Fachhochschule
Münster University of
Applied Sciences

Fachbereich
Elektrotechnik und Informatik

Bachelorarbeit

über das Thema

Mobile Security Awareness
Entwicklung einer Software zur Sensibilisierung

Autor: Eugen Seiz
eugen.seiz@gmx.de

Betreuer: M.Sc. Hendrik Schwartke

Prüfer: Prof. Dr. Sebastian Schinzel

Unternehmen: OpenSource Security Ralf Spenneberg

Abgabedatum: 23.03.2015

Danksagung

An dieser Stelle möchte ich die Gelegenheit wahrnehmen, mich bei allen Menschen zu bedanken, die mich während der Fertigstellung dieser Arbeit und während meines Studiums unterstützt haben.

Zunächst möchte ich Herrn Prof. Dr. Sebastian Schinzel für seine Unterstützung und die Betreuung dieser Arbeit danken.

Im Weiteren bedanke ich mich bei Herrn M.Sc. Hendrik Schwartke, meinem Betreuer in der Firma OpenSource Security - Ralf Spenneberg, der mich insbesondere in schwierigen und komplizierten Entwicklungsphasen meiner Praxisphase und Bachelorarbeit unterstützt hat und auch für kleinere Probleme immer ein freies Ohr hatte.

Daneben gilt mein Dank Claudia und Ralf Spenneberg, die beide meine Arbeit Korrektur gelesen haben. Sie wiesen auf Schwächen hin und hielten auch oft direkte Verbesserungsvorschläge bereit. Herrn Spenneberg möchte ich außerdem dafür danken, dass er mir die Möglichkeit gegeben hat meine Praxisphase und Bachelorarbeit in seinem Unternehmen zu absolvieren.

Mein besonderer Dank gilt weiterhin meinen Eltern, Lilli und Anatoli Seiz und meiner Schwester Irene, auf die ich mich zu jeder Zeit verlassen konnte und auch gewiss in Zukunft wieder kann. Diese drei Personen haben durch ihre moralische Unterstützung wesentlich zum Erfolg dieser Arbeit und meines Studiums beigetragen.

Darüber hinaus bedanke ich mich bei allen meinen Freunden, die mir bewusst aber auch oft unbewusst die Stange gehalten und mich somit bei der Anfertigung dieser Arbeit unterstützt haben.

Kurzfassung

Fast täglich erreichen die Endverbraucher und Nutzer mobiler Geräte neue Studien über die Zunahme und rasante Verbreitung mobiler Schadsoftware. Fast jeder, vom Schulkind bis zum Senior, besitzt mittlerweile ein solches Gerät und kann somit zum Opfer derartiger Software werden. Aus diesem Grund ist eine Sensibilisierung und Aufklärung der Bürger auf diesem Gebiet erforderlich, denn nur wer Gefahren und Risiken kennt, kann sich auch effektiv dagegen schützen und nebenher der Verbreitung entgegenwirken.

Die Bachelorarbeit beschäftigt sich mit der Entwicklung einer Schulungssoftware zu Demonstrationszwecken des Gefährdungs- und Risikopotenzials von Applikationen auf mobilen Geräten mit dem Android OS. Diese Software soll bei Schulungsveranstaltungen in Unternehmen eingesetzt werden und typische Angriffe auf die Privatsphäre simulieren, um User bzw. Mitarbeiter im Umgang mit mobilen Geräten zu sensibilisieren.

Inhaltsverzeichnis

II Abbildungsverzeichnis

III Tabellenverzeichnis

1 Einleitung

1.1 Unternehmensprofil

OpenSource Security - Ralf Spenneberg ist ein Dienstleistungsunternehmen, das sich auf den Support von anderen Unternehmen im Bereich IT-Security, sowie Schulungen von Mitarbeiterpersonal spezialisiert hat. Wie man schon dem Firmennamen entnehmen kann, beschäftigt man sich hier mit OpenSource Software, die kundenspezifisch angepasst, vor Ort in Betrieb genommen und gewartet wird. Des weiteren gehört vor allem in den letzten Jahren auch immer mehr die Forschung in fremden, jedoch verwandten, IT-Bereichen zum Einsatzgebiet des Unternehmens. Dabei werden z.B. Steuerungsanlagen für die Industrie oder Funkschließanlagen unter Betrachtung sicherheitstechnischer Kriterien der IT untersucht.

1.2 Hintergrund und Motivation

Im heutigen Zeitalter verdrängen mobile Endgeräte wie Tablets, Smartphones etc., immer mehr die traditionellen Systeme wie PC oder Laptop, aus dem privaten Haushalt. Der Grund für diese Tatsache ist, dass fast alle Aufgaben, wie z.B. im Internet surfen, online einkaufen, online Banking betreiben, E-Mails abrufen und versenden, Dokumente bearbeiten usw., mit diesen Geräten erledigt werden können und das ohne Wartezeiten für das Hoch- und Herunterfahren oder die lokale Bindung an einen Arbeitsplatz. Auch in Unternehmen wird das Mitarbeiterpersonal vermehrt mit solchen Geräten ausgestattet, um beispielsweise die Kommunikation oder den Informationsaustausch zwischen Abteilungen oder Mitarbeitern zu verbessern.

Die Zunahme mobiler Geräte führt auch dazu, dass die Anzahl der dafür geschriebenen Schadsoftware rasant ansteigt. Seit dem ersten Auftreten im Jahr 2004 ist mobile Schadsoftware seit ca. 2011 zum ernstzunehmenden Problem angewachsen, denn in diesem Jahr fand man die gleiche Anzahl an Bedrohungen, wie im gesamten Zeitraum von 2004-2010. Der Großteil aller Bedrohungen, immerhin über 90%, zielt dabei auf Geräte mit dem Android OS ab. [Vgl. [1]] Aus diesem Grund beschäftigt sich diese Arbeit mit mobiler Schadsoftware, die auf der Android Plattform läuft.

Benutzer speichern eine große Anzahl an sensitiven Daten wie Kontakte, Fotos, Nachrichten, unterschiedlichste zum Teil vertrauliche Dokumente, E-Mails und noch vieles mehr auf ihren Geräten. Für die Erzeugung, Bearbeitung und Speicherung dieser Daten werden Apps eingesetzt unter denen sich, aufgrund oben genannter Zahlen, auch immer häufiger Applikationen mit Malware-Inhalt befinden.

Das Verwenden von Speicher, Sensoren, Kamera, Mikrofon und anderer Ressourcen wird einer App durch die vom Android Betriebssystem zur Verfügung gestellten Schnittstellen ermöglicht. Den Zugriff auf diese Schnittstellen steuern sogenannte Berechtigungen, die ein Entwickler einmalig beim Implementierungsvorgang in der Manifest-Datei festlegt und die dann beim Installationsvorgang vom Benutzer bestätigt werden müssen. Hat sich ein Benutzer für eine App entschieden, muss er alle beim Installationsvorgang sichtbaren Berechtigungsanfragen akzeptieren, ansonsten kann er diese nicht installieren. Dieses Berechtigungskonzept ist ziemlich einschränkend, denn man kann die von der App geforderten Berechtigungen nur als Ganzes akzeptieren und nicht nach eigenem Ermessen einzeln ein- oder ausschalten, um beispielsweise den Zugriff auf Ressourcen und Hardwarekomponenten selbst zu steuern. Diese Tatsache bedingt, dass einige Entwickler ihren Anwendungen mehr Berechtigungen einverleiben, als diese für die Ausübung ihrer Funktion benötigen, um Zugriff auf oben genannten Ressourcen zu erhalten, um somit auf private Daten des Benutzers zuzugreifen.

Die Suche nach dem Grund für diesen Missbrauch des Berechtigungskonzeptes, lässt Raum für Spekulation. Ist es einfach die Unerfahrenheit, Faulheit oder das gezielte Ausspähen von Telefondaten das primäre Ziel mancher Entwickler? Zumindest der letzte Punkt hat sich in jüngster Vergangenheit immer wieder bestätigt. Denn zahlreiche IT-Sicherheitsexperten haben sich mit dem Thema beschäftigt und sind sich einig, dass zahlreiche Apps zu viele Berechtigungen einfordern mit deren Hilfe sie befähigt werden sensitive Daten vom Gerät auszuspähen. Nachlesen kann man dies unter anderem in dem Paper *"Wenn Android Apps mehr Berechtigungen verlangen als nötig"* von Trend Micro Inc. [Vgl. [2]] oder in den Artikeln *"Welche Apps ihre Daten ausspähen"* [Vgl. [3]] und *"Selbstbedienungsladen Smartphone"* [Vgl. [4]].

Hinzu kommt, dass das Sicherheitsbewusstsein der Benutzer im Umgang mit diesen Geräten, ob im privaten oder im unternehmerischen Umfeld, sich oft auf einem nicht ausreichendem bis mangelhaftem Niveau befindet. Dieser Mangel öffnet "zwielichtigen" Entwicklern Tür und Tor für den Einsatz und Verbreitung ihrer Malware. Für diesen Sachverhalt gibt es verschiedene Gründe. Einer davon ist, dass ein Teil der Benutzer die teilweise allgemein formulierten Berechtigungen nicht komplett versteht bzw. sich nicht darüber informiert was sie bewirken. Dadurch sind sie nicht in der Lage die Handlungsmacht einer Applikation und die damit verbundenen Risiken richtig einzuschätzen. Ein anderer Grund ist, dass viele User Anwendungen, die ihnen aufgrund der vielen Berechtigungsanfragen verdächtig vorkommen, trotzdem installieren, weil sie deren "tolle" Funktionen der Sicherheit ihrer Daten vorziehen oder einfach keine alternative Anwendung mit weniger Anfragen finden. Außerdem lassen sie sich oft vom grafischen Design oder der Ersparnis, beim kostenlosen

Erwerb einer kostenpflichtigen App, blenden.

Vor diesem Hintergrund stellt sich die Frage, welche Sicherheitsvorkehrungen oder -maßnahmen, sowohl im privaten als auch im unternehmerischen Umfeld, getroffen werden können, um das Sicherheitsbewusstsein im Umgang mit mobilen Geräten langfristig zu erhöhen. Kann eine Demonstrationssoftware entwickelt werden, die Gefahren und Risiken mobiler Schadsoftware verständlich und nachvollziehbar darstellt?

1.3 Desktop- vs. Appmalware

Die Sensibilisierung hinsichtlich der Informationssicherheit bei der Benutzung von Desktop Rechnern und Notebooks, ist ziemlich ausgeprägt. D.h. jeder Anwender weiß zumindest, dass man ein Antivirus Programm auf seinem Gerät haben und nicht jedes beliebige Programm aus unsicheren Quellen installieren sollte, um vor möglichen Angriffen geschützt zu sein. Hardware Komponenten, wie integrierte Kameras bei Notebooks, werden deaktiviert oder zugeklebt, um Angreifern den Einblick ins Privatleben zu verwehren.

Die enorme Anzahl an Zusatzfunktionen bei mobilen Geräten, wie der Möglichkeit zum Telefonieren, Fotos und Videos zu machen, Bestimmen der Aufenthaltskoordinaten via GPS, Austauschen von Dateien per Bluetooth, WLAN oder NFC usw., erhöhen das Einsatzgebiet, die Vielfalt der anfallenden Daten und somit die Preisgabe von privaten Dingen und Ereignissen gegenüber herkömmlichen Systemen. Somit ist es offensichtlich, dass mobile Geräte, durch die oben genannten Eigenschaften, perfekt zur Überwachung oder Ausspionierung eingesetzt werden können und deshalb ein besonders attraktives Ziel für Angreifer bilden.

Des weiteren setzen manche Benutzer ihre Geräte, sowohl privat als auch bei der Arbeit ein. Auf diese Weise kann ein Angreifer leicht an unternehmensinterne Daten gelangen, falls diese auf dem infizierten Gerät gespeichert werden. Es besteht die Gefahr, dass Schadsoftware quasi als blinder Passagier von Mitarbeitern in die Unternehmen eingeschleust wird und dort, ungehindert von typischen Sicherheitsvorkehrungen der IT-Infrastruktur, unternehmensinterne Daten ausspioniert oder mit geeigneter Technik weitere Desktop-Arbeitsplätze, per USB- und andere Schnittstellen, infiziert.

Umso paradoxer erscheint die Tatsache, dass das Sicherheitsbewusstsein im Umgang mit mobilen Geräten und den damit verbundenen steigenden Gefährdungen sich den neuen Gegebenheiten nicht anpasst oder komplett auf der Strecke bleibt.

In der folgenden Übersicht sind die wichtigsten charakteristischen Merkmale mobiler Schadsoftware aufgelistet, die deren zusätzliches Gefährdungspotenzial gegenüber herkömmlicher Software verdeutlichen.

- Eine Funktion für das Anrufen kostenpflichtiger Nummern oder das Versenden kostenpfl. Kurzmitteilungen ist implementiert und wird ohne die Zustimmung des Benutzers ausgeführt.
- Schädlichen Applikationen können mit Hilfe von Benachrichtigungen oder Kurzmitteilungen Nachrichten verschicken, die Links zu verseuchten Internetseiten oder manipulative Aufforderungen oder Hinweise beinhalten. (Also jegliche Art von Phishing-Angriffen.)
- Ohne Wissen, d.h. irgendeine Interaktion des Benutzers mit dem Gerät, wird heimlich auf Hardware des Geräts zugegriffen, um z.B. mit Hilfe der Kamera Fotos oder Videos zu machen, das Mikrofon als Wanze fungieren zu lassen und Anrufe oder andere Gespräche zu belauschen.
- Handelt es sich bei dem Gerät um ein Smartphone, so könnte dieses auch als Bot missbraucht werden, indem es automatisiert Betrugsanrufe tätigt oder Kurzmitteilungen mit betrügerischem Inhalt an alle Rufnummern im Adressbuch verschickt.
- Sensordaten/irgendwelche Daten (über Netzwerkumgebung, Ausführung/Installation von Apps, Anrufprotokolle, Kontaktliste, Dateien aus dem Speicher, ...) werden ohne ersichtlichen Nutzen für die Funktionalität einer App gesammelt, ausgewertet und/oder an Dritte übermittelt.
- Daten/Dateien werden über ungesicherte Schnittstellen wie USB, WLAN, NFC oder Bluetooth ohne Wissen oder Zustimmung des Nutzers übertragen.
- Firewall und andere Sicherheitsmechanismen können in manchen Fällen leicht umgangen werden, indem das Prinzip *"BYOD"* im Unternehmen durchgesetzt bzw. erlaubt wird.
- Schadsoftware kann nur schwer erkannt werden, da sie oft mit sinnvollen Anwendungen beliebiger Kategorien kombiniert wird.

1.4 Zielsetzung

Im Rahmen eines Schulungsprojektes, soll im Auftrag von *OpenSource Security - Ralf Spenneberg* eine Software entwickelt werden, die typische Eigenschaften einer mobilen Malware aus dem vorherigen Kapitel enthält. Diese soll eine Möglichkeit bieten, den entstehenden Verlust von Daten und damit einhergehende Gefährdungen und Risiken demonstrativ darzustellen. Die Ergebnisse der Software, die den Schulungsteilnehmern präsentiert werden und das Problem verdeutlichen, sollen in einer Art und Weise dargestellt werden, dass auch technisch nicht versierte Benutzer leicht die vorliegende Problematik nachvollziehen können. Ziel der Schulungen ist es, das Sicherheitsbewusstsein der Verbraucher bzw. Mitarbeiter im Umgang mit mobilen Geräten nachhaltig zu steigern.

Das grobe Konstrukt der Software sieht vor, dass verschiedene Daten von einem mobilen Gerät ausgelesen und an den Dozenten übermittelt werden. Des weiteren soll ein Frontend existieren mit dem der Dozent befähigt wird, die auf dem mobilen Gerät laufende Schadsoftware zu steuern und die ausspionierten Daten einzusehen und zu bearbeiten.

2 Bedrohungsanalyse

2.1 Gefährdungsübersicht

Stetig werden Ergebnisse aus Forschungsstudien veröffentlicht, laut denen neue mobile Spyware entdeckt wird. Immer öfter zielt diese auf Geräte mit dem Android OS. Gefährdungen, die bei der Nutzung mobiler Geräte entstehen können, haben unterschiedliche Ursachen. Diese Übersicht beinhaltet Gefahren, die von einer installierten Schadsoftware verursacht werden können. Andere Punkte, wie etwa das Rooting, fehlende Sperr- bzw. Entsperrmechanismen, Nutzung unsicherer Installationsquellen, Verschlüsselung von Datenträgern oder fehlende Sicherheitsupdates für Apps und Betriebssystem werden hier nicht behandelt.

In der folgenden Übersicht sind die wesentlichen, für diese Arbeit relevanten, Gefährdungen aufgelistet:

1. Eine Fremdsteuerung/-bedienung vorhandener Hardware (Kamera, Mikrofon) und anderer Komponenten kann vom Angreifer durchgeführt werden.
2. Sensordaten wie GPS, Accellerometer, Gyroskop, etc. können abgegriffen werden.
3. Der Angreifer kann den SMS-Speicher auslesen, einzelne SMS empfangen, manipulative SMS schreiben und verschicken.
4. Das Auslesen/Schreiben von Daten jeglicher Art (Kontaktliste, Anrufprotokoll, Kalendereinträge, Notizen) ist möglich.
5. Das Auslesen/Schreiben von Dateien jeglicher Art (Foto, Audio, Video, Dokumente, etc.)ist möglich.
6. Mit der Maleware kann ein Up- und Download von Daten/Dateien über das Netzwerk durchgeführt werden.
7. Informationen über installierte und laufende Apps können ermittelt werden.
8. Die Software kann Informationen über Netzwerke(WLAN-Netz mit dem das Gerät verbunden ist, WLAN-Netze in der Umgebung) ermitteln.
9. Das Surfverhalten des Benutzers kann über das Auslesen der Chronik- und Lesezeicheneinträge des Browsers analysiert werden.
10. Der Angreifer ist in der Lage, mit Hilfe des Benachrichtigungsmechanismus, dem Benutzer manipulative Nachrichten zukommen zu lassen.
11. Manipulative E-Mails können unbemerkt bzw. ohne die dafür vorgesehenen Standart-App von infizierten Gerät verschickt werden.

2.2 Risiken

Die Gefährdungsübersicht verdeutlicht, welche Gefahren eingeschleuste Schadsoftware auf einem Gerät mit sich bringt. In diesem Kapitel werden die Auswirkungen erläutert, die durch diese Gefährdungen entstehen können. Häufig werden die entstandenen Risikoszenarien durch die Kombination von oben genannten Gefährdungen verstärkt oder erst ermöglicht. Diese Eigenschaft wird durch die in Klammern stehenden Gefährdungsnummern dargestellt.

(1)
Durch Fremdsteuerung der Kamera, könnten (intime) Fotos gemacht werden. Über ein Netzwerk an einen Angreifer übertragen, könnte dieser die Dateien zu Erpressungszwecken oder Rufschädigung verwenden. Umgebung könnte abfotografiert/aufgenommen werden, um Kriminellen beispielsweise Informationen über eingesetzte Alarmanlagen oder andere technische Sicherheitsmaßnahmen eines Gebäudes zu liefern.

(4+1)
Denkbar ist die Ermittlung wichtiger Termine eines Mitarbeiters, wie z.B. Konferenzen, Geschäftsessen, etc. durch Auslesen der Kalendereinträge. Während solcher Veranstaltungen könnte der Angreifer zusätzlich das Mikrofon des kompromittierten Geräts einsetzen, um an weitere Informationen zu kommen.

(1+3)
Auch Anrufe könnten, zumindest teilweise, auf dem Gerät mit der installierten Schadsoftware abgehört werden. Des weiteren wäre der Angreifer in der Lage unbemerkt kostenpflichtige Kurzmitteilungen zu versenden.

(5)
Durch das Öffnen und Lesen von Dateien könnte ein Angreifer ebenfalls an geheime Informationen gelangen. Wenn die Schadsoftware darüber hinaus Schreib- und Löschfunktionalität besitzt, kann durch Manipulation wichtiger Dokumente, wie Rechnungen, Forderungen, etc., finanzieller oder sonstiger Schaden entstehen. Außerdem könnte ein Datenleck, bei gewissen Unternehmen, zu einem Imageverlust führen.

(3+4)
Das Lesen der Kontakte und die Möglichkeit Kurzmitteilungen zu versenden, könnten zum Versenden von Nachrichten genutzt werden, die Links zu verseuchten Internetseiten oder andere manipulative Nachrichten enthalten. Das Lesen des SMS-Speichers und Empfangen eingehender SMS könnte sensitive Inhalte offenbaren.

(4)

Realistisch ist auch die Manipulation von Kontaktdaten durch den Angreifer, so dass dann unter falscher Identität Informationen vom Opfer erschlichen werden können.

(2)
Ausgelesene und ausgewertete Sensordaten, beispielsweise vom GPS-Sensor, könnten dem Angreifer dazu dienen Bewegungsprofile zu erstellen oder den Aufenthaltsort des Opfers zu ermitteln.

(3+9+10+11)
Die Analyse des Surf-Verhaltens, durch Auslesen der Lesezeichen und Chronik des Browsers, könnte ermöglichen, Benutzerprofile für Werbezwecke zu erstellen. Außerdem wäre der Angreifer theoretisch in der Lage Login-Daten zu erschleichen. Dazu müsste er dem Opfer, unter Verwendung von SMS, E-Mail oder Benachrichtigungen, Links von manipulierten Login-Seiten schicken, z.B. von web.de oder gmx.de etc., nachdem er sich informiert hätte, welche Login-Seiten vom Opfer besucht wurden.

(7) Durch die Analyse installierter oder ausgeführter Applikationen könnte der Angreifer gezielt nach Anwendungen mit bekannten Sicherheitslücken suchen oder auf bestimmte Aktivitäten des Users horchen. Beispielsweise auf die Ausführung einer Banking-App.

(8)
Auch die Sammlung und Auswertung von Netzwerkinformationen des betroffenen Geräts, würde unter bestimmten Umständen eine Angriffsfläche für den Angreifer bilden. So könnte man anhand des WLAN-Netzes mit dem das Gerät verbunden ist, ebenfalls ein ungefähres Bewegungsprofil erstellen. Auch ein Statusprofil wäre denkbar, dazu müsste der Angreifer nur auswerten wann und wie lange das Opfer am häufigsten online ist. Informationen, wie Art der Verschlüsselung, SSID, BSSID usw., von anderen WLAN-Netzen aus der Umgebung, könnten ermittelt und dazu genutzt werden, schwach gesicherte oder gar unverschlüsselte Router ausfindig zu machen.

2.3 Bedrohungslage

Die Motivation mobile Schadsoftware zu schreiben hat in den meisten Fällen zwei Ursachen. Im ersten Fall versuchen die Kriminellen durch den Erwerb der Daten/Dateien, die der Benutzter auf seinem Gerät speichert, an Informationen zu kommen, die ihnen einen wirtschaftlichen Nutzen versprechen. Dazu zählen z.B. abgefangene Kurznachrichten, die z.B. mTANs (mobile Transaktions-Authentifizierungsnummern) für das Online-Banking enthalten. Ebenfalls bekannt ist der Einsatz von mobiler Malware, die das Versenden kostenpflichtiger Kurznachrichten, das Anrufen kostenpflichtiger Nummern oder die Sammlung von Daten, wie GPS-Koordinaten, Einträge der Kontaktliste, Informationen über

Netzwerkverbindungen, etc. ermöglicht. Im letzteren Fall werden die Daten im Nachhinein ausgewertet und für weitere Angriffe verwendet oder an Interessenten verkauft.

Im zweiten Fall steht der finanzielle Aspekt nicht mehr im Vordergrund. Hier wird Software eingesetzt, die den Betroffenen überwachen und kontrollieren soll. Dabei werden Daten ermittelt, die z.B. etwas über den Aufenthaltsort, Gewohnheiten, soziales Umfeld oder den Informations- bzw. Datenaustausch mit anderen Personen aussagen.

Aufgrund der unterschiedlichen Motivation ist auch das Täterprofil der "Kriminellen", die solche Schadsoftware schreiben, breit gefächert. Von einzelnen Entwicklern, die kleine Apps programmieren und dabei Features einbauen, um Daten auszuspähen, zu sammeln und diese dann selber für Angriffe zu verwenden oder an Dritte weiter zu verkaufen. Über ganze Entwicklerteams, wie etwa dem italienischen *Hacking Team*, deren Produkte über wesentlich komplexere Überwachungstechniken verfügen und vordergründig zur Ausspionierung sensitiver Daten von Privatpersonen, Unternehmen oder Behörden oder für deren Überwachung eingesetzt werden. Bis hin zu staatlichen Institutionen, wie dem BND oder der NSA, die bei weitem das größte Know How auf diesem Gebiet haben.

Vor allem wenn es um professionell erstellte Schadsoftware handelt, bleiben die Beweggründe der Hersteller oft fragwürdig. Die jüngste Vergangenheit hat jedoch ziemlich deutlich gezeigt, das diese Art von Software, entgegen allen Behauptungen zum Trotz, nicht nur zur Bekämpfung und Ausspionierung krimineller Vereinigungen eingesetzt wird.[Vgl. [9]]
Die Zielpersonen, die Opfer solcher Software werden, kommen praktisch aus allen Schichten der Gesellschaft. Vom einfachen Bürger über Journalisten, Menschenrechtler und andere regimekritische Personen bis hin zu Regierungsbeamten kann praktisch jeder, der ein Smartphone oder Tablet-PC besitzt, Opfer von mobiler Spyware werden. Nicht zu vergessen sind Unternehmen, die wegen ihrem technischem Know-How und anderen wertvollen Informationen, schon immer beliebte Ziele für Angriffe im IT-Sektor dargestellt haben. Sie werden in der heutigen Zeit gezwungen sein, ihre Sicherheitsmaßnahmen zu überdenken und anzupassen, wenn sie der zusätzlichen Gefahr, die von mobilen Geräten ausgeht, trotzen wollen.

So vielseitig wie das Täterprofil der Hersteller, so unterschiedlich ist auch die Schadsoftware selbst. Theoretisch kann jede beliebige App, ganz egal ob man sie der Sparte Spielen, Nachrichten, Social Media, Musik, Sport, Finanzen, praktische Widgets usw. zuordnet, Mal-/Spyware sein oder zumindest solche Komponenten enthalten. Besonders betroffen ist das Betriebssystem Android, auf das immerhin über 90 % der gesamten Spyware ab-

zielt.[Vgl. [1]]

Eine interessante Gegenüberstellung zu starken Verbreitung von Spyware unter Android findet man im Paper von den *SnoopWall mobile security experts*. Dort befindet sich unter anderem eine Übersicht, der aktuell im Umlauf befindenden Taschenlampen-Apps mit zahlreichen und unnötigen Berechtigungsanfragen und den von ihnen ausgehenden Gefährdungen.[Vgl. [5]].

3 Grundlagen von Android

In diesem Kapitel werden einige grundlegende Schlüsselwörter sowie Programmierkonzepte des Android Betriebssystems erläutert. Diese werden, für ein besseres Verständnis und Nachvollziehen der einzelnen App-Komponenten und deren Zusammenarbeit in späteren Kapiteln, benötigt.

3.1 Android Sandbox und Berechtigungsmodell

Berechtigungen oder im englischen Permissions werden in Betriebssystemen dazu verwendet, den Zugriff auf Ressourcen und Dateien auf einem System zu regulieren, um beispielsweise eine klare Trennung der Ressourcenfreigabe und -verwendung für Programme, die im Kernel- oder User Mode laufen, zu schaffen. Diese Grenze ist notwendig, da das im Kernel-Mode laufende OS mit all seinen Aufgaben, wie Prozessverwaltung, Speicherverwaltung usw. den vollen Zugriff auf die Ressourcen benötigt, um seine Arbeit vollständig erledigen zu können. Im Gegensatz dazu benötigt ein normaler User, der z.B nur bestimmte Programme ausführen und Dateien einsehen bzw. bearbeiten will, nur beschränkte Zugriffsrechte auf Ressourcen. Es gibt jedoch Schnittstellen mit deren Hilfe ein User auf OS-interne Funktionen zugreifen kann, um sich somit einen erweiterten Zugang zu Ressourcen zu ermöglichen.

Das Android OS verfügt über zwei getrennte jedoch miteinander kooperierende Berechtigungsmodelle.

Als erstes zu nennen ist der Linux Kernel, der Zugangsberechtigungen von Applikationen mit Hilfe von User- und Group-IDs realisiert. Vereinfacht dargestellt wird dabei jeder App eine eigene eindeutige User- oder Group-ID zugewiesen. Der Zweck dieses Modells ist, das auf dem Gerät laufende Apps isoliert und daran gehindert werden miteinander zu kommunizieren. Somit können sie z.B. keine Signale mehr untereinander verschicken oder auf den internen Speicher des jeweils anderen zugreifen. Dieses Modell wurde von Linux übernommen und wird als *Androids Sandbox* bezeichnet.

An zweiter Stelle steht das, unter dem Punkt Motivation schon angeschnittene, *Android Permission Modell*. Dieses ermöglicht den Apps mit Hilfe der vom Betriebssystems zur Verfügung gestellten Schnittstellen den Zugriff auf Speichermedien und die restlichen Hardwarekomponenten. Mit Hilfe von diesem Modell ist es möglich, die App-Funktionalität zu beschränken bzw. ihr nur gewisse Bereiche und Ressourcen des OS zuzuteilen. Das Setzen der Permissions übernimmt der Programmierer während des Entwicklungsprozesses.

3.2 Komponenten von Android-Anwendungen

3.2.1 Activity

Eine Activity ist immer der sichtbare Teil einer App, mit deren Hilfe der Benutzer in der Lage ist die verschiedenen Bedienelemente der graphischen Oberfläche zu bedienen, Informationen der App einzusehen und innerhalb der App zu navigieren. Eine Applikation kann aus mehreren Activities bestehen, wobei zur Laufzeit immer nur genau eine von diesen aktiv sein kann, die anderen befinden sich in jeweils einem bestimmten Zustand.

Zur Verwaltung wird der sogenannte Activity Stack eingesetzt. Die aktuell ausgeführte Activity, auch als aktiv oder im Vordergrund bezeichnet, befindet sich ganz oben auf dem Stack. Die restlichen liegen darunter und befinden sich im Pausenzustand .

Mögliche Zustände einer Activity können aus dem folgenden Zustandsdiagramm entnommen werden:

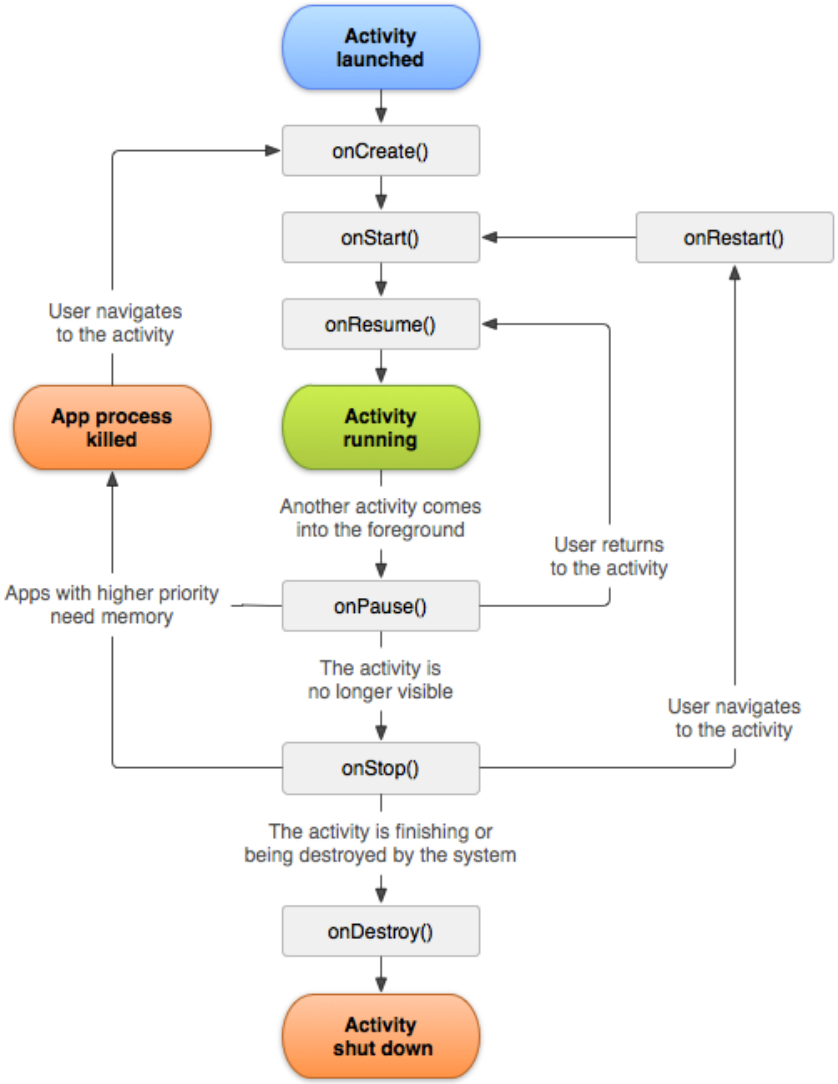


Abbildung 1: activity lifecycle[Vgl. [6]]

3.2.2 Service

Ein Service ist im Gegensatz zu einer Activity nicht sichtbar und muss erst durch eine Activity aufgerufen werden, um mit seiner Arbeit zu beginnen. Man unterscheidet zwischen zwei Arten von Services:
Zum einen existiert der sogenannte “Started Service”. Dieser Service wird von einer anderen Applikation-Komponente gestartet und verrichtet seine Aufgabe im Hintergrund.
Zum anderen gibt es den ”Bounded Service”, der ebenso von einer App-Komp. gestartet wird, jedoch in bestimmten Zeitabständen auf Nachfrage einer anderen App-Komponente oder bei bestimmten Systemereignissen, seine Ergebnisse mit der aufrufenden. Komp. synchronisiert.
Ein Beispiel für ein Service könnte ein Programmteil einer App sein, die ein Onlinespiel darstellt. Während der Benutzer über Activities das Spiel bedient könnte der Service im

Hintergrund ausgehende und eingehende Daten zum/vom Server verwalten.

3.2.3 BroadcastReceiver

Ein Broadcast Receiver ermöglicht einer App Broadcast-Nachrichten vom System, anderen Apps oder zwischen Komponenten der eigenen App zu empfangen. Solche Nachrichten werden auch als Ereignisse (engl. Events) bezeichnet und können die App über diverse Vorgänge auf dem System informieren.

Beispielsweise kann sich eine App für die Events *Intent.ACTION_BATTERY_LOW* oder *Intent.ACTION_BATTERY_OKAY* registrieren und somit über den Stand des Akkus informiert werden.

In der folgenden Tabelle sind einige Events aufgelistet, die für die Funktionalität der erstellten App eine Rolle spielen. Wie der Einsatz dieser Events im Detail aussieht und was deren Verarbeitung in der App bewirkt, wird an entsprechender Stelle in den Kapiteln *"SpyAction-Interface und -Klassen"* und *"Verschleierung der App-Aktivität"* erklärt.

Ereigniss	**Beschreibung**
Intent.ACTION_SCREEN_ON	Der Bildschirm des Geräts ist an.
Intent.ACTION_SCREEN_OFF	Der Bildschirm des Geräts ist aus.
Intent.ACTION_USER_PRESENT	Der User ist aktiv und das Gerät nicht im Ruhemodus.
android.provider.Telephony.SMS_RECEIVED	Eine SMS wurde empfangen.

Tabelle 1: Übersicht der verarbeiteten Events in der Android Anwendung.

3.2.4 Manifest- und Layoutdateien

Jede Android-Applikation enthält eine Manifest.xml Datei. Wie man an der Endung der Datei erkennen kann, handelt es sich um eine xml-Datei. Unter anderem wird hier der *package name* deklariert, dies ist eine ID, mit der jede Applikation ausgestattet wird und mit deren Hilfe die Applikation eindeutig auf dem laufenden System identifiziert werden kann. Des weiteren deklariert und beschreibt man hier auch die verschiedenen Komponenten, wie Activity, Service, BroadcastReciever etc., der Applikation. Auch die App-Berechtigungen(engl. Permissions) werden im Manifest platziert und somit der Zugriff auf verschiedene Systemressourcen der App geregelt.

In den Layoutdateien wird, ebenfalls in einem xml-Format, das grafische Design der Activities bestimmt. Diese Dateien sind hierarchisch aufgebaut, d.h. es gibt eine Verschachtelung von Eltern- und Kindelementen. So bildet z.B. sehr häufig das Layout-Tag das

Elternelement für nachfolgende Tags, wie TextView, Button, etc,. Das Aussehen der Activities variiert, aufgrund der unterschiedlichen Funktionalität, oft sehr stark. Deshalb besitzt jede Activity ihre eigene Layoutdatei, in der die graphischen Elemente, gemäß der Anforderung an die Activity, platziert, skaliert und ineinander verschachtelt werden.

4 Das Projekt - VirusDoc

4.1 Anforderung

1. Entwicklung einer App für mobile Geräte, die folgende Merkmale aufweist:

- Die App soll auf Geräten mit dem Android-Betriebssystem laufen
- Harmloses Aussehen bzw. eine Scheinfunktionalität soll mit Hilfe einer graphischen Oberfläche vorgetäuscht werden.
- Dauerhafter Betrieb im Hintergrund soll möglich sein auch nach dem Beenden der sichtbaren/graphischen Oberfläche.
- Die Software sollte modular aufgebaut sein, so dass die Anwendung leicht um weitere Komponenten erweitert werden könnte.

Des weiteren soll die App folgende Daten vom Gerät ermitteln bzw. Funktionalität zur Verfügung stellen können:

- Aktuelle GPS-Koordinaten des Geräts sollen ermittelt und graphisch dargestellt werden können.
- Hardware wie Kamera, Mikrofon oder Sensoren(Gyroskop- oder Accelerometersensor) sollen unbemerkt vom Benutzer verwendet und/oder gesteuert werden können.
- Unterschiedliche sensitive Daten wie z.B. Einträge aus der Kontaktliste, dem Kalender oder dem Anrufprotokoll sollen abgegriffen werden.
- Es soll die Möglichkeit bestehen Dateien vom/in internen/externen Speicher zu lesen und zu schreiben.
- Die Verzeichnisstruktur soll angezeigt werden können.
- Der Angreifer soll die Möglichkeit haben E-Mails vom infizierten Gerät zu verschicken.
- Das Lesen, Schreiben, Empfangen und Verschicken von SMS soll möglich sein.
- Netzwerk- und Browser-Informationen (WLAN-Netze in der Umgebung, das WLAN-Netz in dem sich das Gerät befindet, Lesezeichen und Chronik) sollen ermittelt werden können.
- Die auf dem Gerät installierten und ausgeführten Apps sollen angezeigt werden. Zusätzlich soll auf die Ausführung bestimmter Apps gelauscht werden.
- Alle ausgespähten Daten sollen über das Internet an den Angreifer übertragen werden können.

2. Entwicklung eines geeigneten Frontends, das einem Angreifer ermöglicht die vom Gerät ausgespähten Daten einzusehen. Es sollte folgende Merkmale aufweisen bzw. Funktionalität zur Verfügung stellen:

- Die graphischen Oberfläche sollte einfach und intuitiv zu bedienen sein.
- Ausspionierte Daten sollen überschaubar und verständlich präsentiert werden, so dass auch nicht versierte Benutzer leicht nachvollziehen können, welche Informationen entwendet werden und was mit diesen geschieht bzw. wozu sie verwendet werden können.
- Es soll die Möglichkeit bestehen, Steuerkommandos an die App zu übertragen.
- Für ausspionierte Daten/Dateien sollte ein Speichermechanismus vorhanden sein.
- Die Software sollte modular aufgebaut sein, so dass die Anwendung leicht um weitere Komponenten erweitert werden könnte.

4.2 Entwurf/Architektur

Beim Entwurf fiel der Fokus auf eine Client-Server-Architektur, wobei die Client-Komponente eine App ist, die auf dem infizierten Gerät läuft und die Server-Komponente auf dem PC des Angreifers. Dieses Software-Konzept wurde gewählt, da Opfer und Angreifer in der Regel weit von einander entfernt sind. Laut der Anforderungen soll aber eine "realtime“ Steuerung des Clients durch den Server und die damit verbundene Datenübertragung von Kommandos oder ausgespähten Daten, existieren.

Beide Komponenten haben jeweils zwei TCP-Verbindungen, die zum Senden und Empfangen von Daten verwendet werden. Der Verbindungsaufbau wird immer vom Client durchgeführt, dadurch wird die Spyware NAT-fähig. Wie die detaillierte Kommunikation der beiden Komponenten funktioniert, wird im Kapitel *"Kommunikation zwischen Client und Server im Detail"* erläutert.

Der Client soll eine grafische Oberfläche besitzen, die aus Activities besteht und dem User eine Möglichkeit zur Interaktion mit der Anwendung bietet. Außerdem ist eine Service-Komponente zu erstellen, die im Hintergrund die einzelnen Komponenten zur Ausspähung der Daten verwaltet und die Netzwerkkommunikation zum Server regelt.

Der Server soll ebenfalls über ein grafisches Frontend verfügen, indem die abgegriffenen Daten dargestellt und bearbeitet werden können. Zusätzlich ist auch hier eine Service-ähnliche Komponente vorgesehen, die die vom Client eingehende Daten entgegen nimmt, sortiert und an die passenden GUI-Elemete weiterleitet. Für die Umsetzung der Serveranwendung wurde die Programmiersprache Java eingesetzt.

4.3 VirusDoc-Client

Der Client besteht aus mehreren Komponenten und bildet den Teil der Anwendung, der auf dem mobilen Gerät läuft. Die enthaltenen Komponenten sind die graphische Ober-

fläche und ein Service, der den unsichtbaren Teil der App ausführt und steuert. Des Weiteren gibt es einen NetworkTask zum Senden und Empfangen von Daten und die SpyAction-Klassen, die genau eine bestimmte Art von Informationen vom infizierten Gerät auslesen. Die Komponenten und deren Funktionsweise werden in den folgenden Kapiteln ausführlich erläutert.

4.3.1 Graphische Oberfläche

Um den Benutzern zu demonstrieren, dass mobile Malware unsichtbar sein und einer beliebigen Kategorie von Apps angehören kann, wurde die Applikation mit einer graphischen Oberfläche ausgestattet. Diese Oberfläche ist der einer AntiVirus-App nachempfunden und enthält typische Bedien- und Informationselemente, wie *Button*, *ProgressBar* oder *TextView*, so dass typische Steuervorgänge imitiert werden können. Sie besteht aus drei Activities, die sich mit Hilfe der auf ihnen befindenden Steuerelemente gegenseitig aufrufen lassen.

Die folgenden zwei Abb. zeigen die drei Activity-Komponenten und verdeutlichen das gegenseitige Aufrufverhalten zwischen ihnen.

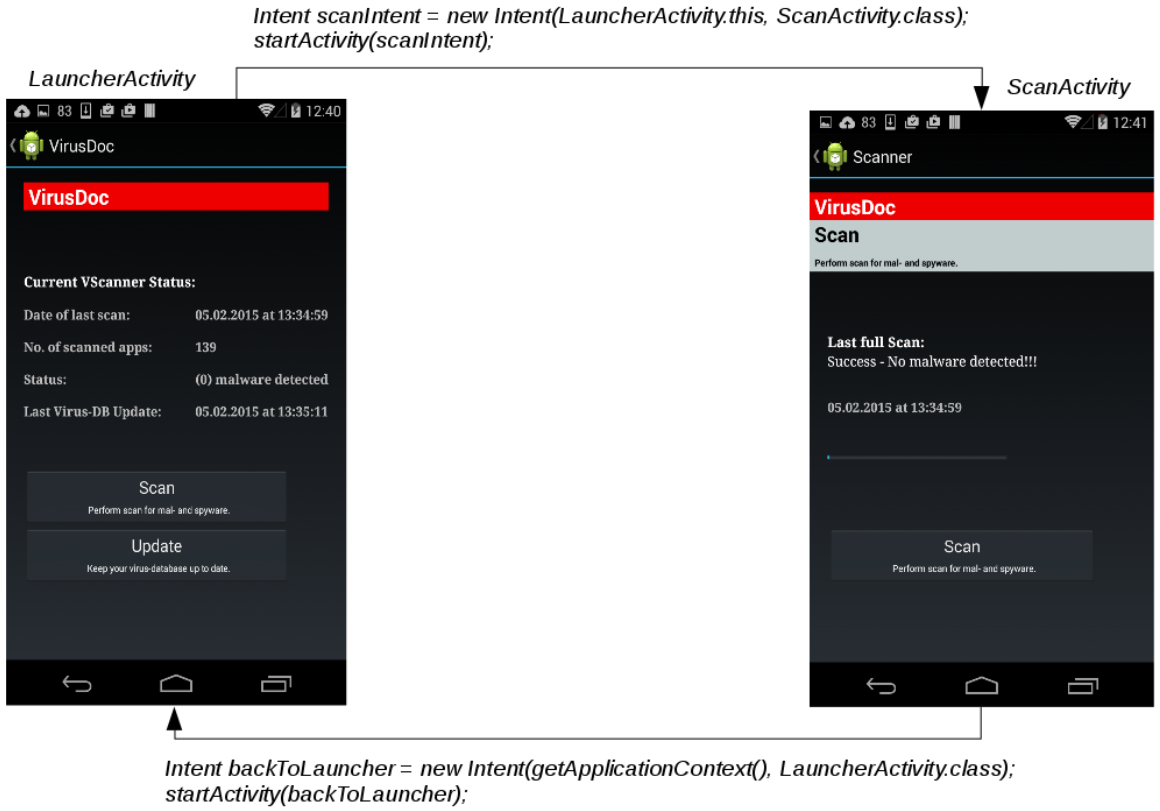


Abbildung 2: LauncherActivity und ScanActivity, die Pfeile deuten die Aufrufrichtung an.

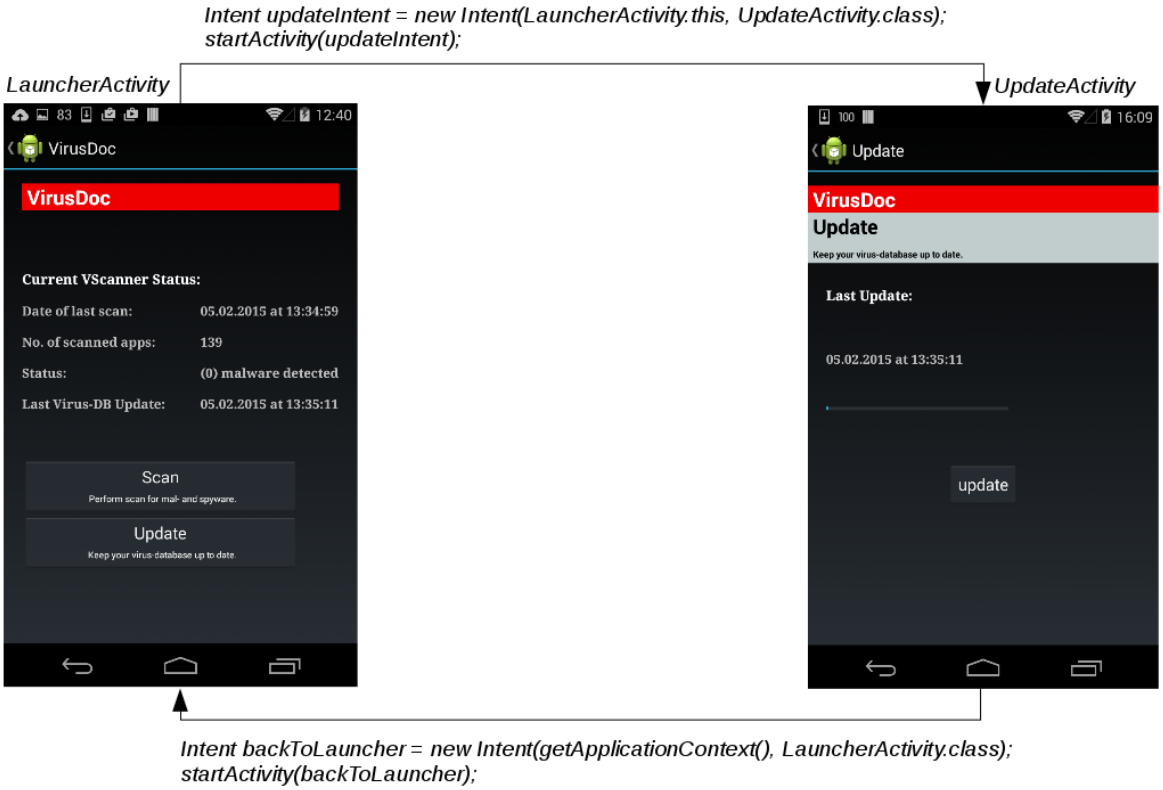


Abbildung 3: LauncherActivity und UpdateActivity, die Pfeile deuten die Aufrufrichtung an.

4.3.2 SpyService

Der SpyService ist, wie der Name schon andeutet, eine Service-Komponente. Dieser Teil der App ist für den User nicht sichtbar, denn er läuft im Hintergrund und kann nur manuell gestoppt werden. D.h. auch wenn der sichtbare Teil der App nicht mehr ausgeführt wird, weil der Benutzer z.B. mit dem Backbutton zurück zum Homescreen navigiert, ist der Service weiterhin aktiv und übermittelt Daten an den SpyServer.

Man kann diesen Teil des Clients auch als dessen Steuerzentrale bezeichnen, denn er enthält die Liste mit allen SpyActions und sorgt mithilfe eines zyklischen Verfahrens in einem Thread, dass jede von ihnen ihre ausgehenden und eingehenden Daten verarbeitet. Die Daten bestehen dabei entweder aus den ausgespähten Informationen, die von den SpyActions abgegriffen, dann an den NetwokTask übergeben und schließlich an den SpyServer weitergeleitet werden oder aus Server-Kommandos. Letztere werden in umgekehrter Reihenfolge zuerst im NetwokTask empfangen und geparst und anschließend im SpyService an die passende SpyAction übermittelt.

4.3.3 NetworkTask

Auch beim NetworkTask deutet der Name schon die Funktionalität an. Diese Komponente ist für das konkrete Senden und Empfangen von Daten auf Netzwerkebene zuständig. Hier werden die passenden In- und Outputsockets erzeugt und ausgehende bzw. eingehende Nachrichten in diese geschrieben bzw. von diesen gelesen. Zur Zwischenspeicherung ausgehender und eingehender Nachrichten enthält der NetwokTask eine Output- und InputQueue. Mit diesen Datenstrukturen wird eine komfortable Bearbeitung der Nachrichten möglich. Außerdem lassen sich so Nachrichten variabler Länge besser verarbeiten.

4.3.4 NetworkMessage-Interface und -Klassen

Das vorherige Kapitel demonstriert unter anderem welche Komponente für das Senden und Empfangen von Nachrichten zuständig ist und wo diese gespeichert werden. Im Verlauf dieses Kapitels wird das Datenformat der Nachrichten und deren Bedeutung erläutert.

Das folgende UML-Klassendiagramm verdeutlicht den modularen Aufbau der NetworkMessage-Klassen.

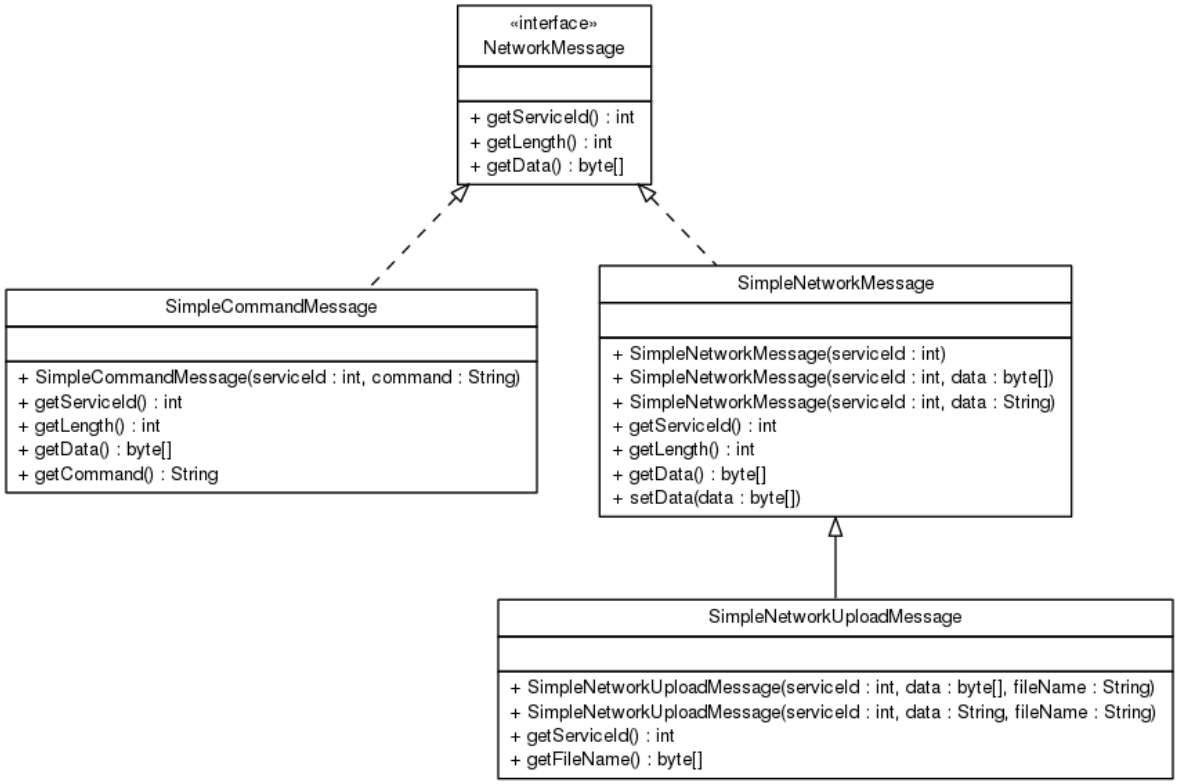


Abbildung 4: Klassendiagramm der NetworkMessage-Klassen

Dem Diagramm kann entnommen werden, dass es zwei unterschiedliche Hauptnachrichtentypen gibt, die durch die beiden Klassen *SimpleCommandMessage* und *SimpleNet-*

workMessage repräsentiert werden.

Die SimpleCommandMessage Klasse wird verwendet, um ein vom Server empfangenes Kommando zu speichern. Sobald das Kommando eintrifft, werden die Kommando-Id und der Inhalt des Kommandos in einer Nachricht abgelegt und zur Weiterverarbeitung in die InputQueue vom NetwokTask eingefügt.

Die SimpleNetworkMessage wird für den Informationsaustausch in entgegengesetzter Richtung verwendet. Sie enthält in Form eines Byte-Arrays, die Service_Id der jeweiligen SpyActions und deren ausgespähte Daten. Diese Nachrichtenpakete werden in die dafür vorgesehene OutputQueue eingefügt und an den Server gesendet. Die von SimpleNetworkMessage abgeleitete SimpleNetworkUploadMessage enthält, neben den Parametern *serviceId* und *data*, zusätzlich den Parameter *fileName* vom Typ String. Dieser wird verwendet, um bei einem Upload von Dateien den Dateinamen mit zu übertragen.

Zusammenfassend kann man sagen, dass die NetworkMessage-Klassen eine Art Header für die übertragenden Daten bilden. Durch die Verwendung dieser Header wird eine einfache jedoch effiziente Zuweisung der Daten an die jeweils dafür vorhergesehenen Empfänger auf beiden Seiten möglich.

4.3.5 SpyAction-Interface und -Klassen

Bei den SpyActions handelt es sich um die Klassen, die tatsächlich gewünschte Daten, wie Kontakte, SMS, Fotos, Videos usw., vom Gerät auslesen oder auf bestimmte Hardwarekomponente, wie Kamera, Sensoren, Mikrofon, zugreifen. Dazu benutzen sie vordefinierte Schnittstellen, die von der Android API angeboten und durch die bestätigten Berechtigungsanfragen freigeschaltet werden.

Das folgende UML-Klassendiagramm verdeutlicht den modularen Aufbau der SpyAction-Klassen und deren Funktionsweise. Siehe Anhang Abb.6.

In dem Diagramm sieht man, dass alle Klassen direkt oder indirekt, über die Klasse *ContextBaseSpyAction*, das Interface SpyAction implementieren. Das Interface enthält die Methoden *getServiceId()* und *execute(...)*. Anhand der ersten wird die *ServiceId* der jeweiligen SpyAction ermittelt, diese ist bei jeder SpyAction eindeutig und dient dazu das Daten, die vom Client kommen, vom Server richtig interpretiert werden können.

In der execute(...)-Methode findet die eigentliche Arbeit einer Spyaction statt, d.h. je nach Anforderung wird die Methode passend überschrieben und kann dann beispielsweise SMS-Speicher, Kontakte, Sensoren etc. vom Gerät auslesen. Die ausgespähten Daten werden am Ende der Methode in die OutputQueue eingefügt und zur Weiterverarbeitung an den NetwokTask übermittelt. Durch Auslesen und Interpretieren des ersten Übergabeparameters,

der das vom Server gesendete Kommando enthält, wird eine Steuerung der SpyAction möglich. Z.B. kann durch das Kommando *"switch cam"*, das durch Drücken eines Buttons in der Gui des SpyServers ausgelöst wird, in der PhotoSpyAction zwischen Rück- und Frontkamera des Geräts geschaltet werden.

In der folgenden Tabelle ist eine Übersicht aller vorhandenen SpyActions dargestellt.

ID	**Name**	**Gegenpart auf Serverseite**	**greift auf HW zu**
1	LocationSpyAction	LocationSpyActionListener	ja
2	SmsSpyAction	SmsSpyActionListener	nein
3	PhotoSpyAction	PhotoSpyActionListener	ja
4	AudioSpyAction	AudioSpyActionListener	ja
5	SmsStorageSpyAction	SmsStorageSpyActionListener	nein
6	ContactsSpyAction	ContactsSpyActionListener	nein
7	UploadSpyAction	UploadSpyActionListener	nein
8	DirSpyAction	DirSpyActionListener	nein
9	CallLogSpyAction	CallLogSpyActionListener	nein
10	CalenderSpyAction	CalenderSpyActionListener	nein
11	ScreenStatusSpyAction	ScreenStatusSpyActionListener	nein
12	NetwokStateSpyAction	NetwokStateSpyActionListener	nein
13	WifiScanSpyAction	WifiScanSpyActionListener	nein
14	ListAppsSpyAction	ListAppsSpyActionListener	nein
15	SmsSendSpyAction	nicht nötig	nein
16	EmailSendAction	nicht nötig	nein
17	NotificationSpyAction	nicht nötig	nein
18	SensorSpyAction	SensorSpyActionListener	ja
	ContextBaseSpyAction	abstrakte Klasse	
	BroadcastSpyAction	abstrakte Klasse	

Tabelle 2: Übersicht der SpyAction-Klassen

Die Namen der SpyActions wurden so gewählt, dass man anhand dieser schon deren Funktionalität erahnen kann. In der folgenden Auflistung werden zum besseren Verständnis Funktionsweise und Besonderheiten der einzelnen SpyAction ausführlich erläutert.

- Die LocationSpyAction liest mit Hilfe des von der von Android API zur Verfügung gestellten LocationManagers die GPS- Koordinaten des Geräts aus. Diese lassen sich auf Serverseite dazu verwenden, die Position des infizierten Geräts festzustellen.

- Mit der SmsSpyAction können eingehende Nachrichten "just in time" an den SpyServer weitergeleitet werden. So kann der Angreifer unmittelbar über Ereignisse, die das Opfer betreffen und dessen mögliche Reaktion darauf, informiert werden.

- Die PhotoSpyAction greift auf die Kameras des Geräts zu, erstellt Fotos und sendet diese an den Server. Bei dieser SpyAction wurde eine Besonderheit bzw. Sicherheitslücke bewusst ausgenutzt. Die Android API sieht nämlich vor, dass Fotos/Videos nur dann gemacht werden können, wenn die App vorher ein Vorschau-Fenster (engl. Preview) erstellt und geöffnet hat, indem die zu fotografierende Szene zu sehen ist. Die Funktionsweise dieses Verfahrens kann man leicht jedem Android-Gerät, anhand vorinstallierter Apps für Fotos und Videos, nachvollziehen. Es wird immer erst das Vorschau-Fenster im engl. Preview geöffnet, bevor Fotos oder Videos gemacht werden können. Dieses Verfahren bietet eine gewisse Sicherheit, da der Benutzer, anhand der Preview immer sieht wann auf die Kamera zugegriffen wird. Nun existieren solche Previews nicht nur für den Zugriff auf die Kamera, sondern auch für andere Elemente, wie z.B. Toast-Nachrichten, die auf dem Bildschirm angezeigt werden, um den Benutzer über mögliche Ereignisse des Systems oder einer App zu informieren. Genau so eine Preview wird in dieser SpyAction verwendet. Es wird eine 1x1 Pixel große Preview für eine Toastnachricht erstellt. Diese wird an die PhotoSpyAction übergeben, die damit in der Lage ist, Fotos aufzunehmen. So kann die Umgebung heimlich fotografiert werden, ohne dass der Benutzer etwas davon merkt.

- Mit Hilfe der AudioSpyAction kann VirusDoc als Wanze agieren und Audiosignale, die mit dem Mikrofon abgehört werden, aufnehmen und übertragen. Diese Übertragung funktioniert als Livestream, sobald man in der Server-Gui unter dem Eintrag *"microfone"* im Menü auf *"start capture"* drückt beginnt die Übertragung. Über den Button *"stop capture"* kann der Stream angehalten werden.

- Der SMS-Speicher des Infizierten Geräts kann mit der SmsStorageSpyAction ausgelesen werden.
- ContactsSpyAction liest alle auf dem Gerät gespeicherten Kontaktinformationen aus und übergibt diese an den Angreifer. Die dabei ermittelten Daten, wie Telefonnummern und E-Mailadressen können dann in EmailSend- oder SmsSendSpyAction zu Social Engineering Angriffen eingesetzt werden.

- Mit DirSpyAction kann die Verzeichnisstruktur (d.h. welche Ordner, Unterordner und Dateien existieren auf dem Gerät) eingesehen werden.

- Mit der UploadSpyAction können auf dem Gerät befindliche Dateien, an den Angreifer übermittelt werden. Dieser Vorgang wird meistens nach der DirSpyAction ausgeführt, da der Angreifer im Normalfall nicht vorher weiß, welche Dateien und

Verzeichnisse sich auf dem Smartphone oder Tablet befinden. Der Pfad für den Upload kann dann einfach aus der mit DirSpyAction ermittelten Liste von Dateien kopiert, ins passende Input-Feld der Gui eingesetzt und per *"download"* Button an den Client übertragen werden.

- CallLogSpyAction liest das gesamte Anrufprotokoll aus und überträgt es ebenfalls an der Server. Diese Daten könnten dazu verwendet werden, das soziale Umfeld des Opfers besser kennenzulernen.

- Mit CalenderSpyAction werden die Kalendereinträge des infizierten Geräts ausgelesen. Ein Angreifer könnte diese Daten für weitere kriminelle Schritte ausnutzen, indem er beispielsweise bei dem Opfer einbricht, wenn dieses laut der Einträge im Kalender aus dem Haus ist. Im unternehmerischen Umfeld könnten diese Einträge über wichtige Termine eines Mitarbeiters informieren. Der Angreifer könnte dann während dieser Termine weitere SpyActions, wie PhotoSpyAction oder AudioSpyAction, verwenden, um an unternehmensinterne Informationen zu kommen.

- Die abstrakte Klasse BroadcastListener und ihre Kindklasse BroadCastSpyAction werden verwendet um Broadcast-Nachrichten vom System zu verarbeiten. Mit ihrer Hilfe können beispielsweise SmsSpyAction, ScreenStatusSpyAction und AudioSpyAction geeignet auf Nachrichten vom System reagieren. Der BroadcastListener wird intern mithilfe eines BrodcastReceiver aus dem Kapitel *"Komponenten von Android Anwendungen"* realisiert.

- Die ScreenStatusSpyAction informiert den Angreifer über den Zustand des Displays. Somit kann dieser anhand der Benachrichtigungen in der Statusleiste der Server-Gui erkennen, ob der User gerade aktiv auf das Gerät zugreift oder nicht. Eine mögliche Reaktion des Angreifers könnte sein, dass er dann mit Hilfe der AudioSpyAction anfängt das kompromittierte Gerät als Wanze zu verwenden. Im Normalfall kann man bei einem ausgeschaltetem Display davon ausgehen, den Benutzer nicht bei der Ausführung einer Anwendung zu stören, die möglicherweise ebenfalls auf das Mikrofon zugreift. Somit kann eine Kollision, aufgrund des Zugriffs auf die selbe Ressource, zwischen VirusDoc und anderen Applikationen vermieden werden.

- Die Anwendung kann durch die Verwendung der NetworkStateSpyAction, Informationen über das WLAN-Netz ermitteln in dem sich das Gerät befindet. Zusätzlich können Lesezeichen- oder Chronikeinträge, die vom Browser gespeichert werden, mit

dieser SpyAction eingesehen werden.

- Mit der WifiScanSpyAction kann nach anderen, in der Umgebung vorhandenen WLAN-Netzen, gescannt werden. Mit dieser SpyAction kann der Angreifer, beispielsweise nach ungesicherten Wlan-Netzen suchen oder nach Routern, mit bekannten Schwachstellen, Ausschau halten.

- Auf dem Gerät installierte und laufende Apps lassen sich mit der ListAppsSpyAction ermitteln.

- SMS-Nachrichten lassen sich mit der SmsSendSpyAction verschicken. Die dafür benötigten Daten, Telefonnummer und die Textnachricht selbst, werden auf Serverseite vom Angreifer als Kommando zusammengebaut und an VirusDoc versendet. Die SpyAction parst zunächst die benötigten Informationen aus dem Kommandostring, fügt diese an der richtigen Stelle im Quellcode ein und verschickt anschließend die Kurznachricht. Mit geeigneten Nachrichten kann der Angreifer nach dem Prinzip von Social Engineering bzw. - Hacking Angriffen, einen Benutzer bzw. Mitarbeiter dazu veranlassen, geheime Informatinen preiszugeben oder bestimmte Handlungen auszuüben. Eine Beispielnachricht findet man in dem Kapitel *Beispielnachrichten für Sms-, EmailSend- und NotificationSpyAction.*

- Die EmailSendAction ermöglicht das Versenden von E-Mails von dem, auf dem Gerät vorhandenem, Google-Mail Konto. Diese SpyAction kann allerdings nur genutzt werden, wenn das Passwort zur entsprechender E-Mailaddresse vorhanden ist. Dieses müsste vorher vom Angreifer mit der SMS-, der NotificationSpyAction oder auf anderem Wege ermittelt werden. Wenn die Zugangsberechtigungen zum Konto vorhanden sind, können Social Engineering bzw. - Hacking Angriffe, ähnlich den der SmsSpyAction, getätigt werden. Eine Beispielnachricht findet man in dem Kapitel *Beispielnachrichten für Sms-, EmailSend- und NotificationSpyAction.*

- Mit der NotificationSpyAction können ebenfalls Social Engineering bzw. - Hacking Angriffe durchgeführt werden, dabei wird eine vom Angreifer konzipierte Nachricht über den Notificationmechanismus der Anroid API an den User verschickt. Die manipulative Information taucht dann in der oberen Benachrichtigungsleiste auf, in der User gewöhnlich auch über abgeschlossene Downloads, entgangene Anrufe, einkommende SMS usw. informiert werden. Vordergründig wurde diese SpyAction konstruiert, um den User über weitere Apps oder Updates vom VirusDoc zu in-

formieren. Damit sollte das Interesse des Benutzers für weitere Malware-Apps des Entwicklers geweckt werden. Eine Beispielnachricht findet man in dem Kapitel *Beispielnachrichten für Sms-, EmailSend- und NotificationSpyAction.*

- Sensordaten von Gyroscope und Accellerometer können mit der SensorSpyAction aufgenommen werden. Diese SpyAction wird aber nur bei der Ausführung bestimmter Apps gestartet. Denn es soll nicht unnötig Rechenleistung des Geräts verschwendet werden. Ein anderer Grund ist, dass diese Daten nicht wirklich aussagekräftig sind, wenn man nicht den Hintergrund betrachtet, in dem sie aufgenommen werden. Die Aufnahme dieser Daten lohnt sich nur, wenn der User irgendwelche Tastatureingaben tätigt, z.B. zum Entsperren des Bildschirms. Dann kann man mit Hilfe der Sensordaten und geeigneten Algorithmen, die diese auswerten, in einigen Fällen auf den Pin rückschließen kann. Auf welche Apps der Angreifer bei der Ausführung von SensorSpyAction horchen möchte kann er selbst bestimmen, indem er sich zunächst mit der ListAppsSpyAction einen Überblick über die installierten Apps verschafft und dann gezielt einige auswählt und an den Client sendet.

4.3.6 Beispielnachrichten für Sms-, EmailSend- und NotificationSpyAction

- **Beispiel einer Nachricht für die SmsSendSpyAction:**
  017632267312/015231791170//Unsere IT-Abteilung hat festgestellt, dass Smartphones und Tablet-PCs, die den Mitarbeitern ausgehändigt wurden, vermehrt installierte Schadsoftware nach dem Gebrauch aufweisen. Bitte seien Sie achtsam im Umgang mit diesen Geräten und installieren Sie nur die notwendigsten Apps. Außerdem werden alle Mitarbeiter angewiesen eine neue Antivirus-Software, die von der IT-Abteilung geprüft und ausgewählt wurde, auf ihren Arbeitsgeräten zu installieren. Sie finden die Installationsdatei und die Installationsanleitung unter den folgenden Links:

  Installationsdatei: https://www.dropbox.com/s/i2wx5kpcvvuj42e/VirusDoc.apk?dl=0
  Installationsanleitung: https://www.dropbox.com/s/ccmuva8hbyweogc/InstallationsanleitungVirusDoc.pdf?dl=0

  Hier sieht man den grundsätzlichen Aufbau einer solchen Nachricht. Zu Beginn stehen die Telefonnummern der Zielpersonen an die die Nachricht gehen soll. Danach kommt der eigentliche manipulative Inhalt der beim Benutzer ein bestimmtes Verhalten auslösen soll.

- **Beispiel einer Nachricht für die EmailSendAction:**
  password//name1@gmx.de/name2@web.de/name3@fh-muenster.de//subject///Wir haben den unternehmensinternen Mailserver umgestellt, damit Sie ihren E-Mail-verkehr problemlos weiter bearbeiten können, müssen sie einmalig ihre Login-Daten auf unserer Webmail Loginseite bestätigen.Dazu melden Sie sich ganz einfach mit ihren vorhandenen Login-Daten auf der aktuellen Login-Seite an.

  Mit freundlichen Grüßen
  Ihre IT-Abteilung

  Im Beispiel sieht man den typischen Aufbau eines Kommandos, das der Angreifer vom SpyServer an die EmailSendAction verschickt. Der Kommandostring besteht aus mehreren Teilen, die mit doppel-Slash voneinander getrennt werden. An erster Stelle steht das Passwort für das E-Mailkonto, danach kommen die E-Mailadressen, an die der Inhalt gehen soll. An dritter Stelle kann der Angreifer einen passenden Subject angeben und zum Schluss kommt der eigentliche Inhalt der Nachricht. Die einzelnen Kommandoteile werden, wie auch bei der SmsSpyAction, an der passenden Stelle im Client-Programm geparst und eingesetzt.

- **Beispiele von Nachrichten für die NotificationSpyAction:**
  Es gibt zwei Arten von Benachrichtigungen, die vom Angreifer verschickt werden können. Eine informiert das Opfer über angebliche Updates und ladet automatisch die APK-Datei der neuen Version in den Download Ordner des Geräts. Mit der anderen versucht der Angreifer, mit Hilfe einer manipulierten Homepage, Login-Daten vom Opfer zu erspähen. Die Nachrichten bestehen jeweils aus mehreren Teilen, die mit einem dreifachem Doppelpunkt ":::"voneinander getrennt werden. Dieses Trennsymbol wird dazu benötigt die einzelnen Teile aus dem Benachrichtigungs-Kommando zu parsen.

  0:::https://www.dropbox.com/s/i2wx5kpcvvuj42e/VirusDoc.apk?dl=0:::vd.apk:::Es ist eine neue Version der Applikation VirusDoc verfügbar. Es wurden zusätzliche Features integriert, um die Sicherheit bei Bearbeitung, Download, Upload und Verwendung von Dateien und Applikationen noch mehr zu erhöhen. Die neue Anwendung finden sie im Download-Ordner ihres Gerätes!!!.

  In der oberen Nachricht sieht man den typischen Aufbau einer Update-Benachrichti-

gung. Diese besteht aus vier Teilen. An der ersten Stelle steht ein Flag, das null oder eins sein kann. Anhand dieses Flags kann auf Clientseite unterschieden werden, um welche Art von Benachrichtigung, Update- oder Fishing-Nachricht, es sich handelt und somit die richtige Interpretierung der weiteren Nachrichtenteile erfolgen. An der zweiten Stelle steht die Url von der die APK-Datei bezogen wird. Es folgen der Dateiname unter dem die Datei auf dem Gerät gespeichert wird und als letztes die eigentlich manipulative Textnachricht, die den Benutzer dazu bringen soll die neue App zu installieren.

1:::facebook:::https://de-de.facebook.com/:::Jemand hat sich unerlaubt auf ihrem Facebook Account eingeloggt. Bitte bestätigen Sie ihre Logindaten, indem Sie auf den unteren Login-Link klicken und sich ganz normal einloggen.

1:::sparkasse:::https://bankingportal.ksk-steinfurt.de/portal/portal/StartenIPSTANDARD?IID=40351060&AID=IPSTANDARD::: Jemand hat sich unerlaubt auf ihrem Sparkassen-Account eingeloggt. Bitte bestätigen Sie ihre Logindaten, indem Sie auf den unteren Login-Link klicken und sich ganz normal einloggen.

1:::google:::https://accounts.google.com/ServiceLogin?service=mail&passive=true &rm=false &continue=https://mail.google.com/mail/&ss=1&scc=1&ltmpl=google-mail&emr=1:::Jemand hat sich unerlaubt auf ihrem Google-Account eingeloggt. Bitte bestätigen Sie ihre Logindaten, indem Sie auf den unteren Login-Link klicken und sich ganz normal einloggen.

Bei den oberen drei Nachrichten handelt es sich um Fishing-Benachrichtigungen. An der ersten Stelle im Aufbau steht ebenfalls das Flag, das im Gegensatz zu einer Update-Benachrichtigung immer eins ist. De zweite Parameter steht für die Kategorie, diese ist für die Präzisierung der Nachricht nötig und sorgt für die Verwendung des richtigen Icons, der zusammen mit der Nachricht in der Benachrichtigungsleiste auftaucht. Es folgt die Url der manipulierten Loginseite und die manipulative Textnachricht.
Beim Anklicken der Benachrichtigung wird vom VirusDoc eine Activity gestartet, die den Browser mit der manipulierten Url öffnet und sich wieder beendet. So kann nicht direkt zurück verfolgt werden welche App die Benachrichtigung gesendet hat. In diesen drei konstruierten Fällen werden zur Veranschaulichung die Original Login-Seiten von Facebook, Google und Sparkasse verwendet. Diese müssten im echten Fall natürlich nachgebildet werden.

Welche Login-Seiten vom Opfer aufgerufen werden kann der Angreifer im Vorfeld, mit Hilfe der ausgelesenen und ausgewerteten Chronik oder Lesezeichen des Browsers, in Erfahrung bringen.

4.3.7 Verschleierung der App Aktivität

Im Android Operating System existiert ein Konstrukt, dass den Zugriff der Applikationen auf Hardware regelt. Im Normalfall kann jeweils nur eine einzige App auf Hardware, insbesondere Kamera oder Mikrofon, zugreifen. In der Regel entstehen dabei keine Probleme, da es meistens die Activities in einer App sind, die sich dieser Hardware bedienen und diese auch wieder freigeben, sobald sie aus einem aktiven in einen passiven Zustand, pause-/stop-/destroy-state, übergehen. Danach kann die Activity einer anderen App in den Vordergrund treten und die Hardware für sich beanspruchen.

Ein Problem entsteht meistens bei der Verwendung von einem Service, wie z.B. dem SpyService in der erstellten Anwendung, der dauerhaft im Hintergrund läuft also immer in einem aktiven Zustand ist. Werden bei seiner Ausführung PhotoSpyAction und/oder AudioSpyAction gestartet greifen diese kontinuierlich auf Kamera und Mikrofon zu. Möchte nun eine andere App des Geräts auf diese Hardware zugreifen, wird ihr der Zugriff, aufgrund des oben beschriebenen Konstruktes, verweigert und es kommt zu einer Fehlermeldung auf dem Gerät. Dies könnte einen User auf Dauer misstrauisch machen und er könnte dies möglicherweise mit der Ausführung von VirusDoc in Verbindung bringen.

Um die Aktivität von VirusDoc auf dem Zielgerät besser zu verschleiern, wurden zwei Mechanismen implementiert, die das Ausführen von AudioSpyAction und PhotoSpyAction nur bei bestimmten Bedingungen erlauben.

Als erstes zu nennen ist die methode *isCameraUsebyApp()* in der PhotoSpyAction, die zu Beginn mit *Camera.open()* versucht eine Instanz der Kamera zu erstellen. Ist die Erstellung erfolgreich wird am Ende der Methode die Kamera mit *camera.release()* wieder freigegeben und der Rückgabewert der Methode auf *false* gesetzt. Läuft der Aufruf von *Camera.open* schief, wird eine RuntimeException von Java ausgelöst und der Rückgabewert bekommt den Wert *true*. Wird diese Methode nun am Anfang der execute-Methode in der PhotoSpyAction aufgerufen und liefert "false“ zurück, d.h. die Kamera wird gerade nicht verwendet, kann die App wie vorgesehen auf die Kamera zugreifen, andernfalls wird an dieser Stelle der weitere Programmdurchlauf unterbrochen und die execute-Methode der SpyAction von vorne gestartet.

Als zweites wurde ein Mechanismus eingebaut, der es erlaubt den Status des Displays abzufragen. Im Normalfall kann man davon ausgehen, dass bei einem ausgeschalteten

Bildschirm der User nicht mehr aktiv ist und somit das Mikrofon des Geräts von der AudioSpyAction verwendet werden kann. Beim Einschalten des Bildschirms wird diese SpyAction gestoppt und gibt die Ressource, in diesem Fall das Mikrofon frei, sodass eine andere Anwendung diese verwenden kann.

4.4 Spy-Server

Der SpySever ist eine Java-Anwendung, die dem Angreifer das Empfangen, die Verarbeitung und Visualisierung der ausgespähten Daten und das Senden von Steuerkommandos an den Client ermöglicht. Die Hauptkomponenten des Servers bilden eine Gui, die dazugehörigen ButtonListener, ein TCP-Server, verschiedene SpyActionListener, die Command-Klassen für die Speicherung von Kommandos und letztendlich die externe Libary JMapViewer in Form einer Jar-Datei, die die Darstellung von GPS-Koordinaten in einer Karte übernimmt. In den folgenden Kapiteln werden die einzelnen Komponenten und deren Funktionsweise ausführlich erläutert.

4.4.1 Graphische Oberfläche

Bei der GUI handelt es sich um eine Swing basierte Anwendung. Sie ist klassisch in drei Funktionseinheiten aufgeteilt. Eine Menübar befindet sich oben, einen Content-Panel im Zentrum und eine Statusbar im unteren Bereich des Fensters. Im Content-Panel ist eine JTabbedPane eingebettet. Diese enthält die Tabs in denen die empfangenen und durch die SpyActionListener aufbereiteten Daten angezeigt werden. In den meisten Fällen werden die Daten eines Listeners in einem eigenem Tab dargestellt. Die Tabs enthalten typische Bedienelemente, wie JButton, JTextArea usw. für die Visualisierung und Bearbeitung der ausgespähten Daten und für die Übermittlung von Steuerbefehlen an den Client. In der Menübar befinden sich Bedienmöglichkeiten für die Dir- und AudioSpyAction und in der Statusbar kann man die Aktivität des ausspionierten Clients verfolgen.

Die folgende Abb. zeigt die gestartete GUI.

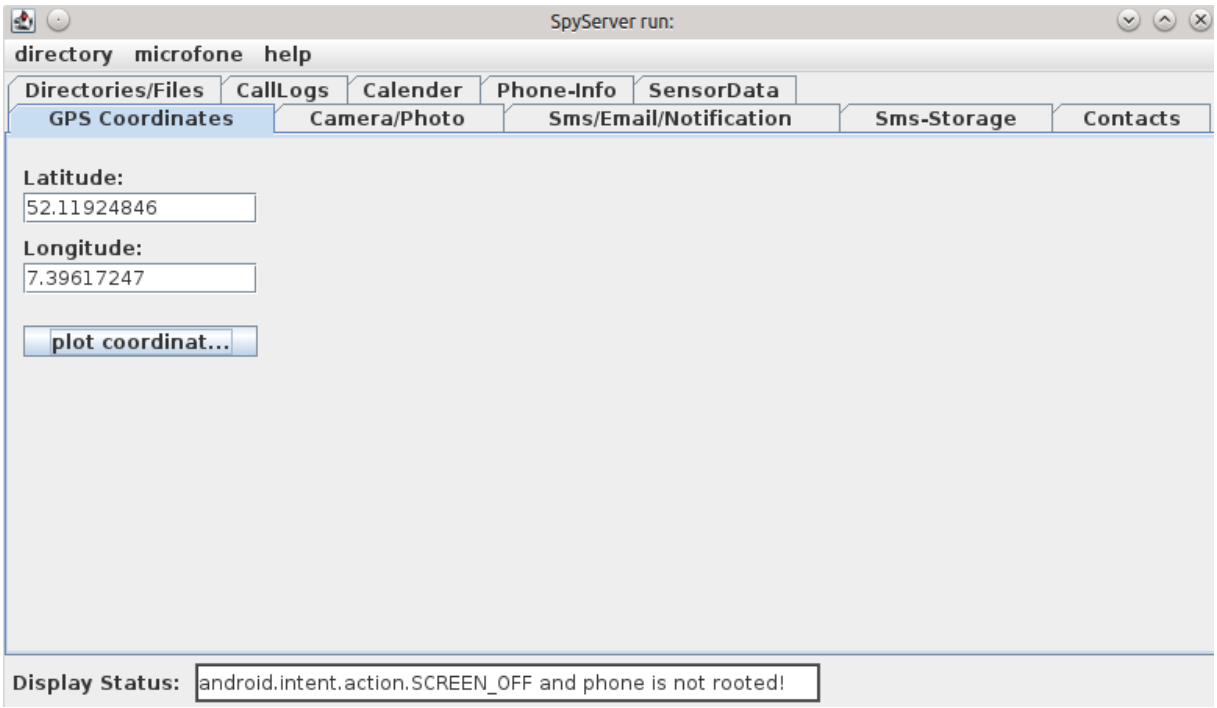


Abbildung 5: Server-GUI

Beim Start wird immer als erstes der Tab geöffnet, der die Daten des LocationSpyAction-Listeners anzeigt.

4.4.2 TCP-Server

Der TCP-Server ist das Pendant zum SpyService im Client und kann somit als die Steuerzentrale des Servers bezeichnet werden. Er enthält eine Liste mit allen SpyActionListenern und sorgt innerhalb der

processInputData(MessageHandler handler, Socket client){ ... }

Methode, dass ankommende Daten dem richtigen Listener zur Verarbeitung zugeteilt werden. Nach der Ermittlung des passenden Listeners anhand der ServiceId wird dieser und das gelesene zu ihm gehörende Datenpaket an einen Handler weitergereicht. Dieser Handler ruft dann innerhalb des GUI-Threads den tatsächlichen Listener auf und übergibt ihm die Daten zur Verarbeitung. Die Daten bestehen dabei aus einem Byte-Array, indem beispielsweise Anrufprotokoll, Kalendereinträge, Fotos usw. enthalten sind.

In einer weiteren Methode

public void start(final MessageHandler handler){ ... }

wird ein Thread angelegt. Innerhalb dieses Threads werden jeweils die Input- und Output-Sockets für eingehende und ausgehende Daten bzw. Kommandos angelegt, auf eingehende Verbindungen geprüft und letztendlich processInputData(...) intern aufgerufen, so dass

der oben beschriebene Prozess der eindeutigen Weiterleitung der Daten an den richtigen Listener immer wieder zyklisch ablaufen kann.

Als letztes zu nennen ist die

sendCommand(String command, int commandId){ ... }

Methode, mit der Kommandos an den Client übertragen werden. Bei diesem Vorgang werden die Befehlsdaten in den oben angesprochenen Outputsocket geschrieben und somit das konkrete Senden realisiert.

4.4.3 SpyActionListener-Interface und -Klassen

Das vorangegangene Kapitel zeigt, dass ausgespähte Daten von sog. SpyActionListenern bearbeitet werden. Diese Komponenten werden jetzt näher erläutert. Dabei wird deren grundsätzlicher innerer Aufbau erklärt und beschrieben für welche Art von Daten sie zuständig sind.

Das folgende UML-Klassendiagramm verdeutlicht den modularen Aufbau der SpyAction-Listener-Klassen und deren Funktionsweise. Siehe Anhang Abb.7.

Das Diagramm zeigt, dass die SpyActionListener-Klassen direkt oder indirekt, über die Klasse *BufferedServiceListener*, das Interface ServiceListener implementieren. Das Interface enthält die Methoden *getServiceId()*, *startMessage()*, *processData(byte[] data)*, *endMessage()*. Mit getServiceId() wird die *ServiceId* der zugehörigen SpyAction zurückgegeben. In startMessage() werden alle, für den Empfang und Weiterverarbeitung der Daten, notwendigen Vorkehrungen getroffen. So kann beispielsweise dort ein Pfad gesetzt werden und zu diesem ein FileOutputStream erzeugt werden. In *processData(...)* findet dann die Verarbeitung der Daten statt, z.B. das Schreiben des übergebenen Byte-Arrays in den vorher erzeugten FileOutputStream. Schließlich werden in *endMessage()* Abschlussarbeiten, wie z.B. das Schließen des FileOutputStreams, getätigt. Listener, die sich von der Klasse *BufferedServiceListener* ableiten, empfangen meistens kleinere Nachrichtenpakete. Um diese Nachrichten überschaubarer in der GUI darstellen zu können, werden sie mit Hilfe eines ByteArrayOutputStream gepuffert und erst dann weiter verarbeitet.

4.4.4 ButtonListener-Klassen

Zu jeder Gui gehören im Normalfall Bedienelemente, die es dem User ermöglichen, mit dem Programm zu interagieren. Auch das Frontend vom SpyServer weist eine Vielzahl an Buttons oder Menüeinträgen auf, die solche Vorgänge ermöglichen. Um auf verschiedene Ereignisse, die von Buttons und anderen Bedienelementen ausgelöst werden, reagieren zu

können, werden diverse *ButtonListener* eingesetzt. Diese implementieren die Schnittstellen *ActionListener* und können so mit der geeigneten Überschreibung der *actionPerformed(Event e)*-Methode auf die unterschiedlichen Ereignisse reagieren.

Nicht für jeden Button existiert ein eigener ButtonListener. Aktionen, die von Buttons mit ähnlicher Funktionalität ausgelöst werden, werden manchmal in einer Klasse zusammengefasst. So gibt es z.B. den store-Button in recht vielen verschiedenen Varianten, wie beispielsweise *"store callLog"*, *"store calender"* usw., in der Gui. Durch das Drücken dieses Buttons wird in den meisten Fällen, der sich in einer *TextArea* befindende Text in einer Textdatei auf der Platte abgelegt. Um individuell auf die unterschiedliche von den store-Buttons ausgelösten Events zu reagieren und Codeduplizierung zu vermeiden, wurde ein *StoreButtonListener* angelegt. In dieser Klasse wird unter Verwendung eines switch-case-Konstruktes, das von dem jeweiligen Button ausgelöste Event geparst und schließlich passend reagiert, indem eine Textdatei, mit treffendem Namen, wie *callLog.txt* oder *calender.txt*, angelegt wird. Das Parsen geschieht dabei mit Hilfe des *event.getActionCommand()* Aufrufs, der die Buttonbezeichnung zurück liefert.

In der folgenden Tabelle befindet sich eine Übersicht aller vorhandenen ButtonListener.

Name des Listeners *(Buttons die diese Schnittstelle verwenden)*	**Aufgabe**
SensorButtonListener *"capture accelerometer", "capture gyroscope"*	Ermöglicht die Auswahl eines Sensors, dessen Daten abgefragt werden sollen.
DirMenuListener *"complete directory", "image directory", "audio directory", "video directory"*	Ermöglicht die Suche nach Dateien in bestimmten Verzeichnissen.
DownloadButtonListener *"download"*	Ermöglicht das Runterladen von Dateien vom Client.
MicrofonMenuListener *"start capture", "stop capture"*	Ermöglicht das Starten und Stoppen der Wanzenfunktion .
PhoneInfoButtonListener *"show network-infos", "scan network", "list current Apps"*	Ermöglicht das Abfragen von Geräteinformationen *(Netzwerkstatus, liste installierter Apps, WLAN-Netze aus der Umgebung)*
SendButtonListener *"send sms", "send email", "send notification"*	Ermöglicht das Versenden von E-Mails, SMS und Benachrichtigungen vom/an infizierten Gerät.
ShowButtonListener *"show contacts", "show calllog", "show events", "show storage"*	Ermöglicht das Anzeigen der ausgespähten Daten in der Gui.
ShowMapButtonListener *"showInMap"*	Ermöglicht die Darstellung von GPS Koordinaten in einer Karte.
StoreButtonListener *"store contacts", "store calender", "store sms", "store log"*	Ermöglicht die Speicherung der angezeigten Daten in Dateien.
SwitchCamButtonListener *"switch camera", "stop"*	Ermöglicht die Steuerung der Kamera.

Tabelle 3: Übersicht der ButtonListener-Klassen

4.4.5 Command-Klassen

Diese Klassen helfen dem Angreifer bei der Steuerung des Clients durch gezielte Kommandos der Serveranwendung. Die sendCommand(...)-Methode, die diese Funktion ausübt, wird im Kapitel TCP-Server besprochen. Dieser Abschnitt führt den internen Aufbau von Kommandos genauer aus, so dass deren Funktions- und Wirkungsweise verständlich wird.

Das folgende UML-Klassendiagramm verdeutlicht den modularen Aufbau der Kommando-Klassen.

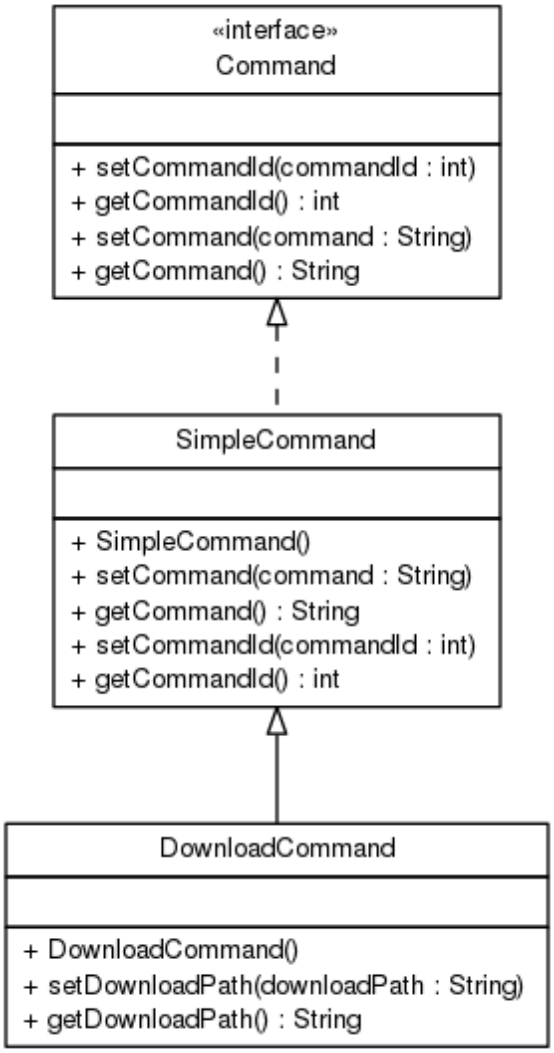


Abbildung 6: Klassendiagramm der Command-Klassen

Aus dem Klassendiagramm kann entnommen werden, dass es vier wesentliche Methoden gibt und zwar die *Setter* und *Getter* für das Kommando und dessen Id. Das Kommando ist vom Datentyp String. Komplexere Kommandos die nicht eindeutige bzw. einfache Befehle enthalten, sondern aus mehreren Teilen bestehen, werden von der Client-Anwendung geparst, um an die gewünschten Kommandoteile heranzukommen. Die *CommandId* ist vom Typ int und ist mit der *ServiceId* in den SpyAction-Klassen zu vergleichen. Sie bezweckt die eindeutige Zuweisung eines Kommandos zu der passenden Spyaction auf der Client-Seite.

Nachdem der Aufbau und die Wirkungsweise der Kommandos erläutert wurden, bleibt noch zu klären wo und wie diese gesetzt und versendet werden.

Dieser Vorgang wird nun exemplarisch anhand des showCallLog-Buttons erklärt. Wird der Button in der Gui gedrückt, so wird der passende ButtonListener dafür aufgerufen. In diesem Fall ist das der ShowButtonListener. Innerhalb von diesem Listener wird als erstes eine neue Kommando-Instanz erzeugt und anschließend mit dem Aufruf:

command.setCommand(event.getActionCommand());

zuerst das Attribut command und dannach mit

command.setCommandId(9);

die *commandId* gesetzt. Zur besseren Verständigung des nächsten Schrittes ist noch zu erwähnen, dass allen ButtonListenern eine Instanz *Tcp-Server* im Konstruktor übergeben wird. Mit dieser Instanz können sie die Methode *sendCommand* aufrufen und somit Kommandos verschicken. Im nächsten Schritt des Listeners wird genau diese Methode aufgerufen und erhält als Übergabeparameter das erstellte Kommando. In *sendCommand(Command command)* werden beide Parameter ausgelesen und nacheinander zuerst *commandId* und dann *command* in den SocketOutputStream geschrieben.

4.5 Kommunikation zwischen Client und Server im Detail

Im Verlauf dieses Kapitels wird die Kommunikation bzw. Datenübertragung zwischen Server und Client erläutert.

Die gesamte Kommunikation, sowohl vom Client zum Server als auch umgekehrt, läuft über zwei TCP-Verbindungen. Diese werden mit Hilfe der von Java zur Verfügung gestellten Schnittstelle *Socket* realisiert. Des weiteren müssen sie mit einer IP, es sollte die des Rechners sein auf der später die Serveranwendung läuft, und dem Port 5555 zum Senden und 5556 zum Empfangen, aus der Sicht des Clients, instantiiert werden. In der unteren Abbildung werden diese Verbindungen als V1, vom Client zum Server und V2 für die umgekehrte Richtung bezeichnet.

Die folgende Abbildung verdeutlicht die Kommunikation zwischen Client und Server. Die Zusammenarbeit und Funktionsweise der einzelnen Komponenten wird weiter unten beschrieben.

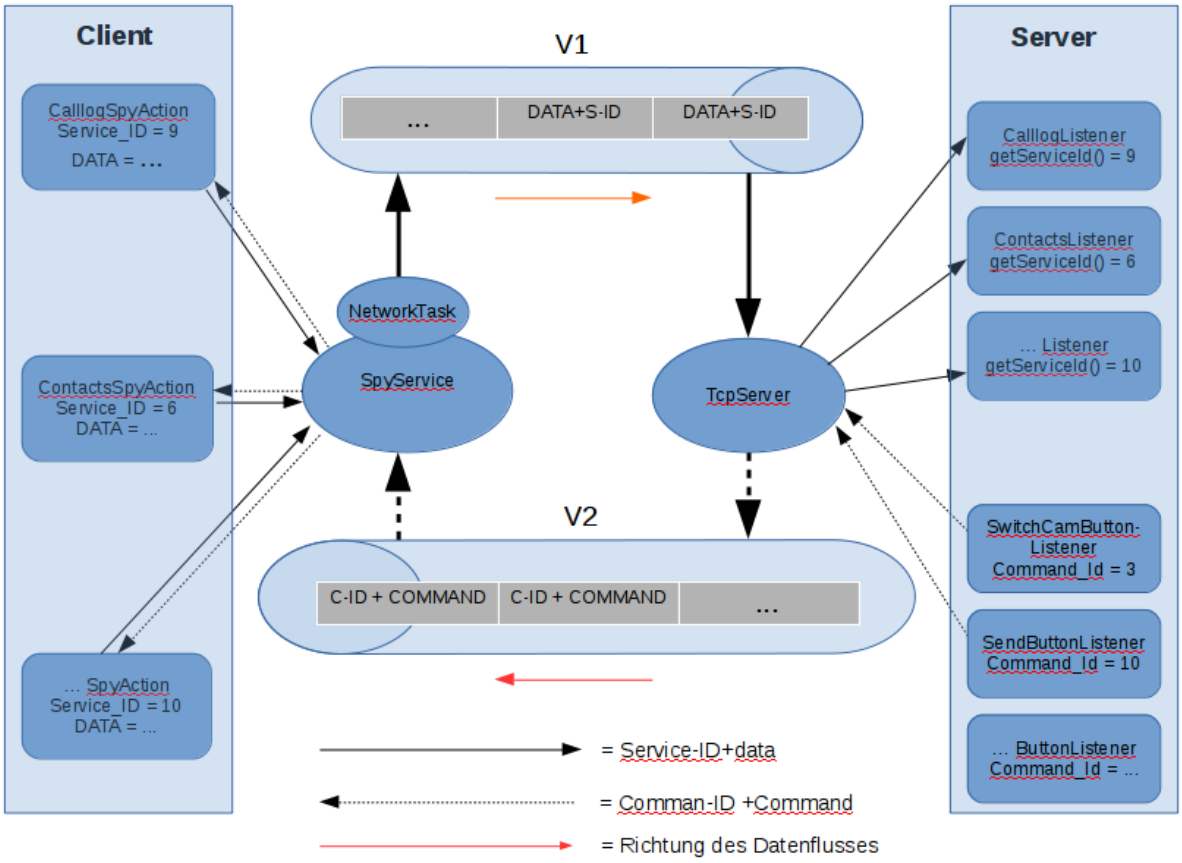


Abbildung 7: Kommunikationsfluss zwischen Client und Server

Kommunikation: Client zum Server

Die Übermittlung der ausgespähten Daten an den Server geschieht über V1.

Schritt 1:

Nachdem eine beliebige SpyAction die gewünschten Daten vom Gerät ermittelt und für die Weiterverarbeitung aufbereitet hat, erstellt sie eine NetworkMessage, die diese Daten und die ServiceId der SpyAction beinhaltet. Als nächstes wird diese in die OutputQueue vom NetworkTask eingefügt. Da der SpyService intern den NetworkTask verwendet, wird dieser Prozess, als langer schwarzer Pfeil von einer SpyAction zum SpyService, dargestellt.

Schritt 2:

Innerhalb des NetworkTasks wird die OutputQueue abgearbeitet. Nacheinander werden die enthaltenen NetworkMessages entnommen und ihre Bestandteile in den OutputStream von V1 geschrieben. Dieser Vorgang wird durch den kleinen schwarzen Pfeil vom SpyService zu V1 dargestellt. Die sequentielle Abarbeitung der Queueelemente ermöglicht, dass jeweils die ServiceId und die dazugehörigen Daten aufeinander folgend auf der Serverseite

ankommen. In der Abb. als graue Blöcke innerhalb von V1 zu sehen.

Schritt 3:

Wenn die Datenblöcke auf der Serverseite ankommen, in der Abb. als schwarzer Pfeil von V1 zum TCP-Server zu sehen, wird im TCP-Server als erstes die ServiceId aus dem Datenblock gelesen. Diese wird mit den Ids aller SpyActionListener abgeglichen. Bei Übereinstimmung werden die darauf folgenden Daten an den passenden SpyAction-Listener weitergeleitet. Dieser Schritt wird durch die langen durchgehenden schwarzen Pfeile, die vom TCP-Server zu den SpyActionListenern im Server zeigen, dargestellt.

Kommunikation: Server zum Client

Die Übermittlung von Steuerkommandos an das infizierte Gerät geschieht über V2.

Schritt 1:

Entscheidet sich der Angreifer ein Kommando abzusetzen, so wird dieses durch die Betätigung des jeweiligen Buttons in der Gui, unter Verwendung des zuständigen ButtonListeners und einer CommandSimpleCommand, zusammengebaut und für den Weitertransport an den TCP-Server übermittelt. Diese Aktion wird in der Abb. durch die gestrichelten Pfeile von den ButtonListenern zum TcpServer dargestellt. Die *SimpleCommand* enthält die Attribute *commandId* und *command*, in denen die jeweiligen Kommandobestandteile gespeichert sind. Die commandId muss die gleiche sein, wie die ServiceId der SpyAction, an die der Steuerbefehl gehen soll. Durch diese Eigenschaft kann das Kommando auf der Gegenseite eindeutig der zuständigen SpyAction zugewiesen werden. Nachdem das Kommando zusammengebaut ist, wird es mittels der *sendCommand(...)*-Methode in den OutputStream von V2 geschrieben. Dieser Schreibvorgang wird durch den dicken gestrichelten Pfeil von TCP-Server zu V2 in der Abbildung.

Schritt 2:

Wenn die Kommandonachrichten, (in der Abb. als graue Blöcke innerhalb von V2 dargestellt), auf der Client-Seite ankommen, werden ihre Bestandteile ausgelesen. In der Abb. ist dieser Vorgang als kleiner gestrichelter Pfeil von V2 zum SpyServer gekennzeichnet. Aus diesen Teilen wird eine vom Client verwendete SimpleCommandMessage erstellt und in die dafür vorgesehene InputQueue eingefügt. Innerhalb des SpyServices wird diese InputQueue sequentiell abgearbeitet. Dabei wird als erstes die enthaltene CommandId entnommen und mit der ServiceId aller SpyActions abgeglichen. Bei einer Übereinstimmung wird dann der eigentliche Steuerbefehl aus der Kommandonachricht ausgelesen und zur Weiterverarbeitung an die SpyAction übermittelt. Die langen gestrichelten Pfeile vom SpyService zur SpyAction im Client stellen diesen Ablauf dar.

Dieser Mechanismus ermöglicht durch die Verwendung eindeutiger Id-Attribute und deren Abgleich eine fehlerfreie und eindeutige Datenübertragung zwischen beiden Kommunikationspartnern.

5 VirusDoc ohne Internet-Permission

Anhand der Anwendung VirusDoc sieht man, dass es mittels einiger weniger Komponenten der Android API möglich ist eine App zu konstruieren, die eine gewisse "sichtbare"Funktionalität vortäuschen und ohne Wissen des Benutzers im Hintergrund Daten von seinem Gerät ausspionieren kann. Damit ist gezeigt, dass ein durchschnittlicher Benutzer nicht in der Lage ist mobile Malware als solche zu erkennen. Die einzige Chance dieser Benutzer ist es eine gewisse Grundskepsis gegenüber mobilen Applikationen zu entwickeln, die sie dazu veranlasst Berechtigungsanfragen beim Installationsvorgang genauer zu betrachten, zu hinterfragen und letztendlich zu entscheiden ob diese durch die Funktionalität der Applikation gerechtfertigt sind oder nicht.

Durch eine geschickte Programmierung und die Ausnutzung einiger Eigenschaften der Android API können auch vorsichtige Benutzer, die sich im Vorfeld über angefragte Berechtigungen einer App und deren Auswirkungen auf dem Gerät Gedanken machen, trotzdem getäuscht werden. Um dies zu verdeutlichen wird eine modifizierte Variante des VirusDoc entwickelt, die kein Internet-Zugang per Berechtigung einfordert, jedoch trotzdem in der Lage ist Daten über das Netzwerk an einen Angreifer zu übermitteln.

5.1 Der Client

Wie auch in der ersten Variante von VirusDoc bildet der Client den Teil der Anwendung, die auf dem mobilen Gerät läuft. Dieser beinhaltet nur zwei Komponenten die graphische Oberfläche zur Interaktion mit dem User und einen Hintergrund-Service zum Auslesen der Daten.

5.1.1 Die graphische Oberfäche

Die graphische Oberfläche setzt sich bei dieser modifizierten Version aus einem Start- und einem Scan-Screen zusammen und unterscheidet sich nur geringfügig zu der Originalversion. Der Screen für den Updatevorgang entfällt aus logischen Gründen, denn eine App ohne Internetzugang kann keinen Updatemechanismus bereitstellen. Der Update- wurde durch den *"Visit our Site"*-Button ersetzt. Dieser neue Button spielt eine zentrale Rolle in der App. Mit seiner Hilfe wird es möglich, den Default-Browser des Geräts aufzurufen und ausgespähte Daten unter Verwendung von Url-Attributen zu verschicken. Andere Komponenten der graphischen Oberfläche dienen, wie auch schon in der vorherigen Version, lediglich zur Ablenkung des Benutzers.

5.1.2 Der DataProviderService

Dieser Service liefert die benötigten Daten für die App. Mit den Methoden *getSmsData()*, *getFileData()* und *getGpsData()* werden SMS-Nachrichten, kleinere Dateien und GPS-Koordinaten vom Gerät ausgelesen. Diese Daten werden in geeigneten Attributen gespeichert und mit dem *Base64* Algorithmus kodiert. Die Kodierung hilft dabei mögliche Sonderzeichen, die in den Daten enthalten seien könnten, durch gewöhnliche Ascii-Zeichen zu ersetzen und ermöglicht somit das Verschicken der Daten als übergebene URL-Parameter. Ein angenehmer Nebeneffekt ist, dass der User keinen Klartext in der Url vorfindet. Schließlich werden die kodierten Daten durch das Senden einer Broadcast-Nachricht, zur Weiterverarbeitung an die *LauncherActivity* weitergeleitet.

In der *LauncherActivity* werden die Daten, von dem für Broadcast-Nachrichten vorgesehenen Receiver, entgegengenommen und durch eingebaute Logik in passende Arrays einsortiert. Sobald der App-User nun den *"Visit our Site"*-Button betätigt, werden durch den programmatischen Aufruf der Browser-App aus der LauncherActivity heraus, mehrere Tabs im Default-Browser des Geräts gestartet, insgesamt so viele wie für das angesammelte Datenvolumen der Arrays erforderlich ist. Ist die Gesamtgröße der augespähten Daten beispielsweise 16 KB, so werden 4 Tabs geöffnet, da pro Url in einem Tab ca. 5 KB verschickt werden können. Jedem Tab wird eine php-Seite in der Url übergeben, an die vorher zwei Parameter angehängt werden. Der erste *"n"* enthält eines der Arrays, mit den kodierten Daten als Inhalt. Im zweiten *"m"* wird der Typ des kodierten Textes übergeben. So kann *"m"*, beispielsweise *datei*, *gps* oder *sms* sein. Der zweite Parameter wird benötigt, um die vom Server entgegengenommenen Daten der Übersicht halber in passenden Dateien wieder zusammenzufügen.

5.2 Der SpyServer

Der SpyServer besteht aus einer HTML-Datei mit PHP und JavaScript Ergänzungen. Der sichtbare HTML-Teil der Seite soll dem User ein Unternehmensprofil oder Produktportfolio präsentieren, so dass der Benutzer glaubt ein reales Unternehmen bzw. dessen Internetauftritt vor sich zu haben.

Im PHP-Teil des Servers werden mit Hilfe des Get-Arrays, die übergebenen Url-Parameter ausgelesen. Der Wert des ersten Parameters *"n"* wird innerhalb einer geeigneten Methode Base64 dekodiert und mit Hilfe des zweiten Parameters *"m"* in eine extra dafür angelegte Datei gespeichert. Es werden insgesamt drei Dateien, jeweils für sms-, file- und gps-Daten angelegt und gefüllt.

Um den Benutzer nicht misstrauisch zu machen, kümmert sich nach der Übertragung der Daten ein javascript-script um die Schließung der geöffneten Browser-Tabs. Dies ge-

schieht in der Regel so schnell, dass ein Benutzer es gar nicht oder aller höchstens als ein unauffälliges und nicht beachtenswertes Fehlverhalten des Browsers wahrnehmen würde. Es werden alle Tabs geschlossen bis auf eins, dies soll dem Benutzer das tatsächliche Gefühl geben eine Seite aufzurufen.

Natürlich lassen sich mit diesem Verfahren nicht massenweise Daten ausspionieren. Auch Zugriff auf sensible Hardware wie Kamera oder Mikrofon oder größere Dateien, wie Videos oder Fotos, kann damit nicht realisiert werden, da das anfallende Datenvolumen viel zu groß wäre, um es in der Url zu übertragen. Allerdings lassen sich so andere wichtige Informationen, wie z.B. Kurznachrichten, Kontakte, GPS-Koordinaten, Browsereinträge, usw., die auch in kleinen Mengen eine große Aussagekraft besitzen und mit deren Hilfe man auf Gewohnheiten, soziales Umfeld und andere Aspekte der Privatsphäre schließen kann, problemlos ermitteln.

6 Sicherheitsmaßnahmen bei der Einführung von Software auf mobilen Geräten

Der Einsatz von Android-Geräten in einem Informationsverbund bringt eine Vielzahl an Gefährdungsstellen mit sich, sodass eine Menge an Sicherheitsaspekten beachtet werden muss, um einen sicheren Umgang mit diesen Geräten zu ermöglichen. Neben sicheren Entsperrmechanismen, Nutzung von bewährten Verschlüsselungsverfahren für Dateien und richtiger Konfiguration zur Vermeidung ungewollter Datenerhebung und -weitergabe durch vorinstallierte Anwendungen ist die Wahl der nachträglich installierten und verwendeten Software von hoher Bedeutung. Die Beachtung des letzten Punktes erhöht die Sicherheit, hinsichtlich dem Verbleib und der Manipulation, der auf dem Gerät befindlichen Daten und trägt allgemein dazu bei, einer Kompromittierung und Verbreitung von Malware entgegen zu wirken.

Aufgrund der in der Bachelorarbeit entwickelten Software werden in diesem Kapitel hauptsächlich Maßnahmen erläutert, die den letzten Punkt betreffen.

- Ein großes Problem ist, dass ein Benutzer/Mitarbeiter einmalig und direkt beim Installationsvorgang einer App entscheiden muss, ob die von ihr angeforderten Berechtigungen zur Erfüllung ihrer Funktionalität zwingend benötigt werden und aus diesem Grund gerechtfertigt sind oder nicht. Aus Mangel an Fachwissen sind viele User nicht in der Lage eine vollständige Abhängigkeit zwischen Berechtigungen und Funktion der App zu erkennen. Hinzu kommt, dass viele Berechtigungen nur oberflächlich bzw. stichpunktartig beschrieben werden, so dass deren komplexe Auswirkungen auf Dateien, Hardware und andere Systemkomponenten nicht leicht ersicht-

lich sind. Die IT-Abteilung bzw. Administration könnte hierbei den Mitarbeitern behilflich sein, indem sie eine Liste mit ausgewählten und geprüften Apps für oft nachgefragte Aufgaben und Dienste pflegt und herausgibt. Zusätzlich könnten interne/externe Schulungen bzw. Informationsveranstaltungen angeboten werden, um auf Gefahren bei der Nutzung dieser Geräte hinzuweisen.

- Um den Schutz vor schädlichen Komponenten und Funktionen einer App durch den Missbrauch von Berechtigungen noch weiter zu erhöhen, könnten Expertenmeinungen und Testergebnisse hinzugezogen werden. Diese könnte dabei helfen mögliche Sicherheitsrisiken, die von einer App ausgehen, aufzudecken und mögliche Alternativen aufzuzeigen.

- Als Installationsquelle für Applikationen sollte nach Möglichkeit nur der Google Play Store genutzt werden. Dieser bietet, aufgrund der durchgeführten Sicherheitstests, bei Aufnahme einer neuen App, ein solides Maß für die Sicherheit der Software, in Bezug auf schädliche Malware Komponenten an. Andere Installationsquellen, wie alternative App Stores oder Downloadplattformen, sollten nicht genutzt werden. Sie bieten keinerlei Maßnahmen zur Prüfung der angebotenen Anwendungen, was die Wahrscheinlichkeit, eine App mit schädlichem Inhalt zu beziehen, drastisch erhöht. Wird Google Play Store als einzige Quelle genutzt, sollte die Einstellung *"Installation von Apps aus unbekannten Quellen zulassen"* deaktiviert bleiben. Aber auch wenn man Apps aus dem Playstore bezieht, sollte man skeptisch bleiben und die Berechtigungsanfragen der Apps hinterfragen. Vor allem wenn eine App auf Ressourcen, wie GPS-Sensor, Webcam, Mikrofon oder Kontaktliste zugreifen möchte, sollte der Benutzer aufmerksam werden. Er sollte sich hinterfragen ob diese Anfragen tatsächlich für die Funktionalität der App benötigt werden und sich nötigenfalls nach einer Alternative umschauen. Der Grund für diese Vorsichtsmaßnahme ist, dass es in der Vergangenheit immer wieder Apps mit Malware-Inhalt in den Google Playstore geschafft haben und erst im Nachhinein, d.h. nachdem schon etliche Benutzter diese installiert und genutzt haben, entdeckt und entfernt wurden [Vgl. [7]]

- Je nachdem wie hoch das Bedürfnis nach Sicherheit ist, sollte bei gegebenen finanziellen, personellen und technischen Ressourcen ein unternehmensinterner App-Store realisiert werden. Der Vorteil dieser Alternative ist, dass Software entwickelt wird, die speziell an die Anforderungen der im Unternehmen vorhandenen Aufgaben zugeschnitten ist und keinen Overhead an Funktionen mit sich bringt, die sie möglicherweise unsicherer machen. Ein weiterer Punkt ist, das alle Anwendungen

vom eigenen vertrauenswürdigen Personal entwickelt werden, dass im Normalfall nicht darauf bedacht ist, dem eigenem Unternehmen Schaden hinzuzufügen. Außerdem kann bei einer solchen Herangehensweise schon beim Entwicklungsprozess der Fokus auf Sicherheit gelegt werden und dadurch eine lange Testphase zur Findung von Sicherheitslücken vermieden werden.

- Stehen die notwendigen Ressourcen nicht zur Verfügung und soll die eingesetzte Applikation trotzdem ein hohes Sicherheitsniveau aufweisen, kann im Vorfeld mit geeigneten technischen Mitteln wie Wireshark etc. das Kommunikationsverhalten von Anwendungen untersucht werden. Des Weiteren existiert eine Menge Tools, um den Quellcode aus den APK-Dateien der Apps zu extrahieren, so dass Quellcodeanalysen durchgeführt werden können. Diese Art von Untersuchungen geben die Möglichkeit Malware-Eigenschaften von Software frühzeitig zu erkennen und somit entsprechende Maßnahmen einzuleiten, um deren Bezug und Verwendung zu verhindern.

- Da die Berechtigungsanfrage für den vollständigen Netzwerkzugriff teilweise umgangen werden und eine Datenübertragung mittels der Browser-App durchgeführt werden kann, sollte das Verhalten dieser Anwendung dementsprechend genau beobachtet werden.

  Folgende Anhaltspunkte könnten dabei helfen:

  - Eine Url bietet die Möglichkeit, zusätzliche Informationen mit Hilfe von Parametern zu übertragen. Diese Parameter werden immer am Ende einer Url mit entsprechenden Werten eingefügt. Diese Eigenschaft ist sinnvoll und wird beispielsweise dazu benutzt, Steuerkommandos an einen Server zu übermitteln. Anhand der modifizierten Variante der VirusDoc App sieht man, dass diese Funktion auch missbraucht werden, um private Daten zu transportieren. Aus diesem Grund sollten Urls, die ein Benutzer automatisch mit Hilfe von Buttons oder Links aus Anwendungen heraus öffnet, immer nachkontrolliert werden. Bei ungewöhnlich langen Urls oder verschlüsseltem Parameterinhalt sollte man skeptisch werden und gegebenenfalls die Applikation, die diese Url aufruft, von seinem Gerät deinstallieren.
  - Beim Aufruf einer bestimmten Seite, mit Hilfe eines Links oder sonstigem Aufrufmechanismus aus einer App heraus, öffnet sich in der Browser-App eine große Anzahl an Tabs und schließt sich automatisch wieder.

  - Die Browser-App öffnet sich automatisch ohne jegliche Interaktion des Benutzers mit dem Gerät.

- Eine allgemeine Lösung bieten sogenannte Mobile-Device-Management Systeme mit deren Hilfe ein oder mehrere Administratoren in der Lage sind, Software- und Datenverteilung sowie Schutz der Daten auf mobilen Geräten vorzunehmen und zentral zu betreiben.

7 Ausblick

Die erste Version der Schulungssoftware steht am Ende meiner Bachelorarbeit fest. Diese kann in zukünftigen Projekten weiter ausgebaut und verbessert werden.
Beispielsweise könnte die Server-Anwendung so erweitert werden, dass eine Kommunikation mit mehreren Clients möglich ist oder sie eine Datenbankanbindung, für eine strukturierte und langfristige Speicherung von ausgespähten Daten, erhält. Weiterhin könnte eine Verschlüsselung der gesamten Kommunikation zwischen Server und Client implementiert werden, um die App-Aktivität besser zu verschleiern. Ein Mechanismus auf der Client-Seite könnte dafür sorgen, dass abgegriffene Daten nicht dauerhaft, sondern periodisch in bestimmten Zeitabständen an den Angreifer sendet. Denkbar wären auch weitere SpyActions, die zusätzliche Daten ausspionieren oder sonstige Angriffe auf das System ermöglichen.

8 Fazit

In den oberen Kapiteln werden die Eigenschaften, Bedrohungen und Risiken mobiler Malware benannt und erläutert. Des Weiteren werden das Täterprofil und dessen Motivation, sowie mögliche Sicherheitsmaßnahmen hinsichtlich der Einführung und Verwendung mobiler Geräte, besprochen. Mit der Implementierung der beiden VirusDoc-Applikationen und den zugehörigen Serveranwendungen, wurde ein Schulungskonzept konzipiert, dass bei Nutzern ein besseres Sicherheitsbewusstsein schaffen soll. In diesem Kapitel werden die verschiedenen Ergebnisse zusammenfasst und kritisch beurteilt.

Die Applikation VirusDoc zeigt, dass es für einen durchschnittlichen Benutzer schwierig ist, mobile Malware zu erkennen. Eine graphische Oberfläche ist kein Garant für eine vorhandene Funktionalität. Jede beliebige App kann, analog zur entwickelten Software, mit Hilfe einfacher grafischer Elemente, wie Fortschrittsbalken, Textfelder, Buttons etc., ein gewisses funktionales Verhalten vortäuschen. Zu erkennen, ob eine tatsächliche oder eben nur fiktionale Funktionalität vorliegt, ist ohne tief gehendem Wissen oder der Unterstützung eines Experten nahezu unmöglich.

Die entwickelte App-Malware wird als eine Antivirus-Anwendung getarnt. Man kann aber auch jede andere Kategorie wie Spiele, Multimedia, Social Media, Sport etc. verwenden, um das Vertrauen eines Users zu erschleichen. Dies ist eine weitere gefährliche Eigenschaft mobiler Malware. Diese Besonderheit erschwert das Aufspüren und Erkennen schädlicher Anwendungen, da man im Vorfeld nicht weiß in welcher Kategorie primär gesucht werden muss.

Um mögliche Auswirkungen der implementierten Malware besser darstellen zu können, wird im ersten Teil vom VirusDoc bewusst eine große Anzahl an Berechtigungen eingefordert, um somit an eine Vielzahl sensitiver Daten zu kommen. Die enorme Menge und Variabilität der Daten und die Tatsache, wie leicht diese ausgespäht werden können, sollen erstens eine Schockwirkung bei einem Schulungsteilnehmer auslösen und zweitens der Software die Eigenschaft verleihen die verschiedenen Gefährdungen und Risiken aus dem Kapitel Bedrohungsanalyse darzustellen.

Die modifizierte Variante der Anwendung VirusDoc demonstriert, dass das Berechtigungskonzept hinsichtlich der Berechtigung “vollständiger Netztwerkzugriff“ umgangen werden kann. Daraus kann man folgern, dass dieses Konzept zumindest teilweise irreführend ist. Es sichert dem Benutzer nämlich fälschlicherweise zu, dass eine kritische, für den Ausspionierungsvorgang fast unentbehrliche und oft verwendete Ressource, wegen fehlender Zugriffsberechtigung nicht genutzt wird. Eine Applikation ist jedoch trotzdem in der Lage, durch den Aufruf der Browser-Anwendung Daten über das Netzwerk an einen Angreifer

zu übertragen, obwohl sie aufgrund nicht vorhandener Berechtigung nicht die Erlaubnis dafür besitzt.

Beim kritischen Rückblick auf die Ergebnisse der Arbeit muss man sagen, dass alle vorgegebenen Anforderungen an das Projekt, in der entwickelten Software umgesetzt werden. Ein Dozent ist mit dieser Schulungssoftware in der Lage, alle Gefährdungen und die daraus resultierenden Risiken, die im Kapitel Bedrohungsanalyse aufgelistet werden, nachzubilden und die dabei entwendeten Daten anschaulich zu präsentieren. Darüber hinaus kann der Unterschied gegenüber üblicher Desktop-Malware verdeutlicht werden, da in den entwickelten Applikationen einige wichtige Merkmale mobiler Schadsoftware aufweist.

Negativ zu vermerken bleibt, dass vor allem die erste Version der App eine sehr umfangreiche Liste an Berechtigungen einfordert, was in der Realität dazu führen könnte, dass ein User aus Sicherheitsbedenken diese App nicht installiert. Der Grund für diese Eigenschaft wurde einige Abschnitte weiter oben genannt und erläutert. Im Nachhinein betrachtet wäre es eventuell besser gewesen, mehrere kleinere Applikationen zu schreiben, die aufgrund ihrer Funktionalität die eingeforderten Berechtigungen nachvollziehbarer gemacht und zusätzlich Daten ausspioniert hätten. Beispielsweise einen kleinen Dateimanager oder ein einfaches Diktiergerät, um Zugriff auf den Speicher oder Mikrofon zu rechtfertigen.

Die Erstellung der Software selbst hat mir viel Spaß gemacht. Der Prozess von der ersten Idee über die Entwicklung eines Konzeptes bis hin zur programmiertechnischen Umsetzung an deren Ende dann z.B. ein neues Spy-Feature der VirusDoc-Apps oder eine neue Steuer-, Bearbeitungs- oder Betrachtungskomponente der Serveranwendungen stand, war spannend und lehrreich. Außerdem erweiterten und verbesserten sich meine persönlichen Kenntnisse, im Hinblick auf die Java- und App-Programmierung.
Die Erkenntnis darüber, wie leicht sensitive Daten von einem Smartphone oder Tablet-Pc mit Hilfe einer App entwendet werden können, hat bei mir ein neues Sicherheitsbewusstsein hinsichtlich dieser Geräte geschaffen.

Abschließend kann man sagen, dass die Nutzung von Applikationen auf mobilen Geräten zahlreiche Risiken in Bezug auf Entwendung und Missbrauch persönlicher Daten birgt, wenn man nicht mit entsprechender Vorsicht und Aufmerksamkeit mit diesen Geräten umgeht und willkürlich alle möglichen, teilweise unnötigen Anwendungen, installiert. Nutzer, die Applikationen aus unsicheren Quellen beziehen, sind besonders gefährdet, denn die so bezogene Software unterliegt keiner Überprüfung bzw. Qualitätssicherung und vergrößert somit die Gefahr einer Infizierung.

Zudem belegt eine Vielzahl an Untersuchungen, dass mobile Malware immer weiter zunimmt und vor allem Geräte mit dem Android OS, auf die immerhin über 90% aller

Angriffe abzielen, sehr stark von dieser Entwicklung betroffen sind. [Vgl. [1]] Wie leicht Angreifer potentielle Opfer umfassend und unentdeckt ausspionieren können hat die in dieser Bachelorarbeit entwickelte Software gezeigt. Vor diesem Hintergrund wird klar, dass es es lohnenswert ist die Entwicklung auf diesem Gebiet der IT-Sicherheit auch aus eigenem Interesse zu verfolgen. Denn fast jeder von uns besitzt und benutzt mobile Geräte, ob privat oder bei der Arbeit, und kann somit Opfer mobiler Schadsoftware werden.

9 Quellenverzeichnis

[1] http://mobiforge.com/research-analysis/mobile-software-statistics-2014

[2] http://www.trendmicro.de/media/br/ebook-when-android-apps-want-more-de.pdf

[3] https://www.test.de/Datenschutz-bei-Apps-Welche-Apps-Ihre-Daten-ausspaehen-4378643-0/

[4] http://www.heise.de/ct/artikel/Selbstbedienungsladen-Smartphone-1464717.html

[5] http://www.snoopwall.com/wp-content/uploads/2014/10/Flashlight-Spyware-Appendix-2014.pdf

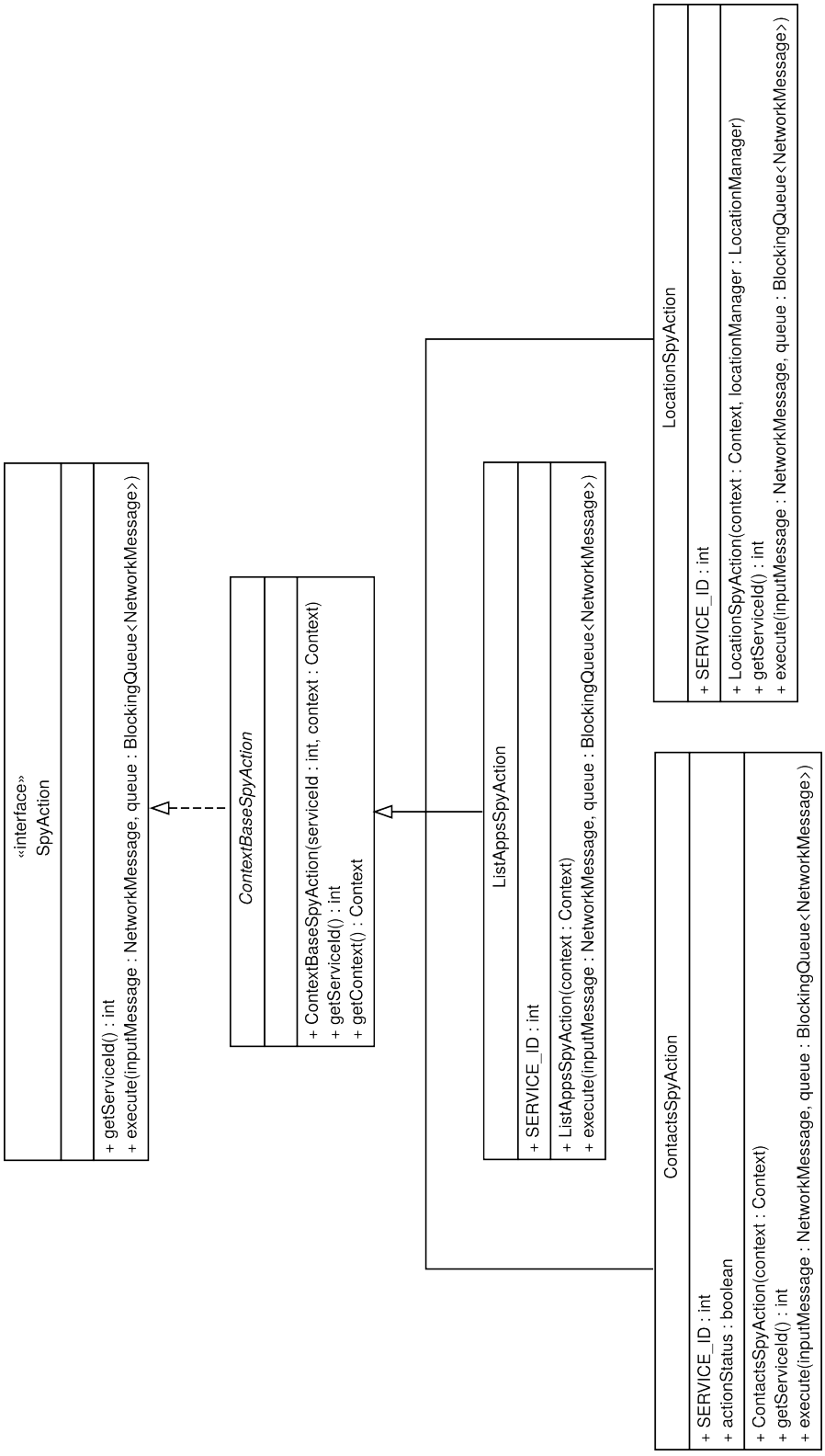
[6] http://developer.android.com/guide/components/activities.html

[7] https://blog.lookout.com/blog/2013/04/19/the-bearer-of-badnews-malware-google-play/

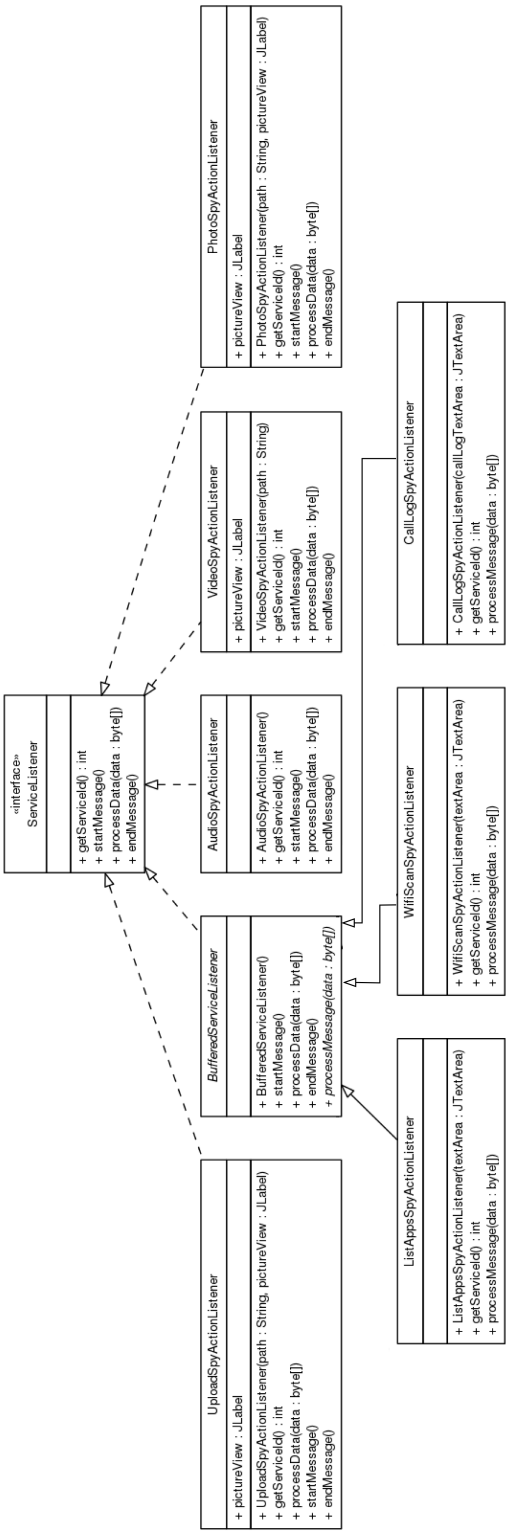
[9] http://surveillance.rsf.org/en/hacking-team/

Alle URL-Adressen wurden am 06.03.2015 auf ihre Erreichbarkeit geprüft.

A Klassendiagramm der SpyAction-Klassen



B Klassendiagramm der SpyActionListener-Klassen



Erklärung

Hiermit versichere ich, dass ich meine Abschlussarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Datum: .....

(Unterschrift)