IT-forensische Erkennung modifizierter Android-Apps

Sebastian Becker[1] · Hans Höfken[1]
Philip Schütz[2] · Marko Schuba[1]

[1]FH Aachen
sebastian.becker4@alumni.fh-aachen.de
{hoefken | schuba}@fh-aachen.de

[2]Landeskriminalamt NRW
philip.schuetz@polizei.nrw.de

Zusammenfassung

Malware auf Smartphones ist ein Problem, dem auch Strafverfolgungsbehörden immer häufiger gegenüberstehen. Insbesondere Telefone, bei denen potentiell schadhafte Apps zu einem finanziellen Schaden geführt haben, finden sich auf den Schreibtischen der Polizei wieder. Dabei müssen die Ermittler möglichst schnell und gezielt erkennen können, ob eine App tatsächlich schadhaft manipuliert wurde, was manipuliert wurde und mit wem die App kommuniziert. Klassische Malware-Erkennungsverfahren helfen zwar bei der generellen Erkennung schadhafter Software, sind aber für die polizeiliche Praxis nicht geeignet. Dieses Paper stellt ein Programm vor, welches gerade die forensischen Fragestellungen berücksichtigt und so für den Einsatz in der Strafverfolgung in Frage kommt.

1 Einleitung

Ein wichtiger Erfolgsfaktor moderner Smartphones ist die Vielzahl verfügbarer Anwendungen. Mit der Vielfalt dieser sogenannten Apps steigt jedoch auch der potentielle Missbrauch, insbesondere dann, wenn Angreifer durch den Missbrauch einen finanziellen Vorteil erlangen können. So gibt es Untersuchungen, dass für 77% der 50 am häufigsten heruntergeladenen Apps im Google Play Store eine gefälschte und meist schädliche Version im Internet vorhanden ist [LuYa14]. Die schädliche Funktionalität reicht von einfachem Datendiebstahl über entfernten Gerätezugriff bis hin zur Nutzung kostenpflichtiger Dienste wie Anrufe oder SMS. Premium-SMS kosten in der Regel bis zu fünf Euro pro SMS. Da viele Nutzer ihre Mobilfunkrechnung erst am Ende des Monats oder sogar überhaupt nicht kontrollieren, kann über einen längeren Zeitraum leicht ein Schaden von mehreren Hundert Euro entstehen, ohne dass dieser bemerkt wird. Die Modifikation von Android-Apps ist dabei vergleichbar einfach: die Original-App wird einfach um schadhaften Code ergänzt, beispielsweise durch die Verwendung entsprechender Tools, und in Umlauf gebracht [Trend14]. Für den Nutzer sichtbare Informationen wie Icons und Name der App bleiben ebenso unverändert wie die Hauptfunktionen des Originals; die schädliche Funktionalität spielt sich im Hintergrund ab und bleibt somit – zumindest vorerst – unentdeckt. Einziges Manko für die Cyberkriminellen ist die in den meisten Fällen notwendige Änderung der Berechtigungen der App [Goog16], dem jedoch die Unbekümmertheit der

P. Schartner (Hrsg.) · D•A•CH Security 2016 · syssec (2016) 120-125.

Benutzer entgegensteht. So installieren viele Anwender Apps, ohne auf die geforderten Berechtigungen zu achten bzw. akzeptieren diese aus Unkenntnis bzw. aus dem Glauben heraus, dass schon nichts passieren wird. Dabei nutzen Angreifer insbesondere aus, dass Benutzer weniger seriöse Bezugsquellen der App und kritische Berechtigungen leichter akzeptieren, wenn der Bezug über den Google Play Store teurer oder nicht mehr möglich ist. Beispielsweise war das Spiel Flappy Bird eines der beliebtesten Spiele im Google Play Store, wurde aber ab einem gewissen Zeitpunkt durch den Entwickler nicht weiter über den Google Play Store angeboten. Das Interesse an dem Spiel war jedoch weiterhin sehr groß und diese Chance wurde von Cyberkriminellen genutzt, um modifizierte Flappy Bird Versionen ins Internet zu stellen, welche z.B. im Hintergrund Premium-SMS verschickten [LuYa14].

Strafverfolgungsbehörden wie das Landeskriminalamt NRW haben täglich mit Fällen zu tun, bei denen Smartphones auf Schadsoftware untersucht werden müssen. Aus IT-forensischer Sicht stellt sich dabei nicht nur die Frage, ob eine App gewisse, für den Benutzer möglicherweise negative Funktionen aufruft; denn auch das Sammeln von Daten bzw. das Versenden von Premium-SMS kann legitim sein. Vielmehr geht es um die besondere Fragestellung, ob dem Benutzer statt der Original-App eine schadhaft modifizierte App untergeschoben wurde und welche möglicherweise schädlichen Funktionen ergänzt wurden. Derzeit muss ein IT-forensischer Ermittler dazu jede App einzeln prüfen und herausfinden, ob diese im Vergleich zum Original manipuliert wurde, auch in Bezug auf die Berechtigungen der App. Hierzu werden die Apps von dem zu überprüfenden Smartphone gesichert und anschließend untersucht. Dieser Prozess ist sehr aufwändig, was daran liegt, dass auf dem Smartphone häufig 20 bis 30 Apps installiert sind. Um Modifikationen zu erkennen, muss der Ermittler die Original-App aus dem Google Play Store mit der zu überprüfenden App vergleichen. Selbst wenn die App noch im Google Play Store angeboten wird, kann dies den Ermittler vor Probleme stellen, da immer nur die allerneuste App-Version im Store angeboten wird, welche nicht mit der App-Version auf dem zu untersuchenden Gerät übereinstimmen muss.

Um die Arbeit der Ermittler zu erleichtern, wurde ein Programm entwickelt, welches die Untersuchung von modifizierten Apps erleichtert. Dazu soll das Programm weitgehend automatisch die offizielle App aus dem Google Play Store mit der zu überprüfenden App vergleichen. Damit das Programm die Historie der offiziellen App-Versionen berücksichtigen kann, werden im Internet verfügbare, seriöse Online-Datenbanken hinzugezogen, die alte Versionen der Google Play Store Apps archiviert haben, z.B. [Upto16]. Die Vergleichs-Apps werden vom Programm in verschiedenen Versionen heruntergeladen und in einer lokalen Datenbank gespeichert. Nach dem Vergleich der mit der potentiell modifizierten App und einer tiefergehenden statischen Analyse generiert das Programm einen Bericht, welcher alle für den Ermittler relevanten Informationen enthält.

2 Erkennung von Malware auf Smartphones

Es gibt verschiedenen Methoden, mit denen Smartphone-Apps auf Schadcode hin überprüft werden können. Einen geräteunabhängigen Überblick über Malware-Erkennung geben beispielsweise Egele et al. [ESKK12], wohingegen Schmidt bzw. Alazab und Batten Verfahren spezifisch für Mobiltelefone erläutern [Schm11], [AlBa15]. Klassisch gehören zu diesen Methoden signatur-basierte Verfahren sowie statische und dynamische Analyseverfahren.

Signatur-basierte Erkennungsmethoden scannen die zu prüfende App auf der Basis von Signaturen bekannter Schadprogramme. Dies kann durch eine entsprechende Anti-Viren-App auf

dem Gerät selbst geschehen oder die App wird an einen entsprechenden Online-Webdienst geschickt und dort geprüft, z.B. [NVIS16] für Android. Signatur-basierte Verfahren geben keine Information darüber, wie eine App manipuliert wurde, ob – abweichend vom Original – zusätzliche Berechtigungen erforderlich sind oder zu welchen Systemen die App Kontakt aufnimmt, weshalb sie für die hier betrachtete IT-forensische Fragestellung ungeeignet sind. Weiterhin sind signatur-basierte Verfahren nicht in der Lage, neue Schadsoftware zu erkennen.

Die Analyse der von einer App angeforderten Berechtigungen war in der Vergangenheit Thema verschiedener Projekte, so zum Beispiel das Programm APK Auditor [TAA15]. Das Programm analysiert alle auf einem Android-Gerät installierten Apps und ihre Berechtigungen, indem die zu einer installierten entsprechende APK-Datei aus dem Google Play Store geladen und analysiert wird. Alle Ergebnisse werden in einer zentralen Datenbank gespeichert, so dass auf die Analyse-Ergebnisse älterer Apps zugegriffen werden kann. Schon auf dem Gerät installierte, manipulierte Apps können nicht erkannt werden, da das Programm diese Funktion nicht anbietet. Zudem muss der APK-Auditor-Client auf dem zu untersuchenden Gerät installiert sein und Zugriff auf das Internet gestattet werden, was eine forensisch korrekte Untersuchung unmöglich macht.

Einen ähnlichen Ansatz verfolgt das Projekt von Almin und Chatterjee [AC15]. Auch hier muss eine App auf dem zu untersuchenden Gerät installiert werden, die Analyse findet jedoch lokal auf dem Gerät statt. Jedoch wurde die App speziell für Android Version 4.2.2 entwickelt, so dass sie nicht auf jedem Gerät genutzt werden kann. Alle Apps des zu untersuchenden Geräts müssten auf einem Test-Gerät oder einen Emulator mit der unterstützten Android-Version installiert werden.

Bei statischen bzw. dynamischen Analysen wird der Code bzw. das Verhalten der App untersucht. Die statische Analyse erfordert normalerweise die Verfügbarkeit des Quellcodes des Programms oder das Reverse Engineering desselben, z.B. durch den Einsatz von Disassemblern wie IDA Pro [IDAP16]). Eine rein manuelle statische Analyse ist sehr zeitaufwändig und daher für einen forensischen Ermittler nicht brauchbar, weshalb nur (teil-) automatisierte Verfahren in Betracht gezogen werden können. Ein entsprechender Ansatz wird von Junaid et al. präsentiert [JLK16], hier werden sogar fortgeschrittene Methoden der statischen Quellcode-Analyse, wie zum Beispiel das Reverse Engineering des Life-Cycle-Modells, diskutiert. Jedoch können im vorgestellten Zustand nur zwei Typen von schädlichem Verhalten erkannt werden.

Eine dynamische Analyse kann entweder auf dem Gerät mit laufender App selbst durchgeführt werden, z.B. durch Überwachung von Funktionsaufrufen, Funktionsparameter-Analyse oder Überwachung des Informationsflusses (vgl. erneut [ESKK12]), oder die Kommunikation des Systems mit der Umgebung wird überwacht, z.B. durch Überwachung der Netzwerkkommunikation der App über die Mobilfunk-Schnittstellen [SBHS13]. Ein Problem dieses Ansatzes ist, dass Malware sich nicht täglich bzw. stündlich offenbart. Eine Malware, die z.B. jeden 13. Tag im Monat eine Premium-SMS sendet, wird somit erst bei einer Langzeitanalyse sichtbar. Die App müsste monatelang untersucht werden, bevor Ergebnisse vorliegen würden, was dieses Verfahren in vielen Fällen unbrauchbar macht.

3 Programm zur Erkennung modifizierter Apps

Aus der Diskussion wird deutlich, dass derzeitig verfügbare Malware-Erkennungsmethoden zur Erkennung modifizierter Apps im Sinne IT-forensischer Ermittlungen nicht immer effizient sind, weshalb ein anderer Ansatz gewählt und in Kooperation mit dem Landeskriminalamt

NRW in einem Programm umgesetzt wurde. Dabei werden spezifisch die in Online-Datenbanken verfügbaren, ursprünglich aus dem Google Play Store stammenden App-Versionen genutzt, um die Modifikation einer zu untersuchenden App mit der Original-App-Version zu erkennen. Anschließend wird eine automatisierte statische Analyse durchgeführt und das Ergebnis in einem Bericht dargestellt. Der Ablauf ist in Abbildung 1 schematisch dargestellt.

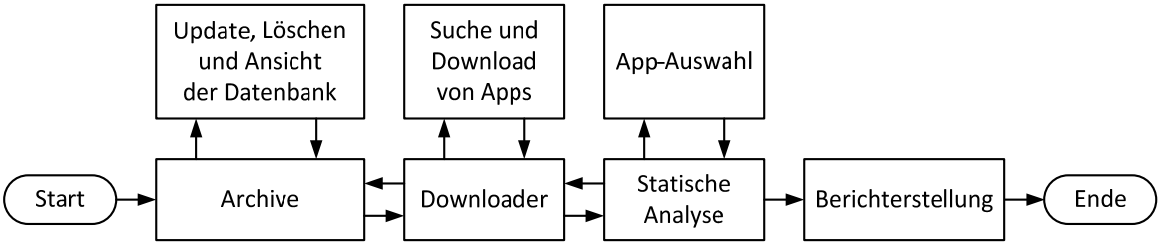

Abb. 1: Ablauf des Programms zur Erkennung und Bewertung modifizierter Apps

Nach dem Programmstart wird zunächst überprüft, ob sich die Vergleichs-App zu der zu untersuchenden App bereits in der Programmdatenbank (dem Archive) befindet. Sollte dies nicht zutreffen, kann die Vergleichs-App in verschiedenen Versionen im sogenannten Downloader heruntergeladen werden. Im Programmteil „Statische Analyse“ können im Anschluss eine oder mehrere verdächtige Apps des zu untersuchenden Geräts mit den jeweiligen Vergleichs-Apps verglichen werden. Dazu werden die Apps jeweils entpackt und die enthaltene Apk-Datei an das Tool Apktool [APKT16] übergeben, welches u.a. die Datei apkTool.yml erstellt, in der Daten wie die App-Version und das App-Package enthalten sind. Daneben generiert das Tool auch die AndroidManifest.xml-Datei, welche später zur Analyse der App-Berechtigungen wichtig ist. Zunächst wird jedoch auf der Basis der App-Version ein Datenbankaufruf generiert, der prüft, ob sich die Vergleichs-App mit gleicher Version in der Datenbank befindet. Ist dies nicht der Fall, kann der Vorgang abgebrochen werden oder die Analyse kann ohne einen Vergleich auf Modifikation weiter ausgeführt werden. Falls eine Vergleichs-App vorhanden ist, wird ein SHA-256 Hashwert generiert und mit der App aus der Datenbank verglichen. Wenn die Hashwerte identisch sind, wird fortgefahren, als wäre keine Vergleichs-App vorhanden gewesen. Sollten die Hashwerte nicht identisch sein, ist die App eine Modifikation des Originals und die Unterschiede zwischen den Apps werden in einer statischen Analyse detaillierter ausgearbeitet.

Zu Beginn der statischen Auswertung werden alle Bild-, Video- und Audiodateien aus der jeweiligen App extrahiert und für eine weitere Analyse in unterschiedlichen Ordnern abgelegt. Nach dem erfolgreichen Extrahieren der APK-Datei, wird die classes.dex-Datei freigelegt. Mit Hilfe der Programme DexToJar [DEXT16] und JDCore [JDCO16], welche auch bei mehrfacher Dekompilierung derselben classes.dex-Datei konstanten Quellcode generieren, wird der Javacode generiert. Anschließend wird der Quellcode der App geparst, wobei verdächtige Elemente (Telefonnummern, IP-Adressen, URLs, Email-Adressen etc.) für eine nachfolgende, manuelle Analyse extrahiert werden. Danach werden die Berechtigungen der App ausgewertet. Hierbei nutzt das Programm die Sicherheitslevel, welche Google in der Entwickler-Dokumentation für Android angibt (vgl. [AnDe16]). Auf Basis dieser Einstufung kann ein Risiko, dass bei der Benutzung der App durch den Anwender besteht, berechnet werden, was dem Ermittler bei der Beurteilung der Gefährlichkeit der App helfen kann. Alle Daten werden in einem Bericht zusammengefasst und abgespeichert (siehe Beispiel in Abbildung 2).

Testprotokoll

FH AACHEN UNIVERSITY OF APPLIED SCIENCES

POLIZEI Nordrhein-Westfalen Landeskriminalamt

App Daten:

AppName	com_devuni_flashlight-5.2.4.apk	**AppVersion**	5.2.4
AppPackage	com.devuni.flashlight	**HashWert**	6df1e147dc8430355ad043369bc40d2121b1294

Das Risiko, dass diese App Malware enthält, liegt bei: 32,727 %

Es wurden 11 Permissions & Packages gefunden...

Hohes Risiko: (18,182%):

Normales Risiko (54,545%):

Unbekannte Permissions & Packages: (27,273%)

Der Quellcode wurde durchlaufen und folgende Elemente gefunden:

Mögliche Telefonnummern:

/com/devuni/flashlight/d.java: +49123456789

Mögliche Ip-Addressen:

/com/google/android/gms/ads/b/a.java: 192.168.178.2

Mögliche URL Adressen:

Mögliche Email Adressen:

Es gab einen Unterschied zwischen der Datenbank und der auszuwertenden App:

Die Datei: res\drawable\screen_icon.png wurde in der zu Überprüfenden APK gelöscht!

Die Datei: res\drawable\test.txt wurde in der zu Überprüfenden APK hinzugefügt!

Die Datei: com\devuni\flashlight\b.javawurde verändert!

Die Datei: com\devuni\flashlight\d.javawurde verändert!

Die Datei: com\devuni\flashlight\MainActivity.javawurde verändert!

Die Datei: com\google\android\gms\ads\b\a.javawurde verändert!

Abb. 2: Beispiel-Bericht für eine modifizierte App

In einem Programmablauf können mehrere verdächtige Apps gleichzeitig bearbeitet werden, abhängig von der Größe der Apps kann der Analyseprozess mehrere Minuten pro App dauern, da pro Handy durchschnittlich 20-30 Apps untersucht werden, benötigt eine Analyse aller Apps eines Handys maximal wenige Stunden und läuft unbeaufsichtigt ab. Würde der Forensiker alle Schritte, die durch das Programm automatisch ablaufen, manuell oder teilautomatisiert ablaufen lassen und als Bericht zusammenfassen, würde die Untersuchung mehrere Tage benötigen.

4 Zusammenfassung

Die Erkennung von modifizierten Smartphone-Apps und ihrer Funktionsweise ist eine wichtige Aufgabe IT-forensischer Ermittler. Derzeit stehen dazu keine geeigneten Tools zur Verfügung, was die Arbeit gerade der polizeilichen Ermittler stark erschwert. Das in diesem Paper vorgestellte Programm erlaubt einem IT-Forensiker modifizierte Apps schnell zu erkennen und gibt erste Ansatzpunkte für eine genauere manuelle Untersuchung und beschleunigt so die Ermittlung in Betrugsfällen mit Smartphones.

Literatur

[LuYa14] S. Luo, P. Yan: Gefälschte Apps – Legitimität vortäuschen, Trend Micro Forschungsbericht (2014).

[Tren14] Trend Micro: Malware in Apps' Clothing: A Look at Repackaged Apps, Online (2014), http://www.trendmicro.com/vinfo/us/security/news/mobile-safety/malware-in-apps-clothing-a-look-at-repackaged-apps (abgerufen am 03/2016).

[Goog16] Google Play-Hilfe: App-Berechtigungen bis Android 5.9 prüfen, Online (2016), https://support.google.com/googleplay/answer/6014972, (abgerufen 03/2016).

[Upto16] Uptodown, Online (2016), http://de.uptodown.com/android (abgerufen 03/2016).

[ESKK12] M. Egele, T. Scholte, E. Kirda, C. Kruegel: A Survey on Automated Dynamic Malware Analysis Techniques and Tools, Journal ACM Computing Surveys, Volume 44, Issue 2 (2012) 1-49.

[Schm11] A. Schmidt: Detection of Smartphone Malware, Dissertation, TU Berlin (2011).

[AlBa15] M. Alazab, L. Batten: Survey in Smartphone Malware Analysis Techniques. In: M. Dawson, M. Omar: New Threats and Countermeasures in Digital Crime and Cyber Terrorism (2015) 105-130.

[NVIS16] NVISO AppScan, Online (2016), https://apkscan.nviso.be/, (abgerufen am 28. März 2016).

[TAA15] K. A. Talha, D. I. Alper, C. Aydin: APK Auditor: Permission-based Android malware detection system, Digital Investigations Magazine, Volume 13 (2015) 1-14

[AC15] S.B. Almin, M. Chatterjee: A Novel Approach to Detect Android Malware, Procedia Computer Science, Volume 45 (2015), 407-417

[JLK16] M. Junaid, D. Liu, D. Kung: Dexteroid: Detecting malicious behaviors in Android apps using reverse-engineered life-cycle models, Computers & Security, Volume 59 (June 2016), 92-117.

[IDAP16] IDA Pro, Online (2016), https://www.hex-rays.com/index.shtml, (abgerufen am 28. März 2016).

[SBHS13] P. Schütz, M. Breuer, H. Höfken, M. Schuba: Malware Proof on Mobile Phone Exhibits Based on GSM/GPRS Traces, Proceedings of CyberSec 2013, Kuala Lumpur, Malaysia, (2013).

[APKT16] Apktool: A tool for reverse engineering Android apk files, Online (2016), https://ibotpeaches.github.io/Apktool/, (abgerufen 03/2016).

[AnDe16] Android Developers: Manifest-permission, Online (2016), http://developer.anroid.com/reference/android/Manifest.permission.html, (abgerufen 03/2016).

[DEXT16] DexToJar: A tool for decompile dex files to jar files, Online (2016) https://github.com/pxb1988/dex2jar, (abgerufen 05/2016).

[JDCO16] JD-Core: A tool for decompile jar files to Javacode, Online (2016), http://jd.benow.ca/, (abgerufen 05/2016).