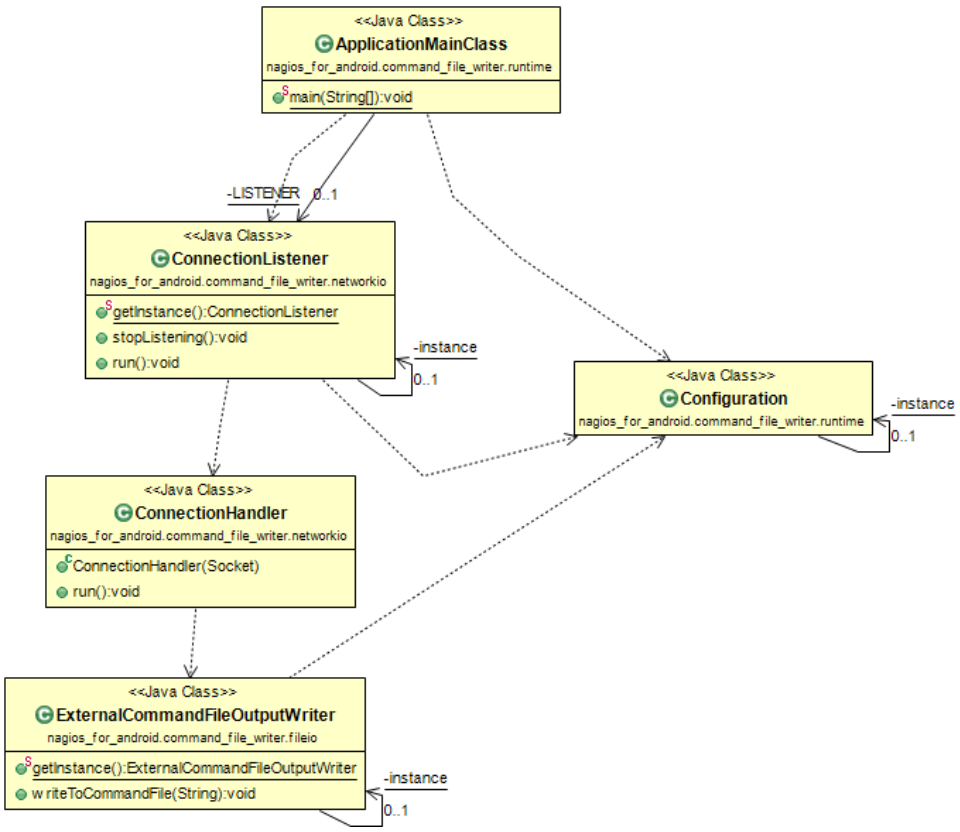

Abbildung 17: Klassendiagramm der Java Anwendung

4.3.2 Empfang von Daten

Tritt während der Parameterüberprüfung kein Fehler auf, wird ein `ConnectionListener` erstellt, der in einem neuen Thread auf Verbindungsversuche der Web Anwendung wartet. Bei einem erfolgreichen Verbindungsaufbau führt eine Instanz der `ConnectionHandler` Klasse die Netzwerkinteraktion mit der Web Anwendung durch. Die empfangenen Daten enthalten ein passives Check Resultat in Form eines Nagios Kommandos und werden beim Aufruf der statischen `writeToCommandFile(final String commandsToWrite)` Methode der `ExternalCommandFileOutputWriter` Klasse als Parameter übergeben. Das Kommando wird hier noch um einen Unix Zeitstempel am Anfang der Zeichenkette ergänzt. Das folgende Listing 7 zeigt ein Beispiel für ein passives Check Resultat.

```
[1411038074] PROCESS_HOST_CHECK_RESULT;AndroidAppApi19;0;
Device location (Mi., 29 Okt 2014 10:05:33 MEZ):
```

```
DeviceInsideRectangle false/LocationProvider gps/LocationAccuracy
    104.0,
ADBState: ADB disabled,
Applications: WhitelistedApplications /, BlacklistedApplications ,
Battery: BatteryLevel 32%/(2014-10-28 12:58:04: Intermediate
    Value: 100%)/ChargingStatus Discharging/BatteryTemperature
    18.0°C,
DateAndTime: 29. Oktober 2014 10:04:57 MEZ,
AndroidVersion: Android 4.0.4 15,
WiFi: WiFi Enabled/SSID "jCGU0DsmXCU8geUIHWrCiq8b9"/
    KeyManagementAlgorithm WPA PSK/
PairCipher CCMP AES/GroupCipher CCMP AES,
Device rooted: false/Application root access: false
```

Listing 7: Beispiel für passives Check Resultat

Der Zeitstempel wird erst jetzt vor dem endgültigen Schreibvorgang ergänzt um sicherzustellen, dass die Java Anwendung und Nagios mit derselben Betriebssystemzeit arbeiten und die korrekte Zeit im Nagios Statusfenster angezeigt wird. Da mehrere Prozesse schreibend auf die „Externe Kommando“ Datei zugreifen können, versucht der `ExternalCommandFileOutputWriter` sie vor dem Schreibvorgang zu sperren um exklusiven Schreibzugriff zu erhalten. Dies wird solange in zufälligen Zeitabständen von maximal fünf Sekunden wiederholt bis die Datei erfolgreich gesperrt werden kann. Erst im Anschluss werden die Resultate von der `ExternalCommandFileOutputWriter` Klasse in die „Externe Kommando“ Datei geschrieben.

5 Bedienungsanleitung

5.1 Android Anwendung

Die Android Anwendung wird ausschließlich über die drei Seiten der Benutzeroberfläche bedient. Sie muss vor der Verwendung auf dem Gerät installiert werden.

5.1.1 Installation

Es gibt verschiedene Wege um die Android Anwendung zu Installieren. Falls sie auf Google Play [17] veröffentlicht ist, kann sie einfach von dort heruntergeladen werden. Eine weitere Möglichkeit ist die Installation durch ein

[17] https://play.google.com/store?hl=de

Android Application Package (APK) in das eine Anwendung verpackt werden kann. Dazu muss die APK Datei, zum Beispiel mittels E-Mail, von einer Internetseite, über ein USB-Kabel oder auf der SD-Karte, auf das Android Gerät übertragen und von dort installiert werden. Voraussetzung dafür ist die Erlaubnis Anwendungen aus unbekannten Quellen zu installieren [Project(2013k)].

5.1.2 Startseite

Die Startseite der Android Anwendung ermöglicht einem Benutzer die Durchführung von Registrierungsvorgängen und die Steuerung der automatischen Registrierung. Abbildung 18 zeigt einen Screenshot der Startseite.

- „Register“ Schaltfläche
  Drückt der Benutzer diese Schaltfläche wird ein Registrierungsvorgang gestartet und eine Anfrage an die Web Anwendung gesendet. Ist die Startseite noch aktiv und im Vordergrund beim Empfang der Antwort, wird das „Status:“ Textfeld mit dem neuen Registrierungsstatus aktualisiert. Bevor eine Registrierung durchgeführt werden kann, muss die URL der Web Anwendung in den Einstellungen festgelegt werden.

- „Status:“ Textfeld
  Zeigt den lokalen Registrierungsstatus der Android Anwendung an. Allerdings entspricht er dem Status der letzten Registrierungsanfrage und kann sich vom wirklichen Status auf der Web Anwendung unterscheiden.

- „Last check:“ Textfeld
  Dieses Textfeld zeigt Uhrzeit und Datum, an dem zuletzt eine Registrierungsanfrage an die Web Anwendung gesendet wurde.

- „Automation“ Schalter
  Mit diesem Schalter kann der Benutzer das automatische Durchführen der Registrierung steuern. Die Anwendung muss sich dabei nicht aktiv im Vordergrund befinden. Nach der Installation der Anwendung ist der Schalter standartmäßig auf „AUS“ gestellt.

- „Interval (Hours, Minutes)“ Auswahl
  Diese Nummernauswahl stellt die Verzögerung zwischen zwei automa-

tischen Registrierungsvorgängen ein. Die maximal einstellbare Verzögerung beträgt Dreiundzwanzig Stunden und Neunundfünfzig Minuten. Diese Begrenzung entsteht durch das verwendete Benutzeroberflächenelement. Die Grundeinstellung für das Interval beträgt fünf Minute.

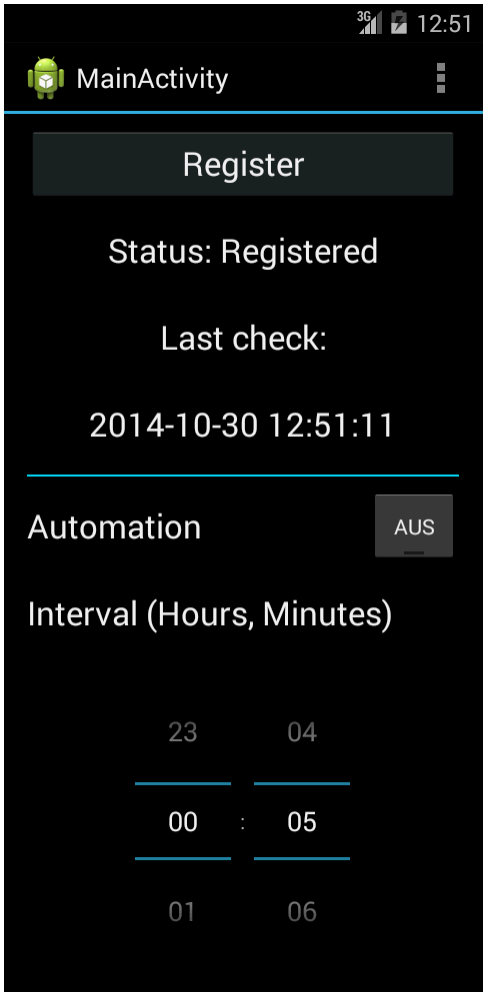

Abbildung 18: Startseite der Android Anwendung

5.1.3 Einstellungen

Diese Seite enthält einige notwendige Einstellungen, die für die Registrierung und die Ausführung der Anwendung notwendig sind. Abbildung 19 zeigt einen Screenshot der Einstellungsseite.

- „Download configuration“ Schaltfläche
  Mit dieser Schaltfläche kann ein Benutzer die Version der lokalen Konfiguration mit der Version der Web Anwendung vergleichen und falls nötig eine aktuellere herunterladen und anwenden.

- „Last check:“ Textfeld
  Dieses Textfeld zeigt Uhrzeit und Datum des letzten erfolgreichen Konfigurationsvergleiches an.

- „Local version:“ Textfeld
  Dieses Textfeld zeigt Uhrzeit und Datum an denen zuletzt eine Konfiguration heruntergeladen und angewendet wurde.

- „Device alias“ Eingabefeld
  Mit diesem Eingabefeld kann dem Gerät und der Android Anwendung ein Name zugewiesen werden. Dieser Name wird in der Nagios Konfiguration als Hostname verwendet um das Gerät zu identifizieren und muss daher einzigartig sein.

- „Web server address“ Eingabefeld
  In diesem Eingabefeld gibt der Benutzer die URL an, unter der die Web Anwendung erreichbar ist.

- „Import server certificate“ Schaltfläche
  Für den Fall, dass die Web Anwendung ein selbst-signiertes Zertifikat verwendet, kann die Android Anwendung keinen Vertrauenspfad herstellen und die Herstellung einer gesicherten SSL Verbindung schlägt fehl. Um das Problem zu lösen, muss das Server Zertifikat manuell in die Liste der vertrauenswürdigen Zertifikate aufgenommen werden. Android verfügt seit Version 1.6 über einen System-eigenen Keystore [Project(2013a)], dieser kann allerdings erst seit Version 4.0 durch Anwendungen verwendet werden. Für die Kompatibilität mit den früheren Versionen verwaltet die Android Anwendung einen privaten Keystore in ihren Anwendungsdaten. Um ein Server Zertifikat zu importieren, legt man es als *.cer Datei auf der SD-Karte des Gerätes ab. Die Anwendung durchsucht die SD-Karte und listet alle gefundenen *.cer Dateien als Einträge in der Dropdown-Liste über der Schaltfläche auf. Der Benutzer kann das entsprechende Zertifikat auswählen und durch drücken der Schaltfläche importieren.

- „Reset to default“ Schaltfläche
  Mit dieser Schaltfläche kann ein Benutzer die Einträge in den lokalen Anwendungsdaten löschen und die Anwendung auf den Ausgangszustand zurückführen. Vor dem Löschen wird der Benutzer in einem Dialog noch um seine Bestätigung gefragt.

5.1.4 Gerätedaten

Die Seite für Gerätedaten ermöglicht die Durchführung von Sammelzyklen und die Anzeige der gesammelten Daten. Abbildung 20 zeigt einen Screenshot der Seite.

- „Collect local“ Schaltfläche
  Mit dieser Schaltfläche löst der Benutzer einen lokalen Sammelzyklus aus. Die gesammelten Daten werden mit dem letzten vollständigen Zyklus verglichen und falls die Seite noch aktiv im Vordergrund ist, angezeigt.

- „Collect and send“ Schaltfläche
  Diese Schaltfläche startet einen vollständigen Sammelzyklus. Dies beinhaltet das Sammeln und Senden der Daten sowie dem Überprüfen der Konfigurationsversion. Falls die Seite aktiv im Vordergrund ist, werden die gesendeten Daten angezeigt.

- „Last upload:“ Textfeld
  Dieses Textfeld zeigt Uhrzeit und Datum des letzten erfolgreichen vollständigen Sammelzyklus an.

- „Automation“ Schalter
  Mit diesem Schalter kann die automatische Durchführung von Sammelzyklen gesteuert werden. Der Zeitplan für die Automatik wird in der Anwendungskonfiguration festgelegt. Die Automatik ist standardmäßig deaktiviert.

- „Device data:“ Textfeld
  Dieses Textfeld wird verwendet um die gesammelten Daten anzuzeigen.

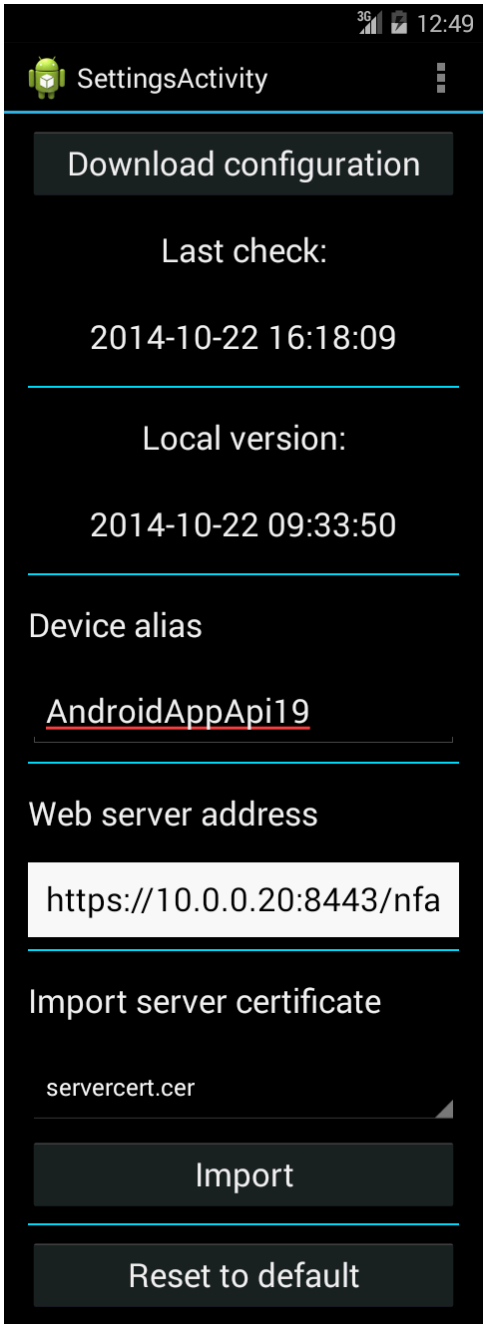

Abbildung 19: Einstellungen der Android Anwendung

5.1.5 Konfiguration

Listing 8 zeigt ein Beispiel für eine vollständige Konfiguration der Android Anwendung mit allen verfügbaren Elementen.

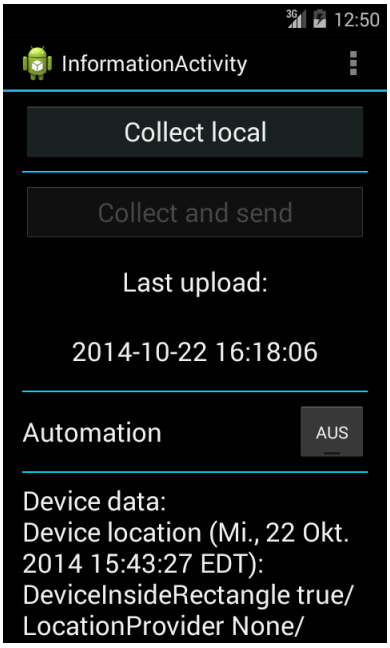

Abbildung 20: Seite für Geräteinformationen

```
<configuration>
  <webapplication url="https://10.0.2.2:8443/nfam" />
  <sendingschedule monday="0800-2200*2M/1" tuesday="0800-2200*2M/1"
       wednesday="0800-2200*2M/1" thursday="0800-2200*2M/1" friday=
      "0800-2200*2M/1" saturday="0800-2200*2M/1" sunday="
      0800-2200*2M/1" />
  <battery batterypositivedeviation="30" batterynegativedeviation="
      30" />
  <bluetooth />
  <location mode="absolute" resolveaddress="true" originlangitude="
      7.0" originlatitude="47.0" absolutemodemaximumdistance="2000"
      />
  <wifi reportintermediateunsafe="true"/>
  <dataconnection />
  <applications blacklist="" whitelist=""/>
  <temperature positivedeviation="20" negativedeviation="20" />
  <usbdebugging />
  <timedate />
  <androidversion />
  <rootavailable />
</configuration>
```

Listing 8: Beispiel für die Konfiguration der Android Anwendung

- Web Anwendung
  Die URL der Web Anwendung kann direkt in den Einstellungen der Android Anwendung oder in der XML Konfiguration über das `url` Attribut im `<webapplication>` Element festgelegt werden. Das Aktualisieren der Konfiguration findet am Ende eines vollständigen Sammelzyklus statt, eine neue URL wird daher erst für die nächste HTTP Anfrage verwendet.

- Sammel- und Sendezeitplan
  Mit dem `<sendingschedule>` Element wird der Zeitplan für die automatische Durchführung der Sammelzyklen festgelegt. Jeder Wochentag kann einzeln über das entsprechende Attribut konfiguriert werden. Wird für einen Tag kein Attribut angegeben, finden an diesem keine Sammelzyklen statt. Die Attributwerte folgen einem festgelegten Format. Der Wert `0800-2200` bedeutet, dass an diesem Tag von Acht bis Zweiundzwanzig Uhr Daten gesammelt werden sollen. `*2M` legt fest, dass alle Zwei Minuten ein Sammelzyklus durchgeführt wird. Wird statt einem „M“ ein „H“ verwendet, ist die Verzögerung in Stunden. Der letzte Teilt des Attributwertes `/1` gibt an wie viele lokale Sammelzyklen zwischen zwei vollständigen durchgeführt werden. Das bedeutet hier das effektiv alle vier Minuten ein vollständiger Sammelzyklus stattfindet.

- Bluetooth, Datenverbindung, USB-Debugging, Zeit und Datum, Android Version, Rootstatus
  Die Elemente `<bluetooth/>`, `<dataconnection/>`, `<usbdebugging/>`, `<timedate/>`, `<androidversion/>` und `<rootavailable/>` enthalten keine zusätzlichen Attribute. Sie teilen der Android Anwendung nur mit, dass diese Werte gesammelt und gesendet werden sollen.

- Standort
  Für die Sammlung der Standortinformationen sind zwei verschiedene Modi verfügbar, die mit dem Attribut `mode` im `<location/>` Element ausgewählt werden. Listing 8 zeigt eine Konfiguration für den Absoluten Modus. Dieser Modus misst die Entfernung des aktuellen Standortes vom, in der Konfiguration angegebenen, Ausgangspunkt in Metern. Zusätzlich wird die Richtung vom Ausgangspunkt zum aktu-

ellen Standort als Winkel in Grad von Norden als Null Grad ausgehend angegeben. Mit den `originlongitude` und `originlatitude` werden Längen- und Breitengrad des Punktes angegeben, von dem aus Entfernung und Richtung berechnet wird. Während eines lokalen Sammelzyklus wird der Wert des `absolutemodemaximumdistance` Attributes als maximal erlaubte Entfernung verwendet. Wird der Wert überschritten, wird den gesendeten Informationen beim nächsten vollständigen Sammelzyklus eine entsprechende Meldung hinzugefügt.
Listing 9 zeigt das `<location/>` Element für den Rechteck Modus. Auch hier wird ein Längen- und Breitengrad als Ausgangspunkt angegeben. Die Attribute `rectangleEndLatitude` und `rectangleEnd Longitude` legen den Endpunkt des Rechtecks fest. Hat das Gerät während eines lokalen Sammelzyklus das Rechteck verlassen wird mit dem `reportrectangleleaving` Attribut bestimmt ob eine Meldung gespeichert und in den Informationen des nächsten vollständigen Sammelzyklus gesendet werden soll.

```
<location mode="rectangle" originlongitude="7.0"
    originlatitude="47.0" rectangleEndLatitude="48.0"
    rectangleEndLongitude="8.0" reportrectangleleaving="true
    " />
```

Listing 9: Beispiel für Standortdaten in Rechteckmodus

- Wi-Fi
  Ist das `<wifi>` Element vorhanden, sammelt die Anwendung Informationen über die Netzwerkkonfiguration falls das Gerät mit einem Wi-Fi Netzwerk verbunden ist. Ist zum Zeitpunkt eines lokalen Sammelzyklus eine Verbindung zu einem unsicheren (WEP, Unverschlüsselt) Wi-Fi Netzwerk vorhanden, legt das `reportintermediateunsafe` Attribut fest, ob eine Benachrichtigung über die Verbindung zu den Informationen im nächsten vollständigen Sammelzyklus hinzugefügt werden soll.

- Anwendungen
  Die Konfiguration legt fest ob ein installiertes Paket in den gesendeten Informationen gemeldet wird. Das `<applications/>` Element enthält jeweils ein Attribut für die White- und Blacklist. Mit dem Attribut `blacklist` kann eine Liste von unerwünschte Anwendun-

gen festgelegt werden, die bei Vorhandensein gemeldet werden. Die Trennung zwischen den Paketnamen erfolgt durch ein „;“ Zeichen. Das `whitelist` Attribut gibt die Liste an Anwendungen an, die auf dem Gerät vorhanden sein müssen und deren Fehlen gemeldet wird.

- Temperatur
  Ist das `<temperature/>` Element in der Konfiguration enthalten, werden Abfragen des Temperatursensors als Teil der Sammelzyklen aktiviert. Ein vollständiger Sammelzyklus enthält den Temperaturwert in Grad Celsius, wenn das Gerät über einen Temperatursensor verfügt. Die Attribute `positivedeviation` und `negativedeviation` legen eine maximale Positive und Negative Abweichung fest. Ist die Abweichung der Temperatur während eines lokalen Sammelzyklus, im Vergleich zum Wert des letzten vollständigen Sammelzyklus, größer als einer dieser Maximalwerte, wird den Informationen eine entsprechende Meldung hinzugefügt.

- Batterie
  Das `<battery/>` Element aktiviert die Sammlung des Batterielevels in Prozent und des Ladestatus (Laden, Entladen). Wie beim Temperaturwert können mit Hilfe der `batterypositivedeviation` und `batterynegativedeviation` Attribute eine maximale Abweichung die für die Vergleiche bei lokalen Sammelzyklen definiert werden. Auch hier wird eine Meldung zu den Informationen hinzugefügt falls einer der Maximalwerte überschritten wird.

5.2 Web Anwendung

Die Benutzeroberfläche der Web Anwendung ermöglicht es die, in der Datenbank gespeicherten Gerätelisten anzuzeigen und diese zu bearbeiten.

5.2.1 Installation

Die Web Anwendung benötigt für ihre Ausführung eine Datenbankverbindung und den Pfad zu ihrer Konfigurationsdatei. Beides wird hier im Anwendungseigenen Context definiert. Dazu wird im `WebContent/META-INF/` Verzeichnis eine `context.xml` Datei erstellt, in der Umgebungsvariablen

und Ressourcen definiert werden können. Ein Beispiel eines Context ist hier 10 zu sehen.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <Resource auth="Container" driverClassName="com.mysql.jdbc.Driver
      " maxActive="15" maxIdle="3" name="jdbc/nagiosforandroid"
      password="<Passwort>" type="javax.sql.DataSource" url="
      jdbc:mysql://localhost:3306/nagios_for_android" username="
      NagiosForAndroid" />

  <Environment name="WEB_APP_CONFIGURATION_FILE" value="<Pfad>\
      WebApplicationConfiguration.xml" type="java.lang.String" />
</Context>
```

Listing 10: Beispiel einer Context Definition.

Die Datenbankverbindung wird als Ressource mit Hilfe des `<Resource>` Elements definiert. Die Attribute enthalten alle benötigten Parameter, die für eine Herstellung einer Datenbankverbindung nötig sind, wie die Datenbank URI, den Benutzernamen und das Password. Der Vorteil ist, dass die Verbindung durch den verwendeten Apache Tomcat Server[18] verwaltet wird und ein Wechsel auf eine andere Datenbank nur eine Änderung der Attributwerte erfordert. Der Programmcode der Web Anwendung selbst muss nicht geändert werden.
Mit dem `<Environment>` Element kann eine Umgebungsvariable für die Web Anwendung definiert werden. Hier enthält die Variable im Attribut `value` den Pfad der Web Anwendungskonfiguration. Die Attribute `name` und `type` definieren den Namen, unter dem der Wert abgerufen werden kann, und den Java Typ. Mehr Informationen über die Definition von Resourcen und Variablen im Context findet sich unter [19].
Die Web Anwendung verwendet SSL um die Kommunikation mit der Android Anwendung zu verschlüsseln. Um SSL zu aktivieren, wird ein Server Zertifikat und die entsprechenden Änderungen in der Server Konfiguration benötigt. Standardmäßig ist nur der unverschlüsselte HTTP Connector auf Port 8080 aktiviert. Der SSL Connector auf Port 8443 ist in der `server.xml` Konfigurationsdatei als Kommentar eingetragen und somit deaktiviert. Listing 11 zeigt das `<Connector />` Element mit entfernten

[18] http://tomcat.apache.org/
[19] http://tomcat.apache.org/tomcat-7.0-doc/jndi-resources-howto.html

Kommentarzeichen. Die einzigen Änderungen gegenüber der Standartkonfiguration sind die `keystoreFile`, `keystorePass`, `truststoreFile` und `truststorePass` Attribute. Sie definieren den Pfad und das Passwort für den Keystore der die Server Zertifikate enthält und den Truststore mit den vertrauenswürdigen Client Zertifikaten. Weitere Informationen zur Konfiguration von HTTP Connectoren und den einzelnen Parametern findet man unter [Foundation(2014)].

```
<Connector SSLEnabled="true" clientAuth="false"
   keystoreFile="<Pfad>/servercert.jks"
   keystorePass="<Passwort>" maxThreads="150" port="8443" protocol
       ="org.apache.coyote.http11.Http11NioProtocol"
   scheme="https" secure="true" sslProtocol="TLS"
   truststoreFile="<Pfad>/servertrust.jks"
   truststorePass="<Passwort>" />
```

Listing 11: Tomcat SSL Konfiguration

5.2.2 Konfiguration

Listing 12 zeigt ein Beispiel für eine Konfigurationsdatei der Web Anwendung.

```
<configuration>
  <androidappconfigurationfile
   configurationfile="<Pfad>\AndroidApplicationConfiguration.xml"
       />
  <keystorefile
   keystorepath="<Pfad>/servercert.jks" keystorepassword="<
       Passwort>"
   certalias="nagiosforandroid.ddns.net" truststorepath="<Pfad>/
       servertrust.jks"
   truststorepassword="<Passwort>" clientstorepath="<Pfad>/
       clientstore.jks" clientstorepassword="<Passwort>" />
  <nagioscommandwriter writeripaddress="<IP Adresse NagiosHost>"
      port="<Portnummer>" useSSLConnection="true" />
  <scheduleCheck delay="1M" maxMissed="5" />
</configuration>
```

Listing 12: Beispiel für die Konfiguration der Web Anwendung

Die vier Elemente unter der Wurzel `<configuration>` enthalten die, für die Ausführung, benötigten Werte in ihren Attributen.

- Konfigurationsdatei für die Android Anwendung
  Die Web Anwendung stellt die XML Konfiguration für die Android Anwendung zur Verfügung. Im Element `<androidappconfiguration file>` gibt man den Pfad zur XML Datei als Wert des Attributes `configurationfile` an.

- Keystorepfade
  Die Web Anwendung generiert für jede Android Anwendung im Laufe einer erfolgreichen Registrierung ein Client Zertifikat. Um vom Server als vertrauenswürdig angesehen zu werden, wird jedes Client Zertifikat mit dem privaten Schlüssel des Server Zertifikates signiert. Im `<keystorefiles>` Element der Konfiguration werden mit den Attributen `keystorepath`, `keystorepassword` und `certalias` Pfad und Passwort des Keystores und der Alias unter dem das Server Zertifikat gespeichert ist, angegeben. Die Attribute `truststorepath` und `truststorepassword` enthalten Pfad und Passwort für den Keystore mit den Trusted Certificate Einträgen, denen bei Herstellung einer SSL Verbindung vertraut wird. Er enthält zum Beispiel einen Eintrag für das Server Zertifikat der Java Anwendung die auf dem Nagios Host ausgeführt wird. Die Attribute `clientstorepath` `clientstorepassword` enthalten Pfad und Passwort für den Keystore, der die Kopien der erstellten Client Zertifikate enthält. Als Alias für die Einträge wird die IMEI+AndroidID Kombination des jeweiligen Gerätes verwendet.

- Verbindung zur Java Anwendung
  Die Java Anwendung ermöglicht die Trennung der Web Anwendung vom Nagios Host. Die Kommunikation zwischen beiden Anwendungen erfolgt über Socket Verbindungen, deren Parameter im `<nagioscommandwriter>` Element festgelegt werden. Die Attribute `writeripaddress` und `port` enthalten die IP-Adresse des Nagios Hosts und den Port, auf dem die Java Anwendungen auf Verbindungen wartet. `useSSLConnection` legt fest ob die Verbindungen verschlüsselt oder unverschlüsselt sein sollen. Die hier zulässigen Attributwerte sind „false“ und „true“.

- Zeitplanüberprüfung
  Für alle Geräte auf der Whitelist wird regelmäßig kontrolliert ob sie

dem in der Android Anwendungskonfiguration festgelegten Zeitplan für das Sammeln und Senden der Gerätedaten folgen. Im `<scheduleCheckDelay>` Element werden die Optionen für diese Überprüfung festgelegt. Der Wert des `delay` Attributes bestimmt die Verzögerung zwischen den Überprüfungen. Der Attributwert muss aus einer Zahl gefolgt von entweder „M“ für Minuten oder „H“ für Stunden bestehen. Es kann verschiedene Gründe geben, warum ein Gerät keine Daten mehr sendet. Mit dem `maxMissed` Attribut wird festgelegt, wie oft ein Gerät auf der Whitelist es versäumen darf seine Daten zu senden, bevor sein Status in Nagios auf „DOWN“ geändert wird.

5.3 Java Anwendung

Um die passiven Check Resultate in die „Externe Kommando“ Datei zu schreiben, wird die Java Anwendung auf demselben Host wie Nagios ausgeführt. Sie wird über die Kommandozeile gestartet und mit Hilfe von Kommandozeilenparametern konfiguriert.

5.3.1 Initialisierung

Die Java Anwendung akzeptiert entweder zwei oder sechs Kommandozeilenparameter. Die folgenden beiden Parameter sind für die Ausführung notwendig und müssen immer definiert sein.

- -commandfile=<Pfad>
  Mit diesem Parameter wird der Anwendung der Pfad zur externen Kommando Datei von Nagios mitgeteilt. Hier muss sichergestellt sein, dass die Java Anwendung die erforderlichen Berechtigungen besitzt, um in die Datei zu schreiben.

- -clientport=<Portnummer>
  Der angegebene Port wird verwendet, um auf Verbindungen der Web Anwendung zu warten. Ist es nicht möglich einen `ServerSocket` für den verwendeten Port zu erstellen, beendet sich die Anwendung selbst.

Werden nur diese beiden Parameter angegeben, erstellt die Anwendung einen normalen `ServerSocket` und die Verbindungen mit der Web Anwendung sind unverschlüsselt. Durch die zusätzliche Angabe aller folgenden vier Pa-

rameter wird stattdessen ein `SSLServerSocket` verwendet und Verbindungen mit der Web Anwendung sind verschlüsselt.

- -keystore=<Pfad>
  Dieser Parameter enthält den Pfad des Keystore der, das von der Java Anwendung und ihrem `SSLServerSocket` verwendete Server Zertifikat enthält.

- -keystorepassword=<Passwort>
  Der nächste Parameter teilt der Anwendung das Passwort für den Keystore mit.

- -truststore=<Pfad>
  Neben dem Keystore muss auch der Pfad zu einem Truststore angegeben werden, der einen Eintrag für das Zertifikat der Web Anwendung enthält, dem bei einem Verbindungsversuch vertraut wird.

- -truststorepassword=<Passwort>
  Der letzte Parameter enthält das Passwort für den Truststore.

5.4 Nagios

Die Installation von Nagios und der erforderlichen Pakete und Bibliotheken erfolgt über die Paketverwaltung der verwendeten Linux Distribution.

5.4.1 Konfiguration

Die Konfiguration von Nagios wird über Konfigurationsdateien erledigt. Um Passive Checks zu aktivieren, sind Änderungen in der Hauptkonfigurationsdatei nötig. Android Geräte werden durch Einträge in den Host Dateien hinzugefügt.

5.4.1.1 Hauptkonfigurationsdatei

Die passiven Check Ergebnisse werden in die „Externe Kommando“ Datei geschrieben. Um sicherzustellen, dass Nagios diese Kommandos verarbeitet, muss die Option `check_external_commands=1` in der Konfigurationsdatei angegeben werden. Der Standort der „Externen Kommando“ Datei wird mit Hilfe der Anweisung `command_file=/var/lib/nagios3/`

rw/nagios.cmd festgelegt. Die Datei wird jedes mal gelöscht, wenn Nagios beendet wird und beim Start wieder neu erstellt. Deswegen ist hier zu beachten, dass eine Anwendung, die Kommandos an Nagios übermitteln möchte, die entsprechenden Schreibrechte bereits auf Verzeichnisebene besitzen muss. Die Informationen aus den Abfragen auf den Android Geräten werden als passive Check Resultate übermittelt, jedoch muss Nagios angewiesen werden, diese zu akzeptieren. Dies geschieht über die Anweisung accept_passive_service_checks=1. Mit der Option command_check_interval=30s kann festgelegt werden in welchem Zeitabstand Nagios die „Externe Kommando“ Datei überprüft. Ein Kommando wird bis zu seiner Verarbeitung in einen Buffer geschrieben. Mit external_command_buffer_slots=4096 kann die Buffergröße festgelegt werden.

5.4.1.2 Hinzufügen von Android Hosts

Bevor Nagios die Informationen eines Gerätes anzeigen kann, muss es in die Liste der Hosts aufgenommen werden. Diese Host Definitionen werden in Dateien mit der *.cfg Erweiterung geschrieben. Für eine einfachere Bearbeitung und mehr Übersicht können sie auf mehrere Dateien aufgeteilt werden. In der Hauptkonfigurationsdatei können mithilfe von cfg_file= und cfg_dir= diese einzelnen Dateien oder komplette Verzeichnisse die, *.cfg Dateien enthalten, angegeben werden. Das folgende Beispiel 13 zeigt eine Definition für ein Android Gerät in einer Host Konfigurationsdatei.

```
#Host definition for Android device

define host{
host_name        Android demo device
alias          Nagios Emulator Device #1
address         192.168.56.101
max_check_attempts   5
active_checks_enabled  0
passive_checks_enabled 1
check_period      24x7
contacts        root
contact_groups     admins
notification_interval  30
notification_period   24x7
}
```

Listing 13: Host Definition für Android Gerät.

Eine komplette Auflistung der möglichen Parameter findet man unter [Enterprises(2014)]. Wichtig hier sind vor allem die Parameter `host_name` und `passive_checks_enabled`.

- host_name
  Der `host_name` dient als kurzer Name für die eindeutige Identifizierung eines Host in Nagios. Ist eine Namensauflösung über DNS möglich kann ein FQDN angegeben werden um den Host zu adressieren. Da der Status eines Android Gerätes nur über passive Checks aktualisiert wird, muss dieser Wert mit dem Namen, der in den Einstellungen der Android Anwendung für das Gerät vergeben wurde übereinstimmen.

- alias
  `alias` erlaubt die Definition eines längeren Namens für einen Host um ihn leichter von anderen Hosts unterscheiden zu können.

- passive_checks_enabled
  Mit der `passive_checks_enabled 1` Option wird die Aktualisierung des Host durch passive Checks aktiviert. Da keine aktiven Checks für die Android Hosts zu erwarten sind, werden diese durch die `active_checks_enabled 0` Option deaktiviert.

- Adresse
  Der `address` Parameter enthält hier die IP-Adresse der Web Anwendung und nicht die Adresse des Host selbst, da der Host selbst nicht durch aktive Checks geprüft wird. Der Parameter erfüllt hier bei rein passiven Checks keine Funktion, er muss aber in einer Host Definition angegeben werden und kann daher nicht weggelassen werden.

6 Evaluierung

6.1 Test des Systems

Die Komponenten können einzeln nur sehr begrenzt getestet werden. Deswegen werden alle Tests mit einem kompletten System durchgeführt.

6.1.1 Testumgebung

6.1.1.1 Android Anwendung

Die Android Anwendung ist kompatibel mit Android Version 2.2 und höher. Aus diesem Grund sind Tests mit verschiedenen Versionen nötig, um die Kompatibilität zu prüfen. Für diese Tests werden zwei Virtuelle Geräte, die mit Hilfe der Entwicklungswerkzeuge erstellt werden, verwendet. Ein virtuelles Gerät verwendet Android Version 2.2 und das Zweite die derzeit aktuellste Version 4.4.2. Zusätzlich wird noch ein Sony Ericsson WT19i mit der Android Version 4.04 verwendet um die Tests auf einem realen Gerät durchzuführen.

6.1.1.2 Web Anwendung

Für die Entwicklung und Tests der Web Anwendung wird ein Apache Tomcat Server in der Version 7.0.54 verwendet. Ausgeführt wird der Server auf einem Windows 8.1 Host mit Java Version 8 Update 5, beides in der 64 Bit Variante. Als Datenbanksystem für die Web Anwendung kommt der MySQL Community Server[20] in der Version 5.6 zum Einsatz.

6.1.1.3 Java Anwendung und Nagios

Für die Ausführung von Nagios wird Linux Mint 15 Olivia in der Cinnamon 32-bit Edition verwendet [21]. Die Distribution ist als virtuelle Maschine in der Oracle VM VirtualBox Version 4.3.12 installiert. Für Nagios wird die kostenlose Core Edition Version 3.4.1 verwendet.

6.1.2 Durchgeführte Tests

6.1.2.1 Setup

Damit die Anwendungen einwandfrei arbeiten können, müssen die Parameter aus XML Konfigurationsdateien und Kommandozeilenparameter korrekt eingelesen werden. Daher wird getestet, ob alle Parameter erfasst und den entsprechenden Variablen zugewiesen werden. Für die Android Anwendung werden zusätzlich die Optionen der lokalen Einstellungen, der Server Zertifikat Import und das Zurücksetzen der privaten Anwendungsdaten getestet.

[20] http://dev.mysql.com/downloads/ (5.11.2014)
[21] http://www.linuxmint.com/index.php (5.11.2014)

Die Konfiguration selbst wird durch eine Anfrage an die Web Anwendung auf Änderungen überprüft oder heruntergeladen. Beides wird getestet sobald die Android Geräte in der Web Anwendung registriert sind.

6.1.2.2 Registrierung

Während er Registrierung wird eine neue Android Anwendung und ihr Gerät in eine der Listen der Web Anwendung aufgenommen. Zusätzlich wird noch die Erstellung und das Herunterladen eines Client Zertifikates getestet. Für die Registrierung sind die folgenden Testfälle definiert.

- Es sind noch keine Geräte registriert und die drei Android Anwendungen senden eine Anfrage. Im Anschluss muss für jedes Gerät ein Eintrag in der Liste der unentschiedenen Anfragen vorhanden sein.
- Die Geräte befinden sich bereits auf der Liste noch unentschiedener Anfragen. Die Web Anwendung soll hier nur die Zeitpunkte der letzten Verbindungsaufnahme für das jeweilige Gerät aktualisieren.
- Die Geräte befinden sich auf der Blacklist. Sendet die Android Anwendung die Registrierungsanfragen mit Hilfe der Automatik, muss diese hier als Ergebnis der Antwort der Web Anwendung ausgeschaltet werden.
- Die Geräte befinden sich auf der Whitelist. Nach dem Erhalt der Antwort der Web Anwendung lädt die Android Anwendung ein Client Zertifikat und anschließend die Konfiguration herunter. Auch hier muss bei einer automatischen Registrierung die Automatik ausgeschaltet werden.

6.1.2.3 Gerätedaten

Das Senden der Gerätedaten wird durch drei Testfälle überprüft.

- Eine lokale Aktualisierung der Daten durch den Benutzer. Dadurch wird nur das Sammeln der Informationen an sich getestet, da sie nur in der Anwendung angezeigt werden. Gleichzeitig wird der Vergleich der Werte mit dem letzten vollständigen Sammelzyklus und das Erstellen der Meldungen, falls ein Wert oder seine Änderung die festgelegte Grenze überschreiten, überprüft.

- Ein vollständiger Sammelzyklus der durch den Benutzer ausgelöst wird. Die Daten werden gesammelt, an die Web Anwendung gesendet und schlussendlich von der Java Anwendung an Nagios übertragen.
- Für das automatische Sammeln der Informationen muss nur noch die Einhaltung des Zeitplans und die Durchführung der einzelnen Aktionen überprüft werden, da Sammeln, Senden und Vergleichen an sich schon getestet wurden.

6.1.2.4 Bearbeitung der Gerätelisten

Diese Tests betreffen hauptsächlich die Web Anwendung. Dabei werden Einträge für Geräte zu den verschiedenen Listen hinzugefügt und wieder entfernt. Auch das Verhalten falls mehrere Benutzer gleichzeitig die Webseiten benutzen soll geprüft werden.

6.1.2.5 Verhalten bei Ausfall einer Komponente

Neben dem Verhalten wenn alle Komponenten einwandfrei arbeiten, muss auch getestet werden was passiert, wenn eine der Komponenten ausfällt oder keine Verbindung hergestellt werden kann. Das bedeutet die Tests betreffen die Operationen, die eine Interaktion zwischen den Komponenten erfordern. Zusätzlich muss jeder Test mit jeweils einer anderen ausgefallen Komponente wiederholt werden.
Ein Ausfall einer laufenden Android Anwendung stellt kein Problem für das System dar. Werden über einen längeren Zeitraum keine Gerätedaten gesendet, ändert die Web Anwendung den Status des Gerätes. Ein Ausfall der Web Anwendung stellt ein größeres Problem dar. Laufende Android Anwendungen funktionieren zwar weiter und folgen ihrem Abfragezeitplan aber sämtliche Netzwerkoperationen enden in einem Timeout. Die Java Anwendung ist von einem Ausfall der Web Anwendung nicht betroffen da sie nur auf Verbindungen wartet. Ein Ausfall der Java Anwendung führt zu Timeouts, wenn die Web Anwendung versucht eine Verbindung aufzubauen. Android Anwendungen sind hier nicht betroffen, da ihre Anfragen von der Web Anwendung verarbeitet werden. Allerdings führen die beiden letzten Fälle zu einem Ausfall des Gesamtsystems, da keine Daten mehr an Nagios gesendet werden. Der Status beider Anwendungen könnte auf herkömmliche Weise mit Hilfe von Nagios kontrolliert werden.

6.2 Speicher- und Akkuverbrauch der Android Anwendung

6.2.1 Szenario

Akku- und Speicherverbrauch sollen anhand verschiedener Messungen ermittelt werden. Für einige der Messungen wird die Android Anwendung automatische Sammelzyklen ausgeführen und die Daten senden, um einen normalen Betrieb zu simulieren. Für die Auswahl der Werte, die während dieser Sammelzyklen erhoben werden, wird ein Szenario definiert, in dem die Arbeitnehmer eines Unternehmens Android Geräte für die berufliche und private Nutzung erhalten. Der Status dieser Geräte und eventuelle Probleme für Sicherheit und Privatsphäre sollen nun überwacht werden. Zu diesem Zweck werden aus der Liste der möglichen Datenquellen die folgenden Werte ausgewählt.

- Standort
  Für die Feststellung des Standortes wird der binäre Rechteckmodus verwendet. Das bedeutet es wird nicht der genaue Standort ermittelt, sondern nur festgestellt ob sich das Gerät auf dem Gelände des Unternehmens befindet oder nicht.
- Wi-Fi
- Akku
- Black- und Whitelist für Anwendungen
- Datum und Zeit
- Root-Status, Android Version, Android Debug Bridge (ADB)

Die Daten werden im Abstand von fünf Minuten abgefragt und bei jeder zweiten Abfrage an die Web Anwendung gesendet.

6.2.2 Ergebnisse der Messung

Der Akkuverbrauch wird in verschiedenen Konfigurationen gemessen. Jede Messung fand über einen Zeitraum von vierundzwanzig Stunden, mit einem voll aufgeladenen Akku und nach einem Neustart des Gerätes statt. Tabelle 1 zeigt die Ergebnisse in Prozent an.

Testkonfiguration	Akkuverbrauch
(1) FM (Flugzeugmodus) an, Wi-Fi aus, GPS aus, AA (Android Anwendung) aus	4%
(2) FM an, Wi-Fi an, GPS aus, AA aus	8%
(3) FM an, Wi-Fi aus, GPS an, AA aus	4%
(4) FM aus, Wi-Fi aus, GPS aus, AA aus	15%
(6) FM an, Wi-Fi aus, GPS aus, AA an	13%
(7) FM an, Wi-Fi aus, GPS an, AA an	45%
(8) FM aus, Wi-Fi an, GPS an, AA an	∼107%

Tabelle 1: Ergebnisse der Messung des Akkuverbrauchs

1. Die Messung zeigt den Verbrauch des Android Betriebssystems im Standby ohne Kommunikations- oder Netzwerktätigkeit. Das Ergebnis zeigt, dass Android selbst nur relativ wenig Energie benötigt.

2. In Messung zwei wird der Wi-Fi Adapter aktiviert und eine Verbindung zu einem Netzwerk hergestellt. Allerdings wird die Verbindung nicht aktiv genutzt bis auf Vorgänge die automatisch im Hintergrund ablaufen wie das Suchen nach Anwendungsaktualisierungen durch Android. Das Ergebnis zeigt das die Wi-Fi Verbindung selbst ohne aktive Nutzung etwa soviel wie das Betriebssystem verbraucht.

3. Für Messung drei wird die Positionsbestimmung mit GPS aktiviert jedoch durch keine Anwendung zur Bestimmung der Position genutzt. Das Ergebnis zeigt keine messbare Auswirkung auf den Akkuverbrauch durch die Aktivierung.

4. Für Messung vier werden der Flugzeugmodus, Wi-Fi und GPS deaktiviert. Das Gerät ist damit wieder mit dem Mobilfunknetz verbunden um den Verbrauch dieser Verbindung zu messen. Das Ergebnis zeigt einen deutlichen Anstieg des Akkuverbrauchs im Vergleich zur ersten Messung.

5. Messung fünf misst den Verbrauch für die Ausführung der Android Anwendung im Hintergrund. Dazu wird der Flugzeugmodus aktiviert, Wi-Fi und GPS deaktiviert und die automatische Abfrage der Gerätedaten in der Android Anwendung eingeschaltet. Dabei werden die Daten alle fünf Minuten gesammelt, jedoch nicht gesendet. Da Wi-Fi und GPS nicht genutzt werden ist der Anstieg des Akkuverbrauchs

relativ gering, selbst mit der hohen Abfragefrequenz.

6. Diese Messung ist, bis auf die Aktivierung der Positionsbestimmung durch GPS, identisch zu Messung fünf. Durch die aktive Nutzung von GPS ist der Akkuverbrauch im Vergleich zur vorigen Messung stark angestiegen.

7. Mit dieser letzten Messung wird der Verbrauch bei normalen Betrieb des Gerätes und der Android Anwendung gemessen. Dazu werden der Flugzeugmodus deaktiviert, Wi-Fi und GPS aktiviert und die automatische Abfrage der Gerätedaten durch die Android Anwendung eingeschaltet. Die Anwendung sammelt die Daten alle fünf und sendet sie alle zehn Minuten. Die Messung endet allerdings nach zirka zweiundzwanzig Stunden da der Akku zu diesem Zeitpunkt vollständig verbraucht war. Die aktive Nutzung von Wi-Fi und GPS haben den Akkuverbrauch sehr stark ansteigen lassen.

Tabelle 2 zeigt wie viel Telefon- und Hauptspeicher von der Anwendung belegt werden. Für die Arbeitsspeicherbelegung werden zwei Werte gemessen. Der private Wert umfasst die Menge an Speicher die nur von der Android Anwendung verwendet wird. Der Gesamtwert berücksichtigt auch Anteile an Seiten, die mit anderen Prozessen geteilt werden [Project(2014b)].

Speicher	Belegung
Arbeitsspeicher privat	∼9 MegaByte
Arbeitsspeicher gesamt	∼11 MegaByte
Telefonspeicher	1.46 MegaByte

Tabelle 2: Speicherverbrauch der Android Anwendung

Die Messungen der Arbeitsspeicherbelegung erfolgten während eines laufenden vollständigen Sammelzyklus und stellen einen Durchschnittswert dar. Die Messung der Telefonspeicherbelegung erfolgte nach einer vollständigen Registrierung und Konfiguration der Anwendung. Das Anwendungspaket der Android Anwendung selbst ist relativ klein und verbraucht wenig Telefonspeicher. Der größte Teil (60-70%) der Arbeitspeicherbelegung entsteht durch Heapspeicher, der von der Anwendung reserviert wird.

7 Zusammenfassung

Das entwickelte Nagios für Android System ermöglicht das passive Monitoring verschiedener Werte und Eigenschaften auf Geräten mit dem Android Betriebssystem. Die Web Anwendung erlaubt die Registrierung neuer und die Verwaltung bereits überwachter Geräte, die ihre Konfiguration von der Web Anwendung in Form einer XML Datei herunterladen. Zusätzlich kontrolliert sie die Einhaltung der Zeitpläne nach denen die Android Anwendung auf den Geräten die Werte abfragt und sendet. Die Interaktion zwischen dem System und der Nagios Monitoring Anwendung wird durch die Java Anwendung ermöglicht.

7.1 Erweiterungsmöglichkeiten

7.1.1 Datenabfrage durch Plug-ins

Die einzelnen Abfragen der Gerätedaten sind in der aktuellen Version direkt in der Android Anwendung implementiert. Um eine Abfrage hinzuzufügen oder zu ändern muss das komplette Anwendungspaket aktualisiert werden. In einer Plug-in Architektur könnten die Abfragen verändert oder erweitert werden ohne die Hauptanwendung selbst zu ändern. Die Web Anwendung könnte erweitert werden um die Plug-ins zu verwalten und sie für die registrierten Android Geräte zur Verfügung zu stellen.

7.1.2 Anpassung für gerootete Geräte

Die Android Anwendung ist für die Benutzung auf nicht gerooteten Geräten ausgelegt. Die möglichen Abfragen und daraus gewonnen Daten werden durch die Android API festgelegt und begrenzt. Einer Anwendung mit Root Level Berechtigungen stehen wesentlich mehr Daten zur Verfügung. Allerdings ermöglicht es auch die Umgehung von Sicherheitsfunktionen und erlaubt zum Beispiel den Zugriff auf Anwendungsdateien durch andere Anwendungen. Hier ist eventuell zu prüfen ob Daten, die von der Android Anwendung gespeichert werden, zusätzlich gesichert werden müssen.

7.1.3 Definition von Service Werten

In der aktuellen Version werden alle Gerätedaten in einem einzigen passiven Check Kommando für den Host übertragen. Werden viele Werte abgefragt

oder ist die Ausgabe der Abfragen umfangreich, kann die Anzeige der Geräte in Nagios unübersichtlich werden. Für eine bessere Übersicht können Service Definitionen in der Nagios Konfiguration erstellt werden. Das passive Check Kommando für die Aktualisierung des Host Status kann so auf mehrere Kommandos für die Aktualisierung des entsprechenden Service Status aufgeteilt werden.

7.1.4 Automatische Host Verwaltung

Um die Administration zu erleichtern, kann das System mit einer automatischen Hostverwaltung erweitert werden. Wird ein Gerät zur Whitelist hinzugefügt oder von ihr entfernt, könnte mit Hilfe eines Skripts der entsprechende Eintrag in der Nagios Host Liste erstellt oder entfernt werden. Um die Änderung der Host Konfiguration abzuschließen ist ein Neustart nötig, der ebenfalls über ein Skript durchgeführt werden kann.

8 Literaturverzeichnis

Literatur

[Association(2013)] GSM Association. Imei allocation and approval process, Oktober 2013. URL http://www.gsma.com/newsroom/wp-content/uploads/2013/12/TS06-v7-0.pdf.

[Corporation(2014a)] Oracle Corporation. Chapter 14 configuring javaserver faces applications (the java ee 5 tutorial), Juli 2014a. URL http://docs.oracle.com/cd/E19159-01/819-3669/bnawo/index.html.

[Corporation(2014b)] Oracle Corporation. web.xml deployment descriptor elements, Juli 2014b. URL http://docs.oracle.com/cd/E13222_01/wls/docs81/webapp/web_xml.html.

[Enck et al.(2011)Enck, Octeau, McDaniel, and Chaudhuri] William Enck, Damien Octeau, Patrick McDaniel, and Swarat Chaudhuri. A study of android application security. In *USENIX security symposium*, 2011. URL http://www.enck.org/pubs/NAS-TR-0144-2011.pdf.

[Enterprises(2013a)] Nagios Enterprises. Nrpe - nagios remote plugin executor, Dezember 2013a. URL http://exchange.nagios.org/directory/Addons/Passive-Checks/NRDP--2D-Nagios-Remote-Data-Processor/details.

[Enterprises(2013b)] Nagios Enterprises. Nrpe - nagios remote plugin executor, Dezember 2013b. URL http://exchange.nagios.org/directory/Addons/Monitoring-Agents/NRPE--2D-Nagios-Remote-Plugin-Executor/details.

[Enterprises(2014)] Nagios Enterprises. Object definitions, März 2014. URL http://nagios.sourceforge.net/docs/3_0/objectdefinitions.html#host.

[Foundation(2014)] Apache Software Foundation. Apache tomcat 7 configuration reference (7.0.54) - the http connector, September 2014. URL http://tomcat.apache.org/tomcat-7.0-doc/config/http.html.

[Maker and Chan(2009)] Frank Maker and Y-h Chan. A survey on android vs. linux. *University of California*, pages 1–10, 2009.

[Project(2013a)] Android Open Source Project. Unifying key store access in ics — android developers blog, März 2013a. URL http://android-developers.blogspot.co.at/2012/03/unifying-key-store-access-in-ics.html.

[Project(2013b)] Android Open Source Project. Android debug bridge — android developers, Dezember 2013b. URL http://developer.android.com/tools/help/adb.html.

[Project(2013c)] Android Open Source Project. Determining and monitoring the docking state and type — android developers, Dezember 2013c. URL http://developer.android.com/training/monitoring-device-state/docking-monitoring.html.

[Project(2013d)] Android Open Source Project. Application fundamentals, November 2013d. URL http://developer.android.com/guide/components/fundamentals.html.

[Project(2013e)] Android Open Source Project. Licenses — android developers, November 2013e. URL http://source.android.com/source/licenses.html.

[Project(2013f)] Android Open Source Project. <manifest>— android developers, November 2013f. URL http://developer.android.com/guide/topics/manifest/manifest-element.html.

[Project(2013g)] Android Open Source Project. Manifest.permission — android developers, November 2013g. URL http://developer.android.com/reference/android/Manifest.permission.html.

[Project(2013h)] Android Open Source Project. Security enhancements in jelly bean — android developers blog, Dezember 2013h. URL http://android-developers.blogspot.co.at/2013/02/security-enhancements-in-jelly-bean.html.

[Project(2013i)] Android Open Source Project. Signing your applications — android developers, November 2013i. URL http://developer.android.com/tools/publishing/app-signing.html.

[Project(2013j)] Android Open Source Project. Vpnservice — android developers, November 2013j. URL http://developer.android.com/reference/android/net/VpnService.html.

[Project(2013k)] Android Open Source Project. Open distribution, September 2013k. URL http://developer.android.com/distribute/open.html.

[Project(2014a)] Android Open Source Project. Adt plugin — android developers, April 2014a. URL http://developer.android.com/tools/sdk/eclipse-adt.html.

[Project(2014b)] Android Open Source Project. Investigating your ram usage — android developers), September 2014b. URL https://developer.android.com/tools/debugging/debugging-memory.html.

[Project(2014c)] Android Open Source Project. Android ndk — android developers, April 2014c. URL https://developer.android.com/tools/sdk/ndk/index.html.

9 Curriculum Vitae

10 Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe. Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Linz, 7. Januar 2015 _____

Christoph Wiesner