Zertifizierte Datensicherheit für mobile Anwendungen

Conference Paper · January 2014

CITATIONS
0

READS
61

9 authors, including:

Karsten Sohr
Universität Bremen
48 PUBLICATIONS **487** CITATIONS
SEE PROFILE

Steffen Bartsch
Technische Universität Darmstadt
24 PUBLICATIONS **83** CITATIONS
SEE PROFILE

Melanie Volkamer
Karlsruhe Institute of Technology
236 PUBLICATIONS **947** CITATIONS
SEE PROFILE

Bernhard J. Berger
Universität Bremen
10 PUBLICATIONS **40** CITATIONS
SEE PROFILE

Zertifizierte Datensicherheit für Android-Anwendungen auf Basis statischer Programmanalysen

Steffen Bartsch[1], Bernhard J. Berger[2], Eric Bodden[3], Achim D. Brucker[4], Jens Heider[3], Mehmet Kus[5], Sönke Maseberg[6], Karsten Sohr[2], Melanie Volkamer[1]

[1]Technische Universität Darmstadt
[2]Universität Bremen
[3]Fraunhofer-Institut für Sichere Informationstechnologie, Darmstadt
[4]SAP AG, Karlsruhe
[5]OTARIS Interactive Services GmbH, Bremen
[6]datenschutz cert GmbH, Bremen

Abstract: Smartphones erfreuen sich einer stetig wachsenden Beliebtheit. Ein Grund hierfür ist die Vielzahl verschiedenster mobiler Anwendungen. Mit den Chancen, die sich hierdurch für den Benutzer, aber auch Organisationen bieten, sind Risiken verbunden, die beispielsweise zu einer Verletzung der Privatsphäre führen können. In diesem Beitrag diskutieren wir, wie statische Programmanalyse dabei helfen kann, Android-Anwendungen bzgl. der Sicherheit zu zertifizieren.

1 Einleitung

Smartphones finden eine immer weitere Verbreitung sowohl im privaten als auch im geschäftlichen Umfeld; laut Bitkom besitzt jeder dritte Deutsche ein Smartphone – Tendenz steigend. Wesentlich für den Erfolg von Smartphones ist die Möglichkeit, mobile Anwendungen (sog. „Apps“) bequem aus Markets zu installieren. Das Angebot im Android Market „Google Play“ umfasst inzwischen mehr als 1 Mio. mobile Anwendungen, die insgesamt über 25 Mrd. Mal heruntergeladen wurden. In jedem Monat kommen ca. 30.000 neue Anwendungen dazu. Auch in speziellen Markets für Geschäftskunden, wie er z.B. von der SAP AG betrieben wird, nimmt die Anzahl von Geschäfts-Apps rasant zu. Mit den Chancen dieser Entwicklung gehen jedoch große Risiken einher, vor allem durch die wachsende Zahl mobiler Anwendungen mit oft unbekannter Herkunft. Hierdurch steigt die Gefahr der Verbreitung von Schadsoftware, die sich beispielsweise als nützliche Anwendung tarnt [1]. Zudem können Schwachstellen in Apps von Angreifern als Einstiegspunkte genutzt werden, um Zugriff auf Firmendaten zu erhalten. Dass Schwachstellen verstärkt in Apps auftreten, zeigen die aktuellen Berichte über Sicherheitsprobleme des WhatsApp-Messengers und die fehlerhafte Implementierung der SSL-Verschlüsselung quer durch einen großen Anteil von sicherheitskritischen Apps [2].

Aufgrund dieser Entwicklung ist es erforderlich, das Thema „Sicherheitsanalyse von mobilen Anwendungen“ **grundlegend und umfassend** zu bearbeiten und eine Analyse-

und Zertifizierungsplattform für Apps zu entwickeln. Nachfolgend skizzieren wir, wie eine solche Plattform aussehen könnte, und beschreiben einen statischen Analyse-Ansatz für Android Apps, der als Basis der Zertifizierungsplattform dienen kann. Wir diskutieren unseren Ansatz anhand von Android, da dieses aktuell die am weitesten verbreitete Smartphone-Plattform ist.

Die eigentlich wissenschaftlich spannende und herausfordernde Aufgabe für eine Sicherheitsanalyse von Android-Apps ergibt sich daraus, dass Android als ein komplexes verteiltes System aufgefasst werden kann, das aus vielen verschiedenen Apps und Komponenten besteht (Multi-Applikationen-System). Die Apps können auf diversen Wegen miteinander kommunizieren oder auf einander zugreifen, insbesondere über Nachrichten (Interprozess-Kommunikation) oder gemeinsam genutzte Speicherbereiche wie z.B. externe Speicherkarten [3]. Hierbei können für den Benutzer unerwünschte Informationsflüsse entstehen, wenn die Apps gewollt oder unabsichtlich fehlerhaft programmiert worden sind. Durch die Notwendigkeit, mehrere Betriebssysteme (z.B. Android, iOS, Windows Phone) zu unterstützen, nimmt die Zahl der Apps, welche plattformunabhängige JavaScript-Anteile mit plattformabhängigen Java (Android) oder Objective-C (iOS) Anteilen kombinieren, zu. Gerade im Zusammenspiel dieser unterschiedlichen Technologien entstehen bisher unbekannte Möglichkeiten für Verletzungen der Datensicherheit. Auch die unterschiedlichen Kommunikationswege eines Smartphones mit der Außenwelt wie z.B. Internet, SMS/MMS, Bluetooth oder Near Field Communication (NFC) machen die Aufgabenstellung komplexer, aber auch interessanter. Da Android eine Multi-Applikationen-Plattform ist, werden Analysen benötigt, die die Kommunikationsstruktur mehrerer Apps analysieren und mit Informationsflussergebnissen für einzelne App-Komponenten kombinieren können, um Lecks privater Daten zuverlässig zu erkennen.

Bei Android-Apps liegt es grundsätzlich im Ermessen des Benutzers, ob eine von der App geforderte Berechtigung angemessen für den Zweck der App ist. Gleiches nehmen wir für die Ergebnisse der innovativen Sicherheitsanalysen der Apps an: Die beste Analyse ist in der Praxis hinfällig, wenn ihre Ergebnisse nicht verstanden werden. Ziel ist hierbei, es auch Sicherheitsadministratoren in Unternehmen oder technisch versierten Benutzern zu ermöglichen, die Analyseergebnisse direkt zu interpretieren. Auf Basis der Analysen können sie dann fundiert die Entscheidung treffen, ob eine App installiert werden darf.

Eine weitere Herausforderung besteht in einem leichtgewichtigen Prozess zur Sicherheitszertifizierung von Software, der der Dynamik der Entwicklung von Android-Apps Rechnung trägt; ein solcher Prozess existiert weltweit noch nicht. Evaluationsschemata wie die Common Criteria sind hierfür zu aufwendig. Perspektivisch bietet sich auch eine Anwendung im Bereich der Zertifizierung von Java-basierter Software von kleinen und mittelständischen Unternehmen (KMU) an, die sich oft eine aufwendige Evaluation wie bei den Common Criteria nicht leisten können.

Der Rest dieses Beitrages ist folgendermaßen gegliedert. Abschnitt 2 diskutiert eine Zertifizierungsplattform für Android-Anwendungen, während die Notwendigkeit von Sicherheitsanalysen als Basis für die Zertifizierungsplattform Gegenstand von Abschnitt

3 ist. In Abschnitt 4 stellen wir unseren Lösungsansatz für einen leichtgewichtigen Zertifizierungsprozess für Apps vor. In Abschnitt 5 geben wir ein Fazit und einen Ausblick.

2 Zertifizierungsplattform für mobile Anwendungen

Abbildung 1 gibt eine Übersicht einer denkbaren Zertifizierungsplattform. Insbesondere wird hier folgendes Szenario unterstützt, das die Lieferkette zur Bereitstellung von Apps abdeckt: Der App-Hersteller lässt sich die Sicherheit seiner App von einer Zertifizierungsstelle auf Grundlage der Prüfung eines Prüflabors bestätigen, das ein entsprechendes Sicherheitswissen über mobile Anwendungen besitzt. Das Prüflabor führt Prüfungen gemäß einem an die App-Dynamik angepassten Zertifizierungsverfahren durch, das sowohl auf statischen als auch dynamischen Analysen beruht. Bei Unbedenklichkeit der App (gemäß vorgegebenen Zertifizierungskriterien/Policy) wird ein Zertifikat vergeben und in die Zertifizierungsplattform eingespielt, das automatisiert ausgewertet werden kann. Marktbetreiber (insbesondere spezieller App-Märkte) wie z.B. die SAP AG erhalten durch die Plattform Zugriff auf das Zertifikat und können automatisiert beim Hochladen einer App in den Market prüfen, ob diese den Sicherheitsrichtlinien des Marktes entspricht und ggf. Nachbesserungen verlangen. Ferner können Werkzeuge von Nutzen sein, die den App-Hersteller aktiv bei der Korrektur gefundener Sicherheitslücken unterstützen.

Im Gegensatz zur beschriebenen automatischen Auswertung der Zertifikate besteht zusätzlich die Möglichkeit, manuelle Überprüfungen vorzunehmen. Dies erfordert eine verständliche Darstellung der Analyseergebnisse. Ziel ist hierbei, auch technisch versierten Benutzern oder Sicherheitsadministratoren in Unternehmen zu ermöglichen, die Analyseergebnisse direkt zu interpretieren. Auf Basis der Analysen können sie dann fundiert die Entscheidung treffen, ob eine App installiert werden darf.

Durch die Anbindung der Zertifizierungsplattform an Device Management-Lösungen werden ferner Großunternehmen unterstützt, die Sicherheitsrichtlinien auf einer Vielzahl von mobilen Endgeräten umsetzen müssen. Basis einer solchen Zertifizierung sind die Sicherheitsanalysen der Apps. Hier bieten sich statische Analysen an, die in den letzten Jahren immer ausgereifter geworden sind [4]. Statische Analysen werden vor der Ausführung eines Programmes wie z.B. einer App durchgeführt und nehmen den Quellcode oder auch den Binärcode als Gegenstand der Untersuchungen. Prinzipiell sind das ähnliche Techniken und Datenstrukturen, wie sie beim Kompilieren und Übersetzen von Programmen eingesetzt werden. Dynamische Analysen werden im Gegensatz dazu bei der Ausführung des Programmes durchgeführt. Nachfolgend geben wir einen Überblick über bereits vorhandene statische und dynamische Sicherheitsanalysen für Android-Apps und leiten den Bedarf für zukünftige Analysen ab.

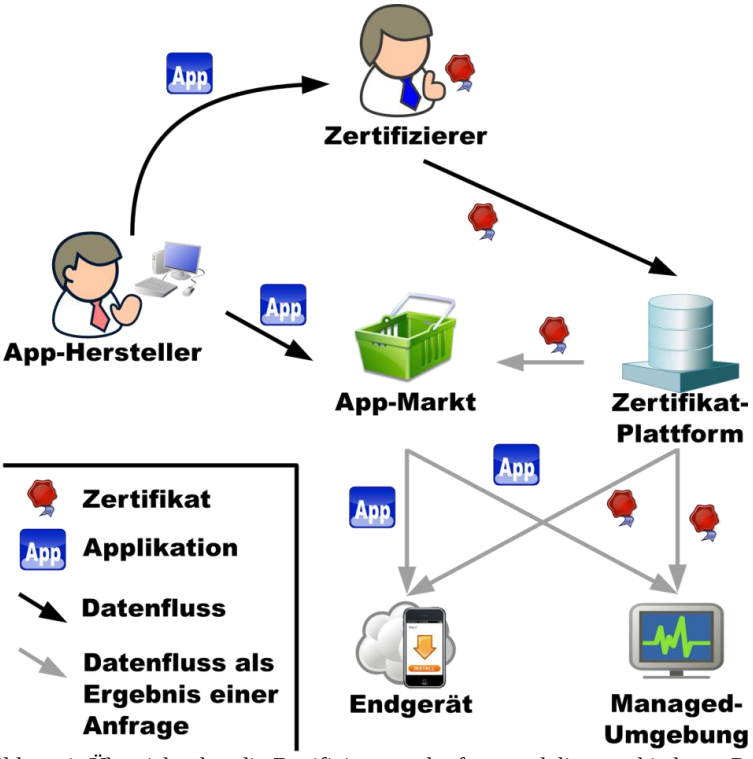

Abbildung 1: Übersicht über die Zertifizierungsplattform und die verschiedenen Rollen im Zertifizierungsprozess.

3 Sicherheitsanalysen von Android-Anwendungen

Google nimmt bereits erste Überprüfungen von Apps in Google Play mittels des Werkzeugs „Bouncer“ vor [5]. Es ist aber nicht transparent, welche Überprüfungen in welcher Tiefe durchgeführt werden. Kürzlich zeigte sich, dass Malware-Apps sich der Erkennung durch den Bouncer entziehen können, indem sie erkennen, wenn der Bouncer sie analysiert und sich zu diesem Zeitpunkt harmlos geben [6] – ein Nachteil rein dynamischer Analysen. In Android 4.2 wurde zudem ein App Verification-Dienst eingeführt. Wie sich jedoch schnell zeigte, ist dieser äußerst unbrauchbar und kann keinerlei Sicherheitsgarantien geben [7]. Der Dienst berechnet lediglich zu einer zu installierenden App einen Hash und gleicht diesen dann mit den Hashes bekannter Malware-Apps ab. Der Hash ist jedoch zu statisch: Selbst leichte (automatisierbare) Änderungen am Code der Malware führen dazu, dass diese nicht mehr erkannt wird.

Sicherheitsanalysen mit der Entwicklung eines machbaren Konzeptes zur Zertifizierung von mobilen Anwendungen existieren bislang nicht am Markt. Vor allem die Interprozess-Kommunikation (IPC), mit der Android-Anwendungen Daten über Prozessgrenzen hinweg austauschen können, lässt sich nicht mittels statischen Analysewerkzeugen, wie derzeit am Markt vorhanden, adressieren. Android-Applikationen setzen sich gemäß Android-Programmiermodell aus verschiedenen Komponenten zusammen. Android-

Komponenten sind u.a. Activities (verantwortlich für die graphische Darstellung und User-Interaktion) und Services (welche die eigentliche Hintergrundarbeit leisten); Broadcast Receiver können Broadcast-Nachrichten abonnieren und darauf reagieren. Diese Komponenten kommunizieren via IPC über sogenannte Intents (spezielle Datenstrukturen zum Nachrichtenaustausch). Eine Analyse muss verstehen, wo ein Intent gesendet wird und welche Callbacks den Intent empfangen. Dies benötigt eine sehr präzise statische Programmanalyse, um komponentenübergreifende Informationsflüsse berechnen zu können. Ferner muss berücksichtigt werden, dass Android eine ganze Reihe solcher IPC-Mechanismen (beispielsweise ein Callback für den Klick eines Buttons) bereitstellt, die allesamt von Analysen unterstützt werden müssen. Mithin sind Standardwerkzeuge wie IBM AppScan [8] und Fortify SCA [9] für solche Analysen nicht geeignet.

Im Forschungsumfeld gibt es Analyse-Werkzeuge, die auf Android zugeschnitten sind. Chin et al. berücksichtigen in ihrem ComDroid-Werkzeug zwar die Interprozess-Kommunikation, führen aber nur einfache statische Analysen direkt auf dem Android-Binär-Code (DEX-Code) aus [10] und können somit keine ausgereiften Zeigeranalysen verwenden, wie sie in Java-/Bytecode-Analyseframeworks wie z.B. Soot [11] implementiert worden sind. Daher ist die Analyse nicht sehr präzise. Grace et al. analysieren System-Apps, die z.B. vom Geräte-Hersteller mitgeliefert werden, mit Hilfe ihres Woodpecker-Werkzeugs auf Sicherheitslücken [12]; da sie nur den Control Flow Graph (CFG) für ihre Analyse verwenden, unterstützen sie keine ausgereiften Datenflussanalysen. Lu et al. haben ein statisches Analyse-Verfahren entwickelt, das Android-Apps auf sog. „Confused Deputy"-Probleme hin untersucht und die Framework-Konzepte von Android berücksichtigt [13]. Confused Deputy-Lücken erlauben es einer bösartigen Anwendung auf eine andere App „aufzuspringen" und kritische Berechtigungen (z.B. Zugriff auf Telefonlisten, Kontaktdaten) auszuführen, obwohl der Angreifer diese Berechtigungen nicht besitzt.

Die eben genannten Werkzeuge sind meist nur als Proof-of-Principles, teilweise durch verschiedene Skripte implementiert, vorhanden. Oft behandeln sie auch nur einen bestimmten Teilaspekt der Android-Sicherheit. Die verschiedenen Werkzeuge untermauern jedoch noch einmal die Relevanz und Aktualität des Themas „Statische Analyse von Android-Apps", auch wenn es sich eher um Ad hoc-Lösungen handelt, die insbesondere nicht den Zertifizierungsaspekt von Apps und die damit verbundenen Prozesse berücksichtigen. Zudem ist keines der bisher verfügbaren Werkzeuge in der Lage, Informationsflüsse zwischen JavaScript- und Java-Komponenten zu analysieren.

4 Lösungsansatz

Die Neuheit unseres Lösungsansatzes besteht in der Analyse der Interprozess-Kommunikation auf Basis von speziellen, auf Android-Apps zugeschnittenen statischen Code-Analysen mit der Entwicklung eines an die App-Dynamik angepassten Zertifizierungskonzeptes. Statische Analyseverfahren für diese Kommunikation sind derzeit zwingend erforderlich, jedoch nicht existent.

Eine weitere wesentliche Neuerung unseres Analyseansatzes besteht darin, für einzelne Android-Anwendungen (und Android-Komponenten) zuerst Analysen separat durchzuführen und die Einzelergebnisse zu „Summaries“ zusammenzufassen. Diese Summaries werden anschließend unter Berücksichtigung der Interprozesskommunikation sowie der Kommunikation zwischen Komponenten unterschiedlicher Technologien (z.B. JavaScript und Java) zu einem Gesamtergebnis zusammengesetzt (Komposition der Einzelergebnisse). Des Weiteren werden die Analysen so gestaltet, dass sie einem Anwender genügend Informationen liefern, um die Analyse-Ergebnisse nachvollziehen und interpretieren zu können (Benutzbarkeitsaspekt). Als Basiswerkzeug für die Analysen kann beispielsweise das bereits oben genannte Soot-Werkzeug verwendet werden. Wir haben bereits einen Prototyp entwickelt, der auf Soot aufsetzt und Analysen unterstützt, die die Android-Framework-Mittel berücksichtigen und auf Basis des Android-Binärcodes (DEX) arbeiten [14]. Auf dieser Basis werden wir weitere Komponenten für eine umfassende App-Analyse entwickeln wie z.B. eine verständliche Darstellung der Analyse-Ergebnisse in Form von architekturellen Beschreibungen.

Im Folgenden beschreiben wir kurz die einzelnen Schritte unseres Analyse-Ansatzes:

1. Wandele den DEX-Code einer oder mehrerer Android-Anwendungen in das Zwischenformat des Soot-Werkzeuges um.
2. Erzeuge mit Hilfe von Soot eine abstrakte Form des Programms, welche die Software-Architektur wie z.B. die einzelnen Android-Komponenten, deren Einstiegs- und Ausstiegspunkte, IPC-Verbindungen sowie Call-Graph-Informationen enthält.
3. Ermittle auf dem Soot-Zwischenformat mit Hilfe von Datenflussanalysen Informationsflüsse zwischen Einstiegs- und Austrittspunkten.
4. Fasse die ermittelten Informationsflüsse pro Komponente zusammen (Summaries).
5. Berechne mit Hilfe dieser Summaries auf der in Schritt 2 erzeugten abstrakten Darstellung alle Informationsflüsse zwischen mehreren Komponenten (bzw. mehreren Anwendungen) und ermittele Informationsflüsse, die vertrauliche Daten nach außen leiten (z.B. Versenden von Kontaktdaten in das Internet oder per MMS).
6. Stelle die Analyseergebnisse in einer für Sicherheitsanalysten verständlichen Form dar und überprüfe, ob die Informationsflüsse durch einzelne Anwendungen bzw. die Kombination von Anwendungen der definierten Policy entsprechen.

Zudem ist es vorgesehen, zusätzliche Analysen für in JavaScript entwickelte Komponenten zu integrieren, so dass in Schritt 5 auch Datenflüsse in hybriden Apps untersucht werden können.

Die Analyseschritte können z.T. getrennt voneinander ausgeführt werden. Dadurch ist es möglich, dass die aufwendigen Schritte (vor allem die Zeigeranalysen in Schritt 3) einmalig durchgeführt werden und ein Nutzer der Analyse nur noch weniger aufwendige Schritte (vor allem 5 und 6) vornehmen muss. Insbesondere werden die Schritte 5 und 6 direkt auf dem Mobiltelefon nutzbar sein. Dies ist wichtig, weil das Schadpotenzial einer App oft vom Vorhandensein anderer Apps oder einer bestimmten Version des Frameworks abhängt. Diese Informationen liegen im App Market nur unzureichend vor. Eine schnelle abschließende Analyse auf dem Smartphone kann hier für Gewissheit sorgen. Die automatische statische Analyse wird somit in Zweifelsfällen durch dynamische Analysen ergänzt, um die Ergebnisse präziser zu machen. Unternehmen können sich

damit ohne eine aufwendige Analyse auf die Angabe der vorhandenen Informationsflüsse verlassen.

Schritt 6 bewirkt eine verständliche Darstellung der durch die statischen und dynamischen Analysen ermittelten Ergebnisse. Eine Möglichkeit besteht darin, diese Ergebnisse in Form von Datenflussdiagrammen (DFDs) darzustellen. DFDs werden von Microsoft beim Threat Modeling, einer Sicherheitsanalyse auf Basis der Software-Architektur, verwendet und sind recht weit verbreitet [15]. Hierdurch können auf Ebene der Software-Architektur kritische Informationsflüsse von Datenquellen (Lokalisierungsdaten, Kontaktlisten, Telefonlisten) zu Datensenken (wie z.B. dem Internet) in übersichtlicher Form dargestellt werden. Ggf. sind die DFDs noch um spezielle Elemente zu erweitern, die Android-spezifische Framework-Mittel berücksichtigen.

4.1 Zertifizierungsprozess für Android-Anwendungen

Die Ergebnisse zu existierenden Informationsflüssen können für eine Zertifizierung verwendet werden, um sich von der Unbedenklichkeit einer Applikation zu vergewissern. Obwohl sich die Evaluierung und Zertifizierung mittels Common Criteria [16] oder gemäß dem Datenschutz-Gütesiegel für eine ganze Reihe von IT-Produkten bewährt hat, ist ein Zertifizierungsprozess für Smartphone-Apps derzeit nicht vorhanden. Zudem existiert kein Zertifizierungskonzept, bei dem Smartphones kontrollieren können, nur zertifizierte Apps zu installieren. Von daher ist dieser Ansatz ein neuartiger und aus Benutzersicht und aus Sicht einer Institution, die Smartphones an ihre Mitarbeiter ausgibt, notwendiger Ansatz. Da der Erfolg des Apps-Konzeptes gerade darin liegt, dass es sich hier oft um Low-Cost-Produkte handelt, ist ein abgestuftes Zertifizierungskonzept wünschenswert, mit einer geringen Prüftiefe (evtl. in Form eines automatischen Dienstes) bis hin zu einer detaillierteren manuellen Analyse für besonders sicherheitskritische und weit verbreitete Apps.

Ein Prüf- und Zertifizierungsschema für mobile Anwendungen auf Basis von statischen und dynamischen Analysen sollte insgesamt folgende Aspekte umfassen:

- den inhaltlichen Umfang der Prüfung auf Basis der Ergebnisse der Projektpartner mit dem zu entwickelnden Analysewerkzeug,
- die formalen Aspekte zur Prüfung und Zertifizierung,
- die Einbeziehung der Methoden internationaler und nationaler Kriterienwerke im Bereich von Datensicherheit,
- ein Validierungssystem zur automatischen Auswertung der Gültigkeit von Zertifikaten,
- den Aufbau von Prüf- und Zertifizierungsstelle mit allen relevanten Prozessen, im Einklang mit anerkannten Standards.

4.2 Verständliche Ergebnisse

Um in der Praxis Anwender tatsächlich zu unterstützen, werden die komplexen Analyseergebnisse und App-Bewertungen verständlich aufbereitet. Sowohl Benutzer ohne oder mit beschränktem technischen Verständnis als auch Sicherheitsexperten sollen die

Ergebnisse einschätzen und basierend darauf fundierte Entscheidungen zu ihrer Sicherheit fällen können. Die Aufbereitung erfolgt in Form von jeweils dem Benutzer angepassten Darstellungen, die das Vorwissen, die mentalen Modelle und die Ziele der Anwender einbeziehen. Es wird unter anderem in Benutzerstudien geprüft, ob die Sicherheit anhand der Darstellung verständlich wird.

5 Fazit

Zusammenfassend betrachtet besteht die Neuheit des vorgeschlagenen Lösungsansatzes darin, ausgereifte Methoden der Programmanalyse im Kontext der Multiapplikationen-Plattform Android anzuwenden. Hierbei wird insbesondere die Semantik des Android-Frameworks mit einbezogen. Einen leichtgewichtigen Zertifizierungsprozess für mobile Anwendungen auf Basis von fundierten Sicherheitsanalysen zu schaffen ist eine herausfordernde, aber dennoch wichtige Aufgabe, um das Vertrauen in IT-Systeme zu stärken. Wir werden diese Aufgabe im Rahmen eines Forschungsprojektes mit einem Konsortium grundlegendend und umfassend bearbeiten, zumal wir die komplette Lieferkette für die Bereitstellung von sicheren und datenschutzgerechten Apps abdecken, von App-Entwicklern, über Prüfstellen bis hin zu Marktplatzbetreibern. Die Forschungspartner werden in diesem Kontext die benötigten Analyseverfahren konzipieren und umsetzen. Wir sind auch optimistisch, dass wir einen grundsätzlichen Beitrag zu einer leichtgewichtigen Zertifizierung von Software liefern können. Gerade kleinere Software-Hersteller können hiervon profitieren.

Danksagung

Diese Arbeit wurde vom Bundesministerium für Bildung und Forschung gefördert (ZertApps-Projekt). Weitere Informationen können Sie unter www.zertapps.de finden.

Darüber hinaus möchten wir uns bei den anonymen Gutachtern bedanken, die wesentlich dazu beigetragen haben, diesen Beitrag zu verbessern.

Literatur

[1] Geek.com: Google Yanks Android Apps From Market Over Malware Concerns. http://www.geek.com/articles/mobile/google-yanks-android-apps-from-market-over-malware-concerns-2011032/

[2] S. Fahl, M. Harbach, T. Muders, M. Smith, L. Baumgärtner, B. Freisleben. Why Eve and Mallory love Android: An analysis of Android SSL (in)security. In: Proceedings of the 2012 ACM conference on Computer and communications security.pp. 50–61. CCS '12, USA , 2012.

[3] W. Enck, D. Octeau, P. McDaniel, S. Chaudhuri. A Study of Android Application Security. In Proceedings of the 14th USENIX Security Symposium, August 2011.

[4] B. Chess., J, West. Secure Programming with Static Analysis. Addison-Wesley, 2007.

[5] Google Inc.: Google Mobile Blog – Android and Security. Februar 2012. Zugreifbar unter: http://googlemobile.blogspot.de/2012/02/android-and-security.html

[6] D. Fisher. Researchers Find Methods for Bypassing Google's Bouncer Android Security. 2012. Zugreifbar unter: https://threatpost.com/researchers-find-methods-bypassing-googles-bouncer-android-security-060412/76643
[7] W. Jiang. An Evaluation of the Application ("App") Verification Service in Android 4.2. North Carolina State University, 2012.
[8] IBM Inc. AppScan-Website. 2013. Zugreifbar unter: http://www01.ibm.com/software/rational/products/appscan/source/.
[9] Fortify Software. Website, 2013. Zugreifbar unter: http://www.fortify.com.
[10] E. Chin, A. Porter Felt, K. Greenwood, D. Wagner. Analyzing Inter-Application Communication in Android. In Proceedings of the 9th International Conference on Mobile Systems, Applica-tions, and Services (MobiSys '11). ACM, USA, 239-252, 2011
[11] R. Vall´ee-Rai, P. Co, E. Gagnon, L. Hendren, P. Lam, V. Sundaresan. Soot - A Java Bytecode Optimization Framework. In CASCON '99: Proceedings of the 1999 Conference of the Centre for Advanced Studies on Collaborative Research. IBM Press, 1999.
[12] M. Grace, Y. Zhou, Z. Wang, X. Jiang. Systematic Detection of Capability Leaks in Stock Android Smartphones. Proceedings of the 19th Network and Distributed System Security Symposium (NDSS 2012). San Diego, CA, February 2012.
[13] L. Lu, Z. Li, Z. Wu, W. Lee, W., G. Jiang. CHEX: statically vetting Android apps for component hijacking vulnerabilities. In: Proc. of the 2012 ACM Conference on Computer and Communications Security. pp. 229–240. CCS '12, 2012.
[14] C. Fritz, S. Arzt, S. Rasthofer, E. Bodden, A. Bartel, J. Klein, Y. le Traon, D. Octeau, P. McDaniel. Highly Precise Taint Analysis for Android Applications, Technical Report TUD-CS-2013-0113, EC SPRIDE, 2013.
[15] S. Hernan, S. Lambert, T. Ostwald, A. Shostack. Uncover security design flaws using the STRIDE approach. MSDN Magazine, Nov 2006. Zugreifbar unter http://msdn.microsoft.com/en-us/magazine/cc163519.aspx
[16] Common Criteria: Common Criteria for Information Technology Security Evaluation—Part 1: Introduction and general model, 2009. Zugreifbar unter: http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R3.pdf