Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 147

Vertrauen Jenseits Datenschutz und Datensicherheit

Mark Aukschlat

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr.-Ing. habil. Bernhard Mitschang
Betreuer/in:	Dipl.-Inf Christoph Stach
Beginn am:	2014-06-19
Beendet am:	2014-12-19
CR-Nummer:	D.4.6, K.6.5

Kurzfassung

Smartphones sind zu einem festen Bestandteil des heutigen Lebens geworden. Besonders auf Grund der breit gefächerten Möglichkeiten durch Software von Drittanbietern, sogenannte Applikationen oder Apps erfreuen sie sich großer Beliebtheit. Doch neben ihrem Nutzen bringen die Apps auch Probleme mit sich, da jeder sie erstellen kann. Manche Apps bieten nur eine mangelhafte Funktionalität, andere richten Schaden an oder stehlen die privaten Daten des Benutzers. Bevor eine App installiert wird, sollte man sich die Frage stellen, ob man dieser App überhaupt vertraut.
Dazu wird in dieser Arbeit ergründet, was es bedeutet einer App zu vertrauen. Es werden bereits bestehende Vertrauenssysteme sowie Metriken betrachtet und mit der *TriMetrik* eine eigene Metrik für das Vertrauen in eine App aufgestellt. Mit *TriTrust* werden zudem Konzept zur Umsetzung dieser Metrik vorgestellt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Ausgangssituation

In der heutigen Zeit sind Smartphones zweifellos ein fester Bestandteil unseres Lebens geworden. Wir tragen diese kleinen Geräte immer bei uns und viele Menschen wirken fast verloren, wenn sie ihr Mobiltelefon nicht bei sich haben. Laut einer eMaketer-Studie[1] wird davon ausgegangen, dass im Jahr 2014 weltweit über 1,75 Milliarden Menschen ein Smartphone besitzen.

Das Leistungsspektrum der Smartphones hat dabei ständig zugenommen und ist inzwischen mit dem von Desktop-PCs zu vergleichen. Ein Smartphone verbindet die Möglichkeiten der Telekommunikation eines herkömmlichen Mobiltelefons mit den Funktionen eines PDA oder Tablett-Computers. Die Erwartungshaltung an ein jedes Smartphone umfasst dabei, dass es auch als mobiles Medienabspielgerät, zur Verwaltung von Terminen und Kontakten, als Navigationsgerät, Spielkonsole, Foto- und Videokamera oder gar mittels NFC zum Bezahlen an der Kasse verwendet werden kann.

Die Vielseitigkeit von Smartphones wird dabei durch ein breites Angebot an Software von Drittanbietern unterstützt. Diese mobilen Applikationen, auch *Apps* genannt, können dabei nicht nur von Firmen stammen, sondern auch von privaten Entwicklern. Alleine der *Google Play Store*[2], der offizielle App-Marktplatz für Smartphones mit dem Android-OS, beinhaltet über 1,3 Millionen Apps. Insgesamt gibt es für Geräte mit dem Android-OS über 30 App-Marktplätze, wobei der *Amazon Appstore*[3] oder der von Samsung speziell für seine eigenen Smartphones gedachte Store *Samsung Apps*[4] zu den bekannteren zählen.

Geräte mit dem Android-OS dominieren dabei den Markt. Laut einer IDC-Studie[5] ist auf 84,4% der im dritten Quartal 2014 verkauften Geräte Android installiert. Das hohe Angebot der Apps wird dabei durch die Vielzahl an Märkten unterstützt, wobei hohe Sicherheitsmaßnahmen bei diesen Märkte nicht garantiert sind. Auch wenn der *Play Store* die Kontrolle von eingestellten Apps verschärft, fallen sie doch im Vergleichsweise zum *App-Store* von Apple noch gering aus. Auf Grund des hohen Marktanteils von Android, der Vielzahl an Apps und des offenen Sicherheitskonzept von Android wird sich diese Arbeit vorwiegend mit Android Apps beschäftigen.

Die Vielfalt an Apps bringt nämlich auch ihre Probleme mit sich. Viele Apps sind niedriger Qualität und bringen nicht die erwarteten Funktionalitäten mit sich. Manche Apps sind sogar schadhafter

[1] http://goo.gl/wpACCI
[2] https://play.google.com/store
[3] http://www.amazon.com/mobile-apps/b?node=2350149011
[4] http://apps.samsung.com
[5] http://www.idc.com/prodserv/smartphone-os-market-share.jsp

Natur. Nicht immer sind sich die Benutzer im Klaren, welche Rechte eine App, die sie auf ihrem Gerät installieren, besitzt und was die App mit diesen Berechtigungen anstellen kann. Dabei kommt es oft zur unerwünschten Verwendung privater Daten wie etwa Fotos, Kontaktdaten oder Nachrichten durch die App. Diesem Missbrauch ist der Benutzer sich meist nicht bewusst oder kann ihn nicht verhindern, solange er die App weiter verwenden will [Bac12].

Die Bedrohungen durch Apps für den Benutzer lassen sich dabei in zwei grundsätzlich unterschiedliche Gefahrenquellen zusammenfassen. Zum einen müssen das System und seine Daten vor Manipulation oder ungewollten Zugriffen geschützt werden. Sicherheitslücken in Apps oder dem mobilen Betriebssystem werden von Angreifern gezielt ausgenutzt, um vom Nutzer unbemerkt Aktionen durchzuführen, die eigentlich nicht legitimiert sind und potenziell Schaden anrichten können. Hierbei spricht man von *Information Security*.

Uscilowski [Usc13] stellte dabei fest, dass das Wachstum vom Schadsoftware für mobile Geräte überproportional stärker ist als für Desktop-PCs. Erst allmählich reagieren die Hersteller von Sicherheitssoftware auf diese neuartigen Gefährdungen. Zunehmend entwickelt sich ein Markt für mobile Sicherheitssoftware. Einige der größten Anbieter für Computer-Security wie etwa Kaspersky[6], Avira[7] und McAfee[8] brachten Programme für mobile Plattformen heraus.

Das andere Problem ist, dass dem Nutzer die Möglichkeit geboten werden muss reglementieren zu können, in welchem Umfang und auf welche Art der Daten eine App legal zugreifen darf. Funktionen dieser Art werden als *Data Privacy* zusammengefasst.

Mit *AppGuard* [BGH+13] oder *Dr. Android and Mister Hide* [JMV+12] gibt es bereits Ansätze dazu, die dem Benutzer erlauben einer App den Zugriff auf Daten zu verweigern und wenn nötig zufällig oder bewusst falsch generierte Daten an die App zu liefern, wodurch keine privaten Informationen verraten werden. Jedes System setzt dabei auf andere Ansätze und Strategien [SM13].

Die bisherigen Lösungsansätze konzentrieren sich meistens jedoch nur auf eine der Gefahrenquellen, was aus Benutzersicht nicht zufriedenstellend ist. Aus diesem Grund führten Stach und Mitschang [Sta13a] [Sta13b] [SM+14] die **P**rivacy **M**anagment **P**latform (PMP), ein kontextsesitives und absturzsicheres Berechtigungssystem für Android ein. Mit PMP lassen sich Berechtigungsanpassungen zur Laufzeit bewerkstelligen, private Daten auf Wunsch beliebig randomisieren und der Nutzer wird jederzeit über die Konsequenzen seiner Einstellungen auf den Leistungsumfang einer App informiert. Mittels eines sicheren Datencontainers (SDC) lässt sich die PMP leicht um wichtige, die Information Security betreffende, Funktionen erweitern.

1.2. Problemstellung

Security- und Privacy-Systeme bieten dem Benutzer also Möglichkeiten, die Berechtigungen einer App einzuschränken. Wenn eine App keine Berechtigungen besitzt, kann sie auch keinen Schaden anrichten. Schränkt man die Berechtigungen einer App – in Android *Permissions* genannt – ein,

[6] http://www.kaspersky.com/de/android-security
[7] http://www.avira.com/android-phone/download-at
[8] http://home.mcafee.com/store/mobile-security

leidet darunter eigentlich immer auch die Funktionalität der App. Wer nicht Gefahr laufen will, dass eine App den eigenen Standort an Dritte sendet, verbietet ihr die Verwendung von Permissions zur Standortbestimmung. Dadurch ist die *Privacy* gewährt, aber etwa eine Navigations-App wird nutzlos, da sie den Benutzer nicht zum Ziel lotsen kann ohne seine Position zu kennen.

Um dieser Problematik entgegen zu wirken, muss man sich die Frage stellen, ob man einer App (und ihrem Entwickler) vertrauen kann. Das Vertrauen spielt in der Informatik bereits in vielen Bereichen wie dem Onlinehandel oder in Netzwerken eine große Rolle [AG07].

Es gilt nun auch eine Lösung zu finden, um das Vertrauen in eine App zu berechnen. Dabei sollte allen Bereiche Beachtung geschenkt werden, die dafür eine Rolle spielen könnten. Neben der Frage nach der Funktionalität der App sollten ihre Bedrohungen für das System und die Daten auf dem Smartphone analysiert werden. Es sollten einerseits Empfehlungen abgegeben werden, ob eine App auf Grund des für sie berechneten Vertrauens installiert werden kann oder ob man es besser lassen sollte. Andererseits könnten auch Überlegungen angestellt werden, sodass begründete *Security*- und *Privacy*-Maßnahmen unternommen werden können, will man die App trotz des niedrigem Vertrauen nutzen.

Yan et al. untersuchten die Auswirkung eines angezeigten Vertrauens-Wertes auf die Bereitschaft von Benutzern eine App weiter zu verwenden. Dabei wurde festgestellt, dass Benutzer bei einem niedrigen Vertrauens-Wert tatsächlich bereit waren die Benutzung einer App nicht fortzusetzen. Außerdem bestand ein nicht zu vernachlässigendes Interesse über die Berechnung des Vertrauens-Wertes. Die Einführung eines solchen spielt auch aus Sicht des gewöhnlichen Smartphone-Nutzers eine Rolle [YLNY13].

In dieser Arbeit wird der aktuelle Stand der Forschung bezüglich Technik zum Thema Vertrauen analysiert und kritisch bewertet. Dabei wird ein eigener Best-of-Breed Ansatz, der allen Anforderungen für die nachvollziehbare Berechnung eines Vertrauens-Wertes entspricht, entwickelt. Außerdem werden eine Metrik, Konzeptvorschläge zur Umsetzung der Metrik und ein Prototyp für eines der Konzept vorgestellt.

Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Definition von Trust: Zunächst wird der Begriff Vertrauen definiert, wie er im gesellschaftlichen Sinne, im weiten Gebiet der Informatik und auf Android App bezogen verstanden wird. Dabei wird abschließend für das Vertrauen eine eigene Definition gegeben.

Kapitel 3 – Related Work: In Kapitel 3 werden bereits bestehende Ansätze und Techniken vorgestellt. Neben Vertrauens- werden auch Security- und Privacy-Systeme aufgezählt, da später eine Kombination dieser mit dem entwickelten Vertrauens-System diskutiert wird.

Kapitel 4 – Anforderungen und Umsetzungen: In Kapitel 4 werden die Anforderungen an eine Vertrauens-Metrik für Android Apps aufgezählt. Dabei wird betrachtet, wie die in Kapitel 3 vorgestellten Systeme das Vertrauen berechnen. Aus diesen Erkenntnissen werden Überlegungen aufgestellt, wie eine eigene Metrik aussehen müsste.

Kapitel 5 – TriMetrik: Kapitel 5 setzt die Überlegungen aus Kapitel 4 in eine eigene Metrik, die *TriMetrik* um.

Kapitel 6 – TriTrust: Es werden in Kapitel 6 Konzepte präsentiert, wie die in Kapitel 5 aufgestellte Metrik mit einem Vertrauens-System (*TriTrust*) realisiert werden kann.

Kapitel 7 – Implementierung: Kapitel 7 beschreibt die Implementierung eines Prototypen für eines der Konzepte aus Kapitel 6.

Kapitel 8 – Bewertung: Abschließend wird eine Bewertung der Metrik durchgeführt, ob sie allen Anforderungen entspricht.

Kapitel 9 – Zusammenfassung und Ausblick: Fasst die Ergebnisse der Arbeit zusammen, vergleicht das Erreichte mit den Zielen und stellt Anknüpfungspunkte für zukünftige Weiterentwicklung von Metrik und Konzept vor.

2. Definition von Trust

Ziel dieser Arbeit ist es, eine Metrik aufzustellen, um das Vertrauen in eine App zu berechnen. Auch wenn primär Android Apps in dieser Arbeit betrachtet werden, ist diese Definition für das Vertrauen in eine App betriebssystemunabhängig.

Bevor jedoch Methoden und Konzepte zur Berechnung und Modellierung von Vertrauen betrachtet werden, muss definiert werden, was überhaupt unter Vertrauen verstanden wird und wie dieses Phänomen auf eine App angewandt werden kann.

Dazu wird in Unterkapitel 2.1 zunächst ergründet, auf welche Arten Vertrauen grundlegend verstanden wird. Außerdem wird betrachtet, welche Eigenschaften Vertrauen besitzt und wie diese verwendet werden können, um eine Definition und später auch die Metrik aufzustellen. Im Anschluss werden in Unterkapitel 2.2 einige Definition betrachten, die bereits für Vertrauen in der Informatik gemacht wurden.

In Unterkapitel 2.3 wird betrachtet, wie in anderen Arbeiten Vertrauen in Appen definiert wird. Mit Hilfe dieser Definitionen, aber auch den in den vorherigen Unterkapiteln gewonnenen Erkenntnissen wird eine eigene Definition für das Vertrauen in eine App gegeben.

Abschließend wird in Unterkapitel 2.4 der Begriff der Reputation definiert, da er in den folgenden Kapiteln eine Rolle spielen wird und oft falsch verwendet oder verstanden wird.

2.1. Grundlegende Definition von Vertrauen

Zahlreiche Definitionen beschäftigt sich mit der Thematik des Vertrauens. Sie alle aufzuzählen würde den Rahmen dieser Arbeit sprengen und viel Redundanz schaffen. Im Folgenden beschränkt sich dieses Arbeit auf drei Definitionen nach Gambetta, Josang sowie von Grandison und Sloman. Diese Definitionen zählen zu den am meisten zitierten und geben einen guten Überblick über das Verständnis von Vertrauen. Aus ihnen werden grundlegende Erkenntnisse über das Vertrauen gewonnen, die später für unsere Problemstellung erweitert werden.

Zunächst betrachten wir die Definition von Gambetta et al. [Gam88]:

Definition 2.1.1 (Vertrauen nach Gambetta et al. [Gam88])
Eine besondere Ebene der subjektiven Wahrscheinlichkeit mit welcher ein Agent annimmt, dass ein anderer Agent oder eine Gruppe von Agenten eine bestimmte Aktion ausführt, in zweierlei Hinsicht, bevor er solche Aktionen beobachten kann (oder unabhängig von seiner Möglichkeit jemals eine solche Aktion beobachten oder herbeiführen zu können) und in einem Kontext in welchem dies seine eigenen Aktionen beeinflusst.

Vertrauen ist nach Gambetta also eine subjektive Annahme, die ein Agent von einem anderen Agenten oder einer Gruppe von Agenten hat. Gambetta definiert diese Annahme direkt in einer „Wahrscheinlichkeit“, also in welchem Maße wir annehmen, dass der Agent so handelt wie angenommen. Diese Annahme wird um die Bedingungen erweitert, dass der Agent der Vertrauen hat, noch nie den anderen Agenten eine solche Aktion hat ausführen sehen und die Aktion des anderen Agenten die Aktionen des vertrauenden Agenten beeinflusst.

Wir haben also als Eigenschaften für das Vertrauen erhalten, dass dieses eine *subjektive Annahme mit Wahrscheinlichkeit* ist, welche die Aktionen anderer Betrifft. Dabei spielt die *Unwissenheit* (der Agent hat nie eine solche Aktion gesehen und könnte sie eventuell auch niemals sehen) genauso wie die Tatsache, dass der Vertrauende selber durch diese Aktion betroffen ist.

Weitere Eigenschaften von Vertrauen lassen sich aus der Definition von Jøsang [Jøs96] ziehen. Dabei sei angemerkt, dass in dieser Definition und in weiteren von Entitäten die Sprache ist, welche sich jedoch nur in der Bezeichnung, nicht aber in der Bedeutung vom Agenten unterscheiden. In diesem Kapitel wird um Kontinuität zu bewahren der Begriff des Agenten weiterhin verwendet werden – die Verwendung des Begriffes Entität wird in den zitierten Definitionen jedoch belassen.

Definition 2.1.2 (Vertrauen nach Jøsang [Jøs96])
Der Glaube dass sie [die Entität in die man Vertrauen hat] in ihrem Verhalten keine bösartige Absichten hat.

In dieser Definition bedeutet Vertrauen die Annahme, dass ein Agent in seinem Verhalten keine bösartigen Absichten hat. Weniger als eine Eigenschaft für Vertrauen direkt ändert diese Definition unseren Kontext. Nach Gambette konnte ein Agent noch eine Handlung durchführen oder eben nicht und über die Konsequenzen ist nichts weiter bekannt. Nun kann klar definiert durch die Aktionen eines Agenten Schaden entstehen. Vertrauen bedeutet also auch zu glauben, dass ein Agent keinen Schaden anrichtet.

Aus der Definition von Grandison und Sloman [GS00] lassen sich noch weitere Erkenntnisse über das Vertrauen gewinnen:

Definition 2.1.3 (Vertrauen nach Grandison und Sloman [GS00])
Der feste Glaube in die Kompetenz einer Entität verlässlich, sicher und zuverlässig in einem spezifizierten Kontext zu agieren.

Diese Definition verfeinert noch die Bedingungen für das Verhalten eines Agenten, aber auch die Umgebung. Es gibt einen klar spezifizierten Kontext, in dem die Fähigkeit des Agenten bewertet wird, verlässlich, sicher und zuverlässig zu agieren. Vertrauen ist in dieser Definition also der Glaube daran, ob und wie der Agent dazu fähig ist. Ob er eine Aktion durchführt und ob diese schädlich sein könnte, wird nun also um die Qualität erweitert, mit welcher der Agent die Aktion durchführt.

Neben diesen Erkenntnissen zum Vertrauen lassen sich noch ein paar grundlegende Eigenschaften für Vertrauen festlegen. Diese entspringen den Definitionen und mögen offensichtlich erscheinen, doch in der späteren Diskussion für eine eigenen Definition in Unterkapitel 2.3 und bei der Aufstellung der Metrik in Kapitel 5 wird auf diese zurückgegriffen und so ist eine formale Festlegung hilfreich.

2.1.1. Eigenschaften von Vertrauen

Gutscher et al. [GHS08] sowie Yan und Holtmanns [YH08] versuchten die Eigenschaften von Vertrauen zu formalisieren. Hier ist eine Auflistung dieser Eigenschaften gegeben, welche sich aus beiden Versuchen zusammensetzt[1].

(1) **Vertrauen ist gerichtet**: Vertrauen ist eine gerichtete Beziehung zwischen einem Vertrauendem (trustor) und einem Vertrautem (trustee).

(2) **Vertrauen ist nicht symmetrisch**: Wenn A Vertrauen in B hat, impliziert das nicht notwendigerweise, dass B Vertrauen in A hat.

(3) **Vertrauen kann transitiv sein**: Aus „A vertraut B“ und „B vertraut C“ kann „A vertraut C“ folgen, muss aber nicht. Dies darf nicht als Fakt genommen werden, sondern muss von Fall zu Fall neu entschieden werden.

(4) **Vertrauen ist subjektiv**: Vertrauen ist eine persönliche Meinung, ein subjektives Phänomen, das von verschiedenen Faktoren und Bedingungen abhängt.

(5) **Vertrauen ist kontextabhängig**: Vertrauen ist eine subjektive Meinung von einer Entität in einem bestimmten Kontext.

(6) **Vertrauen ist messbar**: Vertrauenswerte können verwendet werden um die verschiedenen Vertrauensgrade zu beschreiben, die eine Entität von einer anderen hat.

(7) **Vertrauen ist abhängig von der Vergangenheit**: Vergangene Erfahrungen können Einfluss auf den aktuellen Vertrauenswert haben.

(8) **Vertrauen ist dynamisch**: Vertrauen ändert sich über die Zeit hinweg durch Erfahrungen oder Informationen, die ich mit der anderen Entität gemacht oder über sie erhalten werden.

(9) **Vertrauen kann eine zusammengesetzte Eigenschaft sein**: Vertrauen ist eine Zusammensetzung vieler Attribute: Verlässlichkeit, Zuverlässigkeit, Ehrlichkeit, Aufrichtigkeit, Sicherheit, Kompetenz und Pünktlichkeit, welche abhängig von der Umgebung in welcher Vertrauen spezifiziert wird, beachtet werden müssen.

Aus der Definition 2.1.1 von Gambetta lässt sich dabei folgende Ergänzung für diese Liste machen, auf die später zugreifen werden wird:

(10) **Unwissen verlangt Vertrauen**: Wenn alle Fakten bekannt sind, wird Vertrauen nicht benötigt oder es ist von geringerer Bedeutung. Je weniger von einem Agenten bekannt ist, desto wichtiger ist, welches Vertrauen man in ihn hat.

[1] Eigenschaften aus Gutscher: (2); (3); (8); Eigenschaften aus Yan: (1); (4); (5); (6); (7); (8); (9)

Eine wichtige Eigenschaft in dieser Liste ist hierbei Eigenschaft (6). Dass Vertrauen messbar ist, ist Voraussetzung dafür, dass man Vertrauen modellieren und berechnen kann. Ohne diese Eigenschaft wäre die Aufstellung einer Metrik eine fruchtlose Arbeit. Alle Arbeiten - wie auch diese hier - die sich mit der Berechenbarkeit und Metriken für Vertrauen beschäftigen, begründen sich auf dieser Eigenschaft.

In den folgenden Unterkapiteln wird auf diese Liste zurückgegriffen, um Definitionen zu be- und ergründen.

Nachdem grundlegende Definitionen von Vertrauen analysiert wurden und die benötigten Eigenschaften von Vertrauen formal aufgestellt wurden, werden im folgenden Unterkapitel 2.2 Definitionen für Vertrauen im Feld der Informatik betrachtet und erörtert welche Informationen daraus für eine eigene Definition gewonnen werden können.

2.2. Definitionen von Vertrauen in der Informatik

Ein Ansatz für Vertrauen Multi-Agent Systeme stammt von Mui et al. [MMH02a]. Sie definieren Vertrauen dabei wie folgend:

Definition 2.2.1 (Vertrauen in Multi-Agent Systemen nach Mui et al. [MMH02a])
Die subjektive Wahrnehmung eines einzelnen Agenten von einem anderen Agenten, die durch die bisherigen Begegnungen der beiden miteinander geprägt wurde.

Die Subjektivität des Vertrauens in dieser Definition ist nichts Neues. Interessanter ist in diesem Fall, dass zur Bildung von Vertrauen bereits Interaktion zwischen den beiden Akteuren geherrscht haben muss. Dies ist problematisch, wenn man es auf die Problemstellung dieser Arbeit beziehen will, da man den Vertrauenswert für eine App auch zu Rate ziehen will, um zu entscheiden, ob man diese überhaupt installieren will.

Hat man also keine eigenen Erfahrungen mit dem anderen Agenten gemacht, muss man sich auf die Erfahrungen anderer verlassen. Hierbei wird Eigenschaft (3) von Vertrauen verwendet. Wenn andere Personen der App vertrauen und diesen Personen vertraut wird, folgt, dass man der App vertrauen kann. Betrachtet man nicht die Meinung einer einzelnen Person, sondern die Meinung der breiten Masse, wird dieses Meinungsbild auch als Reputation aufgegriffen. Eine ausführliche Erläuterung zur Reputation und welche Rolle sie für das Vertrauen spielt, wird in Unterkapitel 2.4 erläutert.

Eine weitere Definition für Vertrauen stammt von Corritore et al. [CKW03] und bezieht sich auf On-line Systeme:

Definition 2.2.2 (Vertrauen in On-line Systemen nach Corritore et al. [CKW03])
Die zuversichtliche Erwartung, dass in einer risikobehafteten Situation online nicht jemandes Schwachstellen ausgenutzt werden.

In dieser Definition findet man wieder die Möglichkeit Schaden zuzufügen und ob dies geschehen wird, als zentralen Punkt des Vertrauens. Während nach Eigenschaft (8) Vertrauen eine zusammengesetzte

Eigenschaft ist, beziehen sich diese Definitionen nur auf einen einzelnen Punkt. Dies kann jedoch daher rühren, dass der Kontext keine anderen Bedingungen verlangt oder manche Bedingungen als selbstverständlich angesehen werden. In der eigenen Definition für Vertrauen in Appen wird später darauf geachtet, eine möglichst genaue Definition aufzustellen, um keinen Spielraum für Mutmaßungen zu zulassen.

Die Rolle von Schadpotenzial und Sicherheit spielt in der Definition für Vertrauen eine große Rolle. Für Yan und Holtmanns [YH08] „geht Vertrauen über die Security hinaus. Vertrauen ist eine erweiterte Lösung für Security." Vertrauen ist also eine Lösung jenseits von unbegründeten Security- und (um der Problemstellung aus Kapitel 1 zu entsprechen) Privacy-Maßnahmen.

Hat man Vertrauen in einen Agenten, dass er einem keinen Schaden zufügen kann, sind die Security- und Privacy-Maßnahmen weniger bis gar nicht von Nöten. Im Umkehrschluss spielt Vertrauen eine geringere Rolle (zumindest unter diesen Bedingungen), wenn man erhöhte Security- und Privacy-Maßnahmen anwenden. Es besteht also ein Zusammenhang zwischen Security, Privacy und Vertrauen. Dies entspricht der Kontext-Eigenschaft (5) von Vertrauen. Natürlich ist es Ziel dieser Arbeit die Richtung zu Betrachten, in der das Vertrauen die wichtigere Rolle spielt, trotzdem werden in Kapitel 5 und 6 Überlegungen angestellt, wie das Vertrauen eine geringere Rolle spielen kann, wenn Securtiy- und Privacy-Maßnahmen erhöht werden oder wie es genutzt werden kann, um diese gezielter einzusetzen.

Nachdem in Unterkapitel 2.1 der Begriff des Vertrauen in seiner grundlegenden Bedeutung ergründet und die Eigenschaften von Vertrauen betrachtet wurden, sowie in diesem Unterkapitel Definitionen für Vertrauen in der Informatik analysiert wurde, wird nun in Unterkapitel 2.3 eine Definition für das Vertrauen in eine App gegeben werden.

2.3. Definition von Vertrauen in eine App

Definitionen für das Vertrauen in App sind noch nicht sehr zahlreich. Für unsere Metrik ist es unerlässlich eine eigene Definition aufzustellen, die als Orientierung für die Faktoren eine Rolle spielt, die in die Metrik einfließen. Um diese eigene Definition aufstellen zu können, wird die Definitionen von Yan et al. [YZD12] betrachtet und dann um die Erkenntnisse erweitert, die in diesem Kapitel gewonnen wurden. Yan et al. definieren das Vertrauen in eine App dabei wie folgt:

Definition 2.3.1 (Vertrauen in eine App nach Yan et al. [YZD12])
Der Glaube des Benutzers an eine App eine Aufgabe wie erwartet zu erfüllen.

In dieser Definition steht ausschließlich die Funktionalität der App im Vordergrund. Doch wenn es um das Vertrauen in eine App geht, spielen andere Faktoren eine Rolle.

Die Definition 2.1.2 von Jøsang für Vertrauen und die Definition 2.2.2 von Corritore et al. für Vertrauen in On-line Systemen beziehen sich wie oben bereits erörtert auf die Möglichkeit und die Bereitschaft des Trustee dem Trustor Schaden zuzufügen. Auch wenn diese es nicht Ziel dieser Arbeit ist eine Security-Lösung zu finden, sollte dieser Aspekt nicht vollständig ignoriert werden. Der Glaube an eine App, kein Schadpotenzial zu besitzen, sollte auf jeden Fall Teil des Vertrauens in eine App sein.

Die Bedenken bezüglich der Sicherheit einer App lassen sich auch auf deren Umgang mit privaten Daten erweitern. In einer Umfrage des *PewResearchCenter* gaben 57% der Teilnehmer an, eine App nicht installiert zu haben, da sie der App zu viel Zugriff auf persönliche Daten gewähren müssten oder eine App wieder deinstalliert zu haben, weil sie herausfanden, wie viele Daten von der App gegen den Willen der Benutzer weitergeleitet wurden, [pew12].

Der Umgang einer App mit privaten Daten spielt also bei vielen Benutzern eine Rolle, wenn es um das Vertrauen in eine App geht. Manche Anwendungen brauchen jedoch den Zugriff auf private Daten, um ihre Funktionalität erfüllen zu können. Da viele kostenlose Appen erst durch Werbung kostenlos bleiben, sind viele Benutzer auch bereit ihre privaten Daten gegen die kostenlose App quasi einzutauschen. Folglich muss der Umgang einer App mit privaten Daten nicht unbeding für das Vertrauen in die App eine Rolle spielen. Es greift dabei Eigenschaft (4) von Vertrauen und zwar, dass dieses subjektiv ist und von den Ansichten des Einzelnen abhängt.

Definition 2.3.2 (Vertrauen in eine App)
Einer App wird vertraut, wenn sie die versprochenen Funktionalitäten bietet, private Daten nicht gegen den Wunsch des Benutzers weitergibt und möglichst geringes Potenzial besitzt, um Schaden anzurichten.

Die *TriMetrik*, welche in Kapitel 5 aufgestellt wird, orientiert sich an dieser Definition 2.3.2. Dabei entspricht die Definition den Eigenschaften des Vertrauen und den Ansprüchen und Vorstellungen, die heutzutage in Bezug auf App herrschen.

Zum Abschluss des Kapitels wird in Unterkapitel 2.4 noch der Begriff der Reputation definiert.

2.4. Definition Reputation

Die Reputation eines Agenten wird gelegentlich mit dem Vertrauen in einen Agenten gleichgesetzt. Dies ist jedoch eine falsche Verwendung des Begriffs und es soll im Folgenden eine klare Definition für die Reputation einer App geliefert werden, um später Verwirrung zu vermeiden.

Dazu wird zunächst ein Vergleich zwischen den Definitionen von Mui et al. [MMH02a] für Vertrauen in einem Multi-Agent System und die Reputation eines Agenten in einem solchen System gezogen. Die Definition für Vertrauen wurde bereits als Definition 2.2.1 gegeben und analysiert. Zur Erinnerung sei sie noch einmal gegeben: „*Die subjektive Wahrnehmung eines einzelnen Agenten von einem anderen Agenten, die durch die bisherigen Begegnungen der beiden miteinander geprägt wurde.*“ Vergleichen wir dies nun mit der Definition für die Reputation:

Definition 2.4.1 (Reputation eines Agenten in einem Mulit-Agent System nach Mui [MMH02a])
Die Wahrnehmung, die von einem anderem Agenten durch dessen vergangene Aktionen über seine Intentionen und Normen gewonnen wurde.

Auffällig bei dieser Definition ist es, dass es bei der Reputation nicht mehr um die Wahrnehmung des einzelnen geht, sondern von allen Agenten, die mit dem Agenten interagiert haben. Während das Vertrauen also die subjektive Wahrnehmung eines Einzelnen ist, ist die Reputation der Eindruck der breiten Masse. Die Reputation ist also der Ruf, den ein Agent bei der Allgemeinheit besitzt, den

er durch seine Taten gewonnen hat. Dies unterscheidet sich klar vom Vertrauen. Ist sein Verhalten einem anderen Agenten gegenüber immer gutartig, hat dieser ein großes Vertrauen in ihm, ist sein Verhalten gegenüber einem weiteren Agenten immer bösartig, hat dieser ein geringes Vertrauen in ihm. Die Reputation des Agenten würde in diesem Fall mittelmäßig sein.

Einen ähnlichen Ansatz lässt sich auch bei Yan et al. [YZD12] finden, wenn man seine Definition für das Vertrauen in eine App mit der Reputation mit der Definition für die Reputation einer App vergleicht. Die Definition für Vertrauen wurde ebenfalls in Definition 2.3.1 gegeben, sie sei aber auch zur Erinnerung wieder gegeben: „Das Glauben des Benutzers an eine App eine Aufgabe wie erwartet zu erfüllen." Unter der Reputation verstehen sie:

Definition 2.4.2 (Reputation einer App nach Yan)
Das öffentliche Vertrauen, berechnet aus direktem oder indirektem Wissen oder Erfahrungen.

Nach dieser Definition wird Reputation mit dem öffentlichen Vertrauen gleichgesetzt, was in diesem Fall dem Glauben, berechnet durch gesammelte Informationen ist, ob die App ihre Funktionalität ist, wenn man beide Definitionen miteinander verknüpft.

Reputation kann also mehr als Vertrauen sein und weniger. Für die Metrik wird die Reputation als Teil des persönlichen Vertrauens betrachtet. Durch Bewertungen und Berichte anderer wie etwa ein Rating in einem App-Marktplatz bildet sich die öffentliche Meinung einer App. Dieser so entstandene Wert wird in diesem Dokument als Reputation einer App verwendet. Dazu sei Definition 2.4.3 gegeben:

Definition 2.4.3 (Reputation einer App)
Die öffentliche Meinung von einer App, entstanden durch Bewertungen oder Berichte über die App.

Für das persönliche Vertrauen spielen jedoch andere Faktoren noch eine Rolle. Reputation beeinflusst das Vertrauen in eine App, jedoch können eigene Erfahrungen, das persönliche Vertrauen in den Entwickler und andere Faktoren eine Rolle spielen. Welche Faktoren dies im Detail sind, darauf wird in Kapitel 5 genauer eingegangen.

In Kapitel 3 werden nun Ansätze und Methoden zum Thema Reputation und Vertrauen, sowie auch inhaltlich verwandte System, die später noch eine Rolle spielen, betrachtet.

3. Related Work

Vertrauen spielt in der Informatik schon lange eine große Rolle. Marsh stellte 1994 eines der ersten Konzepte eines berechenbaren Vertrauens-Wertes vor und seitdem wurden viele weitere Ansätze in diesem Gebiet geliefert [Mar94].

In diesem Kapitel werden einige der bisherige Ansätze zu Vertrauen und verwandten Themen betrachtet. Zunächst werden in Unterkapitel 3.1 Reputationssysteme betrachtet, welche Stärken und Schwächen diese besitzen und wo sie bereits im Einsatz sind. Anschließend werden Frameworks und andere Ansätze für Vertrauens-Systeme in Unterkapitel 3.2 vorgestellt.

Die darauf folgenden Unterkapitel beschäftigen sich mit Ansätzen für das Android System und dafür geschriebene Apps. Dabei wird zunächst in Unterkapitel 3.3 die Sicherheit des Systems und der Daten vor Zugriffen durch Apps durch Security- und Privacy-Systeme analysiert. In Unterkapitel 3.4 werden Systeme behandelt, welche sich mit dem Vertrauen in eine App direkt beschäftigen. Abschließend wird der Aspekt des Vertrauens in den Entwickler einer App und welche Aufgaben es dort zu bewältigen gibt in Unterkapitel 3.5 behandelt.

3.1. Reputationssysteme

Geht es um die Berechnung von Vertrauenswerten, spielen Reputationssysteme oft eine Rolle. In Definition 2.4.2 auf Seite 19 wird die Reputation einer App als ihr öffentliches Vertrauen bezeichnet. Fallen andere Einflussfaktoren, wie sie in Unterkapitel 2.4 etwa für eine App erläutert wurden, weg, kann es sogar sein, dass Vertrauen und Reputation äquivalent sind.

Mui et al. geben in [MMH02b] eine Übersicht über die verschiedenen Typologien von Reputation. Außerdem präsentieren sie ein Framework für den Test der verschiedenen Typologien in Hinblick auf die Überlebensfähigkeit eines Agenten in einer evolutionären Version des unvollständige-Information-Spiels.

Die selben Autoren beschäftigen sich mit bestehenden Vertrauens- und Reputationsmodelle. Dabei stellen sie fest, dass keine Unterscheidung zwischen dem Vertrauens- und dem Reputations-Wert gemacht werden und dass die soziologischen Eigenschaften von Vertrauen nicht beachtet werden. Ergänzend präsentieren sie ein eigenes Modell, welches zwischen Reputation und Vertrauen differenziert und somit dieser Schwäche beikommt [MMH02a].

Hoffman et al. [HZNR09] und Fraga et al. [FBM12] geben eine Taxonomie über die Angriffsmethoden und Ziele auf Reputations- und Vertrauenssysteme. Hoffman et al. überprüfen zusätzlich existierende Systeme auf ihre Anfälligkeit für die vorgestellten Angriffe.

Malik und Bouguettaya entwarfen mit RATEWeb ein Reputationssystem für Web-Services in Hinsicht darauf, wie hoch die Qualität der Services ist, die diese anbieten. Dabei wird für Nutzer, welche Bewertungen von Services abgeben ein Glaubwürdigkeitswert berechnet. Gibt eine Benutzer viele Bewertungen ab, die als Fake identifiziert werden, sinkt die Glaubwürdigkeit des Nutzers und somit der Einfluss seiner Bewertungen auf die Reputation eines Service [MB09].

Hussain et al. stellen Mechanismen vor, wie etwa das *FC direct trust value-based decision making methodology* und die *FC reputation-based trust decision making methodology*, mit deren Hilfe ein Vertrauens- beziehungsweise ein Raputationswert für einen Agenten berechnet werden kann. Außerdem können Aussagen darüber getroffen werdem, wie sich dieser Wert zu einer bestimmten Zeit entwickeln wird [HCH08].

3.2. Vertrauenssysteme

Liu et al. präsentieren mit *StereoTrust* ein gruppenbasiertes Vertrauensmodel. In diesem werden unbekannte Agenten auf Grund von ähnlichen Eigenschaften (Stereotypen genannt) in Gruppen mit Agenten eingeteilt, mit denen bereits Interaktion bestand. Das Vertrauen, das man in die bekannten Agenten setzt, wird dabei auf die anderen Agenten der Gruppe angewandt, wenn man einen Vertrauenswert braucht, da man mit diesen interagieren will. Stehen Informationen von Dritten über die Agenten zur Verfügung, wird die Erweiterung *d-StereoTrust* verwendet, welche mit Hilfe dieser Informationen jede Gruppe in eine gute und schlechte Subgruppe einteilen und eine noch genauere Einschätzung des Vertrauenswertes zulassen [LDRL09].

Trust ME von Huerta-Canep et al. ist ein Vertrauens-System für mobile Umgebungen. Dabei wird der Vertrauenswert in einen anderen Agenten aus drei Quellen berechnet. Erstens persönlichen Erfahrungen mit dem anderen Agenten. Zweitens Informationen dritter Agenten über den anderen Agenten (Reputation). Und drittens Ähnlichkeit von Interessen mit dem anderen Agenten, die man durch direkte Kommunikation mit diesem erfährt. Diese Vielfältigkeit an Quellen zeichnet diesen Ansatz aus [HCLH11].

Jiang et al. entwickelten ein Framework, welches ebenfalls Vertrauenswert in mobilen Umgebungen berechnet. Das Vertrauensmodell erweitert dabei den vorhandenen Security Manager. Für die Berechnung des Vertrauenswertes werden Erfahrungen und Empfehlungen verwendet [JK07]

3.3. Vertrauen in das Android-System

Im Folgenden werden die Ansätze, die sich mit Android im Speziellen beschäftigen, betrachtet. Auch hier gibt es bereits eine große Zahl an Ideen und Methoden. Dabei unterteilt man Systeme in drei große Bereiche, wie das Vertrauen in eine App gesteigert werden kann. Dies wären Privacy (das Vertrauen in die eigene Urteilskraft), Security (das Vertrauen in das System) und Vertrauen (in eine App). Auch wenn es wie bereits angesprochen das Ziel dieser Arbeit ist, nach Lösungen jenseits von Security und Privacy zu suchen, wird auf diese beiden Gebiete eingegangen und mögliche brauchbare Ansätze für ein Vertrauens-System analysiert.

Diese werden betrachtet, da ein Einsatz dieser Systeme das Vertrauen in Apps bezüglich Gefahrenpotenzial und Umgang mit privaten Daten in sofern steigern kann, dass die App dann keine Möglichkeit mehr hat Schaden anzurichten, da sie überwacht. Dies steigert vielleicht nicht das Vertrauen in die App, aber man muss sich um diesen Aspekt keine Gedanken mehr machen.

3.3.1. Privacy - Vertrauen in die eigene Urteilskraft

Mit einem Privacy-System wird dem Nutzer ein Berechtigungssystems gegeben. Mit diesem muss der Nutzer nur die „richtigen“ Entscheidungen treffen und er kann den Apps vertrauen. Wenn er beispielsweise nicht will, dass eine App seinen Standort kennt - er also allen Apps, die diese Berechtigung beinhalten misstraut -, sein Berechtigungssystem ihm aber gestattet dieses Permissions zu entziehen oder falsche Daten zu senden, dann geht von der App keine Gefahr mehr, bezogen auf diesen Punkt aus.

Beresford et al. setzen mit *MockDroid* [BRSS11] direkt am Android OS an und modifizieren dieses. So werden Zugriffe auf Permissions auf Systemebene im Package Manager abgefangen und wenn gewünscht mit falschen Daten, sogenannter *Mocked Data*, getäuscht. Der Benutzer wird über die Notificationbar informiert, wenn eine App auf eine Permission zugreift, welche Mocked Data zurückliefert. So kann eine Verbindung zwischen möglicher fehlender Funktionalität und Mocked Data hergestellt werden und die Permissions gegebenenfalls angepasst werden, wenn man Funktionalität höher als Privacy schätzt.

Das Problem von *MockDroid* ist, dass das Betriebssystem umgeschrieben werden muss. Um das Android OS modifizieren zu können, muss ein Benutzer Root-Rechte besitzen. Android basiert zwar auf dem quelloffenen Linux-Kernel, ist ein Open-Source-System und seine Anpassbarkeit wird als einer der größten Vorteile genannt, doch frei von Problemen ist ein Root-Zugriff nicht. Meist geht ein Garantieverlust mit dem Rooten einher, Malware mit Root-Zugriff kann höheren Schaden anstellen und ein fehlerhafter Root-Vorgang kann sogar das Gerät zerstören. Gerade für unsichere oder unwissende Benutzer wäre deswegen ein System sicherer, das keine Root-Rechte oder eine Veränderung der Firmware benötigt. Solche Systeme werden im Folgenden betrachtet. Diese Systeme modifizieren nicht das OS sondern die App an sich.

AppGuard von Backes et al. [BGH+13] [BGH+12] nutzt Inline Reference Monitoring (IRM) um eine vordefinierte Anzahl von Policies durchzusetzen. Dazu werden die Binärdatein der App umgeschrieben, sodass vor jeder sicherheitsrelevanten Operation ein Sicherheitsmonitor aufgerufen wird. Dieser Monitor gleicht dann mit den gegebenen Policies ab und entscheidet, ob die Funktion durchgeführt wird oder ein alternativer Code ausgeführt wird, der dann gegebenenfalls Mocked Data zurückliefert, damit die App nicht abstürzt.

Xu et al. entwickelten mit *Aurasium* [XSA12] ein System, welches ebenfalls eine App umschreibt, um Funktionsaufrufe der Android API abzufangen und mit den Policies abgeglichen wird. Die App wird damit in eine Sandbox gepackt. Dabei wird Aurasium im selben Prozess wie die App ausgeführt, die sie überwacht und hat keinen eigenständigen Manager in einem seperaten Prozess.

Dr. Android and Mr. Hide von Jeon et al.[JMV+12] baut explizit auf einen Service namens Mr. Hide, der in einem eigenen Prozess läuft. Apps werden mit Hilfe von Dr. Android (Dalvic Rewriter für

Android) umgeschrieben, sodass all ihre API Aufrufe an Mr. Hide umgeleitet werden. Dort wird dann mit Hilfe der dort definierten Policies bestimmt, welche Informationen und Funktionen die App verwenden darf. Dafür wurden ebenfalls die fine-grained (zu dt. fein granulare) Permissions eingeführt. Mit diesen lassen sich detailliertere Bestimmungen über die Rechte von Apps machen, als mit den herkömmlichen Permissions.

Ein weiterer Ansatz ist die von Stach und Mitschang entwickelte ***P**rivacy **M**anagment **P**latform* (PMP) [SM+14] [Sta13a] [SM13] [Sta13b]. Die PMP führt dabei *Ressources* ein, welche die herkömmlichen Permissions von Android ersetzen und dient dabei als Handler zwischen App und OS. Will eine App auf solche Ressourcen zugreifen, muss sie beim PMP nachfragen, welches die Anfrage weiterleitet und Daten zurückgibt oder diese verweigert, beziehungsweise falsche Daten zurücksendet, die keine privaten Informationen preisgeben. Es können neue Ressourcen leicht in die PMP App geladen werden, wodurch sich eine feinere Granularität und eine größere Varianz als mit den statischen Android Permissions erreichen lässt. Der Nachteil aus aktueller Sicht ist jedoch, dass Entwickler ihre Apps auf diese Ressourcen ausgelegt schreiben müssen. Eine jede beliebige App lässt sich mit PMP nicht kontrollieren.

3.3.2. Security - Vertrauen in das System

Wenn entweder sichergestellt werden kann, dass keine Schadsoftware auf dem Gerät ausgeführt werden kann oder dass das System einen Angriffe frühzeitig erkennt und verhindert, dann kann man den Apps in Hinsicht darauf vertrauen, dass sie keinen Schaden anrichtet.

Sun und Tan [ST14] entdeckten, dass Native Libraries von Androids Sicherheitsüberprüfung nicht ausreichend beachtet werden. Mit *NativeGuard* schufen sie ein Sicherheits-Framework, welches Native Libraries von den anderen Komponenten in einer Android App isoliert, indem es sie in einer eigenen App laufen lässt, welche nicht die Permissions der eigentlichen App besitzt und somit nicht auf diese zugreifen kann. Dafür müssen weder Veränderungen am OS noch am Quellcode der App vorgenommen werden.

TrustDroid von Bugiel et al. [BDD+11] ist eine „praktische und leichtgewichte Domänenisolation in Android“. Im Android OS ist keine Isolation zwischen Daten und Appen gegeben, was zum Beispiel bei einem Smartphone welches privat und geschäftlich verwendet wird, wünschenswert wäre. *TrustDroid* überwacht dabei die Inter-Process Comunication (IPC), den Zugriff auf Dateien und den Zugang zum Netzwerk. Dadurch werden Apps und Daten aus einer Domäne vor Apps aus einer anderen Domäne geschützt. Einen ähnlichen Ansatz verfolgen Russello et al. mit *MOSES* [RCCF12]. Sie setzen jedoch auf noch mehr Feingranularität bei der Festlegung welche Daten etc. in welche Domäne fallen, beziehungsweise in welches Security Profile, wie es dort heißt. Dabei ist *MOSES* kontextsensitiv und kann automatisch das Security Profile dem Kontext anpassen.

QUIRE von Dietz et al. [DSP+11] verwendet Provenance um die Herkunft eines Aufrufs in der IPC zu überwachen. Sollte eine App mit wenigen oder keinen kritischen Permissions versuchen auf eine App zuzugreifen, die jene Permissions gegeben hat, schreitet *QUIRE* ein. Die App, auf welche zugegriffen wird, kann diesen Aufruf dann nur mit jenen Permissions bearbeiten, die der aufrufenden App zur Verfügung stehen.

Die Ausnutzung einer App durch eine andere, die weniger Permissions hat, nennt sich *Privilege Escalation Attack* und wird in 3.4.2 noch weiter betrachtet.

Das nächste Unterkapitel beschäftigt sich nun mit Systemen, die zur Bestimmung des Vertrauens in eine App dienen.

3.4. Vertrauen in die App

Um das Vertrauen eine App zu bestimmen gibt es zwei Möglichkeiten. Entweder man vertraut der App, weil dieser App andere Nutzer vertrauen (Unterkapitel 3.4.1) oder man vertraut der App, da man diese analysiert (Unterkapitel 3.4.2 und 3.4.3). Außerdem werden in diesem Unterkapitel bereits bestehende Vertrauensmetriken aufgeführt (Unterkapitel 3.4.4).

3.4.1. Reputationssysteme für Android Apps

Reputationssysteme wurden in Unterkapitel 3.1 ausführlich besprochen. Da die meisten App-Marktplätze über ein eigenes Rating-System verfügen, spielen Reputationssysteme für Android Apps eine eher geringe Rolle. Ein erwähnenswerter Ansatz wäre *AppAware*.

Girardello und Michahelles entwickelten *AppAware*, ein System für ein implizites Rating von Android Apps. Dieser implizite Wert für die Apps berechnete sich daraus, wie viele Benutzer eine App installierten und diese dann updateten oder wieder deinstallierten. Gerade bei Apps mit noch wenigen Downloads erhielten sie dadurch mehr Feedback als durch das Rating-System des Google Play Store [GM10b].

3.4.2. Analysesysteme

Ein möglicher Ansatz um festzustellen, ob man in eine App Vertrauen haben kann oder nicht, sind Analysesysteme, in denen das Verhalten einer App betrachtet wird. Dabei beobachten wir, ob die App Aktionen durchführt, die als schädlich oder ungewollt gelten. Solche Aktionen führen zu einem Vertrauensverlust.

App Analysen werden dabei einerseits auf dem Kontrollfluss einer App ausgeführt, andererseits lässt sich auch der Informationsfluss analysieren.

Ein bekanntes Problem in Android stellen *Privilege Escalation Attacks* dar, welche von Davi et al. erstmals beschrieben wurden. In diesen wird eine Schwäche von Android ausgenutzt, in der eine App mit weniger Permissions Zugriff auf die Komponenten einer privilegierten App hat - also eine App, die über mehr Permissions verfügt. Die zugreifende App kann auf diese Weise Permissions benutzen, die ihr nicht gewährt wurden. Dadurch kann Schaden entstehen, der sich rein durch die Analyse der Permissions wie in Unterkapitel 3.4.3 beschrieben, nicht erahnen lässt. Außerdem kann eine App auf möglicherweise sensitive Daten einer anderen App zugreifen [DDSW11].

Zhongyang et al. entwickelten mit *DroidAlarm* ein statisches Analysewerkzeug für den Kontrollfluss zwischen Apps, um die Gefahr von *Privilege Escalation Attacks* auf Grund von Capability Leaks herauszufinden [ZXMX13]

TaintDroid ist ein von Enck et al. entwickeltes Werkzeug zur effizienten, systemweiten Überwachung des Informationsflusses. Private oder sensitive Daten werden dabei markiert und wenn diese Daten von einer App weitergegeben werden, wird der Benutzer darüber informiert [EGC+14].

3.4.3. Risikoabschätzung durch Permissions

Neben der dynamischen Flussanalyse lässt sich noch eine statische Analyse bezüglich der Permissions durchführen, welche eine App besitzt.

Felt et al. führten eine Studie durch, wie weit Smartphone-Nutzer auf Permissions achten und sich mit diesen auskennen. Sie stellten dabei fest, dass über der Hälfte der Benutzer nicht einmal bewusst war, dass Informationen über die Permissions während der Installation im Store angezeigt werden. Lediglich 20% der Teilnehmer wussten, was ein Großteil der Permissions, die Privacy- oder Security-Risiken mit sich ziehen können, überhaupt bedeuten. Knapp ein Viertel der Probanden gab an, sich über unbekannte Permissions in Reviews der App informiert zu haben. Felt sieht deshalb Reviews, im Idealfall von einem kleinen Expertenkreis geschrieben, als wichtigen ersten Schritt zur Verständlichkeit der Bedeutung von Permissions für den durchschnittlichen Benutzer [FGW11].

Chia et al. stellten fest, dass eine positive Korrelation zwischen der Anzahl der Permissions, die eine App verlangt, und ihrer Popularität sowie einem guten Rating besteht. Dies führen sie darauf zurück, dass mehr Permissions für eine breitere und bessere Funktionalität sorgen und die Beliebtheit einer App sich mehr auf die Funktionalität, denn die Sicherheit bezieht. Sie sehen daher ernste Probleme, dass Benutzer dazu neigen Permissions einfach zu akzeptieren, da sie daran gewöhnt sind, dass Apps viele Permissions verlangen. Somit würde das ganze Permission-System seinen Sinn verlieren [CYA12].

Auf Grund der angezweifelten Effektivität des Permission-Systems wurden einige Versuche gestartet die Permissions zu analysieren, die von Apps verlangt werden und eine Erkenntnis daraus zu gewinnen, ob die Apps ein Risiko darstellen.

Enck et al. entwarfen *Kirin*, ein System, welches dem Benutzer während des Installationsprozesses Empfehlungen gibt, ob er diesen wirklich fortsetzen will. Um mögliche schadhaften Apps identifizieren zu können, wurde ein neunteiliger Regelsatz[1] aufgestellt, welche Kombinationen von Permissions eine App nicht besitzen darf, da diese Permissions als potenzielle Angriffspunkte für schadhaftes Vorgehen dienen [EOM09].

Sarma et al. schlugen eine Risikoerkennung für Apps durch Permissions vor, die über einen einfachen Regelsatz hinaus geht. Sie betrachteten dabei nicht nur die Permissions der zu überprüfenden App, sondern auch jene Permissions anderer Apps in dieser Kategorie. Dadurch werden seltene Permissions in den Kategorien identifiziert, welche als Risikosignal dienen [SLG+12].

[1] Zwei dieser Regeln sind nicht mehr aktuell, da es die enthaltenen Permissions nicht mehr gibt.

Peng et al. führten eine Risk-Score für Apps ein, die ebenfalls auf deren Permissions und jenen Permissions von anderen Apps in der Kategorie beruht. Zur Berechnung der Risk-Score werden Probabilistische Generative Modelle verwendet. Monotonie ist bei der Berechnung für Peng et al. dabei wichtig, sodass die Risk-Score immer durch das Entfernen einer Permissions gesenkt wird. Zusätzlich lässt sich mit Hilfe der Risk-Score ein Ranking unter den Apps berechnen, sodass das Risiko der App im Vergleich zu anderen Apps gesehen werden kann [PGS+12].

Felt et al. analysierten die Android API und stellten ein Mapping zwischen Permissions und API Aufrufen im Quellcode her. Mit diesem Hilfe dieses Mappings entwickelten sie *Stowaway*, ein Werkzeug um Apps zu finden, welche mehr Permissions verlangen, als sie tatsächlich benötigen. In ihrem Testsatz verwendeten etwa ein Drittel der Apps mehr Permissions als sie brauchten, was Felt et al. darauf zurückführen, dass die Android API nicht ausreichend dokumentiert ist [FCH+11].

3.4.4. Vertrauensmetriken für eine App

Um eine Vertrauensmetrik für Android Apps aufzustellen wurden bereits mehrere Ansätze gemacht. Die hier kurz angeschnittenen Ansätze werden in Kapitel 4 weiter vertieft.

Yan et al. entwickelten mit *TruBeRepec* ein Analysesystem, welches die Nutzung einer App durch Benutzer misst und dadurch das Vertrauen des Benutzers in diese App zu messen. Um die Nutzungsdaten in einen Vertrauenswert umzuwandeln, wurde eine Studie durchgeführt, wie das Vertrauen der Benutzer in eine App und ihre Benutzung dieser zusammenhängen [YLNY13].

MAETROID von Dimi et al. welches den Entscheidungsprozess des Analytic Hierarchy Process nutzt, um das Vertrauen von Apps zu bestimmen. Dazu analysieren sie zum einen statische Eigenschaften, schlagen aber auch ein Feedback-System bezüglich Fehlverhalten der App vor. Mit Hilfe dieser Rückmeldungen wird das statisch berechnete Ergebnis gegebenenfalls angepasst [DMM+12] [DMM+13].

Kuehnhausen und Frost berechnen das Vertrauen in eine App aus den Ratings, Reviews und Permissions, welche eine App erhalten hat, beziehungsweise besitzt. Dabei schlagen sie Zuversicht-Metriken für die drei Bereiche vor, um daraus den Vertrauenswert berechnen zu können [KF13].

Wenn einer App vertraut wird, dann muss sichergegangen werden, dass es sich auch wirklich um diese App handelt, wenn man sie herunterläd. Zhou et al. weisen darauf hin, dass viele populäre Appen mit ähnlichem oder gleichen Namen in abgeänderter Form auf Third Party Stores hochgeladen werden. Diese Apps stammen von anderen Entwicklern und verlangen oft mehr Permissions als die original Apps. Um solche umpaketierten Apps zu entdecken, entwickelten sie DroidMoss, welches eine Fuzzy Hashing Technik verwendet, um Unterschiede vom Umpaketieren der Apps zu finden [ZZJN12].

3.5. Vertrauen in den Entwickler

Einer App kann auch vertraut werden, wenn man Vertrauen in ihren Anbieter oder Entwickler hat. In diesem Falle gilt es jedoch sicherzustellen, dass die App auch garantiert von diesem Entwickler stammt.

Barrera et al. haben das Signierungsverfahren für Apps unter Android analysiert und ein eigenes, sichereres alternatives Verfahren vorgestellt [BMCO14] [BCMO12].

Im nächsten Kapitel werden die in Unterkapitel 3.4, speziell in Unterkapitel 3.4.4 genauer betrachtet, um die Anforderungen und Möglichkeiten für eine Vertrauens-Metrik und ein Vertrauens-System für Android Apps zu ergründen.

4. Anforderungen und Umsetzungen

In Kapitel 3 wurde einen Überblick geliefert, welche Systeme und Theorien bereits bezüglich Reputation und Vertrauen bestehen. Außerdem wurden verwandte Ansätze sowie nützliche Werkzeuge vorgestellt. Ziel dieses Kapitel ist es die vielversprechendsten Ansätze zu analysieren, welche sich mit dem Vertrauen in eine App beschäftigen und die jeweiligen Metriken kritisch zu bewerten. Daraus werden die Anforderungen für eine Vertrauensmetrik gewonnen und Überlegungen angestellt, welche Werte in eine eigene Metrik einfließen können. Die Umsetzung der Entscheidungen für die Metrik erfolgt nicht in diesem sondern in Kapitel 5.

Zunächst werden in Unterkapitel 4.1 die Systeme und Metriken grundlegend betrachtet. Welchen Wert liefern die Metriken dem Benutzer zurück, welche Eigenschaften einer App werden dafür betrachtet und welche Aussage kann dadurch über das Vertrauen in eine App getroffen werden?

In Unterkapitel 4.2 werden die jeweiligen Faktoren betrachtet, die in die Metrik einfließen können und welche Bereiche sie beeinflussen. Ausführlich wird dann in den Unterkapiteln 4.3 bis 4.5 untersucht, wie die Faktoren Reputation, Nutzung und verlangte Permissions einer App in die Metrik einfließen. Den Ansatz der Kontroll- und Datenflussanalyse, welcher in keiner der bestehenden Vertrauensmetriken Verwendung findet, wird in Unterkapitel 4.6 untersucht.

Abschließend wird in Unterkapitel 4.7 überlegt, wie andere Elemente wie der Entwickler oder der App-Marktplatz, aus welchem die App bezogen wird, das Vertrauen in die App beeinflussen können.

4.1. Grundaussage einer Vertrauensmetrik

Die erste Überlegung bezüglich einer Metrik sollte sein, welche Aussage mit ihr getroffen werden soll. Eine Metrik für das Vertrauen in eine App sollte dem Betrachter eine Aussage geben können, ob er diese App installieren will oder es lieber lassen sollte. Die Entscheidung des Installierens oder nicht Installieren wird aber auch oft dadurch beeinflusst, dass sich viele andere Apps finden lassen, welche dieselbe Aufgabe erfüllen. Es gilt also zu ergründen, wie der Wert einer Metrik aussehen sollte, damit er auf beide Probleme angewendet werden kann.

Als nächstes stellt sich natürlich die Frage, was das Vertrauen in eine App bedeutet und wie eine Metrik dieses berechnen könnte. In Kapitel 2 wurde in der Definition 2.3.2 auf Seite 18 für den Ansatz diese Arbeit bestimmt, dass einer App vertraut wird, wenn sie die versprochenen Funktionalitäten bietet, private Daten nicht gegen den Wunsch des Besitzers weitergibt und möglichst niedriges Potenzial besitzt, um Schaden anzurichten. Um dafür eine Metrik aufstellen zu können, müssen messbare Werte - im Folgenden Faktoren genannt - gefunden werden, durch welche sich Werte für das Vertrauen in die drei Bereiche aus der Definition errechnen lassen.

In diesem Unterkapitel wird die Fragestellung der am besten geeigneten Werteskala für die Metrik und welche Aussage aus dem Ergebnis der Metrik getroffen werden kann behandelt. Außerdem werden Überlegungen angestellt, wie die drei Bereiche aus der Definition zueinander in Relation gestellt werden und der individuellen Wichtigkeit eines Benutzers angepasst werden können. Mit den Faktoren beschäftigt sich, wie in der Einleitung des Kapitels angekündigt, erst das nächste Unterkapitel 4.2.

4.1.1. Die Werteskala

Um eine ideale Werteskala für unsere Metrik zu finden, werden nun einige Überlegungen angestellt. Es wird aber auch auf die bereits in Unterkapitel 3.4.4 vorgestellten Metriken eingegangen werden und diese auf ihre Tauglichkeit überprüft.

Der einfachste Ansatz für das Ergebnis der Metrik wäre eine binäre Entscheidung, die *Vertrauen* oder *kein Vertrauen* zurückliefert. Dies ist jedoch ein sehr simpler Ansatz und es fällt schwer eine Grenze zu ziehen. Wo setze ich diesen absoluten Bruch an? Sollen wir eher einer App misstrauen oder riskieren zu naiv zu sein? Und wie verhalten sich zwei Apps zueinander, denen beiden vertraut wird? Für welche sollte ich mich entscheiden? Eine binäre Entscheidung liefert für unsere Anforderungen keine zufriedenstellende Werteskala.

Dini et al. [DMM+13] gehen lediglich einen Schritt weiter. Sie unterteilen das Maß des Vertrauens in eine trinäre Entscheidung. Einer App wird entweder als *vertrauenswürdig* (d.h.: „Die App funktioniert korrekt und sollte keine bösartigen Funktionalitäten verbergen.“), *nicht vertrauenswürdig* (d.h.: „Die App könnte die Sicherheit des Smartphones gefährden.“) oder *trügerisch* (d.h.: „Weder funktioniert die App korrekt, noch ist sie sicher“). Hier ergibt sich wie bei einer binären Entscheidung das Problem, dass die einzelnen Kategorien zu weit gefasst sind.

Der Vorteil einer groben Kategorisierung ist jedoch, dass die Metrik eine klare und unmissverständliche Aussage trifft. Dies dient also als eine einfache Orientierung für den Betrachter des Ergebnisses.

Die am feinsten gegliederte Skala der aufgeführten Metriken ist jene im Prozentbereich, wie sie Yan et al. [YLNY13] beziehungsweise Kuehnhausen und Frost [KF13] verwenden. Sie lässt uns einen großen Spielraum bei der Bewertung des Vertrauensmaßes. Jedoch muss hier darauf geachtet, dass beim Benutzer keine Verwirrung entsteht. Was bedeutet beispielsweise ein Vertrauen von *60%*? Ist dies ein guter Vertrauenswert oder ein schlechter?

Die in dieser Arbeit verwendete Lösung ist ein Hybrid aus einer Einteilung in Kategorien und einem parallel berechneten prozentualen Vertrauenswert. Die Kategorien geben einen Anhaltspunkt wie eine App einzuordnen ist und welche Gründe für einen gesenkten Vertrauenswert gesorgt haben. Der prozentuale Wert sorgt für eine feinere Granularität, um innerhalb der Kategorien eine Unterscheidung zu ermöglichen.

Diese Lösung verbindet die Vorteile beider Ansätze: Der Benutzer kann einschätzen, was der Vertrauenswert bedeutet, den er bekommt. Aber Apps, die nur aus einem einzelnen oder wenigen Gründen in eine schlechtere Kategorien gerutscht sind, lassen sich leichter als solche erkennen. Außerdem wird dem Benutzer eine Möglichkeit geboten Apps zu vergleichen, welcher eher zu vertrauen ist, selbst wenn beide in einer guten Kategorie eingeordnet sind.

Im *Google Play Store* müssen Apps in vier Bewertungsstufen eingeteilt werden[1]. Diese beschäftigen sich jedoch mit dem Inhalt einer App wie anzügliche Inhalte und Glücksspiel. Einzig das Abrufen und Veröffentlichen des Standortes spielt in dieser Bewertung und auch in unserer eine Rolle. Ansonsten sind diese Bewertungen für unseren Ansatz nicht zu gebrauchen.

Deswegen werden zum Zweck der Einordnung fünf eigens aufgestellte Kategorien verwendt (siehe Abbildung 4.1), welche die Vertrauenswürdigkeit einer App repräsentieren sollen. Diese Kategorien sind:

- *K1: Volles Vertrauen*: Aus keinem der einzelnen Bereiche gibt es Grund Bedenken bezüglich des Vertrauens in die App zu haben. Um in diese Kategorie fallen zu können, muss die App durchweg sehr gut bewertet sein und darf keinerlei Permissions besitzen, welche die Security oder Privacy gefährden können.
- *K2: Hohes Vertrauen*: Die App hat keine perfekte Bewertung oder besitzt kritische Permissions bezüglich Privacy und Security. Die Nutzung und die Bewertung lassen trotzdem darauf schließen, dass die Funktionalität der App gut ist und ein erhöhtes Gefahrenpotenzial oder eine Weitergabe privater Daten wurde nicht festgestellt.
- *K3: Geringe Bedenken*: Funktionalität, Gefahrenpotenzial oder Umgang mit privaten Daten geben Anlass zu geringen Bedenken. Das Rating und die Nutzung kann Hinweise auf kleine Mängel an der Funktionalität geben. Die Permissions der App können verdächtig erscheinen. Daten wie Standort und Telefon-ID werden an Werbeserver weitergegeben.
- *K4: Größere Bedenken*: Diese Kategorie besagt, dass entweder die Funktionalität wahrscheinlich deutliche Mängel aufweist, ein erhöhtes Schadpotential besteht oder private Daten, die über Werbezwecke hinaus gehen wahrscheinlich an Dritte weitergegeben werden.
- *K5: Geringes Vertrauen*: In dieser Kategorie landen Apps mit wahrscheinlich stark eingeschränkter Funktionalität, mit gemeldeten Fehlverhalten, was das Schadpotenzial angeht und der wahrscheinlichen Weitergabe privater Daten wie persönlichen SMS oder den gespeicherten Kontakten im Smartphone.

Es ist dabei eher unwahrscheinlich, dass eine App in Kategorie *K1* landet, da viele Apps kritische Permissions benötigen, um ihre Funktionalität umsetzen zu können. Ein volles Vertrauen kann einer App jedoch nicht geschenkt werden, wenn sie auch nur das geringste Potenzial besitzt, Schaden anzurichten. Diese Kategorie dient primär dazu, dass eine vollständige Definition des Wertebereichs besteht.

Realistisch gesehen ist Kategorie *K2*, welche ein hohes Vertrauen bedeutet, der beste Wert den eine App erreichen kann. Auf Grund verschiedener Faktoren kann eine App in die Kategorien *K3* bis *K5* eingeordnet werden. Kategorie *K5* kommt dabei der Empfehlung gleich, die App in keinem Fall zu installieren. Um eine solche Empfehlung geben zu können, sollten begründete Zweifel vorliegen der App nicht zu vertrauen. Deswegen wird dieser Kategorie nur erreicht, wenn eine aktive Rückmeldung zum Fehlverhalten der App vorhanden ist und nicht ausschließlich auf Grund von Vermutungen. Dazu mehr in Unterkapitel 4.2.

[1] http://goo.gl/Bd7tm0

Abbildung 4.1.: Die Kategorien, in welche eine App fallen kann.

Eine App fällt dabei in die schlechteste Kategorie, in die sie in einem der drei Bereiche eingeordnet wurde. Besitzt eine App beispielsweise gute Ratings und wird viel genutzt, fällt sie also in Kategorie *K2* bezüglich der Funktionalität. Gibt die App den Standort zu Werbezwecken an Werbeserver weiter landet sie in Kategorie *K3* bezüglich dem Umgang mit privaten Daten. Wurde häufiger Guthabenverlust gemeldet, wodurch der Verdacht sich verstärkt, dass die App heimlich im Hintergrund SMS an Premiumnummern schickt, fällt sie im Gefahrenpotenzial in Kategorie *K5*. Damit fällt die App im gesamten Vertrauen in Kategorie *K5*.

Aus diesem Beispiel wird ersichtlich, dass kein klares Mapping zwischen Vertrauenswert und der Vertrauenskategorie, in welche die App fällt, möglich ist. Durch ihr relativ gutes Abschneiden in Hinblick auf Funktionalität und Umgang mit privaten Daten kann der Vertrauenswert der App immer noch vergleichsweise hoch sein. Aus diesem Grund spielt in solchen Sonderfällen die Vertrauenskategorie eine sehr wichtige Rolle.

Da diese Metrik nicht den Anspruch stellt als Sicherheitssoftware zu agieren, sondern nur den Verdacht liefert, dass eine App Schaden anrichten kann, bleibt es darüber hinaus dem Benutzer überlassen der App trotzdem zu vertrauen. Hierbei ist jedoch die Kommunikation wichtig, aus welchem Grund die App in die schlechte Kategorie fällt und warum ihr Vertrauenswert trotzdem so hoch ist.

Um den Einfluss der einzelnen Bereiche möglichst stark zu halten wird für den Vertrauenswert der App die Summe der einzelnen Vertrauenswerte in die Bereiche verwendet. Für Funktionalität, Gefahrenpotenzial und Umgang mit privaten Daten wird also ebenfalls ein Vertrauenswert auf der Skala von 0 bis 100% berechnet.

Nachdem mit dieser Hybridlösung der Wertebereich der Skala festgelegt wurde, wird im nächsten Unterkapitel über die Gewichtung diskutiert, mit welcher die drei Bereiche Einfluss auf den Vertrauenswert nehmen.

4.1.2. Individualität des Vertrauens

Wie bei der Aufstellung der Definition für Vertrauen in eine App schon festgestellt wurde, ist Vertrauen eine individuelle Ansicht. Die drei Bereiche Funktionalität, Gefahrenpotenzial und Umgang mit privaten Daten können dabei für jeden Benutzer von unterschiedlicher Wichtigkeit sein. Manche Benutzer legen mehr Wert auf hohe Funktionalität als auf ein niedriges Gefahrenpotenzial, andere sind darauf bedacht, dass ihre privaten Daten nicht in die Hände Dritter gelangen. Deswegen sollte dem Nutzer die Möglichkeit geboten werden die Bereiche zu gewichten.

Für die Metrik in dieser Arbeit stehen dem Benutzer vier Stufen der Wichtigkeit zur Verfügung:

- *W1: Sehr wichtig*: Dieser Bereich ist dem Benutzer sehr wichtig. Er wird deutlich stärker gewichtet.
- *W2: Wichtig*: Der Bereich spielt für den Benutzer bei einer App eine Rolle. Dies ist die Standardeinstellung und sorgt für eine gleichmäßige Gewichtung, wenn keine Anpassungen an der Gewichtung durch den Benutzer durchgeführt wurden.
- *W3: Weniger Wichtig*: Der Bereich ist dem Benutzer weniger wichtig als die anderen. Er wird bei der Berechnung des Vertrauenswertes schwächer gewichtet.
- *W4: Unwichtig*: Fließt nicht in die Berechnung mit ein. Sorgt damit auch nicht für die Einordnung in die Kategorien.

Die Gewichtungen *W1* bis *W3* bestimmen dabei primär den Einfluss, den ein Bereich auf den Vertrauenswert nimmt. Die Gewichtungen stehen für den Multiplikationsfaktor, mit welchem die Bereiche bei der Aufsummierung verrechnet werden.

Lediglich wenn ein Bereich mit *W4* gewichtet ist sorgt er nicht mehr dafür, dass eine App auf Grund der Kategorie in diesem Bereich in diese Gesamtkategorie abrutscht. Selbst wenn aus dem obigen Beispiel das Gefahrenpotenzial mit *W3* gewichtet wäre und beide anderen Bereiche mit *W1*, würde die App trotzdem in Kategorie *K5* fallen. Ein anderes Vorgehen würde die Idee hinter den Kategorien zunichtemachen.

Nachdem die Werteskala für Vertrauenswert und Vertrauenskategorien aufgestellt wurde und die Verrechnung und Gewichtung der einzelnen Bereiche dafür bestimmt wurde, werden nun die Faktoren betrachtet, die Einfluss auf das Vertrauen in eine App beziehungsweise einen der drei Bereiche der App haben.

Metrik von	Ratings	Permissions	Nutzung	Problemmeldungen
Yan et al. [YLNY13]	ja	nein	ja	nein
Kuehnhausen et al.	ja	ja	nein	nein
Dini et al. [DMM+13]	ja	ja	nein	ja

Tabelle 4.1.: Übersicht welche Vertrauens-Faktoren in den betrachteten Metriken ein Rolle spielen.

4.2. Die Faktoren und Bereiche

In den betrachteten Metriken spielen unterschiedliche Faktoren eine Rolle bezüglich des Vertrauens in eine App. Aus ihnen lassen sich Werte bezüglich des Vertrauens in eine App berechnen. Eine kurze Übersicht, welche Faktoren von wessen Metrik verwendet wurden, liefert Tabelle 4.1.

Diese Faktoren werden im Folgenden ausführlich erklärt und Überlegungen angestellt, welche unserer drei Bereiche (Funktionalität; Gefahrenpotenzial; Zugriff auf private Daten) durch sie beeinflusst werden.

Ratings spielen bei Yan et al. [YLNY13] wie auch bei Kuehnhausen und Frost [KF13] eine Rolle. Über diese subjektive Bewertung der App durch Benutzer lassen sich primär Schlüsse über die Funktionalität der App ziehen. Eine schadhafte App tendiert dazu ein schlechteres Rating zu haben, aber es fällt eher schwer dies bewerten zu können. Über den Umgang der App mit privaten Daten lassen sich ebenfalls eher keine Schlüsse ziehen, da die wenigstens Nutzer Einsicht in diese Vorgänge haben und so ein Einfluss auf ihre Bewertung für die App unwahrscheinlich erscheint. Eine ausführliche Überlegung zu Ratings findet sich in Unterkapitel 4.3.1.

Permissions werden von Kuehnhausen und Frost [KF13] sowie Dini et al. [DMM+13] in der Metrik verwendet. Man könnte die Permissions verwenden, um eine Aussage über die Funktionalität der App zu machen (zum Beispiel wäre es für eine Foto-App hinderlich nicht die *CAMERA* Permission zu besitzen), doch dieser Ansatz ist sehr wage und wird nicht weiter verfolgt. Einen besseren Anhaltspunkt liefern die Permissions für das Gefahrenpotenzial. Auch lassen sich Schlüsse für den Umgang mit privaten Daten ziehen, da ein Missbrauch hier nur möglich ist, wenn die App dazu auch die nötigen Permissions besitzt um darauf zugreifen zu können. Wie dies in der Metrik umgesetzt wird, wird in Unterkapitel 4.5 erläutert.

Durch die **Nutzung** der App versuchen Yan et al. [YLNY13] auf die Funktionalität der App zu schließen. Wird eine App viel genutzt, kann davon ausgegangen werden, dass sie die Funktionalität bietet, die der Nutzer von ihr erwartet. Hier greift für Gefahrenpotenzial und Missbrauch privater Daten dieselbe Argumentation wie bei der Reputation. Wenn Benutzer das eine oder andere feststellen hören sie eventuell auf die App zu nutzen, [pew12]. Jedoch ist es unmöglich festzustellen, ob dies die Gründe dafür sind. Wie die Nutzung in die Metrik einfließt, erörtert Unterkapitel 4.4.

Um die Unzufriedenheit von Benutzern mit einer App genauer ergründen zu können, reicht es nicht aus nur die Reputation oder die Nutzung zu betrachten. Deswegen sollte man ihm die Möglichkeit der **Problemmeldung** gewähren. Dini et al. [DMM+13] lassen in ihrem Ansatz die Benutzer zu

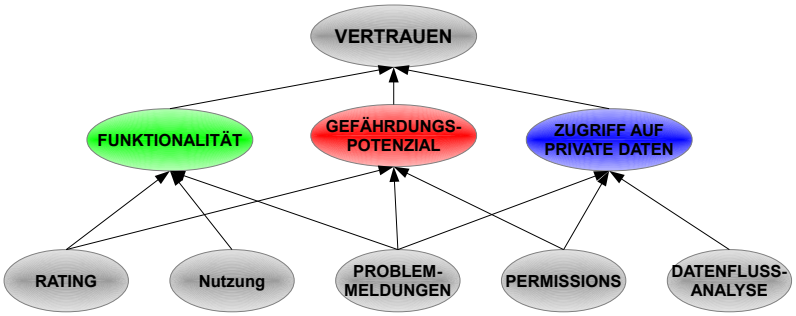

Abbildung 4.2.: Faktoren, die das Vertrauen in eine App beinflussen und mit welchen messbaren Werten sie zusammenhängen.

verschiedenen Fehlverhalten der App eine Rückmeldung machen. Je nachdem, welche Möglichkeiten zur Problemmeldung dem Benutzer gegeben werden, können dieses alle drei Bereiche beeinflussen. Gedanken dazu, welche Fragen beim Feedback gestellt werden und wie diese dann in der Metrik umgesetzt werden, finden sich in Unterkapitel 4.3.3.

Ein Ansatz, der bei keiner der betrachteten Metriken eine Rolle spielt ist die **Flussanalyse**. Mit Hilfe der *Datenflussanalyse* von *TaintDroid* [EGC+10] [EGC+14] lassen sich private Daten markieren (orig. taint Data), sodass ein ungewollter Zugriff und Missbrauch dieser Daten bemerkt wird. Solche Zugriffe verletzen eventuell den Wunsch des Benutzers, wie die App mit seinen privaten Daten umgeht. Mit der *Kontrollflussanalyse* lassen sich die in Unterkapitel 3.4.2 erwähnten *Privilege Escalation Attacks* entdecken, über welche Daten und Permissions von Apps genutzt werden können, die eigentlich keinen Zugriff darauf haben sollten, was das Gefahrenpotenzial wie den Umgang mit privaten Daten betrifft. Wie die Entdeckungen durch *Flussanalyse* in die Metrik einfließen können, wird in Unterkapitel 4.6 beschrieben.

Insgesamt bieten sich also fünf messbare Faktoren an, die für unsere Metrik eine Rolle spielen. All diese Faktoren beeinflussen, wie bereits beschrieben, einen oder mehrere unserer drei Bereiche. Abbildung 4.2 fasst diese Zuordnung noch einmal zusammen.

Nun werden die verschiedenen Faktoren ausführlich analysiert. Begonnen wird mit der Reputation einer App, welche Ratings, Reviews und Problemmeldungen umfasst.

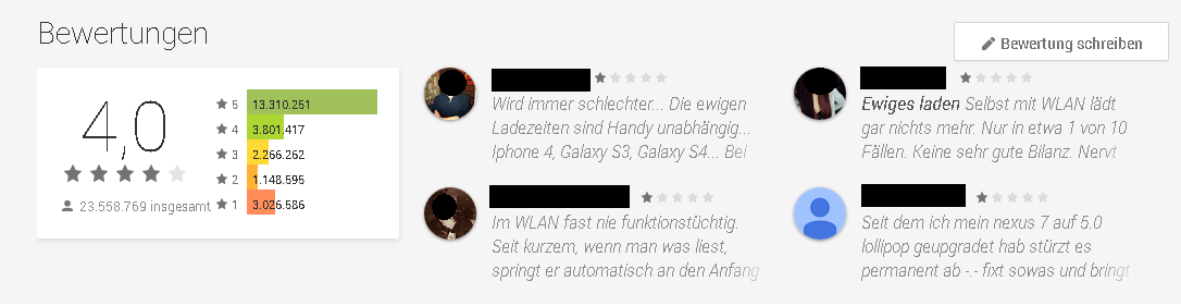

Abbildung 4.3.: Anzeige von Ratings und Reviews im Google Play Store. Diese können als Quelle für eine Vertrauensmetrik verwendet werden.

4.3. Reputation

Reputation ist ein wichtiger Faktor, um das Vertrauen in eine App berechnen zu können. Drei Faktoren lassen sich für die Reputation in eine App zurate ziehen: Ratings, Reviews und Problemmeldungen. Diese lassen sich in strukturierte und unstrukturierte Daten unterteilen, wobei Ratings und Problemmeldungen in erstere und Reviews in letztere Kategorie fallen. Ratings werden in den meisten Fällen auf einer Skala von einem bis fünf Sternen abgegeben, Problemmeldungen sind vorwiegend eine binäre Entscheidung (eine Problemmeldung wurde abgegeben oder nicht). Reviews im Gegensatz sind Fließtexte, denen kein direkter Wert entnommen werden kann. Eine tiefgehende Analyse des Textes muss durchgeführt werden, um eine Erkenntnis daraus gewinnen zu können.

Ratings und Reviews lassen sich leicht aus den verschiedenen App-Marktplätzen beziehen, etwa dem *Google Play Store* (Siehe Abbildung 4.3). Für Problemmeldungen besteht keine solche Quelle, vor allem, wenn man selber auswählen will, welche Probleme betrachtet werden sollen. Hierfür ist ein eigenes System beziehungsweise Konzept nötig. Für Ratings und Reviews könnte auch ein eigenes System verwendet werden, doch gerade die Vielzahl an Bewertungen in den App-Marktplätzen stellen einen Vorteil da, der mit einem eigenen System nur schwer zu erreichen wäre.

Dass Reputationssysteme einige Schwächen haben wurde in Unterkapitel 3.1 bereits angesprochen. Angriffe, um die Bewertung einer App gezielt besser oder schlechter zu machen stellt ein Problem dar. Es gibt Möglichkeiten ein eigenes System gegen solche Angriffe robuster zu machen und Ansätze, Daten anderer Systeme zu interpretieren, wenn über deren Sicherheit gegen solche Angriffe keine Aussage gemacht werden kann. Wie diese Probleme in den bereits bestehenden Systemen behandelt wurden, wird im Folgenden beschrieben, wenn auf die einzelnen Werte eingegangen wird.

Zunächst werden nun Ratings betrachtet und wie sie Einfluss auf das Vertrauen in eine App nehmen können.

4.3.1. Rating

Wie bereits erwähnt stellt ein App-Store eine gute Quelle für die Ratings einer App dar. Natürlich lässt sich hier nur schwer feststellen, wie und ob das System gegen Angriffe gesichert ist. Um eine

Bewertung abgeben zu können, muss man nur ein Google-Profil besitzen und die App auf einem Gerät installiert haben. „Bad Mouthing“, das Senken der Wertung einer App durch zahlreiche Abgabe fälschlich schlechter Bewertungen, und „Ballot-Stuffing“, die Erhöhung der Wertung einer App durch bewusst zu gute Bewertungen, stellen dadurch durchaus mögliche Angriffsmethoden dar.

Was jedoch für eine Quelle wie den *Play Store* spricht, ist seine hohe Anzahl an Nutzern. Je mehr Personen eine App bewerten, desto mehr „falsche Bewertungen“ werden benötigt, um eine Auswirkung zu haben. Diese Überlegung lässt sich mit dem Ansatz von Kuehnhausen und Frost [KF13] kombinieren. Sie berechnen, welche Zuversicht man in das Rating haben kann und verwenden dazu unter anderem die Anzahl der Ratings für eine App. Dafür verwenden sie die Studentsche t-Verteilung und kommen zu guten Ergebnissen.

Als zweites Maß für die Zuversicht in das Rating wird die Verteilung der Ratings von Kuehnhausen und Frost [KF13] verwendet. Dabei wird betrachtet, ob die Ratings sich gleichmäßig verteilen (nicht zu hohe Zuversicht in den Wert des Rating), gegen einen Wert (hohe Zuversicht) korrekt oder gegen zwei Werte (kaum Zuversicht) tendieren.

Die Werte für die Zuversicht in das Rating werden anschließend mit dem durchschnittlichen Rating der App verrechnet und daraus ergibt das Vertrauen in eine App bezüglich der Ratings. Diese Metrik kommt dem Problem von falschen Bewertungen teilweise bei und gibt durch die Analyse der Verteilung der Ratings einen guten Anhaltspunkt, ob der Wert der Ratings aussagekräftig ist. Aus diesem Grund wird die Metrik für Ratings von Kuehnhausen und Frost [KF13] für die Metrik in dieser Arbeit übernommen.

Angemerkt werden sollte, dass eine Metrik bezüglich der Ratings immer an Aussagekraft verliert, wenn nicht genug Ratings für eine App vorliegen. Laut der Analyse des Google Play Stores von AppBrain.com gab es Ende November 2014 über eine halbe Millionen Apps mit weniger als drei Ratings, was vierzig Prozent der Apps ausmacht. Von diesen lässt sich eher kein verlässlicher Wert für das Vertrauen gewinnen. Kuehnhausen und Frost [KF13] wählen in ihrer Metrik die Mindestanzahl von sieben Ratings für eine App. Ansonsten ist das Vertrauen in eine App auf Grund der Ratings in ihrer Metrik 0.

Einer App nicht zu vertrauen, da eine Aussage über ihre Vertrauenswürdigkeit nicht möglich ist, ist besser, als ihr unbegründet zu vertrauen. Ist das Vertrauen in die Funktionalität einer App durch diesen Grund 0, sollte dem Benutzer jedoch auf jeden Fall kommuniziert werden, warum dies der Fall ist.

Im Folgenden werden einige Gedanken zu Reviews angestellt und wie sie für das Vertrauen einer App eine Rolle spielen könnten.

4.3.2. Reviews

Kuehnhausen und Frost [KF13] verwenden Reviews als Faktor für ihren Vertrauenswert. Sie analysieren die Reviews auf Schreibfehler und die Anzahl der lobenden und kritisierenden Worte. Mit ersterem stellen sie fest, ob das Review selber vertrauenswürdig ist, wobei eine bessere Rechtschreibung ein höheres Vertrauen bewirkt. Mit der Anzahl der lobenden oder kritisierenden Worte wird die Stimmung des Reviews erfasst und aus dieser das Vertrauen in die App aufgrund des Reviews berechnet.

Diese Berechnung ließe sich noch erweitern. Im *Google Play Store* etwa lassen sich Reviews von anderen Benutzern bewerten. Einem sehr positiv bewertetem Review könnte trotz Schreibfehlern Vertrauen geschenkt werden. Eine stärkere Gewichtung eines solchen Reviews wäre auch eine Überlegung.

Ebenfalls könnten die Reviews nach bestimmten Schlagworten wie zum Beispiel „Virus“ oder ähnlichen Warnungen gesucht werden und bei einer entsprechend hohen Anzahl der Vertrauenswert bezüglich Gefahrenpotenzial gesenkt werden.

Die Reviews leiden jedoch unter demselben Problem wie die Permissions, nicht jede App besitzt Reviews. Dabei wiegt die Zahl der Apps ohne Reviews noch schwerer. In der Testmenge von Kuehnhausen und Frost [KF13] besaßen über 58% der Apps keine Reviews.

Auf Grund des großen Mangels an Reviews und der Tatsache, dass unstrukturierte Daten schwerer zu erfassen sind und keine für diese Arbeit zufriedenstellenden Werte lieferte, wurde entschieden, dass Reviews nicht in die Metrik mit einfließen. Aus diesem Grund finden sie auch nur an dieser Stelle der Vollständigkeit halber Erwähnung.

4.3.3. Problemmeldungen

Reviews stellen für den menschlichen Benutzer gesehen eine gute Informationsquelle dar. Die textuelle Beschreibung kann hinweise auf Schwächen einer Apps hinweisen und auch aus anderen Gründen vor ihr warnen. Die automatische Analyse solcher Texte liegt jedoch jenseits des Umfang dieser Arbeit.

Eine Möglichkeit gezielte Informationen über eine App beziehen zu können bieten dabei Problemmeldungen. *MAETROID*, das von Dini et al. [DMM+13] entworfene System, verwendet Problemmeldungen, um den statisch berechneten Vertrauenswert in eine App dynamisch anpassen zu können.

Zu diesem Zweck kann ein Benutzer zu fünf möglichen, festgestellten Fehlverhalten der App Rückmeldung geben, ob diese gar nicht, selten oder oft aufgetreten sind. Die möglichen Fehlverhalten wurden dabei danach ausgewählt, dass sie gute Indikatoren für Malware Apps darstellen. Diese möglichen Fehlverhalten sind dabei *Abstürze*, *erhöhter Batterieverbrauch*, *Usability*, *Guthabensverlust (durch SMS)* und *Bugs*.

Ziel dieser Metrik sollte es jedoch sein ein Fehlverhalten einem der drei Bereiche zuweisen zu können und nicht eine allgemeine Aussage zu tätigen. *Abstürze*, *Usability* und *Bugs* lassen sich dabei der Funktionalität der App zuordnen. *Erhöhter Batterieverbrauch* und *Guthabensverlust (durch SMS)* betreffen das Gefahrenpotenzial der App. Für dieses lassen sich noch die Möglichkeiten *Versteckte Installation oder Deinstallation von Apps* sowie *Gebühren durch Netzwerkaufkommen* hinzufügen.

Die Datenflussanalyse durch *TaintDroid* lässt sich verwenden, um Problemmeldungen bezüglich des Umgangs mit privaten Daten abzugeben. Da der Benutzer ohne ein Werkzeug keine Einsicht besitzt was mit seinen Daten geschieht, benötigt er ein Hilfsmittel um bessere Rückmeldung geben zu können. Die Möglichkeit von *TaintDroid* werden nicht an dieser Stelle, sondern in Unterkapitel 4.6 behandelt.

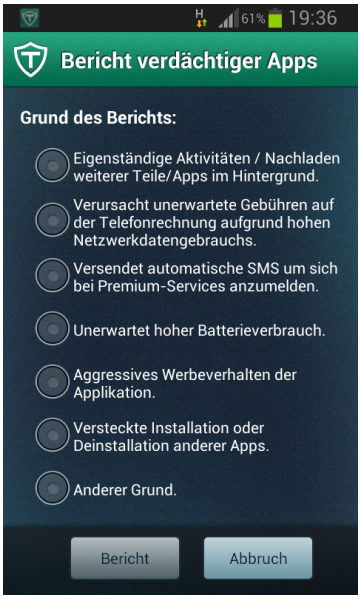

Abbildung 4.4.: Möglichkeiten in der TrustGo App entdeckte Probleme einer App zu melden.

Durch diese direkte Bewertung bezüglich des Fehlverhaltens einer App kann guter Rückschluss auf die App in den drei Bereichen gezogen werden und das Vertrauen der App kann in Kategorie *K4* fallen, denn hier verlassen wir das Feld der Spekulationen wie bei den Ratings und Reviews. Trotzdem bleibt es natürlich in den meisten Fällen die subjektive Wahrnehmung von Benutzern.

Für die Metrik dieser Arbeit wurden nun spezielle Rückmeldungen festgelegt. Da es kein System gibt, das genau diese abfragt, sollte ein eigenes geschaffen werden. Dazu gibt es zwei verschiedene Möglichkeiten.

Der eine Ansatz ist es dem Benutzer eine Meldemöglichkeit zu gewähren, die dieser wahrnehmen kann, wenn er will. Ein Beispiel dafür wäre die Meldefunktion der App TrustGo[2] wie in Abbildung 4.4 zu sehen. Bei dieser kann mit einer Meldung ein Fehlverhalten der App gemeldet werden.

Im Gegensatz zur ausschließlichen Meldung von Fehlverhalten beinhaltet *MAETROID* die Möglichkeit auch abzusenden, dass alles mit dem System in Ordnung ist. Dadurch besitzt die von Dini et al. [DMM+13] vorgeschlagene Rückmeldung auch die Möglichkeit des positiven Feedbacks. Dazu muss eine App jedoch bei jeder Meldung auf alle vorhandenen Rückmeldepunkte bewertet werden und ein Benutzer kann eine App nur ein einziges Mal bewerten.

Für die Metrik dieser Arbeit wird der Ansatz von Dini et al. [DMM+13] gewählt. Will ein Benutzer eine App bewerten, wird er zumindest zu allen Möglichkeiten aus einem der drei Bereiche befragt, zu dem er ein Problem melden will. Pro Benutzer gibt es auch nur eine Rückmeldung, sollte der Benutzer eine neue Problemmeldung senden, werden die Werte der alten Problemmeldung denen der neuen angepasst.

[2] http://www.trustgo.com/

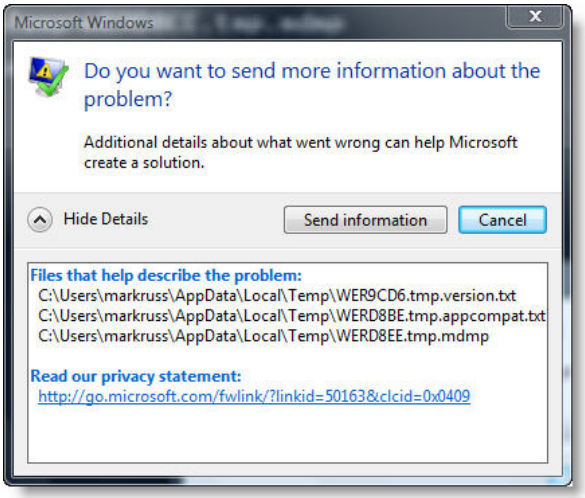

Abbildung 4.5.: Ein Beispiel für eine Windows Problemmeldung, bei welcher nach der Erlaubnis des Benutzers zusätzliche Informationen mitgesendet werden.

Überlegenswert wäre auch eine Problemmeldung, welche vorgefallene Probleme automatisch sammelt und nachdem der Benutzer es legitimierte an einen Server schickt. Vergleichbar mit der Windows Problemmeldung (siehe Abbildung 4.5).

Dazu können die Ergebnisse von TaintDroid für private Daten oder logcat-Informationen[3] bezüglich Abstürzen einer App mitgeschickt werden. Dadurch lassen sich präzisere Aussagen treffen, als nur durch die Vermutungen des Benutzers.

Das Problem ist, dass sich jedoch nicht alle vorgeschlagenen Probleme systematisch feststellen lassen. Es ist für die möglichen messbaren Werte auf jeden Fall ein guter Ansatz, der weiter verfolgt werden sollte. In dieser Arbeit wird er jedoch keine Verwendung finden.

Angriffe auf das System mit unberechtigten Problemmeldungen sollten natürlich verhindert werden. Dazu gibt es zwei wichtige Verteidigungsmechanismen. Zum einen sollte immer klar sein, von wem die Problemmeldung kommt. Dazu empfiehlt sich zum Beispiel die Verknüpfung mit der Problemmeldungen mit dem Google-Account (ohne welchen die Person sowieso keine Android Apps verwenden könnte). Darüber hinaus schlagen Dini et al. [DMM+13] einen Zeitfaktor als Bremse vor. Werden massenhaft Problemmeldungen in kurzer Zeit abgegeben, ist davon auszugehen, dass es sich dabei um kein gewöhnliches Benutzerverhalten, sondern um einen Angriff handelt. Die Effektivität dieses Ansatzes bewiesen sie in mehreren Experimenten.

Eine weitere Möglichkeit, um vor falschen Meldungen zu schützen, stellt die Überprüfung dar, ob eine Problemmeldung überhaupt Sinn macht. Guthabensverlust durch von der App versendete SMS ist nicht möglich, wenn diese App gar nicht die Permissions besitzt SMS versenden zu können. Mit diesen Vorkehrungen sind die Problemmeldungen gegen die meisten Angriffe robust.

[3] siehe: `http://developer.android.com/tools/help/logcat.html`

Wie aus dem Nutzerverhalten einer App Schlüsse auf deren Funktionalität gezogen werden können, wird im nächsten Unterkapitel betrachtet.

4.4. Nutzerverhalten der App

Einen Anhaltspunkt für die Qualität und Funktionalität einer App gibt das Nutzerverhalten. Wenn eine App gut ist und die Funktionalitäten bietet, die sie verspricht, wird ein Benutzer sie auch verwenden, so der Grundgedanke. Eine App benutzen, heißt Vertrauen in sie haben. Daraus lässt sich ein Wert gewinnen.

Eine sehr grobe Lösung bieten Girardello und Michahelles [GM10b] [GM10a]. Sie entwickelten eine App namens *AppAware*, welche von Benutzern auf ihrem Smartphone installieren wird. Diese App überwacht, wenn Apps installiert, deinstalliert und upgedatet werden. Entsprechend der letzten Aktion eines Benutzers bei einer App bekommt diese einen Wert zugewiesen (*deinstalliert* 0,0; *installiert* 0,9; *Update* 1,0). Aus den Rückmeldungen aller Benutzer für eine App wird der Mittelwert berechnet.

Wenn auch simpel, stellt die Überlegung von Girardello und Michahelles einen guten Ansatz dar. Für die Metrik würde es sich also anbieten zu messen, wie oft eine App wieder deinstalliert oder geupdatet wurde und daraus auf die QUalität und Funktionalität der App zu schließen. Jedoch weisen Girardello und Michahelles selber auf die Schwäche hin, dass ihre Metrik darauf baut, dass schlechte Apps auch wirklich deinstalliert werden. Da die meisten Updates von Apps über den Play Store komplett automatisch oder ohne allzu großes Mitwirken des Benutzers vonstattengehen, ist das Update einer App auch nicht zwangsweise ein gewichtiger Faktor für das Vertrauen in die App. Für die Metrik ist dieser Ansatz weniger zu gebrauchen.

Eine Verfeinerung sollte also in Betracht gezogen werden, in welcher analysiert wird, ob eine App nach der Installation auch wirklich genutzt wird und wenn ja wie oft und wie viel. Mit dieser Fragestellung beschäftigt sich wie bereits erwähnt *TruBeRepec* von Yan et al. [YLNY13]. Sie führten eine großangelegte Benutzerstudie durch, um herauszufinden, wie das Nutzerverhalten mit dem Vertrauen in eine App zusammen hängt und stellten mit Hilfe dessen eine komplexe Metrik aufgestellt.

Für das Nutzerverhalten einer App fließen etwa die Anzahl der Nutzungen, die Zeit der Nutzung und die Nutzungshäufigkeit der jeweiligen App, aber auch aller anderen Apps als Vergleichspunkt ein. Informationen wie genutzte und vorhandenen Feature einer App spielen ebenfalls eine Rolle. Dieses Herangehen benötigt eine Vielzahl an Informationen über die App und das Verhalten des Nutzers.

Für diese Arbeit verwenden wir einen simpleren Ansatz als Yan et al. [YLNY13]. Dazu wird in einer abgewandelten Funktion ihres Ansatzes für Nutzungsverhalten (siehe Gleichung (5.21) auf Seite 56) die relative Nutzung einer App durch einen Benutzer berechnet. Unterschiedliche Benutzer verwenden ihr Smartphone unterschiedlich oft und lange, deswegen stellt ein absoluter Wert der Nutzung einer App kein gutes Maß dar. Es sollte also betrachtet werden, wie oft ein Benutzer eine App im Vergleich zu anderen Apps verwendet. Dadurch ist es egal, ob ein Benutzer sein Smartphone ständig in der Hand hat oder nur wenige Male am Tag benutzt.

Ein Problem stellt hier dar, dass manche Apps nur eine bestimmte Zeit am Tag verwendet werden, wie etwa eine Wecker-App, die meistens nur einmal am Abend gestellt und sonst nicht verwendet

wird, egal wie oft man das Smartphone verwendet. Dadurch ergibt sich für einen Viel- und einen Wenig-Nutzer der gleiche absolute aber ein unterschiedlicher relativer Wert.

Bei der Aufstellung dieser Metrik machen wir die Annahme, dass sich das Verhältnis der Smartphone-Nutzer-Typen bei ähnlichen Apps ähnlich verhält. Diese muss jedoch nicht den Tatsachen entsprechen und sollte mit einer Studie unterstützt oder widerlegt werden. Sollte sie widerlegt werden, wäre eine Lösung die Benutzer in verschieden Kategorien einzuteilen und ihren relativen Nutzungswert mit einem Parameter zu versehen, sollte zwischen den einzelnen Benutzerkategorien eine eindeutige Differenz bestehen. Dadurch läuft man jedoch wieder Gefahr den Wert zu verfälschen. Aus diesem Grund geht diese Arbeit dem naiven Ansatz nach.

Um die relative Nutzung in einen aussagekräftigen Wert umzuwandeln wird zunächst der Mittelwert der relativen Nutzung aller Benutzer, welche die App installiert haben berechnet. An dieser Stelle kann die Deinstallation einer App mit einbezogen werden. Dazu in Unterkapitel 4.4.1 mehr.

Die relativen Nutzung einer App mit der irgendwelcher Apps zu vergleichen würde kein aussagekräftiges Ergebnis liefern. Vergleicht man eine Wetter-App mit einer Spiele-App, wird letztere im Normalfall eine höhere Nutzung aufweisen. Deswegen sollte eine App mit jenen Apps aus der selben Kategorie verglichen werden. Dies führt zu einem Ranking unter den Apps, wobei der relativ am meisten genutzten App am meisten vertraut wird und der am wenigsten genutzten App am wenigsten.

In jeder Kategorie gibt es also eine App geben, der auf Grund ihrer Nutzung zu fast 100% in ihre Funktionalität vertraut wird. Dieser Wert wird jedoch durch die Ratings und Problemmeldungen verrechnet. Dass eine App also zu einem Zeitpunkt der Berechnung scheinbar volles Vertrauen erhält, nur weil sie besser als der Rest abschneidet wird wieder relativiert durch andere Faktoren. Würden diese jedoch nicht mit eingerechnet werden, müsste man bei einem Ranking sehr vorsichtig sein. Ist eine App vertrauenswürdiger als der Rest, bedeutet dies nicht, dass sie vertrauenswürdig ist.

Im nächsten Unterkapitel wird die Deinstallation und die Nichtnutzung einer App in deren Vertrauenswert bezüglich der Nutzung einfließen können und ob zwischen diesen eine Unterscheidung gemacht werden sollte.

4.4.1. Deinstallation und Nichtnutzung der App

Insgesamt wurden zwei Gründe ausgemacht, dass das Vertrauen in eine App auf Grund ihrer Nutzung gegen Null strebt oder diesen Wert auch erreicht. Dies wäre eine Deinstallation der App durch den Benutzer oder wenn dieser die App lange Zeit nicht mehr verwendet. Die Frage, die sich stellt ist, ob beide Fälle gleich stark gewichtet werden sollen, oder ob sogar innerhalb der Fälle Unterscheidungen gemacht werden sollte.

Für eine stärkere Gewichtung der Deinstallation spricht, dass dies ein aktives Vorgehen ist, mit welchem man sich der App entledigt. Es muss Gründe geben, dass die App wieder deinstalliert wurde. Jedoch sind diese Gründe nicht unbedingt klar. Dass es sich auf fehlende Funktionalität oder andere Gründe zurückführen lässt, muss nicht der Fall sein. Dass eine App direkt deinstalliert wird und der Benutzer sie nicht einfach vergisst und nicht mehr nutzt, ist in diesem Fall eine Spekulation.

Diesem Problem ließe sich mit einer Nachfrage beikommen. Wenn der Benutzer eine App deinstalliert kann eine Abfrage stattfinden, warum er dies tut. In Frage kämen Auswahlmöglichkeiten wie:

- App bietet nicht die versprochene/erwartete Funktionalität.
- App ist schadhaft.
- App missbraucht private Daten.
- Der Speicherplatz wurde benötigt.
- Kein bestimmter Grund.
- Keine Angabe.

Die ersten drei Möglichkeiten könnten auch detaillierter gehalten werden, vergleichbar mit den Problemmeldungen aus Unterkapitel 4.3.3 und 4.6.2. Wichtig ist auch, dass dem Benutzer die Möglichkeit gegeben werden sollte, keine Angabe machen zu können, wenn er nicht will - wobei die Unterscheidung, zwischen keine Angabe machen wollen und keinen bestimmten Grund haben, hervorzuheben ist (letzteres ist immer noch eine Aussage für manche).

Wenn ein Benutzer längere Zeit eine App nicht verwendet, könnte ihm auch eine solche Nachfrage gestellt werden. Die Punkte der schadhaften App und missbrauchten Daten würden dann wahrscheinlich herausfallen, da eine solche App mit hoher Wahrscheinlichkeit sofort deinstalliert wird, wenn der Benutzer sie entdeckt.

Dieses Herangehen hat jedoch seine Probleme. Die für die Metrik vorgeschlagenen Problemmeldungen sind ein passives Meldesystem. Der Benutzer kann sich entscheiden ein Fehlverhalten der App zu melden, wenn er dies für angemessen hält. Die Abfrage zur Deinstallation oder bei Nichtnutzung einer App sind jedoch aktiv und können dem Benutzer das Gefühl geben, dass sie ihm aufgezwungen werden. Gerade wenn eine App lange nicht mehr genutzt wird, kann eine Nachfrage in diesem Fall als sehr aufdringlich empfunden werden oder gar als Aufforderung die App mehr zu verwenden missverstanden werden kann.

Der Standpunkt dieser Arbeit ist es, dass der Benutzer gewillt sein muss, Probleme mit der App zu melden und er nicht das Gefühl haben soll, dass er dazu gezwungen wird. Folglich wird keine Abfrage bezüglich der Gründe für Deinstallation oder Nichtnutzung einer App stattfinden. Da sich dann nur über die Gründe spekulieren lässt, werden Deinstallation und Nichtnutzung gleich stark gewichtet, wenn sie in den Vertrauenswert einfließen.

Nachdem die Nutzung einer App und deren Einfluss auf das Vertrauen in die Funktionalität einer App analysiert wurde, werden nun die Erkenntnisse behandelt, die aus den Permissions einer App gezogen werden können.

Metrik von	Skala	Einbezogene Werte
Peng et al.	Ranking mit Prozentangabe	Permissions der App; Wert anderer Apps mit denen verglichen wird
Enck et al.	binäre Entscheidung	Neunteiliger Regelsatz (Zwei davon nicht mehr aktuell)
Sarma et al.	binäre Entscheidung	26 Permissions (als kritisch eingestuft); Anforderung kritischer Permissions durch andere Apps

Tabelle 4.2.: Übersicht über die verschiedenen Ansätze für Metriken für das Risiko durch Permissions

4.5. Permissions

Permissions geben einen Anhaltspunkt, welcher Schaden durch eine App entstehen kann. Es bietet sich also an, zu betrachten, welche Permissions Gefahrenpotenzial beherbergen oder dazu genutzt werden können, um die privaten Daten des Benutzers zu entwenden. Eine Liste dieser Permissions, die aus [SLG+12] übernommen und leicht modifiziert ist, findet sich in Anhang A.1.

Wie Chia et al. [CYA12] jedoch feststellten steigt die Anzahl der verlangten Permissions mit der Funktionalität einer App. Somit ist eine reine Betrachtung der verlangten Permissions kein gutes Maß. Welche Aussage für die Metrik jedoch gemacht wird, ist dass eine App, die eine kritische Permission besitzt, nicht in Vertrauenskategorie *K1* fallen kann.

Es wurden bereits einige Versuche unternommen, um das Risiko durch eine App auf Grund ihrer Permissions zu bewerten. Enck et al. [EOM09] stellten zum Beispiel neun Regeln auf, welche Permissions oder Kombinationen dieser enthielten, die oft von schadhaften Apps verwendet wurden.

Der aktuell am meisten genutzte Ansatz ist es, Permissions zu identifizieren, welche auf alle Apps oder auf die Apps in einer Kategorie bezogen selten sind. Peng et al. [PGS+12] verwenden Bayes Modelle, um Apps zu identifizieren, deren Zusammensetzung der verwendeten Permissions selten ist. Sarma et al. [SLG+12] identifizieren seltene kritische Permissions und Kombinationen seltener Permissions, um eine App als verdächtig zu markieren.

Für die Metrik in dieser Arbeit gehen wir einen ähnlichen, wenn auch simpleren Ansatz. Es bietet sich an zu betrachten, wie wahrscheinlich die einzelnen kritischen Permissions für eine App in ihrer jeweiligen Kategorie sind und daraus einen Mittelwert zu berechnen.

Ein weiterer Punkt, der ein Gespür dafür gibt, ob eine App mehr Permissions verlangt, als sie braucht ist die Anzahl der durchschnittlichen kritischen Permissions in der Kategorie. Dazu bietet sich die Metrik für das Vertrauen in die Anzahl der Permissions von Kuehnhausen und Frost [KF13][KF13].

Aus dieser Kombination der Wahrscheinlichkeit der kritischen Permissions und der Anzahl der kritischen Permissions, die eine App verlangt, lässt sich ein guter Wert dafür gewinnen, wie es um ihr Gefahrenpotenzial steht und ob sie mehr Zugriff auf private Daten verlangt, als in ihrer Kategorie gewöhnlich ist.

Um das Gefahrenpotenzial zu kategorisieren, wird betrachtet, wie selten die kritischste Permission einer App in der Kategorie ist. Dafür müssen Werte für die Zuordnung festgelegt werden. Der Ansatz von Sarma et al. schlägt dafür vor, von der Installation einer App abzuraten, wenn sie eine Permissions besitzt, die weniger als 2% der restlichen Apps die Permissions verlangen. Dies wird jedoch über alle Apps im Appshop gesehen. Das Ziel von Sarma et al. ist es auch möglichst wenige Meldungen zu erzielen. Außerdem wollen Sarma et al. wie beschrieben nur eine binäre Entscheidung, ob eine App bösartig oder nicht ist.

Durch das Gefahrenpotenzial kann eine Aufteilung auf die drei Kategorien *K2*, *K3* und *K4* stattfinden. Es stehen primär Zahlen für die Verteilung von Permissions über alle Apps und nicht jene in einer speziellen Kategorie zur Verfügung. Darüber hinaus kann sich die Wahrscheinlichkeitsverteilung über die Permissions verändern. Für diese Arbeit wird die Verteilung: Wahrscheinlichkeit der Permissions $\leq 5\% \mapsto$ *K2*; $5\% <$ Wahrscheinlichkeit der Permissions $\leq 50\% \mapsto$ *K3*; sonst $\mapsto$ *K4*. Die Aussagekraft dieser Verteilung sollte mit einer Studie ergründet werden. Werden keine kritischen Permissions verlangt, kommt die App in Kategorie *K1*.

Die Einteilung bezüglich des Umgangs mit privaten Daten findet in anderer Weise statt. Auch wenn die Permissions, die benötigt werden, um auf Daten zuzugreifen, in das Vertrauen einfließen, wird die Kategorie durch die Rückmeldungen bestimmt, die durch die Datenflussanalyse ermöglicht werden.

Als nächstes wird, die angesprochene Datenflussanalyse besprochen, aber auch ein Blick auf die Kontrollflussanalyse geworfen.

4.6. Kontroll- und Datenflussanalyse

4.6.1. Kontrollfluss

Der Kontrollfluss kann genutzt werden, um die bereits in Kapitel 3 geschilderten *Privilege Escalation Attacks* aufzudecken, wobei der primäre Ansatzpunkt die Entdeckung von Apps ist, welche diese Attacken erst ermöglichen. *DroidChecker* von Chan et al. bietet etwa eine statische Analysmöglichkeit, um Apps zu finden, die *Privilege Escalation Attacks* erst möglich machen, indem sie den Zugriff anderer Apps zulassen, [CHY12].

Eine Möglichkeit wäre alle Apps mit *DroidChecker* zu überprüfen und sollte sich bei einer App Angriffsmöglichkeiten ergeben, müsste das Vertrauen in das Gefahrenpotenzial dieser App sehr niedrig eingeordnet werden. Der Anteil an Apps mit solchen Lücken fällt jedoch vergleichsweise gering aus (in ihrem Experiment weniger als 0,2%, was bei der großen Anzahl an Apps im *Play Store* immer noch sehr viele sein könnten).

In diesem Fall muss eine Abwägung gemacht werden, ob und wann eine App überprüft wird, ob sie Angriffspunkte liefert. Da diese Analyse statisch stattfinden kann, könnte sie etwa von einem App-Marktplatz automatisch ausgeführt werden, wenn eine App neu eingestellt wird. Diese Möglichkeit hängt sehr von den Umständen ab, weswegen sie hier erwähnt sei, in der Metrik jedoch nicht detailierter darauf eingegangen wird.

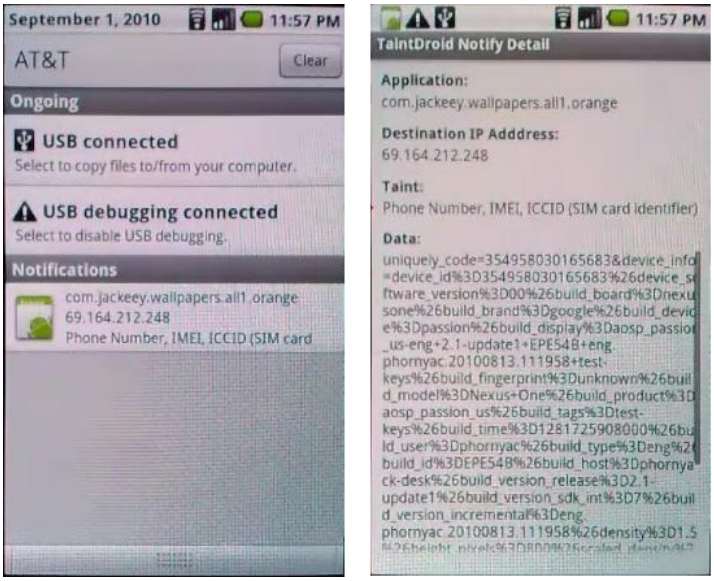

Abbildung 4.6.: Anzeige der weitergeleiteten privaten Daten die von TaintDroid entdeckt wurden.

4.6.2. Datenfluss

Das große Problem in Bezug auf den Missbrauch privater Daten ist, dass der Benutzer diesen oft nicht bemerkt. Während die Fehlverhalten einer App, die bezüglich Funktionalität und Gefahrenpotenzial gemeldet werden können, vom Benutzer relativ gut wahrnehmbar sind, bedarf es bei den privaten Daten der Hilfe eines Werkzeugs. Geeignet dafür ist *TaintDroid* von Enck et al., [EGC+10] [EGC+14].

*TaintDroid*informiert den Benutzer, wenn eine App auf private Daten zugreift und von diese gesendet werden. Ein Beispiel dafür liefern die Screenshots aus dem TaintDroid Demo Video[4] in Abbildung 4.6.

Aus den möglichen Erkenntnissen, die *TaintDroid* dem Benutzer liefern kann, werden für diese Metrik folgende verwendet:

- Standort an einen Werbeserver
- Telephon Information an Content Server (IMSI, ICC-ID)
- Geräte ID an Content Server (IMEI)
- Kalender
- Kontakte
- SMS-Speicher

[4]Quelle: `http://appanalysis.org/demo/index.html`

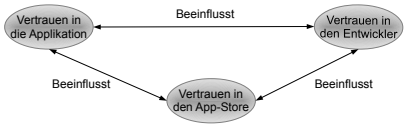

Abbildung 4.7.: Das Vertrauen in eine App, einen Entwickler und einen App-Marktplatz beeinflussen sich gegenseitig.

Die ersten drei Optionen sind Informationen, welche während den Tests von Enck et al. häufig festgestellte wurden und Informationen über den Benutzer liefern. Die anderen drei Optionen sollten selbsterklärend sein, warum ihre Weitergabe das Vertrauen in eine App senken sollte. Die Meldung dieses Missbrauchs privater Daten findet wie in Unterkapitel 4.3.3 statt.

Ein Punkt, den Enck et al. ansprechen ist, dass manche Apps in ihrer EULA deklarieren, dass Daten wie die IMEI von der App verwendet werden dürfen. Für diese Metrik wird die Ansicht verfolgt, dass nur weil für eine App zugegeben wird, dass sie private Daten weitergibt, dadurch nicht das Vertrauen in sie höher ist. Außerdem gilt die Frage: Wie viele Nutzer lesen und verstehen die EULA wirklich?

Nachdem *TaintDroid* als Werkzeug zur Erkennung des Missbrauchs privater Daten vorgestellt wurde, werden im nächsten Unterkapitel die Elemente des Entwicklers und des App-Marktplatzes behandelt und wie sie Einfluss auf das Vertrauen in die App haben können.

4.7. Vertrauen in andere Elemente

Das Vertrauen in eine App kann auch von dem Vertrauen in ihren Entwickler und in den App-Marktplatz beeinflusst, in welchem die App zum Download angeboten wird. Gleiches kann auch in die andere Richtung und zwischen Marktplatz und Entwickler gelten (siehe Abbildung 4.7). Es lohnt sich also zu betrachten, welche Faktoren das Vertrauen in einen App-Entwicklern (Unterkapitel 4.7.1) und einen App-Marktplatz (Unterkapitel 4.7.2) beeinflussen könnten.

4.7.1. Der Entwickler

Es lassen sich eine Handvoll Anhaltspunkte für das Vertrauen in einen Entwickler finden. Der beste ist natürlich, seine Apps zu betrachten. Wie hoch ist das Vertrauen in diese Apps, ist bekannte Malware dabei oder wählt er für seine Apps bewusst täuschende Namen (siehe [CYA12]). Es sollte jedoch davon ausgegangen werden, dass ein guter App-Marktplatz Entwickler löscht, welche Malware in diesen einstellen.

Überlegenswert wären auch zu betrachten wie viele Apps der Entwickler in den App-Marktplatz gestellt hat oder wie viele Downloads seine Apps in der Gesamtheit haben. Jedoch ergibt sich auch daraus kein Ergebnis. Ein Entwickler kann viele Apps, die wenig vertrauenswürdig sind, veröffentlichen und generiert dadurch viele Downloads während ein anderer Entwickler nur eine vertrauenswürdige App veröffentlicht, diese aber kaum heruntergeladen wird. Zu diesen Ansätzen fehlen Studien und Analysen, die eine weitere Überlegung für eine Vertrauensmetrik ermöglichen.

Bewertungen der Entwickler, zum Beispiel durch den App-Marktplatz oder eine andere Quelle stellen auch eine Möglichkeit dar. Der Google Play Store[5] verleiht seine Entwickler beziehungsweise deren Apps zwei Abzeichen, die *Editors Choice* und den *Top Developer*, welche beide von Mitarbeitern des Play Stores gewählt werden. Die *Editors Choice* ist eine Sammlung von Apps, in die Apps mit hoher Qualität, guter graphischer Oberfläche, Langzeitbeliebtheit und innovativer Benutzung von Android Features aufgenommen werden. Den *Top Developer* bekommen Entwickler verliehen, von denen alle oder die meisten Apps eine gute Qualität aufweisen und über lange Zeit hohe Anerkennung genießen. Der Anteil dieser Entwickler sollte jedoch sehr gering ausfallen und so ist dies keine nützliche Aussage für die Allgemeinheit.

Es fällt also schwierig das Vertrauen in einen Entwickler zu berechnen, weswegen es nicht in diese Metrik einfließt.

4.7.2. Der App-Marktplatz

Neben den großen, bekannten App-Marktplätze gibt es auch zahlreiche kleinere App-Marktplätze von Drittanbietern. Laut einer Symantec Analyse [Usc13] gibt es App-Marktplätze von Drittanbietern, bei welchen jede zehnte neu hinzugefügte App Malware enthält. Teilweise gibt es App-Marktplätze , die beinahe ausschließlich Malware enthalten. Auch wenn dies eher Einzelfälle sind und die App-Marktplätze nach kurzer Zeit wieder offline gehen, gibt es doch viele App-Marktplätze, die eine hohe hohe Anzahl von Ad- und Malware aufweisen. Die Eigenschaft eines Marktplatzes, für seinen hohen Malware-Anteil bekannt zu sein, wirkt sich negativ auf das Vertrauen der Apps aus, die dort dargeboten werden.

Ein weiteres Maß für das Vertrauen in einen Marktplatz könnten seine Sicherheitsvorkehrungen sein. Wie weit und gut überprüft der Shop eingestellte Apps auf Malware, besitzt er einen Kill-Switch, um nachträglich als schadhaft festgestellte Apps von den Smartphones zu entfernen, nachdem sie einmal installiert wurden und die Bedingungen dafür, dass man Apps in den Store stellen kann.

Dini et al. [DMM+13] unterscheiden in ihrer Metrik für das Vertrauen in eine App zwischen „offiziellen“ und „inoffiziellen“ Marktplätze, sowie der Möglichkeit, dass kein Shop vorhanden war, man also die .apk-Datei aus einer anderen Quelle hat. Dieses Maß ist sehr grob und verallgemeinernd.

Insgesamt gibt es einige Faktoren, die für die Vertrauenswürdigkeit eines App- Marktplatzes zurate gezogen werden können. Jedoch gewähren gerade inoffizielle Shops eher keine Auskunft über ihre Sicherheitsmechanismen. Ein Ansatz wäre ein Reputationssystem für App- Marktplätze aufzubauen, in welches auch die Erkenntnisse von Analysen wie jener von Symatec einfließen. Ein solches System

[5] http://goo.gl/bdZU4e

liegt jedoch jenseits des Umfangs dieser Arbeit. Es kann keine befriedigender Wert für das Vertrauen in den Marktplatz errechnet werden, wodurch auch von dieser Seite nichts in die Metrik einfließen kann.

5. TriMetrik

In Kapitel 4 wurden Ansätze zu Vertauens-Metriken und Systemen für Android Apps untersucht. Daraus wurden Erkenntnisse für eine eigene Metrik getroffen. In diesem Kapitel wird nun die *TriMetrik* präsentiert.

Dazu wird in Unterkapitel 5.1 ein Überblick über die Zusammenführung des Vertrauens in die einzelnen Bereiche gegeben, deren Berechnung des Vertrauens in Unterkapitel 5.2 behandelt wird. Im Folgenden wird betrachtet, wie die Ratings (Unterkapitel 5.3), die Nutzung einer App (Unterkapitel 5.4), die Permissions, die eine App verlangt (Unterkapitel 5.5) und die Problemmeldungen für eine App (Unterkapitel 5.6) konkret in die Metrik einfließen.

5.1. Ergebnis

Final liefert die Metrik zwei Werte für eine überprüfte App app_i, den prozentualen Vertrauenswert (Gleichung 5.2) in die App und die Vertrauenskategorie (Gleichung 5.3), in welche die App fällt.

Der prozentuale Vertrauenswert ergibt sich als Summe der gewichteten Vertrauenswerte für die Funktionalität der App (F_{app_i}), das Gefahrenpotenzial der App (G_{app_i}) und den Umgang der App mit privaten Daten (D_{app_i}).

Die Gewichtungsfaktoren α, β und γ berechnen sich aus der gewählten Gewichtung für die drei Bereiche wie in Gleichung 5.1 gezeigt. Dabei liefert die Funktion $Gewichtung(Bereich)$ Werte abhängig von der jeweiligen gewählten Wichtigkeit der Bereiche durch den Benutzer (*W1* = 2; *W2* = 1; *W3* = 0,5; *W4* = 0). Für den Wert von α ergibt sich damit:

$$\alpha = \frac{Gewichtung(F)}{Gewichtung(F) + Gewichtung(G) + Gewichtung(D)} \tag{5.1}$$

Analog werden β und γ, mit jeweils Gewichtung(G) und Gewichtung(D) als Nenner, berechnet.

Solange nicht alle drei Bereiche mit $W4$ gewichtet wurden, gilt dabei $\alpha + \beta + \gamma = 1$. Wurden alle drei Bereiche vom Benuzter als *Unwichtig* festgelegt, liefert die Metrik eine Fehlermeldung, da keine Berechnung möglich ist. Für diese Werte wird die Berechnung der Gewichtungsfaktoren durch Gleichung 5.1 nicht aufgerufen, weil sonst ein Division-Durch-Null-Fehler auftreten würde. Dem Benutzer muss dann vom Vertrauenssystem kommuniziert werden, dass eine Berechnung keinen Sinn macht, wenn alle Faktoren keine Rolle spielen.

Für das Vertrauen in eine App erhält man also die Gleichung 5.2. Auf das Zustandekommen des Vertrauens der einzelnen Bereiche wird in den Gleichungen 5.4 für $Vertrauen(F_{app_i})$, Gleichung 5.5 für $Vertrauen(G_{app_i})$ und in Gleichung 5.6 für $Vertrauen(D_{app_i})$ eingegangen.

(5.2) $$Vertrauen(app_i) = \alpha \cdot Vertrauen(F_{app_i}) + \beta \cdot Vertrauen(G_{app_i}) + \gamma \cdot Vertrauen(D_{app_i})$$

Das Ergebnis für das Vertrauen in eine App liegt im Intervall [0, 1] und repräsentiert das prozentuale Vertrauen in die App. Durch die Aufsummierung der drei Bereiche kann ein App noch einen vergleichsweise hohen Vertrauenswert erhalten, selbst wenn sie in einem der Bereiche sehr schlecht abschneidet. Dies ist aber wie beschrieben gewünscht, da die Metrik einen Gesamtüberblick (gemäß der Wichtigkeit für den Benutzer) über alle Qualitäten der App liefern soll.

Damit ein schlechtes Abschneiden in einer der Kategorien nicht unbeachtet bleibt, wurden die Kategorien eingeführt. Hier wurde bestimmt, dass die Vertrauenskategorie der App der schlechtesten Kategorie eines der drei Bereiche entspricht. Auf die Kategorien sei die Ordnungsrelation „$\leq$“ definiert, sodass die Kategorien sich nach dem Maß des Vertrauens ordnen lassen, welches sie ausdrücken sollen (*K5 < K4 < K3 < K2 < K1*). Die Funktion $Kat(F_{app_i}$ liefert die Kategorie, in welche die App im Bereich Funktionalität fällt, außer der Bereich wurde mit *W4* gewichtet, in diesem fall wird automatisch *K1* zurückgegeben, da diesere Bereich für das Vertrauen in die App nicht in Beachtung gezogen wird und *K1* in diesem Fall das neutrale Element darstellt.

Unter diesen Bedingungen gilt für die Gesamtkategorie einer App:

(5.3) $$Kat(app_i) = min(Kat(F_{app_i}), Kat(G_{app_i}), Kat(D_{app_i}))$$

Damit ist nun gegeben, wie sich das Gesamtvertrauen und die Gesamtkategorie einer App berechnet. Im nächsten Unterkapitel wird nun betrachtet, aus welchen Werten sich das Vertrauen in den drei Bereichen zusammensetzt.

5.2. Bereiche

Um das Vertrauen in die einzelnen drei Bereiche zu erhalten, werden dort wiederum die unterschiedlichen Faktoren betrachtet, welche sie beeinflussen.

Das Vertrauen in die Funktionalität einer App berechnet sich aus den Ratings der App (Unterkapitel 5.3, Gleichung 5.7), der Nutzung der App (Unterkapitel 5.4, Gleichung 5.23) und den Problemmeldungen bezüglich der Funktionalität der App (Unterkapitel 5.6, Gleichung 5.36).

(5.4) $$Vertrauen(F_{app_i}) = Ratings(app_i) \cdot Nutzung(app_i) \cdot Problemmeldungen_F(app_i)$$

Für das Gefahrenpotenzial und das damit einhergehende Vertrauen in eine App spielen das Riskrating (Unterkapitel 5.5, Gleichung 5.24) und Problemmeldungen bezüglich des Schadpotenzial (Unterkapitel 5.6, Gleichung 5.37) eine Rolle.

(5.5) $Vertrauen(G_{app_i}) = Riskrating_G(app_i) \cdot Problemmeldungen_G(app_i)$

Der Umgang mit privaten Daten schließlich wird ebenfalls von Riskrating (Unterkapitel 5.5, Gleichung 5.24) und dem Problemmeldungen bezüglich privater Daten (Unterkapitel 5.6, Gleichung 5.38) beeinflusst.

(5.6) $Vertrauen(D_{app_i}) = Riskrating_D(app_i) \cdot Problemmeldungen_D(app_i)$

In welche Vertrauenskategorie einer der Bereiche eingeordnet wird, entscheidet sich auch durch diese Faktoren. Es gilt, wie bei der Gesamtkategorie, dass die App in den einzelnen Bereichen jeweils in die niedrigste, auf Grund eines Faktors erreichte, Vertrauenskategorie fällt. Wie sich die Faktoren auf die Vertrauenskategorien auswirken, wird ebenfalls in den einzelnen Unterkapiteln erwähnt.

5.3. Ratings

Für das Rating gilt, dass bei weniger als 10 Ratings der Wert von 0 zurückgegeben wird, da sich mit zu wenigen Ratings kein verlässlicher Vertrauenswert berechnen lässt. Bei 10 Ratings oder mehr lässt sich das Vertrauen aufgrund der Ratings wie folgt berechnen:

(5.7) $Ratings(app_i) = Rating_{Skaliert}(app_i) \cdot AnzahlRatings(app_i) \cdot VerteilungRatings(app_i)$

Für die *TriMetrik* wird die Metrik für das Vertrauen auf Grund der Ratings eins zu eins von Kuehnhausen und Frost [KF13] übernommen. Auch wenn der Ansatz von Kuenhausen und Frost [KF13] sich auf ein Rating mit 5 Sternen bezieht, lässt es sich theoretisch auch für andere Skalen verwenden. Die gängigste Bewertung ist jedoch das 5 Sterne System.

Steht das durchschnittliche Rating für die App nicht direkt zur Verfügung, muss dieses noch von der Metrik berechnet werden. Sei R_1 die Anzahl der Ratings der App mit einem Stern, R_2 die Anzahl der Ratings mit zwei Sternen und so weiter bis R_{max} dem bestmöglichen Rating. Außerdem ist n die Anzahl der insgesamt abgegebenen Ratings. Das durchschnittliche Rating $\bar{R}_{app_i}$ einer App ergibt sich dann als:

(5.8) $\bar{R}_{app_i} = \frac{\sum_{x=1}^{max} R_x}{n}$

Zunächst müssen die Ratings auf unsere Skala von 0% bis 100% also auf das Intervall [0, 1] gebracht werden. Sei dazu RM die Anzahl der möglichen Werte, die das Rating annehmen kann, dann gilt für das skalierte Rating:

$$Rating_{Skaliert}(app_i) = \frac{\bar{R}_{app_i} - 1}{RM - 1} \tag{5.9}$$

Die Verringerung der beiden Werte um eins stammt daher, dass die meisten Ratings mit einem Stern und nicht mit null Sternen beginnen. Startet bei die ursprüngliche Skala der Ratings bei null, werden beide Werte unmodifiziert gelassen.

Dieser skalierte, aber ansonsten unmodifizierte Wert drückt nun das Vertrauen in die Funktionalität auf Grund der Ratings aus, wenn man diesen zur Gänze vertrauen würde. Wie von Kuehnhausen und Frost [KF13] vorgeschlagen werden die Studentsche t-Verteilung und ein Algorithmus als Maß zur Verteilung der Ratings angewendet, um herauszufinden, welche Zuversicht man in die Werte des Ratings haben kann. Kann man den Ratings nicht vertrauen, vertraut man auch der App nicht. Dabei wird der Ansatz verfolgt lieber kein Vertrauen als falsches Vertrauen zu haben.

Für das Vertrauen in die Anzahl der Ratings ergibt sich die folgende Gleichung:

$$AnzahlRatings(app_i) = 1 - \frac{ST(n)}{\sqrt{n-1}} \tag{5.10}$$

Dabei ist ST(n) der Wert der inversen Distributionsfunktion der Studentschen t-Distribution mit $n-1$ Freiheitsgraden und einem beidseitigen Konfidenzintervall von 95%.

$$ST(n) = CDF^{-1}_{T(n-1)}(0,975) \tag{5.11}$$

Das Ziel der Metrik ist es herauszufinden, ob die Ratings gegen zwei unterschiedliche Werte tendieren, womit die Zuversicht in die Aussagekraft der Ratings schwinden würde. Dazu werden die Ratings in die Anzahl der hohen Ratings R_{high} und niedrigen Ratings R_{low} unterteilt. Steigt die Anzahl der Ratings, muss hier eventuell feiner differenziert werden. Im Folgenden wird lediglich die Variante mit fünf Sternen betrachtet.

$$R_{low} = R_1 + R_2 + R_3 \qquad\qquad R_{high} = R_3 + R_4 + R_5 \tag{5.12}$$

Gibt es die Sonderfälle $R_{low} = 0$ oder $R_{high} = 0$, also nur hohe Bewertungen mit 4 und 5 Sternen oder nur niedrige Bewertungen mit 1 und 2 Sternen, ergibt sich für die *gewichtete Mittelwertdifferenz*:

$$wmd(R_{app_i}) = \frac{R_4 \cdot |R_{app_i} - 4|}{n} + \frac{R_5 \cdot |R_{app_i} - 5|}{n} \tag{5.13}$$

Falls $R_{low} = 0$ gilt, ansonsten wenn $R_{high} = 0$ gilt, dann erhält man:

$$wmd(R_{app_i}) = \frac{R_1 \cdot |R_{app_i} - 1|}{n} + \frac{R_2 \cdot |R_{app_i} - 2|}{n} \tag{5.14}$$

Tritt keiner der Sonderfälle ein, werden das durchschnittliche Rating der hohen und niedrigen Ratings berechnet (Gleichung 5.15+5.16) und die Anzahl der Ratings mit ihnen verrechnet (Gleichung 5.18+5.17).

$$\bar{R}_{high} = \frac{R_3 + R_4 + R_5}{R_{high}} \tag{5.15}$$

$$\bar{R}_{low} = \frac{R_1 + R_2 + R_3}{R_{low}} \tag{5.16}$$

$$wlow = \frac{R_{low}}{R_{high} + R_{low}} \tag{5.17}$$

$$whigh = \frac{R_{high}}{R_{high} + R_{low}} \tag{5.18}$$

Schließlich wird die *gewichtete Mittelwertdifferenz* in Anbetracht des gesamten durchschnittlichen Ratings berechnet:

$$wmd(R) = w_{low} \cdot \left|\bar{R}_{app_i} - \bar{R}_{low}\right| + w_{high} \cdot \left|\bar{R}_{app_i} - \bar{R}_{high}\right| \tag{5.19}$$

Dadurch ergibt sich nach Kuehnhausen und Frost [KF13] ein Vertrauen in die Ratings auf Grund der Verteilung der Ratings:

$$VerteilungRatings(app_i) = 1 - \frac{wmd(app_1)}{2} \tag{5.20}$$

Mit Hilfe der Ratings lässt sich die Vertrauens-Kategorie der App bezüglich der Funktionalität bestimmen. In ihrem Testsatz erhielten Kuehnhausen und Frost [KF13] eine relativ gleichmäßige Verteilung in das Vertrauen der Apps, mit einer Spitze von 25% bei nicht vorhandenem Vertrauen, was auf zu wenige Ratings zurückzuführen ist.

Für die Metrik verwenden wir eine gleichmäßige Aufteilung auf die drei Kategorien, wodurch eine App mit 33% Vertrauen oder weniger in Kategorie $K4$, mit 34% bis 66% in Kategorie $K3$ landet und von 67% bis 99% in Kategorie $K2$ landet. Keine App erreichte die 100%, aber der Vollständigkeit halber sei definiert, dass eine solche App in Kategorie $K1$ landen würde.

5.4. Nutzung

Bei der Berechnung des Vertrauenswertes auf Grund der Nutzung einer App wurde festgelegt, dass diese App immer in Relation zu anderen Apps betrachtet werden sollte. Dazu müssen zwei Submengen von Apps betrachtet werden. Zum einen die bereits eingeführte Menge aller Apps in der selben App $A_{Kategorie}$, zum anderen die Menge der vom jeweiligen $Benutzer_j$ installierten Apps, $A_j \subseteq A$.

Zunächst wird berechnet, wie oft ein Benutzer eine App im Vergleich zu anderen Apps nutzt. Diese relative Nutzung gibt einen besseren Wert, als ein absoluter Wert, da er für Personen mit unterschiedlichen Nutzungszeiten ihres Smartphones besser widerspiegelt. Für diese individuelle Nutzung eines $Benutzer_j$ betrachten wir über einen Zeitabschnitt, wie oft und wie lange dieser die App genutzt hat.

Die Anzahl der Nutzungen der app_i durch den $Benutzer_j$ im Zeitabschnitt t sei dabei $AN_{(j,app_i)}(t)$ und die Zeit in welcher die App genutzt wurde im Zeitabschnitt t: $ZN_{app_i,j}(t)$. Auf die Gesamtheit der von $Benutzer_j$ installierten Apps bezogen, ist $AN_{A_j}(t)$ deren Nutzung im Zeitabschnitt t und analog $ZN_{A_j}(t)$ die Zeit aller genutzten Apps des Benutzers.

Die relative Nutzung der app_i des $Benutzer_j$ im Zeitabschnitt t wird dann durch Gleichung 5.21 beschrieben, welche eine Abwandlung der Gleichung von Yan et al. [YLNY13] darstellt, wobei die Nutzungsfrequenz und die festgestellten Features herausfallen:

$$Nutzung_{Individuel}(app_i, t, j) = \frac{AN_{app_i,j}(t)}{AN_{A_j}(t)} \cdot \frac{ZN_{app_i,j}(t)}{ZN_{A_j}(t)} \tag{5.21}$$

Aus der individuellen Nutzung aller Benutzer, welche die App installiert haben, lässt sich dann ein Mittelwert bilden. An dieser Stelle lässt sich die Information über die Deinstallation einer App einbringen. Alle Benutzer, welche die App installiert hatten und wieder deinstallierten fließen in die Berechnung des Mittelwertes mit dem Wert 0 ein. In diesem Fall ist das Nichtverwenden einer App im Abschnitt t gleichbedeutend mit einer Deinstallation der App zu irgend einem beliebigem Zeitpunkt.

$$Nutzung_{Gesamt}(app_i) = \frac{\sum Nutzung_{Individuel}(app_i, t, Benutzer_j)}{A_{Kategorie(app_i)}} \tag{5.22}$$

Mit der durchschnittlichen Nutzung einer App aller Nutzer lässt sich ein Ranking der Apps erstellen, wie viel eine App im Vergleich zu anderen Apps in ihrer Kategorie genutzt wird:

$$Nutzung(app_i) = \frac{|\{app \in A_{Kategorie} \mid Nutzung_{Gesamt}(app) \leq Nutzung_{Gesamt}(app_i)\}|}{|A_{Kategorie}|} \tag{5.23}$$

Hierbei sei anzumerken, dass diese Metrik gewisse Apps benachteiligt. Bei einem Wecker möchte der Benutzer möglichst schnell die Weckzeit einstellen und kommt so bei einem guten Wecker auf eine

geringe Nutzungsdauer. Bei einer schlechten Wecker App dauert die Einstellung der Weckzeit eine längere Zeit. In diesem Falle würde eine hohe Nutzung nicht für eine gute Funktionalität, sondern für eine schlechte Usability stehen. Eine Möglichkeit diesem Problem entgegen zu wirken wäre es, die Nutzung in Gleichung 5.4 schwächer zu gewichten, je niedriger der relative Nutzen der App in Gleichung 5.22 ausfällt.

Da die Nutzung relativ gesehen wird und besagtes Problem besteht, fließt das durch die Nutzung berechnete Vertrauens lediglich in den Vertrauenswert der Funktionalität ein, aber beeinflusst nicht die Vertrauens-Kategorie.

5.5. Permissions und Riskrating

Um das Vertrauen in eine App auf Grund ihrer Permissions zu berechnen, stellen sich zwei Fragen. Welche Permissions können Schaden anrichten? Ist die App nicht vielleicht auf diese Permissions angewiesen, um ihre Funktionalität umzusetzen?

Im Anhang finden sich die Listen für kritische Permission bezüglich des Schadpotenzials einer App (Tabelle A.1 auf Seite 81) und ihres Zugriffs auf Daten des Benutzers (Tabelle A.2 auf Seite 82). Dadurch definieren sich die Mengen $P_G \subset P$ als Menge der kritischen Permissions, die für ein erhöhtes Schadpotenzial sorgen. Außerdem ergibt sich die Menge $P_D \subset P$ mit kritischen Permissions, die Zugriff auf die Daten liefern, als Untermengen aller Permissions P, die eine App verlangen kann.

Da die Permissions einer App mit den Permissions ähnlicher Apps verglichen werden sollen, wird $A_{kat_l} \subset A$ als die Menge aller Apps in der Kategorie kat_l definiert, wobei A die Menge aller existierenden Apps sei.

Für unser Riskrating einer App wird nun zum einen betrachtet, wie viele andere Apps aus der Kategorie ebenfalls die von der App verlangten kritischen Permissions anfordern, und zum anderen, wie viele kritische Permissions durchschnittlich in der Kategorie verlangt werden. Dadurch ergibt sich für das Riskrating einer App die Gleichung:

$$Riskrating_x(app_i) = AnzKritPerm_x(app_i) \cdot WahrschKritPerm_x(app_i) \tag{5.24}$$

Die Berechnung von $Riskrating_G$ und $Riskrating_D$ ist bis auf den Fakt, dass bei ersterem die Menge P_G und bei letzterem P_D verwendet wird, äuqivalent. Im Folgenden wird die Berechnung nur für $Riskrating_G$ gezeigt, um unnötige Redundanz zu vermeiden. Zu beachten ist, dass sollten keine kritischen Permissions verlangt werden, automatisch der Vertrauenswert von 100% zurück geliefert wird, da keine erkennbare Bedrohung von der App ausgehen kann.

Zunächst wird die Funktion Besitzt(app_i, $perm_m$) festgelegt. Diese ist eine binäre Abfrage, ob die App app_i die Permissions $perm_m$ verlangt:

$$Besitzt(app_i, perm_m) = \begin{cases} 1, \text{falls die App diese Permissions verlangt} \\ 0, \text{sonst} \end{cases} \tag{5.25}$$

Mit Hilfe dieser Funktion berechnet sich die Anzahl der kritischen Permissions einer App app_i mit der Gleichung:

$$P_G(app_i) = \sum_{perm_m \in P_G} Besitzt(app_i, perm_m) \tag{5.26}$$

Die durchschnittliche Summe der kritischen Permissions pro Kategorie erhält man mit der Gleichung:

$$\bar{P}_{G,kat_l} = \frac{\sum_{app_i \in A_{kat_l}} P_G(app_i)}{|A_{kat_l}|} \tag{5.27}$$

Um das Vertrauen in die Anzahl der verwendeten kritischen Permissions zu berechnen, wird die von Kuehnhausen und Frost [KF13] vorgeschlagene Poission Distribution verwendet (Gleichung 5.28). Da die Anzahl der verlangten kritischen Permissions eher gering ausfällt (maximal 11 für P_G und 12 für P_D), ist die heavy-tailed Poisson Distribution für Kuehnhausen und Frost [KF13] am besten geeignet, um als Maß für die Anzahl der kritischen Permissions zu dienen.

$$AnzKritPerm_G(app_i) = \frac{\bar{P}_{G,kat_l}!}{P_G(app_i)!} \cdot \bar{P}_{G,kat_l}^{P_G(app_i) - \bar{P}_{G,kat_l}} \tag{5.28}$$

Neben der Anzahl der kritischen Permission wird nun die Wahrscheinlichkeit dieser Permissions betrachtet, dass sie von Apps in einer Kategorie angefordert werden. Dazu berechnet sich die relative Häufigkeit einer Permissions $perm_m$ in der Kategorie kat_l nach der Gleichung:

$$Haeufig(perm_m, kat_l) = \frac{\sum_{app_i \in kat_l} Besitzt(app_i, perm_m)}{|A_{kat_l}|} \tag{5.29}$$

Die Wahrscheinlichkeit einer App wird dabei aus dem Durchschnitt der Wahrscheinlichkeit der von ihr angeforderten Permissions berechnet. Dabei erhält jede App den „Internet-Bonus“. Die über alle Apps im Marktplatz hinweggesehen am meisten verlangte Permission ist die INTERNET Permission mit etwa 80%. In der Metrik wird die Wahrscheinlichkeit der kritischen Permissions mit dem Faktor $\frac{5}{4}$ verrechnet, sodass eine App, die lediglich die INTERNET Permission verlangt (was heutzutage sehr normal ist), 100% Vertrauen besitzt[1]. Auch wenn diese Modifikation auf den ersten Blick zu gutmütig erscheint, bewirkt sie doch im niedrigen Bereich, in dem sich die Häufigkeit der meisten Permissions bewegt, keinen allzu großen Anstieg des Vertrauens.

[1] In Kategorie *K1 fällt sie dadurch trotzdem nicht.*

Für das Vertrauen in die Wahrscheinlichkeit der kritischen Permissions einer App ergibt sich also die Gleichung:

(5.30)

$$WahrschKritPerm_G(app_i) = min((\frac{Durchschnitt}{P_G(app_i)} \cdot \frac{5}{4}), 1)$$
$$Durchschnitt = \sum_{perm_m \in P_G} (Besitzt(app_i, perm_m) \cdot Haeufig(perm_m, kat_l))$$

Über den Einfluss der kritischen Permissions auf die Vertrauens-Kategorie wurde bereits in Unterkapitel 4.5 ausführlich diskutiert.

Ist die minimale Wahrscheinlichkeit einer kritischen Permissions kleiner 5% landet sie in Kategorie *K2*, ist sie größer 5% aber kleiner 50%, landet sie in Kategorie *K3*, sonst ist sie in Kategorie *K4*. Werden keine kritischen Permissions verlangt, kommt die App in Kategorie *K1*.

Dies gilt wie angesprochen nur für $Kategorie(G_{app_i})$ und nicht für $Kategorie(D_{app_i})$.

5.6. Problemmeldungen

Problemmeldungen dienen dazu, dass der Benutzer gezielte Rückmeldung über die Probleme mit einer App geben kann. Dini et al. [DMM+13] schlagen eher eine Rückmeldung über die App als eine Problemmeldung vor, wodurch mit positiven Rückmeldungen das Vertrauen in eine App gesteigert werden kann. Auch wenn die übernommene Funktion für den Rückmeldewert eine positive Rückmeldung ermöglicht, indem gemeldet wird, dass es keine Probleme gibt, wird doch davon ausgegangen, dass ein Benutzer eher geneigt ist eine negative Rückmeldung zu geben.

Aus diesem Grund ist der initiale Wert bezüglich der Rückmeldungen volles Vertrauen (also einen Wert von 1, beziehungsweise einem Meldewert, von 4 für die Funktionalität und das Gefahrenpotenzial und 5 für die privaten Daten – vgl. Gleichungen 5.36 bis 5.38), da über die Problemmeldungen uns kein Grund geliefert wurde, der App nicht zu vertrauen. Durch die Multiplikation mit dem jeweiligen Riskrating besitzt der Problemmeldungswert dadurch keinerlei Auswirkung auf das Gesamtvertrauen in die einzelnen Bereiche.

Darüber hinaus wird die Rückmeldefunktion der von Dini et al. [DMM+13] verwendete Metrik leicht abgeändert. Sie verrechnen alle möglichen Gründe für ein Fehlverhalten miteinander und geben dadurch eine Gesamtübersicht über die App. Für *TriMetrik* findet eine Aufteilung in die drei Bereiche statt, um Problemmeldungen den einzelnen Bereichen zuordnen zu können. Tabelle 5.1 bietet dazu eine Übersicht der möglichen Rückmeldungen und ihrer Werte.

Diese Liste wird in dieser Zusammensetzung für die *TriMetrik* verwendet, lässt sich aber eventuell noch um weitere Meldungen erweitern. Genauso sind die Werte überlegt gewählt, doch sie können nicht den Anspruch erheben die Meinung der Mehrheit zu repräsentieren. Dini et al. argumentieren, dass sie ihre Meldemöglichkeiten nach Problemen gewählt haben, welche bekannte Anzeichen von

Malware sind. Eine andere Herangehensweise, welche die subjektive Ansicht des Benutzers besser modellieren würde, wäre es zu ergründen, welche Probleme den Benutzer stören und wie schwer sie sein Vertrauen in eine App verringern würden. Dazu müsste eine empirische Benutzerstudie durchgeführt werden, die jenseits des Rahmen dieser Arbeit liegt.

Der Benutzer kann dabei zu einem der drei Bereiche eine Rückmeldung machen, ob und wie stark oder mit welcher Gewissheit er ein Fehlverhalten der App wahrgenommen hat. Daraus ergeben sich die Gleichungen 5.31 für das Vertrauen auf Grund einer Rückmeldung bezüglich der Funktionalität, Gleichung 5.32 für eine Rückmeldung bezüglich des Gefahrenpotenzials und schließlich Gleichung 5.33 für *TaindDroid*-gestützte Problemmeldung bezüglich dem Umgang mit privaten Daten.

$$RF(app_i, k) = max(0, (4 - RF_1 - RF_2 - RF_3)) \tag{5.31}$$

$$RG(app_i, k) = max(0, (4 - RG_1 - RG_2 - RG_3 - RG_4)) \tag{5.32}$$

$$RD(app_i, k) = max(0, (5 - RD_1 - RD_2 - RD_3 - RD_4 - RD_5)) \tag{5.33}$$

Aus den einzelnen Rückmeldungen muss wiederum die Gesamtrückmeldung berechnet werden. Diese Berechnung findet für alle drei Bereiche analog statt, hier sei die Berechnung für das Vertrauen bezüglich der Funktionalität gegeben:

$$RSF(app_i, k) = (1 - \delta(k)) \cdot RSF(app_i, k-1) + \delta(k) \cdot RF(app_i, k) \tag{5.34}$$

$\delta(k)$ dient dabei als Faktor, nach wie vielen Problemmeldungen ein Vertrauenswert mit dem übergebenen Wert dieser Meldungen übereinstimmen soll. Dini et al. [DMM+13] setzen dafür die Anzahl der Benutzer an, welche den Vertrauenswert für die App von *MAETROID* erfragten. Da, je nach Umsetzung, kein solcher Wert zur Verfügung steht, muss ein anderer Weg gefunden werden. Möglichkeiten für die Anzahl der benötgiten Prüblemmeldungen wäre zum Beispiel die Zahl der Nutzer, die ein Rating für die App abgegeben haben oder vielleicht auch nur ein Zehntel dieser. Dieser Wert lässt sich ohne die Informatione, wie die Metrik als System umgesetzt wird, schwer bestimmten.

Für $\delta(k)$ gilt die Gleichung:

$$\delta(k) = \frac{\tau}{k} \cdot (1 - \frac{1}{\Delta t}) \tag{5.35}$$

τ berechnet sich dabei aus der Differenz zwischen $RSF(app_i, k)$ und dem initialen Meldewert (4 im Fall der Funktionalität). Δt ist der zeitliche Unterschied zwischen diesem und der letzten

Problemmeldung. Je weiter zwei Problemmeldungen zeitlich bei einander liegen, desto schwächer wird der Einfluss auf den Gesamtmeldewert.

Abschließend muss der Meldewert noch auf das Intervall [0,1] skaliert werden:

$$Problemmeldungen_F(app_i) = \frac{RSF(app_i, k)}{4} \tag{5.36}$$

$$Problemmeldungen_G(app_i) = \frac{RSG(app_i, k)}{4} \tag{5.37}$$

$$Problemmeldungen_D(app_i) = \frac{RSD(app_i, k)}{5} \tag{5.38}$$

Für die Kategorien findet bezüglich der Funktionalität und der Problemmeldungen ein direktes Mapping zwischen dem Vertrauenswert und der Vertrauenskategorie statt. Liegt RSF oder RSG für eine App zwischen 0 und 1, fällt die App in dem jeweiligen Bereich in die Vertrauenskategorie $K5$, liegt der Wert zwischen 1 und 2 fällt die App in $K4$, zwischen 2 und 3 fällt die App in $K3$ und liegt sie zwischen 3 und 4 fällt die App in $K4$. Wurden keine Problemmeldungen abgegeben oder nur solche, die zurückgeben, dass keine Probleme vorliegen, fällt die App in Kategorie $K5$.

Die Kategorie bezüglich des Umgangs mit privaten Daten hängt mit dem Inhalt der Problemmeldungen zusammen ($RD_1 \mapsto K2$; $RD_2 \mapsto K3$; $RD_3 \mapsto K4$; $RD_3 \mapsto K5$; $RD_3 \mapsto K5$; $RD_3 \mapsto K5$).

Nachdem in diesem Kapitel die Metrik aufgestellt wurde, werden in Kapitel 6 Konzepte vorgestellt, wie diese Metrik sich umsetzen lässt.

Problemmeldung	Rückmeldungen		
Funktionalität			
RF_1: Abstürze	Keine (0)	Wenige (1)	Viele (2)
RF_2: Usability	Gut (0)	Eingeschränkt (1)	Schlecht (2)
RF_3: Bugs	Keine (0)	Selten/Wenige (1)	Oft/Viele (2)
Gefahrenpotenzial			
RG_1: Erhöhter Batterieverbrauch	Keinen (0)	Leicht erhöht (1)	Deutlich erhöht (2)
RG_2: Versteckte Installation oder Deinstallation von Apps	Keine (0)	Vielleicht (1)	Festgestellt (2)
RG_3: Verursacht Gebühren aufgrund hohem Netzwerkgebrauch	Keine (0)	Vielleicht (1)	Festgestellt (3)
RG_4:Guthabensverlust (durch SMS)	Keiner (0)	Vielleicht (1)	Festgestellt (4)
Umgang mit privaten Daten			
RD_1: GPS/Standort an Werbeserver	Nein (0)		Festgestellt (2)
RD_2: Telephon Information an Content Server (IMSI, ICC-ID)	Nein (0)		Festgestellt (2)
RD_3: Geräte ID an Content Server (IMEI)	Nein (0)		Festgestellt (2)
RD_4: Adressbuch	Nein (0)		Festgestellt (5)
RD_5: SMS-Speicher	Nein (0)		Festgestellt (5)
RD_6: Kalender	Nein (0)		Festgestellt (5)

Tabelle 5.1.: Übersicht über die möglichen Problemmeldungen und ihre Werte für die Funktion.

6. TriTrust

In Kapitel 5 wurde die *TriMetrik* vorgestellt, um den Vertrauenswert in eine App zu berechnen. In diesem Kapitel werden mit *TriTrust* nun mögliche Umsetzungen der Metrik vorgestellt. Dazu werden zunächst in Unterkapitel 6.1 der Ist-Zustand betrachten. Anschließend wird in Unterkapitel 6.2 diskutiert an welchen Stellen sich die Metrik implementieren lässt und es werden drei verschiedene Konzepte vorgestellt. Abschließend wird in Unterkapitel 6.3 überlegt, wie sich das System um Privacy-Komponenten erweitern lässt und wie sich diese auf den Vertrauens-Wert auswirken.

6.1. Ist-Zustand

Abbildung 6.1 zeigt den aktuell Ist-Zustand. Apps werden von einem Entwickler geschrieben und in den App-Marktplatz gestellt. Alternativ kann das .apk-File auch an anderer Stelle zur Verfügung gestellt werden. Ein Benutzer kann dann über den App-Marktplatz oder die andere Quelle die App herunterladen. Über die App-Marktplätze erhält er Feedback zur App via Ratings und Reviews. Diese Bewertung findet durch andere Benutzer statt.

Der Benutzer hat dabei im System mit vier Agenten zu tun, denen er mehr oder weniger vertrauen kann. Dies wären andere Benutzer, der App-Marktplatz oder eine andere Quelle von der Apps bezogen werden können, die Entwickler von Apps und natürlich die Apps selber. Siehe dazu Abbildung 6.1.

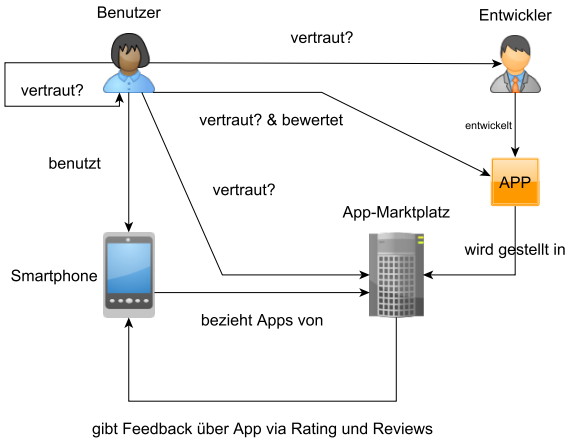

Abbildung 6.1.: Der Ist-Zustand der Interaktion und des Vertrauens ohne Erweiterungen der Systeme.

Andere Benutzer bewertet Apps durch Ratings und Reviews. Dies kann in App-Marktplätzen direkt passieren oder aber auch in inoffiziellen Blogs und Foren[1]. Dabei ist die Frage, ob man den Bewertungen der anderen Benutzer vertrauen kann. Man kann zwar ihre Ratings und Reviews meistens einsehen, doch Anhaltspunkte, wie diese einzuschätzen sind, gibt es wenige.

Die Quellen von Apps - in den meisten Fällen App-Marktplätzen - können auch das Vertrauen von Benutzern genießen. Offizielle App-Marktplätze wie der Google Play Store, der App Store von Amazone oder der Samsung-eigene Marktplatz genießen eher das Vertrauen, dass sie schädliche Apps ausfiltert. Einem Drittanbieter-Marktplatz könnte dabei eher Misstrauen entgegengebracht werden. Genaue Informationen über einen Store sind dabei schwer zu bekommen und solange er auf keine Reporte wie jenen von Uscilowski [Usc13] Zugriff hat, ist es schwer sich einen Eindruck zu machen.

Des weiteren kann der Benutzer Vertrauen in den Entwickler haben, weil man ihn beispielsweise persönlich kennt oder bereits Apps von ihm verwendet, mit denen man zufrieden ist, beziehungsweise denen man vertraut. Großen Entwicklern gegenüber kann man durch Gerüchte und Berichte Vertrauen entgegenbringen oder nicht. Bei einem Entwickler, mit dem man noch keine Erfahrungen hat und der nicht sehr bekannt ist, kann Vertrauen nur schwer aufgebaut werden.

Als Letztes kann der Benutzer natürlich Vertrauen in die Apps selber haben. Dieses kann durch die Ratings und Reviews, Berichte oder eine Betrachtung der Permissions entstehen. Wie diese zu bewerten sind, kann gerade weniger informierten Benutzern schwer fallen.

In Unterkapitel 6.2 wird ergründet, welches Framework aufgebaut werden kann, um die TriMetrik umzusetzen. Dazu werden mehrere Ansätze ergründet, wie sich TriTrust umsetzen lässt.

6.2. Mögliche Konzepte

Als Ansatz für die Implementierung stehen uns drei Möglichkeiten offen. Die erste wäre eine Modifikation des Betriebssystem des Smartphones oder eine App, die auf diesem läuft (**M1**, siehe Abbildung 6.2). Alternativ lässt sich ein Service implementieren (**M2**, siehe Abbildung 6.3), der auf einem Server läuft. Außerdem kommt ein PlugIn für einen App-Marktplatz in Frage (**M3**, siehe Abbildung 6.4).

In den Unterkapiteln 6.2.1 bis 6.2.3 werden die einzelnen Konzepte beschrieben, sowie ihre Vor- und Nachteile diskutiert. Eine große Rolle spielt dabei, wie die Informationen bezogen werden können, die für die Metrik gebraucht werden. Eine Übersicht dazu liefert Tabelle 6.1. Dabei wird unterschieden, ob die Werte direkt vorhanden sind oder sie von einer anderen Quelle bezogen werden müssen.

6.2.1. M1 - Erweiterung des Systems

Eine Möglichkeit die vorgeschlagene Metrik zu implementieren stellt eine Implementierung direkt auf dem dar. Wann immer der Benutzer sich bei der Installation einer App nicht sicher ist, kann er diese dann zunächst von *TriTrust* prüfen lassen.

[1]Etwa: http://www.sebastian-pertsch.de/1911/die-besten-android-apps.html

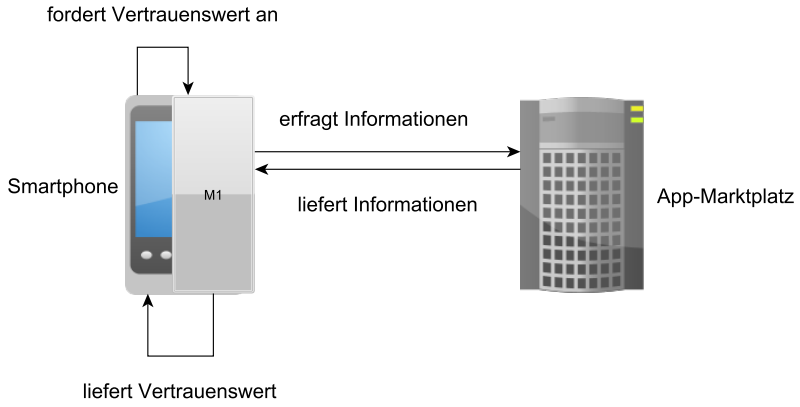

Abbildung 6.2.: M1 - Umsetzung als Erweiterung des Systems

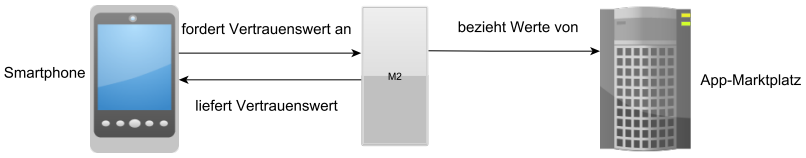

Abbildung 6.3.: M2 - Umsetzung als Service.

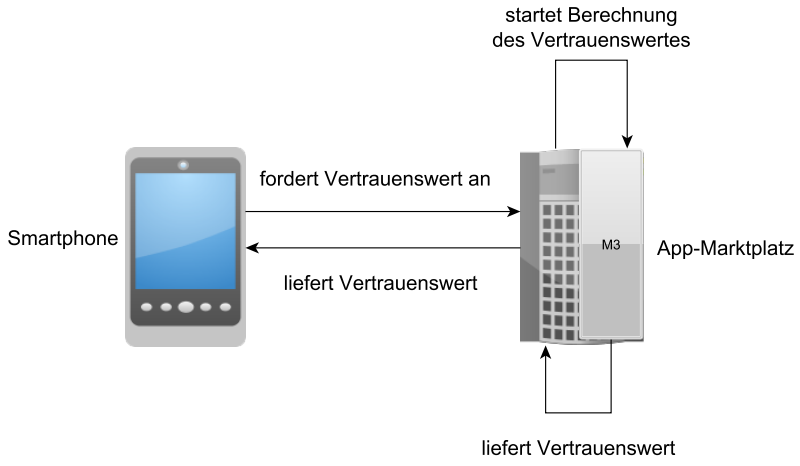

Abbildung 6.4.: M3 - Umsetzung als Erweiterung eines App-Marktplatzes.

Modul [a]	Reputation		Permissions		Nutzung	Informations-fluss
	Rating	Feedback	Genutzte Permissions	Vergleich mit anderen Apps		
M1	indirekt	indirekt	direkt	indirekt	direkt	direkt
M2	indirekt	direkt	direkt	indirekt	indirekt	indirekt
M3	(in)direkt	direkt	direkt	direkt	(in)direkt	indirekt

Tabelle 6.1.: Übersicht welches Modul auf welche Faktoren für die Metrik direkt oder indirekt zugreifen können.

[a] Verwende Modulbezeichnungen aus Abschnitt 6.2

Der Vorteil ist, dass wenn das System direkt auf dem Smartphone sitzt, Informationen wie die Nutzung von Apps direkt gemessen werden können. Wenn die, für die Zukunft geplante Analyse der Erkenntnisse von *TaintDroid* direkt in die Metrik einfließt, könnte *TaintDroid* direkt in das System[2] integriert werden.

Der offensichtliche Nachteil ist jedoch, dass beinahe alle Daten von externen Quellen bezogen werden müssen. Für die Problemmeldungen wurde bereits festgelegt, dass ein eigenes System mit Server geschaffen werden muss. Dieser Server muss dann jedoch um andere Werte wie die Nutzung aller Benutzer, bereits errechnete Vertrauens-Werte oder die Informationen über die genutzten Permissions der Apps erweitert werden. Damit wird ein System benötigt, welches dem Modul *M2* schon sehr ähnlich ist. Informationen über das Rating oder die Permissions der anderen Apps müssen von externen Quellen bezogen werden. Die Möglichkeit *TriTrust* direkt auf dem Smartphone zu implementieren ist also keine ideal Lösung.

6.2.2. M2 – Service

Ein Service in Form einer Website oder einer Online-Applikation wäre eine weitere Alternative. Wenn ein Benutzer sich bei einer App unsicher ist, kann er online nachsehen, wie es um ihren Vertrauenswert steht.

Hierbei stehen zwei Möglichkeiten zur Verfügung, wann eine App von *TriTrust* bewertet wird. Entweder wird eine App bewertet, wenn das erste Mal für sie ein Vertrauenswert angefordert wird oder von einem oder mehreren App-Marktplätzen werden neue Apps immer direkt bewertet, wenn sie zur Verfügung gestellt werden. Gegen die Berechnung des Vertrauenswertes beim ersten Zugriff spricht, dass dieser erste Benutzer eventuell eine Zeit auf sein Ergebnis warten muss. Der klare Nachteil alle Apps zu erfassen ist ein großes Aufkommen an unnötigen Daten. Die große Menge an Apps im Play Store sorgt dafür, dass bei vielen Apps eine hohe Wahrscheinlichkeit besteht, dass sich niemand für den Vertrauenswert dieser Apps interessiert.

[2] Hierfür muss nach aktuellem Stand dann jedoch das Betriebssystem verändert werden, da *TaintDroid* dies benötigt. Die Lösung als App fällt folglich heraus.

Für die Umsetzung von *TriTrust* als Service spricht, dass, wie im vorherigen Unterkapitel erwähnt, einige Informationen sowieso global zwischengespeichert werden müssen und für die Problemmeldungen in jedem Fall ein eigenes System benötigt wird. Außerdem ermöglicht dies einen leichten Zugriff für jede Person auf die Informationen.

Auch bei einem Service bleibt das Problem, dass Informationen über das Rating von externen Quellen bezogen werden muss. Außerdem benötigt es eine Möglichkeit, um an Informationen über das Nutzerverhalten der Apps zu gelangen. Diese Informationen lassen sich durch eine App auf dem System gewinnen. Genauso müsste auf den Smartphones selber *TaintDroid* installiert werden.

Der Service bringt also Vorteile mit sich, doch sie hat auch ihre Nachteile, weshalb sie auch nicht die perfekte Lösung darstellt.

6.2.3. M3 - Erweiterung des App-Marktplatzes

Als letzte Möglichkeit bietet sich noch die Erweiterung eines App-Marktplatzes an. Dem Benutzer kann zu jeder App neben den üblichen Informationen zusätzlich der Vertrauenswert angezeigt werden. Wie bei *M2* kann alternativ darüber nachgedacht werden, dass dieser Wert erst berechnet wird, wenn er erstmals angefordert wurde.

Der Vorteil der App-Marktplatz-Erweiterung ist, dass einige Informationen wie ein Rating und die verlangten Permissions bereits zur Verfügung stehen. Der *Google Play Store* zumindest misst auch das Nutzerverhalten der von ihm bezogenen Apps. Darüber hinaus steht im Normalfall eine Hardware-Infrastruktur (Server etc.) bereit, die genutzt werden kann.

Doch auch dieser Ansatz hat Nachteile. Wenn der Marktplatz kein geeignetes Benutzersystem aufweist und nur wenige Ratings zur Verfügung stehen, sinkt hier der Verlass in die Korrektheit des Ratings. Wenn dann nicht wie beim *Play Store* Informationen über die Nutzung gesammelt werden, ergibt sich hier das selbe Problem wie beim Service. *TaintDroid* muss natürlich auch in diesem Fall vom Benutzer installiert werden.

Darüber hinaus werden bei diesem Ansatz wahrscheinlich nur die Apps aus diesem Marktplatz bewertet. Es sollte als eher unwahrscheinlich betrachtet werden, dass ein Marktplatz Bewertungen für die Apps eines Konkurrenten durchführen will. Dadurch schränkt sich für den Benutzer die Menge der Apps ein.

Die Erweiterung des App-Marktplatzes stellt die aussichtsreichste Möglichkeit dar, doch auch sie ist auf Grund der genannten Nachteile nicht die perfekte Lösung.

Eine Kombination aus einer lokalen Komponente auf dem Smartphone, welche die Nutzung von Apps misst und *TaintDroid* mit sich bringt und entweder einem eigenständigen Service oder einer Erweiterung für einen App-Marktplatz stellte die Lösung für *TriTrust* dar.

Alternativ besitzen aber auch App-Marktplätze wie der Google Play Store eine Möglichkeit die Nutzung einer App zu messen. Wenn Benutzer keine zusätzliche Software auf ihrem Smartphone installieren wollen, würde in diesem Fall $M3$ auch alleine arbeiten können. Gerade zur Einführung und zu ersten Test bietet sich diese Möglichkeit an, da sie den wenigsten Aufwand für den Benutzer

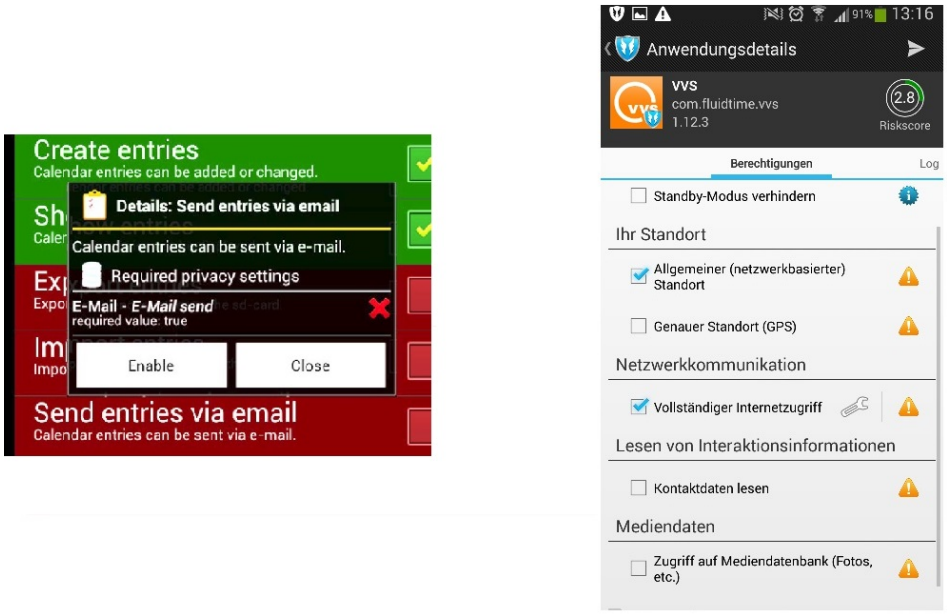

Abbildung 6.5.: Mit PMP und AppGuard lassen sich die Resources beziehungsweise die Permissions einer App einschränken..

bedeutet. Auch aus diesem Grund wurde diese Variante für den Prototypen im nächsten Kapitel gewählt.

Bevor die Implementierung des Prototypen vorgestellt wird, wird zunächst im folgenden Unterkapitel abschließend überlegt, wie Privacy-Systeme das Vertrauen in eine App beeinflussen können.

6.3. Kooperation mit Privacy-Systemen

In Kapitel 3 wurden bereits einige Privacy-Systeme aufgeführt. Werden diese auf dem Smartphone installiert, kann das Vertrauen in eine App bezüglich des Gefahrenpotenzials oder des Umgangs mit privaten Daten indirekt steigern, indem sie etwa die Permissions der App einschränken.

Ein Beispiel hierfür wären die bereits vorgestellte *PMP* oder *AppGuard* (siehe Abbildung 6.5). Vor allem der Umgang mit privaten Daten lässt sich dadurch stark verbessern, wenn dem Benutzer kommuniziert wird, welche Daten weitergesendet werden. Entzieht er der App die dafür benötigten Rechte, steigert sich indirekt das Vertrauen in die App, da sie keine Bedrohung mehr für die Daten darstellt.

Die *TriMetrik* überprüft bezüglich des Riskratings kritische Permissions und die meisten Problemmeldungen lassen sich ebenfalls auf entsprechende Permissions mappen. Dadurch kann dem Privacy-System eine Empfehlung geliefert werden, welche Permissions verboten werden sollten. Diesen Vorschlag muss der Benutzer dann nur noch bestätigen.

Auf die Funktionalität einer App kann mit diesen Systemen kein Einfluss genommen werden. Es ist eher möglich, dass sich diese durch die Einschränkung der Permissions verschlechtert. Natürlich muss auch klar sein, dass das tatsächliche Vertrauen in die App an sich dadurch auch nicht steigt. Das eigentliche Ziel dieser Arbeit sollte es ja sein einen Ansatz jenseits verschärfter und Privacy-Maßnahmen zu finden.

In diesem Kapitel wurden die möglichen Konzepte für *TriTrust*, die Umsetzung der *TriMetrik* vorgestellt und ihre Vor- und Nachteile diskutiert. Außerdem wurde ein Zusammenspiel zwischen einem Vertrauenssystem und Privacy-Systemen diskutiert. Im nächsten Kapitel 7 wird nun die Implementierung eines Prototypen des Konzepts *M3* behandelt.

7. Implementierung

Für die Implementierung eines Prototypen von *TriTrust* wurde eine *html5* Web Applikation gewählt, welche als Erweiterung für einen online App-Marktplatz dienen kann. Dazu wird in Unterkapitel 7.1 beschrieben, wie die Erweiterung im App-Marktplatz eingesetzt werden kann und welche Schnittstellen zwischen der Erweiterung und dem Marktplatz bestehen. Unterkapitel 7.2 gibt tiefere Einsicht in den Aufbau des Prototypen.

7.1. Ansatz und Schnittstellen

Die Erweiterung eines App-Marktplatzes in Form einer Web Applikation bietet einen guten Ansatzpunkt. Dank *html5* ist der Prototyp plattformunabhängig einsetzbar und kann leicht von allen Online-App-Marktplätzen verwendet werden. Dabei bietet es sich an, etwa wie in Abbildung 7.1 gezeigt, den Vertrauenswert neben anderen Informationen, wie den Ratings, anzuzeigen. Für weitere Details muss der Benutzer dann auf die Anzeige klicken und wird weitergeleitet.

Für die Berechnung des Vertrauenswertes und der Kategorien wurde die Metrik mit Hilfe von Javascript implementiert. Das dafür benötigte Input wie die Ratings, die Permissions und (wenn vorhanden) die Nutzung werden an die Skripte übergeben und der Vertrauenswert für die drei Bereiche berechnet und kann dann auf den Servern des App-Marktplatzes gespeichert werden.

Die Eingabe der Gewichtung für die drei Bereiche und das Absenden von Problemmeldungen finden bei dem Prototypen über die Web Applikation statt, könnten aber auch an anderer Stelle im App-Marktplatz eingebaut werden.

Im nächsten Unterkapitel werden die Interaktionsmöglichkeiten des Benutzers mit dem Prototypen beschrieben.

Abbildung 7.1.: Möglichkeit das Rating im Google Play Store anzuzeigen.

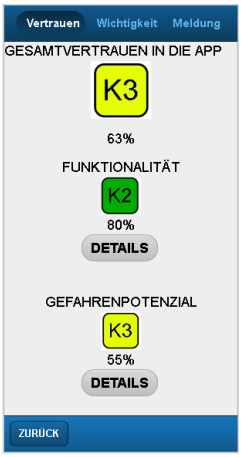

Abbildung 7.2.: Die Hauptanzeige gibt eine Übersicht des gesamten Vertrauenswertes und der einzelnen drei Bereiche.

7.2. Features

Der Hauptbildschirm der Hauptansicht (siehe Abbildung 7.2) bietet eine Übersicht des Vertrauenswertes sowie der Kategorie der App und auch der einzelnen Bereiche. Dort kann über die Tabs der Hauptansicht die Gewichtung für die Vertrauensberechnung festgelegt und Problemmeldungen bezüglich der App abgesendet werden.

Über die *Detail*-Buttons gelangt man in ein ausführlichere Begründung, wie der Vertrauenswert zustande kam. Ein Beispiel für das Riskrating einer App mit Begründung findet sich in Abbildung 7.3.

Die Studie von Yan et al. [YLNY13] zeigt, dass wenn für eine App ein Vertrauenswert angezeigt wird, oft auch Interesse besteht, wie dieser zustande kam. Dabei können beim Gefahrenpotenzial und dem Umgang mit privaten Daten konkrete Hinweise bezüglich der Permissions und der Problemmeldungen gegeben werden, während die Funktionalität (abgesehen von den Problemmeldungen) eher nur Spekulationen zulässt, wenn es um Ranking und Nutzung geht.

Die Anzahl der Bildschirme ist überschaubar und die Navigation sollte dem Benutzer leicht fallen. Dadurch ist eine schnelle aber doch detaillierte Information des Benutzers über den Vertrauenswert möglich.

Im nächsten Kapitel findet eine Bewertung der *TriMetrik* statt, ob die gesetzten Ziele dieser Arbeit erfüllt wurden und wo es noch offene Punkte gibt.

Abbildung 7.3.: Zu den einzelnen Bereichen gibt es ausführliche Informationen, wie der Vertrauenswert zustande kam.

8. Bewertung

Das Ziel dieser Arbeit war es eine Vertrauens-Metrik für Apps aufzustellen. In diesem Kapitel soll nun betrachtet werden, wie weit die Metrik die Anforderungen erfüllt, die sich an die Eigenschaften von Vertrauen, welche in Kapitel 2.1.1 beschrieben wurden, stellen. Dies geschieht in Unterkapitel 8.1. Außerdem findet in Unterkapitel 8.2 ein Vergleich mit der aufgestellten Definition2.3.2 für das Vertrauen in eine App statt, ob auch diese umgesetzt wurde.

8.1. Erfüllung der Eigenschaften

Zehn Eigenschaften für Vertrauen wurden in Unterkapitel 2.1.1 aufgestellt. Im Folgenden wird nun betrachtet, wie weit die *TriMetrik* diese erfüllt.

(1) **Vertrauen ist gerichtet**: Der Benutzer vertraut einer App, die er benutzen will, sich aber nicht sicher ist. Die Richtung des Vertrauens ist klar in der *TriMetrik*.

(2) **Vertrauen ist nicht symmetrisch**: Die Rückrichtung, dass eine App dem Benutzer vertraut ist in der *TriMetrik* nicht vorgesehen, also kann keine Symmetrie bestehen.

(3) **Vertrauen kann transitiv sein**: Im Falle der Ratings und der Problemmeldungen vertraut der Benutzer den Einschätzungen anderer Benutzer. Wie weit dieses transitive Vertrauen gerechtfertigt ist, wird dabei in der *TriMetrik* mit berechnet.

(4) **Vertrauen ist subjektiv**: Das Vertrauen in die App ist subjektiv in Hinsicht auf die Ratings und die meisten Problemmeldungen, die bezüglich der App abgegeben werden. Hinzu kommen in der *TriMetrik* aber auch Analysewerte von den Permissions und der Flussanalyse.

(5) **Vertrauen ist kontextabhängig**: Jeder Benutzer kann andere Vorstellungen haben, welcher der drei Bereiche der *TriMetrik* für ihn wichtig ist. Durch die Gewichtung kann er darauf Einfluss nehmen und den Wert so in Richtung seiner Vorstellungen anpassen.

(6) **Vertrauen ist messbar**: Die *TriMetrik* berechnet einen Vertrauenswert und Vertrauenskategorien für eine App aus messbaren Eigenschaften wie Werte der Ratings, Anzahl der Permissions etc.

(7) **Vertrauen ist abhängig von der Vergangenheit**: Vergangene Erfahrungen mit der App können von Benutzern über die Problemmeldung oder das Rating in den Vertrauenswert der *TriMetrik* einfließen.

(8) **Vertrauen ist dynamisch**: Durch neue Ratings oder erhaltene Problemmeldungen kann der Vertrauenswert einer App den Erfahrungen der Benutzer mit der App von der *TriMetrik* entsprechend angepasst werden und ändert sich damit dynamisch.

(9) **Vertrauen kann eine zusammengesetzte Eigenschaft sein**: Das Vertrauen in die App setzt sich in der *TriMetrik* aus drei Bereichen zusammen, welche wiederum von unterschiedlichen Faktoren bestimmt werden.

(10) **Unwissen verlangt Vertrauen**: Der Hauptgedanke hinter der *TriMetrik* ist es, einem Benutzer eine Empfehlung geben zu können, ob er eine App installieren soll oder nicht. Da er die App davor noch nicht verwendet hat, hat er keine Ahnung wie sich diese verhalten wird und so dient ihm der Vertrauenswert als Hilfsmittel zur Entscheidungsfindung.

Alle aufgestellten Eigenschaften von Vertrauen werden also von der *TriMetrik* abgedeckt oder sind durch die Umstände gegeben. Die *TriMetrik* entspricht also soweit den Anforderungen.

8.2. Erfüllung der Definition

Neben den Eigenschaften für Vertrauen wurde in Kapitel 2 auch eine Definition für das Vertrauen in eine App aufgestellt. Ob die *TriMetrik* mit den Anforderungen der Definition übereinstimmt soll nun in diesem Unterkapitel betrachtet werden.

Zur Erinnerung sei hier noch einmal die Definition gegeben: „Einer App wird vertraut, wenn sie die versprochenen Funktionalitäten bietet, private Daten nicht gegen den Wunsch des Benutzers weitergibt und möglichst geringes Potenzial besitzt, um Schaden anzurichten."

Ob eine App die versprochene Funktionalität bietet, erfährt der Benutzer aus der Analyse der *TriMetrik* und des Vertrauenswertes beziehungsweise der Vertrauenskategorie der Funktionalität der App.

Über den Umgang mit privaten Daten erhält der Benutzer ebenfalls Rückmeldung von der *TriMetrik*. Zusätzlich wurde in Kapitel 6 diskutiert, wie sich durch die Kooperation mit einem Privacy-System, wie beispielsweise PMP, die Weitergabe der Daten einschränken lässt und der App dann insofern vertraut werden kann, dass es ihr nicht mehr möglich ist diese Daten weiterzusenden.

Das Potenzial einer App Schaden anzurichten wird ebenfalls von der *TriMetrik* berechnet und dem Benutzer eine Warnung über ungewöhnliche Permissions oder gemeldete Fehlverhalten der App gibt. Kritische Permissions können dabei wiederum von einem Privacy-System eingeschränkt werden, um das Vertrauen zu steuern. Jedoch muss auch von Seite des Betriebssystems ein gewisser Grad an Sicherheit garantiert werden.

Die *TriMetrik* erfüllt also auch die Anforderungen, welche der Definition entnommen wurden.

9. Zusammenfassung und Ausblick

Zunächst wurden in dieser Arbeit Definitionen von Vertrauen betrachtet, um seine Eigenschaften zu ergründen. Mit Hilfe dieser wurde eine eigene Definition für das Vertrauen in eine App aufgestellt, welche den allgemeinen Eigenschaften von Vertrauen, wie auch den speziellen Anforderungen, die sich in Bezug auf eine App stellen, entspricht. Dabei wurden die drei Bereiche *Funktionalität*, *Gefahrenpotenzial* und *Umgang mit privaten Daten* als für das Vertrauen in eine App elementar erkannt.

Nachdem eine Übersicht über verwandte Arbeiten geliefert wurde, wurden anschließend bestehende Vertrauenssysteme tiefgehend betrachtet und zusammen mit Analysesystemen bezüglich der Permissions von Apps und dem Kontroll- sowie Datenfluss auf ihre Brauchbarkeit für eine eigene Metrik untersucht. Um das Vertrauen in eine App für den Benutzer brauchbar darzustellen, wird auf eine Kombination aus prozentualer Skala und Kategorien gebaut. Für die Berechnung des Vertrauenswertes wurden verschieden Faktoren wie Ratings, Permissions und Problemmeldungen den drei Bereichen zugeordnet.

Mit der *TriMetrik* wurde dann eine Metrik aufgestellt, um diesen Vertrauenswert berechnen zu können. Dazu wurden unterschiedliche Funktionen von den vorgestellten Metriken entliehen, aber auch eigene Funktionen aus den gewonnenen Erkenntnissen aufgestellt.

Die drei Bereiche Funktionalität, Gefahrenpotenzial und Zugriff auf private Daten werden allesamt gesondert betrachtet und für jeden separat der Vertrauenswert und die Vertrauenskategorie berechnet. Diese Aufteilung hebt die *TriMetrik* von allen in dieser Arbeit untersuchten Ansätzen ab.

Für die drei Bereiche wurden Faktoren erörtert, die genutzt werden können um das Vertrauen in diese zu ergründen. Dabei wurden Vertrauensmetriken entliehen, welche an anderer Stelle bereits Verwendung fanden und deren Effektivität bestätigt wurde.

An manchen Punkten wurden die Metriken angepasst und da große Studien den Rahmen dieser Arbeit sprengen, kann nur gemutmaßt werden, ob die Effektivität erhalten bleibt. Gerade in Bezug auf das Riskrating und die Nutzung einer App ist der eigene Ansatz eine deutlich einfachere Version im Vergleich zu anderen Metriken und sollte sich dieser als untauglich erweisen ist darüber nachzudenken, eine komplexere Metrik zu verwenden.

Für das Rating einer App und die Problemmeldungen werden die hauptsächlich subjektiven Wahrnehmungen anderer Benutzer als Maß genommen. Dabei können auch die bekannten Probleme von Reputationssystemen auftreten. Die Metrik baut darauf auf eine hohe Anzahl von Ratings und Problemmeldungen zu bekommen, wie sie etwa die große Benutzerzahl des *Google Play Store* bietet, wodurch ein Angreifer eine sehr große Zahl an falschen Bewertungen abgeben muss, um ins Gewicht zu fallen. Außerdem werden Problemmeldungen mit einem Zeitfaktor versehen, sodass viele Bewertungen in kurzer Zeit, wobei es sich um ein typisches Angriffsschema handelt, kaum einen Einfluss

auf den Vertrauenswert besitzen. Diese Maßnahmen dienen um offensichtliche Angriffe abzuwehren. Es liegt jenseits der Möglichkeiten dieser Arbeit subtiler Angriffe festzustellen.

Mögliche Konzepte zur Umsetzung der Metrik wurden mit *TriTrust* präsentiert. Dabei wurde über die Vor- und Nachteile der Umsetzung als Erweiterung des OS auf dem Smartphone, als Middleware oder als Erweiterung eines App-Marktplatzes diskutiert. Eine Kombination aus einer lokalen Komponente und einer globalen Komponente in Form der Middleware oder dem App-Marktplatz wurde als ideal betrachtet.

Ein Prototyp für die Erweiterung eines App-Marktplatzes wurde dann in Form einer Web App vorgestellt.

Abschließend fand eine Bewertung der Metrik statt, ob sie sie den aufgestellten Anforderungen und Definitionen entspricht. Das Hauptziel, eine Metrik, welche die drei Bereiche abdeckt und dem Benutzer eine brauchbare Rückmeldung gibt wurde erfüllt. Bezüglich einiger Funktionen für die Berechnung der Metrik muss jedoch noch Untersucht werden, wie zuverlässig und korrekt die Ergebnisse sind.

Ausblick

Auch wenn die *TriMetrik* an einigen Stellen auf andere Metriken zurückgreift, deren Aussagekraft bereits getestet wurde, wurden für einige Faktoren neue Berechnungen vorgeschlagen. Wie bereits genannt wurden etwa für die Nutzung oder das Riskrating neue Funktionen aufgestellt und deren Aussagekraft muss durch ausführliche Studien gestützt werden, die den Rahmen einer Bachelorarbeit sprengen.

Informationen, die dabei herausgefunden werden sollten, wären:

- Wie sieht die Wahrscheinlichkeit von kritischen Permissions in den Kategorien aus und wie kann man damit das Vertrauen auf Grund ungewöhnlich seltener Permissions berechnen?
- Ergibt sich durch die Nutzung einer Permission ein nachvollziehbarer Vertrauenswert und sollte die Nutzung bei bestimmten Kategorien eine geringere oder gar keine Rolle spielen?
- Mit wie vielen Problemmeldungen für eine App ist nach einer akzeptablen Zeit zu rechnen, sodass sich der Vertrauenswert durch Problemmeldungen einpendeln kann?
- Spiegelt die Gewichtung der gemeldeten Fehlverhalten einer Problemmeldung die Vorstellungen der breiten Nutzermasse wieder?

Mit Hilfe von empirischen Studien ließe sich die Effektivität der App beweisen, beziehungsweise weiter verfeinern, um eine genauere Aussage treffen zu können.

Eine Beeinflussung des Vertrauens in eine App durch den Entwickler dieser oder den App-Marktplatz, auf welchem sie angeboten wird, könnte weiter untersucht werden, nachdem es in dieser Arbeit nur angesprochen wurde.

Auch wenn Veränderungen der Permissions von Android eher selten sind, können diese doch von Zeit zu Zeit auftreten, wie an dem veralteten Regelsatz von *Kirin* zu sehen war. Eine Anpassung der kritischen Listen an geänderte Permissions sollte durchgeführt werden, um die Metrik aktuell zu halten.

Ebenfalls eine stetige Weiterentwicklung findet bei *TaintDroid* statt und so sollte diese beobachtet werden. Über die direkte Einbindung in das System wurde bereits diskutiert und durch diese automatische Analyse wäre die Abhängigkeit von Problemmeldungen der Benutzer kleiner. Andererseits könnte der Benutzer sich auch überwacht fühlen. Eine Benutzerstudie zur Bereitschaft ein solches System auf seinem Smartphone zu installieren, aber auch allgemein *TaintDroid* zu nutzen, könnte für Klarheit sorgen.

Eine weitere Möglichkeit für die Zukunft stellt die Erweiterung für das *PMP*-System dar. In diesem fallen die Permissions heraus und werden wie angesprochen durch Resources ersetzt. Da sich der Satz der Resources ständig dynamisch erweitert, muss über eine Möglichkeit nachgedacht werden, wie diese sich in kritische Kategorien einordnen ließen und ein Vergleich gezogen werden kann.

A. Anhang

A.1. Kritische Permissions

Dies ist eine Übersicht über die kritischen Permissions, welche verhindern, dass eine Applikation in den Bereichen Gefahrenpotenzial und Umgang mit privaten Daten in die Kategorie *K1* fallen kann, respektive einen Vertrauenswert von 100% in diesen Bereichen erreicht. Die Permissions und ihre Bedeutung wurden der Website android.com[1] entnommen, Stand 01. Dezember 2014 für Android 4.4.

CALL_PHONE[a]	make a phone call w/o user's confirmation
INTERNET[a]	open network sockets
MOUNT_UNMOUNT_FILESYSTEMS	mount / unmount file sys for removable storage.
SEND_SMS[a]	send SMS messages
WAKE_LOCK	Allows using PowerManager WakeLocks to keep processor from sleeping or screen from dimming
WRITE_CALENDAR	write the user's calendar data.
WRITE_CONTACTS	write the user's contacts data.
WRITE_HISTORY_BOOKMARKS[b]	write the user's browsing history and bookmarks.
WRITE_SMS	write SMS messages.
WRITE_EXTERNAL_STORAGE	write to external storage
NFC	perform I/O operations over NFC
GET_ACCOUNTS	access the list of accounts in the Accounts Service
BLUETOOTH	connect to paired bluetooth devices
BLUETOOTH_ADMIN	discover and pair bluetooth devices

Tabelle A.1.: Übersicht der kritischen Permissions, die eine Bedrohung für die Sicherheit des Smartphones darstellen können.

[a] Diese Permissions können den Benutzer Geld kosten.
[b] Diese Permission wird von vielen Schadhaften aber kaum bis keinen gewöhnlichen Apps verlangt, [SLG+12].

[1] http://developer.android.com/reference/android/Manifest.permission.html

ACCESS_COARSE_LOCATION	access to coarse (e.g., Cell_ID, WiFi) location
ACCESS_FINE_LOCATION	access to fine (e.g., GPS) location
PROCESS_OUTGOING_CALLS	monitor, modify, or abort outgoing calls.
READ_CALENDAR	read the user's calendar data.
READ_CONTACTS	read the user's contacts data.
READ_HISTORY_BOOKMARKS[a]	read the user's browsing history and bookmarks.
READ_PHONE_STATE	read only access to phone state.
READ_SMS	read SMS messages.
RECEIVE_MMS	monitor, record, or process MMS msgs.
RECEIVE_SMS	monitor, record, or process SMS msgs.
RECORD_AUDIO	record audio.
RECEIVE_WAP_PUSH	monitor incoming WAP messages.
READ_LOGS	read low_level log msgs.
READ_CALL_LOG	Allows an application to read the user's call log.
READ_VOICEMAIL	Allows an application to read voicemails in the system.

Tabelle A.2.: Übersicht der kritischen Permissions, die eine Bedrohung für privaten Daten des Benutzers auf dem Smartphone darstellen können.

[a] Diese Permission wird von vielen Schadhaften aber kaum bis keinen gewöhnlichen Apps verlangt, [SLG+12]

Literaturverzeichnis

[AG07] D. Artz, Y. Gil. A survey of trust in computer science and the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58 – 71, 2007. doi: http://dx.doi.org/10.1016/j.websem.2007.03.002. URL http://www.sciencedirect.com/science/article/pii/S1570826807000133. Software Engineering and the Semantic Web. (Zitiert auf Seite 11)

[Bac12] M. Backes. New Approach Uncovers Data Abuse on Mobile End Devices. Technischer Bericht, University Saarland, 2012. (Zitiert auf Seite 10)

[BCMO12] D. Barrera, J. Clark, D. McCarney, P. C. van Oorschot. Understanding and Improving App Installation Security Mechanisms Through Empirical Analysis of Android. In *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '12, S. 81–92. ACM, New York, NY, USA, 2012. doi:10.1145/2381934.2381949. URL http://doi.acm.org/10.1145/2381934.2381949. (Zitiert auf Seite 28)

[BDD+11] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A.-R. Sadeghi, B. Shastry. Practical and Lightweight Domain Isolation on Android. In *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '11, S. 51–62. ACM, New York, NY, USA, 2011. doi:10.1145/2046614.2046624. URL http://doi.acm.org/10.1145/2046614.2046624. (Zitiert auf Seite 24)

[BGH+12] M. Backes, S. Gerling, C. Hammer, M. Maffei, P. von Styp-Rekowsky. AppGuard - real-time policy enforcement for third-party applications. Technischer Bericht, SaarlÃ¤ndische UniversitÃ¤ts- und Landesbibliothek, Postfach 151141, 66041 SaarbrÃ¼cken, 2012. URL http://scidok.sulb.uni-saarland.de/volltexte/2012/4902. (Zitiert auf Seite 23)

[BGH+13] M. Backes, S. Gerling, C. Hammer, M. Maffei, P. von Styp-Rekowsky. AppGuard– Enforcing User Requirements on Android Apps. In *Tools and Algorithms for the Construction and Analysis of Systems*, S. 543–548. Springer, 2013. (Zitiert auf den Seiten 10 und 23)

[BMCO14] D. Barrera, D. McCarney, J. Clark, P. C. van Oorschot. Baton: Certificate Agility for Androids Decentralized Signing Infrastructure. In *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*, S. 1–12. ACM, 2014. (Zitiert auf Seite 28)

[BRSS11] A. R. Beresford, A. Rice, N. Skehin, R. Sohan. MockDroid: Trading Privacy for Application Functionality on Smartphones. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, HotMobile '11, S. 49–54. ACM, New York, NY, USA, 2011.

doi:10.1145/2184489.2184500. URL http://doi.acm.org/10.1145/2184489.2184500. (Zitiert auf Seite 23)

[CHY12] P. P. Chan, L. C. Hui, S. M. Yiu. DroidChecker: Analyzing Android Applications for Capability Leak. In *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WISEC '12, S. 125–136. ACM, New York, NY, USA, 2012. doi:10.1145/2185448.2185466. URL http://doi.acm.org/10.1145/2185448.2185466. (Zitiert auf Seite 45)

[CKW03] C. L. Corritore, B. Kracher, S. Wiedenbeck. On-line trust: concepts, evolving themes, a model. *International Journal of Human-Computer Studies*, 58(6):737 – 758, 2003. doi:http://dx.doi.org/10.1016/S1071-5819(03)00041-7. URL http://www.sciencedirect.com/science/article/pii/S1071581903000417. Trust and Technology. (Zitiert auf Seite 16)

[CYA12] P. H. Chia, Y. Yamamoto, N. Asokan. Is This App Safe?: A Large Scale Study on Application Permissions and Risk Signals. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, S. 311–320. ACM, New York, NY, USA, 2012. doi:10.1145/2187836.2187879. URL http://doi.acm.org/10.1145/2187836.2187879. (Zitiert auf den Seiten 26, 44 und 47)

[DDSW11] L. Davi, A. Dmitrienko, A.-R. Sadeghi, M. Winandy. Privilege Escalation Attacks on Android. In M. Burmester, G. Tsudik, S. Magliveras, I. Ilić, Herausgeber, *Information Security*, Band 6531 von *Lecture Notes in Computer Science*, S. 346–360. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-18178-8_30. URL http://dx.doi.org/10.1007/978-3-642-18178-8_30. (Zitiert auf Seite 25)

[DMM+12] G. Dini, F. Martinelli, I. Matteucci, M. Petrocchi, A. Saracino, D. Sgandurra. A Multi-criteria-Based Evaluation of Android Applications. In C. Mitchell, A. Tomlinson, Herausgeber, *Trusted Systems*, Band 7711 von *Lecture Notes in Computer Science*, S. 67–82. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-35371-0_7. URL http://dx.doi.org/10.1007/978-3-642-35371-0_7. (Zitiert auf Seite 27)

[DMM+13] G. Dini, F. Martinelli, I. Matteucci, M. Petrocchi, A. Saracino, D. Sgandurra. Evaluating the Trust of Android Applications through an Adaptive and Distributed Multi-criteria Approach. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*, S. 1541–1546. 2013. doi:10.1109/TrustCom.2013.189. (Zitiert auf den Seiten 27, 30, 34, 38, 39, 40, 48, 59 und 60)

[DSP+11] M. Dietz, S. Shekhar, Y. Pisetsky, A. Shu, D. S. Wallach. QUIRE: Lightweight Provenance for Smart Phone Operating Systems. In *USENIX Security Symposium*. 2011. (Zitiert auf Seite 24)

[EGC+10] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, A. N. Sheth. TaintDroid: An Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, S. 1–6. USENIX Association, Berkeley, CA, USA, 2010. URL http://dl.acm.org/citation.cfm?id=1924943.1924971. (Zitiert auf den Seiten 35 und 46)

[EGC+14] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, A. N. Sheth. TaintDroid: An Information Flow Tracking System for Real-time Privacy Monitoring on Smartphones. *Commun. ACM*, 57(3):99–106, 2014. doi:10.1145/2494522. URL http://doi.acm.org/10.1145/2494522. (Zitiert auf den Seiten 26, 35 und 46)

[EOM09] W. Enck, M. Ongtang, P. McDaniel. On Lightweight Mobile Phone Application Certification. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, S. 235–245. ACM, New York, NY, USA, 2009. doi:10.1145/1653662.1653691. URL http://doi.acm.org/10.1145/1653662.1653691. (Zitiert auf den Seiten 26 und 44)

[FBM12] D. Fraga, Z. Bankovic, J. M. Moya. A Taxonomy of Trust and Reputation System Attacks. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, S. 41–50. IEEE, 2012. (Zitiert auf Seite 21)

[FCH+11] A. P. Felt, E. Chin, S. Hanna, D. Song, D. Wagner. Android Permissions Demystified. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, S. 627–638. ACM, New York, NY, USA, 2011. doi:10.1145/2046707.2046779. URL http://doi.acm.org/10.1145/2046707.2046779. (Zitiert auf Seite 27)

[FGW11] A. P. Felt, K. Greenwood, D. Wagner. The effectiveness of application permissions. In *Proceedings of the 2nd USENIX conference on Web application development*, S. 7–7. USENIX Association, 2011. (Zitiert auf Seite 26)

[Gam88] D. Gambetta. Trust: Making and breaking cooperative relations. 1988. (Zitiert auf Seite 13)

[GHS08] A. Gutscher, J. Heesen, O. Siemoneit. Possibilities and Limitations of Modeling Trust and Reputation. *WSPI*, 332:50–61, 2008. (Zitiert auf Seite 15)

[GM10a] A. Girardello, F. Michahelles. AppAware: Which Mobile Applications Are Hot? In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '10, S. 431–434. ACM, New York, NY, USA, 2010. doi:10.1145/1851600.1851698. URL http://doi.acm.org/10.1145/1851600.1851698. (Zitiert auf Seite 41)

[GM10b] A. Girardello, F. Michahelles. Explicit and Implicit Ratings for Mobile Applications. In *GI Jahrestagung (1)*, S. 606–612. 2010. (Zitiert auf den Seiten 25 und 41)

[GS00] T. Grandison, M. Sloman. A survey of trust in internet applications. *Communications Surveys Tutorials, IEEE*, 3(4):2–16, 2000. doi:10.1109/COMST.2000.5340804. (Zitiert auf Seite 14)

[HCH08] F. K. Hussain, E. Chang, O. Hussain. A Robust Methodology for Prediction of Trust and Reputation Values. In *Proceedings of the 2008 ACM Workshop on Secure Web Services*, SWS '08, S. 97–108. ACM, New York, NY, USA, 2008. doi:10.1145/1456492.1456507. URL http://doi.acm.org/10.1145/1456492.1456507. (Zitiert auf Seite 22)

[HCLH11] G. Huerta-Canepa, D. Lee, S. Y. Han. Trust ME: A Trust Decision Framework for Mobile Environments. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, S. 464–471. 2011. doi:10.1109/TrustCom.2011.60. (Zitiert auf Seite 22)

[HZNR09] K. Hoffman, D. Zage, C. Nita-Rotaru. A Survey of Attack and Defense Techniques for Reputation Systems. *ACM Comput. Surv.*, 42(1):1:1–1:31, 2009. doi:10.1145/1592451.1592452. URL http://doi.acm.org/10.1145/1592451.1592452. (Zitiert auf Seite 21)

[JK07] Z. Jiang, S. Kim. Trust Model for Mobile Devices in Ubiquitous Environment. In *Proceedings of the 1st International Conference on Network-based Information Systems*, NBiS'07, S. 426–434. Springer-Verlag, Berlin, Heidelberg, 2007. URL http://dl.acm.org/citation.cfm?id=1776510.1776563. (Zitiert auf Seite 22)

[JMV+12] J. Jeon, K. K. Micinski, J. A. Vaughan, A. Fogel, N. Reddy, J. S. Foster, T. Millstein. Dr. Android and Mr. Hide: Fine-grained Permissions in Android Applications. In *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '12, S. 3–14. ACM, New York, NY, USA, 2012. doi:10.1145/2381934.2381938. URL http://doi.acm.org/10.1145/2381934.2381938. (Zitiert auf den Seiten 10 und 23)

[Jøs96] A. Jøsang. The Right Type of Trust for Distributed Systems. In *Proceedings of the 1996 Workshop on New Security Paradigms*, NSPW '96, S. 119–131. ACM, New York, NY, USA, 1996. doi:10.1145/304851.304877. URL http://doi.acm.org/10.1145/304851.304877. (Zitiert auf Seite 14)

[KF13] M. Kuehnhausen, V. Frost. Trusting smartphone Apps? To install or not to install, that is the question. In *Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2013 IEEE International Multi-Disciplinary Conference on*, S. 30–37. 2013. doi:10.1109/CogSIMA.2013.6523820. (Zitiert auf den Seiten 27, 30, 34, 37, 38, 44, 53, 54, 55 und 58)

[LDRL09] X. Liu, A. Datta, K. Rzadca, E.-P. Lim. StereoTrust: A Group Based Personalized Trust Model. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, S. 7–16. ACM, New York, NY, USA, 2009. doi:10.1145/1645953.1645958. URL http://doi.acm.org/10.1145/1645953.1645958. (Zitiert auf Seite 22)

[Mar94] S. P. Marsh. *Formalising trust as a computational concept*. Thesis or dissertation, University of Stirling, 1994. URL http://hdl.handle.net/1893/2010. (Zitiert auf Seite 21)

[MB09] Z. Malik, A. Bouguettaya. RATEWeb: Reputation Assessment for Trust Establishment Among Web Services. *The VLDB Journal*, 18(4):885–911, 2009. doi:10.1007/s00778-009-0138-1. URL http://dx.doi.org/10.1007/s00778-009-0138-1. (Zitiert auf Seite 22)

[MMH02a] L. Mui, M. Mohtashemi, A. Halberstadt. A computational model of trust and reputation. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, S. 2431–2439. 2002. doi:10.1109/HICSS.2002.994181. (Zitiert auf den Seiten 16, 18 und 21)

[MMH02b] L. Mui, M. Mohtashemi, A. Halberstadt. Notions of Reputation in Multi-agents Systems: A Review. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1*, AAMAS '02, S. 280–287. ACM, New York, NY, USA, 2002. doi:10.1145/544741.544807. URL http://doi.acm.org/10.1145/544741.544807. (Zitiert auf Seite 21)

[pew12] Privacy And Data Managment On Mobile Devices, 2012. URL http://www.pewinternet.org/files/old-media//Files/Reports/2012/PIP_MobilePrivacyManagement.pdf. (Zitiert auf den Seiten 18 und 34)

[PGS+12] H. Peng, C. Gates, B. Sarma, N. Li, Y. Qi, R. Potharaju, C. Nita-Rotaru, I. Molloy. Using Probabilistic Generative Models for Ranking Risks of Android Apps. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, S. 241–252. ACM, New York, NY, USA, 2012. doi:10.1145/2382196.2382224. URL http://doi.acm.org/10.1145/2382196.2382224. (Zitiert auf den Seiten 27 und 44)

[RCCF12] G. Russello, M. Conti, B. Crispo, E. Fernandes. MOSES: Supporting Operation Modes on Smartphones. In *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*, SACMAT '12, S. 3–12. ACM, New York, NY, USA, 2012. doi:10.1145/2295136.2295140. URL http://doi.acm.org/10.1145/2295136.2295140. (Zitiert auf Seite 24)

[SLG+12] B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, I. Molloy. Android Permissions: A Perspective Combining Risks and Benefits. In *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*, SACMAT '12, S. 13–22. ACM, New York, NY, USA, 2012. doi:10.1145/2295136.2295141. URL http://doi.acm.org/10.1145/2295136.2295141. (Zitiert auf den Seiten 26, 44, 81 und 82)

[SM13] C. Stach, B. Mitschang. Privacy Management for Mobile Platforms – A Review of Concepts and Approaches. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, Band 1, S. 305–313. 2013. doi:10.1109/MDM.2013.45. (Zitiert auf den Seiten 10 und 24)

[SM+14] C. Stach, B. Mitschang, et al. Design and Implementation of the Privacy Management Platform. In -. Auenwald: IEEE Computer Society Conference Publishing Services, 2014. (Zitiert auf den Seiten 10 und 24)

[ST14] M. Sun, G. Tan. NativeGuard: protecting android applications from third-party native libraries. In *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*, S. 165–176. ACM, 2014. (Zitiert auf Seite 24)

[Sta13a] C. Stach. How to Assure Privacy on Android Phones and Devices? In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, Band 1, S. 350–352. 2013. doi:10.1109/MDM.2013.54. (Zitiert auf den Seiten 10 und 24)

[Sta13b] C. Stach. Wie funktioniert Datenschutz auf Mobilplattformen? In *GI-Jahrestagung*, S. 2072–2086. 2013. (Zitiert auf den Seiten 10 und 24)

[Usc13] B. Uscilowski. Mobile Adware and Malware Analysis, 2013. (Zitiert auf den Seiten 10, 48 und 64)

[XSA12] R. Xu, H. Saïdi, R. Anderson. Aurasium: Practical Policy Enforcement for Android Applications. In *USENIX Security Symposium*, S. 539–552. 2012. (Zitiert auf Seite 23)

[YH08] Z. Yan, S. Holtmanns. Trust modeling and management: from social trust to digital trust. *Computer security, privacy and politics: current issues, challenges and solutions*, S. 290–323, 2008. (Zitiert auf den Seiten 15 und 17)

[YLNY13] Z. Yan, C. Liu, V. Niemi, G. Yu. Exploring the Impact of Trust Information Visualization on Mobile Application Usage. *Personal Ubiquitous Comput.*, 17(6):1295–1313, 2013. doi: 10.1007/s00779-013-0636-4. URL http://dx.doi.org/10.1007/s00779-013-0636-4. (Zitiert auf den Seiten 11, 27, 30, 34, 41, 56 und 72)

[YZD12] Z. Yan, P. Zhang, R. Deng. TruBeRepec: a trust-behavior-based reputation and recommender system for mobile applicationsnull. *Personal and Ubiquitous Computing*, 16(5):485–506, 2012. doi:10.1007/s00779-011-0420-2. URL http://dx.doi.org/10.1007/s00779-011-0420-2. (Zitiert auf den Seiten 17 und 19)

[ZXMX13] Y. Zhongyang, Z. Xin, B. Mao, L. Xie. DroidAlarm: An All-sided Static Analysis Tool for Android Privilege-escalation Malware. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS '13, S. 353–358. ACM, New York, NY, USA, 2013. doi:10.1145/2484313.2484359. URL http://doi.acm.org/10.1145/2484313.2484359. (Zitiert auf Seite 26)

[ZZJN12] W. Zhou, Y. Zhou, X. Jiang, P. Ning. Detecting Repackaged Smartphone Applications in Third-party Android Marketplaces. In *Proceedings of the Second ACM Conference on Data and Application Security and Privacy*, CODASPY '12, S. 317–326. ACM, New York, NY, USA, 2012. doi:10.1145/2133601.2133640. URL http://doi.acm.org/10.1145/2133601.2133640. (Zitiert auf Seite 27)

Alle URLs wurden zuletzt am 04. 08. 2014 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift