Risikobasierte Metapolitiken für spontane Kooperationen

Marius Schlegel

Technische Universität Ilmenau
marius.schlegel@tu-ilmenau.de

Zusammenfassung

Mobile Geräte haben sich in den letzten Jahren zu universellen, leistungsfähigen Werkzeugen entwickelt. Zum Schutz vertraulicher Informationen fokussieren sich aktuelle Technologien für mobile Plattformen auf politikorientierte Ansätze. Aufgrund der Dynamik mobiler Systeme werden die eingesetzten Sicherheitspolitiken neben klassischen Zugriffsrechten oft basierend auf externen, kontextuellen sowie lokalen Risiken formuliert, die das Verletzen von Sicherheitsanforderungen beschreiben. Beim Einsatz in zahlreichen Anwendungsszenarien kooperieren mobile Geräte zunehmend in temporär, sporadisch und spontan geformten Gruppen. Hierbei führen die verschiedenen Sicherheitsziele und -risiken der Kooperationsteilnehmer oftmals zu Konflikten zwischen den involvierten Politiken und somit zu gegensätzlichen Zugriffsentscheidungen. Dieses Papier stellt einen Ansatz zur Lösung von Politikkonflikten in spontanen Kooperationen mittels risikobasierter Metapolitiken vor. Deren Konstruktion beruht auf der regelgesteuerten Komposition der risikobasierten Sicherheitspolitiken und der zugehörigen Sicherheitsdomänen der einzelnen Gruppenteilnehmer. Die praktische Umsetzbarkeit der entwickelten Methode zeigt eine Implementierungsstudie unter Verwendung gegenwärtig verfügbarer Technologien.

1 Einleitung

In den letzten Jahren haben sich mobile Geräte, wie Smartphones und Tablets, zu universellen Werkzeugen entwickelt. Beim Einsatz in Unternehmen, Regierungsinstitutionen sowie zivilen und militärischen Organisationen speichern die Geräte zwangsläufig vertrauliche Informationen, beispielsweise interne Unternehmensdaten, Passwörter, kryptographische Schlüssel oder Positionsdaten. Die Wahrung der Sicherheitseigenschaften Vertraulichkeit, Integrität und Verfügbarkeit dieser Informationen ist meist von entscheidender Bedeutung.

Zur Durchsetzung komplexer Sicherheitsanforderungen verwenden IT-Systeme bereits seit mehr als zwei Jahrzehnten *Sicherheitspolitiken*, welche präzise Regeln zum Schutz von Informationen definieren. Zudem wurden Methoden und Technologien entwickelt, um Sicherheitspolitiken zu modellieren, zu analysieren, zu spezifizieren [ZaMa04, JLTW+08, AmKP14] und direkt in politikgesteuerten Betriebssystemen durchzusetzen [LoSm01, WFMV03, Fade06]. Dieses technologische Fundament politikorientierter Ansätze wird derzeit für mobile Plattformen adaptiert [RCCF12, SmCr13, BuHS13, RALB14, AmCD15].

Die Dynamik in mobilen Systemen erfordert im Gegensatz zu stationären IT-Systemen, Sicherheitspolitiken an die Umgebung und gegenwärtige Situation anzupassen, um Zugriffe auf Ressourcen nur dann zu gewähren, wenn ein akzeptabel niedriges Risiko vorliegt. Hierfür ermöglichen die Paradigmen risikobasierter Zugriffssteuerung, Politiken nicht nur durch klassische Zugriffsrechte, sondern zusätzlich basierend auf Risiken zu formulieren [JAS04, Chou05, LuKa11].

P. Schartner (Hrsg.) · D•A•CH Security 2016 · syssec (2016) 188-201.

Risiken können externe, kontextuelle Faktoren (z. B. Position, Umweltfaktoren) sowie lokale Faktoren (z. B. Nutzerverhalten, Gerätemerkmale) einbeziehen. Je nach deren Bewertung können aufgrund eines nicht tolerierbar hohen Gesamtrisikos Zugriffe verweigert (z. B. der Zugriff auf hochvertrauliche Strategiepapiere außerhalb der Unternehmensräume) oder zusätzliche Berechtigungen gewährt werden (z. B. die Delegation von Rechten an niedriger privilegierte Hierarchieebenen im Katastrophenfall), um im Extremfall Gefahren von Menschenleben abzuwenden.

Die Leistungsfähigkeit gegenwärtiger mobiler Geräte befördert deren Einsatz in sensiblen und kritischen Anwendungsszenarien, beispielsweise als Hilfsmittel zur Katastrophenkoordination und -bewältigung [ReLP14]. Dabei kooperieren die Geräte oftmals innerhalb temporär, sporadisch und spontan hergestellten Gruppen in hochdynamischen Umgebungen unter teilweise nicht vorhersehbaren Bedingungen. Diese Umstände verlangen weitaus flexiblere Paradigmen als jene der klassischen politikbasierten Zugriffssteuerung, um Widersprüche zwischen Zugriffsentscheidungen bezüglich geteilter Informationen zu lösen, einen bestmöglichen Kompromiss zwischen Vertraulichkeit und Verfügbarkeit dieser Informationen zu erreichen und damit die Sicherheitsanforderungen einer Kooperationsgruppe gesamtheitlich zu wahren.

Zur Lösung derartiger Probleme werden *Metapolitiken* eingesetzt, welche potentiell in Konflikt stehende Sicherheitspolitiken mittels eigener Regeln koordinieren. Es existieren bereits Methoden, um Metapolitiken ad hoc aus den individuellen Domänen und Politiken der in einer Kooperation involvierten Parteien zu erzeugen [AmKü13, AmKü15]. Sind die einzelnen Sicherheitspolitiken jedoch risikobasiert, stoßen diese Methoden an ihre Grenzen, denn es ist nicht möglich, Risiken auf Metapolitikebene einfließen zu lassen, Risikofaktoren verschiedener Politiken gegeneinander abzuwägen und so die finale Zugriffsentscheidung zu beeinflussen. Die vorliegende Arbeit widmet sich der Lösung dieser Problemstellung.

Ziel dieses Papiers ist es, Risiken explizit in Metapolitiken zu berücksichtigen, um die Risikofaktoren verschiedener Politiken abwägen zu können. Das vorliegende Papier leistet dazu die folgenden Beiträge: (1) die Ergänzung von Metapolitiken um Risikoausdrucksmittel, (2) die Entwicklung einer Methode zur Konstruktion risikobasierter Metapolitiken und (3) die Erarbeitung einer Implementierungsstudie auf Basis gegenwärtiger Technologien, um die praktische Umsetzbarkeit der entwickelten Methode zu zeigen.

2 Ein Beispielszenario

Der vorliegende Abschnitt beschreibt beispielhaft ein Szenario, das die Notwendigkeit der Berücksichtigung von Risiken bzw. Risikofaktoren in Metapolitiken verdeutlicht. Das Szenario orientiert sich an praktischen Gegebenheiten von Ölbohrplattformen [FlSS96, RRFG06, Wip13]. Ausschnitte des Szenarios werden im weiteren Papier referenziert.

Auf einer Ölbohrplattform kooperieren für den Betrieb der *Oilrig Installation Manager* (OIM), Techniker und das zentrale Plattformmanagementsystem, die *Rig Management Environment* (RME), miteinander. Jeder Kooperationsteilnehmer besitzt zur Wahrung individueller Sicherheitsanforderungen eine eigene Sicherheitspolitik. Die RME verwaltet neben Förderungsinformationen auch hochvertrauliche Informationen, z. B. die Zugangssicherungen verschiedener Plattformbereiche und die Konstruktionspläne der Plattform. Hierbei ist die konstruktiv schwächste Stelle eine besonders kritische Information. Jegliche Zugriffe auf solche geteilte Ressourcen werden durch die gemeinsame Metapolitik der Kooperationsgruppe reguliert.

Die Situation verschärft sich plötzlich: im Kontrollraum brechen Flammen aus, zusätzlich bedrohen Erdbeben und Flutwellen die Statik der Plattform. Um die Auswirkungen schnellstmöglich zu mindern, wurden die auf der Plattform befindlichen Rettungskräfte verständigt sowie ein Notruf nach externen Hilfskräften zur Flammenbekämpfung und Evakuierung abgesetzt. Mit deren Eintreffen werden diese in die Gruppe aufgenommen und deren eigene Politiken und Zugriffsregeln für das Teilen notwendiger Krisenmanagementinformationen (z. B. Positionsdaten, Ausstattungsmerkmale, Kamerasichten) in die Metapolitik der Kooperationsgruppe integriert.

Unter den Hilfskräften befinden sich Ingenieure, die zur Ermittlung der konkreten Problemstellen Simulationen durchführen müssen. Dies erfordert Lesezugriff auf die hochvertraulichen Konstruktionspläne, der jedoch den Ingenieuren durch die Zugriffsregeln der RME verweigert wird und nur dem OIM gewährt ist. Der OIM ist durch seine Qualifikation zur Koordination in Krisensituationen berechtigt, Informationen zielführend sowie nach Notwendigkeiten zu teilen, und möchte den Zugriff mittels eigener Regeln gewähren. Durch die widersprüchlichen Entscheidungen liegt hier ein Konflikt zwischen den Politiken von RME und OIM vor. Wird der Zugriff verweigert, die Vertraulichkeit der Konstruktionspläne gewahrt und das Risiko eines Blowouts hingenommen *oder* wird der Zugriff gewährt und das Risiko der Verletzung der Vertraulichkeit der Pläne eingegangen, um die Plattform und Menschenleben zu retten?

Im Szenario sind die Sicherheitspolitiken der Kooperationsteilnehmer basierend auf Risiken formuliert, auf deren Grundlage adaptiv Zugriffsberechtigungen gewährt werden. Folglich ist für die Lösung der beschriebenen Politikkonflikte eine spezielle Klasse von Metapolitiken, welche für präzise Zugriffsentscheidungen Risikofaktoren gegeneinander abwägen und bewerten, notwendig: *risikobasierte Metapolitiken*. Im Szenario ließe sich dann die Gefahr für Menschenleben und Umwelt durch eine risikobasierte Metapolitik wesentlich höher gewichten als die temporäre Verletzung der Vertraulichkeit der Konstruktionspläne und damit Risikofaktor b (Safety-Aspekt) gegenüber Risikofaktor a (Security-Aspekt) priorisieren. Anzumerken ist, dass sich das Abwägen von Risikofaktoren nicht auf „Safety vs. Security“ verallgemeinern lässt.

3 Methode

Dieses Papier stellt eine Methode zur Spezifikation und Konstruktion risikobasierter Metapolitiken für politikkontrollierte spontane Kooperationen vor. Dafür werden Arbeiten von Amthor und Kühnhauser [AmKü13, AmKü15] zur regelgesteuerten Erzeugung von Metapolitiken aufgegriffen und zu einer auf die Paradigmen risikobasierter Zugriffssteuerung spezialisierten Methode weiterentwickelt. Die Methode beruht auf der Komposition risikobasierter Sicherheitspolitiken und den durch diese kontrollierten Sicherheitsdomänen anhand spezifizierter algebraischer Regeln. Die Kompositionsregeln sind durch eine einfache, universelle Politikalgebra beschrieben, wobei die Grundmenge die Menge der involvierten Sicherheitspolitiken ist und die Operationen die Politiken zur Metapolitik kombinieren. Im Ergebnis ermöglicht der Ansatz eine automatisierte, effiziente und transparente Generierung risikobasierter Metapolitiken ohne Nutzerinteraktion.

Bei der Herstellung einer spontanen Kooperation zwischen mobilen Geräten werden die der Trusted Computing Base jedes Systems zugehörigen Regeln untereinander ausgetauscht und jeweils lokal für die Komposition zur Metapolitik angewandt. Außerdem wird eine Sicherheitsdomäne erzeugt, die die Gruppenmitglieder und deren Entitäten (z. B. Anwendungen, Datenobjekte und Ressourcen) umfasst, und die resultierende Metapolitik dieser Domäne zugeordnet. Beispielsweise könnten die Kompositionsregeln beschreiben, dass für jegliche Zugriffsaktionen, alle Politiken zustimmen müssen und damit ein für alle Parteien akzeptables Risiko vorliegen muss.

Zusammenfassend ist es die Kernidee, die Politikkompositionsregeln einmalig durch qualifiziertes Personal, z. B. den IT-Sicherheitsadministrator eines Unternehmens, zu implementieren, in die Sicherheitsarchitektur der mobilen IT-Systeme zu integrieren und jedes Mal bei der Bildung einer spontanen Kooperationsgruppe anzuwenden. Die Auswertung dieser Regeln erzeugt dann die risikobasierte Metapolitik, die mit der gruppenzugehörigen Sicherheitsdomäne assoziiert wird und für jegliche Zugriffsentscheidungen innerhalb dieser Domäne herangezogen wird.

Die folgenden Abschnitte erläutern die Algebra zur Beschreibung der Kompositionsregeln und die Methode zur Konstruktion risikobasierter Metapolitiken. Anschließend werden die Ergebnisse auf das erläuterte Beispielszenario angewendet, um die Problemstellung des Szenarios zu lösen.

3.1 Kompositionsalgebra für risikobasierte Metapolitiken

Die Kompositionsalgebra definiert die Semantik der Politikkompositionsregeln. Deren Elemente sind Sicherheitspolitiken und Operationen, welche präzise beschreiben, wie diese Politiken für die Bildung einer Metapolitik kombiniert werden. Der vorliegende Ansatz zur algebraischen Konstruktion risikobasierter Metapolitiken legt den Fokus auf eine spezielle Klasse von Sicherheitspolitiken, und zwar auf Zugriffssteuerungspolitiken. Diese Klasse von Politiken ist universell einsetzbar zur Wahrung von Vertraulichkeit (Kontrolle von Lesezugriffen), Integrität (Kontrolle von Schreibzugriffen) und Verfügbarkeit (Kontrolle von Zugriffen auf Ressourcen).

Zugriffssteuerungspolitiken definieren üblicherweise eine *Zugriffssteuerungsfunktion* f und einen Zuständigkeitsbereich, ihre *Politikdomäne*, die durch eine von ihnen kontrollierte Menge von Entitäten E repräsentiert wird. Die Zugriffssteuerungsfunktion f_P einer Politik P ist eine Abbildung $f_P : {E_P}^n \times A_P \rightarrow \{true, false\}$ [Lamp74]. Eine Aktion $a \in A_P$ unter n Entitäten $e_i \in E_P$ ist genau dann erlaubt, wenn $f_P(\bar{e}, a)$ auf *true* abbildet bzw. als zutreffend ausgewertet wird. Es gilt $\bar{e} = (e_i)_{i \in \{1,\ldots,n\}} = (e_1, \ldots, e_n) \in {E_P}^n$ mit $n \in \mathbb{N}$. Damit beschreibt f_P vollständig und präzise die Zugriffsentscheidung einer Politik. In heutigen Betriebssystemen und Datenbankmanagementsystemen sind Zugriffssteuerungsfunktionen beispielsweise durch Zugriffssteuerungslisten oder Capability-Listen implementiert.

Diese Darstellung ist für risikobasierte Zugriffssteuerungspolitiken jedoch noch nicht ausreichend, denn oftmals stehen Risikofaktoren verschiedener Politiken in einer sich gegenseitig ausschließenden Beziehung zueinander (z. B. Vertraulichkeit vs. situationsbedingte Verfügbarkeit), sodass Konflikte zwischen Politiken resultieren können. Um diese Konflikte zu lösen, einen Faktor gegenüber einem anderen priorisieren zu können und eine Gewichtung der letztlichen Zugriffsentscheidungen zu erreichen, müssen die bisher politiklokal ausgewerteten Risikofaktoren neben den resultierenden, binären Zugriffsentscheidungen auf Ebene der Metapolitik repräsentiert und komponiert werden.

Eine risikobasierte Zugriffssteuerungspolitik wird daher nicht ausschließlich durch eine Zugriffssteuerungsfunktion, sondern zusätzlich durch eine zweite Komponente zum Ausdruck ihrer Risikofaktoren in Form der *Risikofunktion* repräsentiert. Die Risikofunktion r_P einer risikobasierten Zugriffssteuerungspolitik P ist eine Abbildung $r_P : {E_P}^n \times A_P \rightarrow [0,1]$. Das Risiko des Gewährens einer Aktion $a \in A_P$ unter n Entitäten $e_i \in E_P$ wird damit anhand des Wertes $r_P(\bar{e}, a) \in [0,1]$, dem *Risikowert*, beschrieben. Es wird „0“ als niedrigstes Risiko und „1“ als höchstes Risiko definiert; Safety-Risiken werden dann für die Vergleichbarkeit mit Security-Risiken immer negiert aufgeführt. Die Art und Weise, wie Risiken lokal ermittelt und bewertet werden, ist hochgradig abhängig vom Anwendungsszenario und zudem für die Politikkomposition aufgrund der präzise definierten Schnittstellen nicht relevant.

Risikofunktionen bilden auf reelle Werte im Einheitsintervall ab, jedoch sind Zugriffsentscheidungen binäre Werte. Die Auswertung von Risikofunktionen erfolgt üblicherweise innerhalb von Zugriffssteuerungsfunktionen. Der ermittelte Risikowert wird dabei mit einem Schwellwert verglichen, anhand dem zwischen tolerierbar niedrigem und nicht tolerierbar hohem Risiko differenziert und so entschieden wird, ob ein Zugriff letztlich gewährt oder verweigert wird. Denkbar ist, dass der Schwellwert durch den Gruppenleiter festgelegt wird, aus einzelnen Schwellwerten der Politiken entsprechend einer Regel gebildet wird oder aus einer Abstimmung hervor geht.

Die Idee der Methode ist es, risikobasierte Zugriffssteuerungspolitiken durch zwei Komponenten darzustellen: eine Zugriffssteuerungsfunktion und eine Risikofunktion. Da die Abbildungen jeweils für unterschiedliche Zielmengen definiert sind, müssen diese auch auf verschiedene Weise verknüpft werden. Die Kompositionsalgebra für risikobasierte Metapolitiken wird in zwei Teilalgebren aufteilt, deren Operatoren auf den Grundmengen $\mathscr{P}_f$ bzw. $\mathscr{P}_r$ definiert sind. Jede Politik P_i ist Element beider Grundmengen und wird in $\mathscr{P}_f$ durch die Zugriffssteuerungsfunktion f_{P_i} bzw. in $\mathscr{P}_r$ durch die Risikofunktion r_{P_i} dargestellt. Zwar sind beide Grundmengen gemäß den Bezeichnern nicht disjunkt, jedoch vereinfachen einheitliche Politikbezeichner die Schreibweise sowie Übersichtlichkeit der Spezifikation und beseitigen somit potentielle Fehlerquellen.

Zur Spezifikation der Regeln für die Komposition von Zugriffssteuerungsfunktionen existiert bereits eine Spezifikationsmöglichkeit mit an die boolesche Algebra angelehnten Operatoren zur Konjunktion, Disjunktion, Negation und Selektion [AmKü13, AmKü15]; diese werden für den ersten Teil der Methode übernommen. Es ist jedoch zu beachten, dass die Kompositionsoperatoren in dieser Arbeit auf der Grundmenge $\mathscr{P}_f$ definiert sind. Aus Platzgründen wird exemplarisch der Konjunktionsoperator „$\wedge_f$“ aufgeführt.

Konjunktion. Eine Politikkomposition, die eine Aktion genau dann erlaubt, wenn alle involvierten Politiken zustimmen, wird mit dem Konjunktionsoperator „$\wedge_f$“ konstruiert: $\wedge_f : \mathscr{P}_f \times \mathscr{P}_f \to \mathscr{P}_f,\ \wedge_f(P_i,P_j) \mapsto P_k,$ wobei $f_{P_k} : {E_{P_k}}^{n_k} \times A_{P_k} \to \{true, false\}$ mit ${E_{P_k}}^{n_k} = {E_{P_i}}^{n_i} \times {E_{P_j}}^{n_j},\ n_k = n_i + n_j,\ A_{P_k} = A_{P_i} \times A_{P_j},\ f_{P_k}(\bar{e}_{P_k}, a_{P_k}) = f_{P_k}((\bar{e}_{P_i}, \bar{e}_{P_j}), (a_{P_i}, a_{P_j})) \mapsto f_{P_i}(\bar{e}_{P_i}, a_{P_i}) \wedge f_{P_j}(\bar{e}_{P_j}, a_{P_j})$.

Für die Spezifikation der Regeln zur Komposition von Risikofunktionen bietet die Teilalgebra auf der Grundmenge $\mathscr{P}_r$ analoge Operatoren zur Konjunktion, Disjunktion, Negation und Selektion. Die Operatoren orientieren sich an denen der Fuzzy-Logik, einer Verallgemeinerung der zweiwertigen booleschen Logik. Die Risiko-Konjunktion beschreibt das Mindestrisiko zweier Politiken anhand des Risikofunktionsminimums und die Risiko-Disjunktion das höchstens vorliegende Risiko anhand des Maximums. Die Risiko-Negation bildet das Komplement des Risikofunktionswertes einer Politik. Die Risiko-Selektion ermöglicht eine „If-then-else“-Selektion mittels relativer Risikobewertung. Exemplarisch wird der Risiko-Konjunktionsoperator „$\wedge_r$“ definiert.

Risiko-Konjunktion. Die Priorisierung des Risikofaktors einer Politik gegenüber dem Risikofaktor einer anderen Politik anhand des niedrigeren Risikowertes beider Risikofunktionen wird durch die Risiko-Konjunktionsoperator „$\wedge_r$“ erreicht: $\wedge_r : \mathscr{P}_r \times \mathscr{P}_r \to \mathscr{P}_r,\ \wedge_r(P_i,P_j) \mapsto P_k,$ wobei $r_{P_k} = \min(r_{P_i}, r_{P_j})$ mit der wie oben definierten Domäne.

In Kombination der genannten Operatoren mit den Gesetzmäßigkeiten der Kommutativität und Assoziativität und dem Existenzquantor sowie dem Allquantor bietet die Kompositionsalgebra mit den beiden Teilalgebren ein flexibles und ausdrucksstarkes Rahmenwerk zur Beschreibung der Komposition von Zugriffssteuerungsfunktionen und Risikofunktionen für die regelgesteuerte und automatische Konstruktion risikobasierter Metapolitiken mittels algebraischer Ausdrücke.

3.2 Konstruktion risikobasierter Metapolitiken

Die Konstruktion einer risikobasierten Metapolitik als gemeinsame Sicherheitspolitik einer Kooperationsgruppe kann je nach Art der Kooperation flexibel erfolgen. Beispielsweise können gleichberechtigte Teilnehmer jeweils eigene Konstruktionsvorgaben einbringen. In hierarchisch strukturierten Gruppen kann auch eine verbindliche Einführung einer Konstruktionsvorschrift durch einen Gruppenleiter erfolgen. In wiederum anderen Anwendungsszenarien müssen Abstimmungen über die für die Gruppe gültigen Konstruktionsregeln durchgeführt werden. Folglich ist es nicht sinnvoll, eine statische und universelle Strategie zur Bildung von Gruppenpolitiken anzuwenden. Vielmehr geht der Ansatz von einer organisierten Offenheit aus, in der jedes Gruppenmitglied eine eigene Strategie zur Bildung der Metapolitik und ihrer Domäne spezifiziert.

In jeder spontanen Kooperation steuern die Parteien eigene Informationen für die Komposition zur Metapolitik und ihrer Sicherheitsdomäne bei. Die für die Konstruktion notwendigen Informationen werden durch Tupel der Form (C,P,D,t) beschrieben, bestehend aus

1. einem algebraischen Ausdruck $C = (C_f, C_r)$, dem Konstruktor, wobei C_f die Komposition von Zugriffssteuerungsfunktionen beschreibt (mittels Teilalgebra auf $\mathscr{P}_f$) und C_r die Komposition von Risikofunktionen darlegt (mittels Teilalgebra auf $\mathscr{P}_r$),
2. einer Zugriffssteuerungspolitik P, die durch eine Zugriffssteuerungsfunktion f_P und eine Risikofunktion r_P formuliert wird,
3. einer Menge von Entitäten D, die der Domäne der Gruppe beigefügt wird, und
4. einem Schwellwert t zur Risikobewertung, anhand dessen zwischen tolerierbar niedrigem und nicht tolerierbar hohem Risiko differenziert wird.

Die Konstruktion der Metapolitik P_G aus den einzelnen Tupeln (C_i, P_i, D_i, t_i) der Gruppenmitglieder $i \in G$ (G enthält die Menge aller Politikbezeichner i) wird wie folgt vorgenommen.

$$P_G = (f_{P_G}, r_{P_G}) = (\bigwedge_{i \in G}{}_f\, C_{f_i}, \bigwedge_{i \in G}{}_r\, C_{r_i}) \quad \text{mit der Domäne} \quad D_G = \bigcup_{i \in G} D_i.$$

Für die Konstruktion des gruppengemeinsamen Schwellwerts zur Risikobewertung t_G existieren, wie für die Gruppenpolitikkonstruktion, verschiedene Möglichkeiten. In Anwendungsszenarien mit Gleichberechtigung können verschiedene Normen, beispielsweise für Mittelwert, Minimum oder Maximum, angewendet werden. In hierarchisch strukturierten Gruppen kann t_G durch den Gruppenleiter festgelegt werden. Des Weiteren ist auch eine Abstimmung über den gruppengemeinsamen Schwellwert denkbar. Aus diesen Gründen ist die Definition des Schwellwerts t_G je nach Anwendungsfall zu bestimmen.

Der Ablauf zur Laufzeit gliedert sich in zwei Phasen: (1) die Gruppenbildungsphase und (2) die Interaktionsphase. In der ersten Phase wird die Bildung der Gruppe, ihrer Metapolitik, ihrer Sicherheitsdomäne und des Schwellwerts zur Risikobewertung anhand der untereinander kommunizierten (C_i, P_i, D_i, t_i)-Tupel jeweils für jede Entität lokal durchgeführt. Die zweite Phase beschreibt das Geschehen während der Interaktion. Für die Entscheidung von Zugriffsanfragen durch die Metapolitik muss jede Politik die im Rahmen dieser Anfrage notwendigen Risikowerte und Zugriffsentscheidungen der anderen Politiken einholen.

Es müssen nicht alle Politiken basierend auf Risiko formuliert sein. Nicht definierte Risikofunktionen und Schwellwerte werden im Falle nicht-risikobasierter Politiken für die Auswertung nicht betrachtet. Um die Risikofunktion der Metapolitik komponieren und im Ergebnis eine risikobasierte Metapolitik erhalten zu können, muss mindestens eine Politik risikobasiert sein.

3.3 Beispielhafte Anwendung

Der vorliegende Abschnitt wendet nun die entwickelte Methode beispielhaft auf die Problemstellung des in Abschnitt 2 skizzierten Beispielszenarios an.

Zentral für den aufgetretenen Konflikt zwischen den risikobasierten Sicherheitspolitiken P_{RME} und P_{OIM} ist die Zugriffsanfrage $access = (Engineers, RigConstructionPlans, view)$, d. h. ob die Ingenieure zur Identifikation und Ermittlung konkreter Problemstellen der durch Flammen und Flutwellen bedrohten Ölbohrplattform auf die hochvertraulichen Konstruktionspläne zugreifen dürfen. Die Politik der RME verweigert zunächst den Zugriff auf die Konstruktionspläne ($f_{P_{RME}}(access) \mapsto false$), denn das Risiko der Verletzung der Vertraulichkeit wird als zu hoch eingestuft (z. B. $r_{P_{RME}}(access) \mapsto 0{,}9$). Für einen erfolgreichen Zugriff müsste das Zugriffsrisiko dem Wert $r_{P_{RME}} \leq 0{,}1$ entsprechen. Die Politik des OIM stellt einen kritischen Plattformzustand fest ($r_{P_{OIM}}(access) \mapsto 0$) und gewährt den Zugriff ($f_{P_{OIM}}(access) \mapsto true$, falls $r_{P_{OIM}}(access) = 0$).

Für die Lösung des Politikkonflikts wird eine risikobasierte Metapolitik mit $f_{P_G} = \bigwedge_f P_i \ \forall i \in G$ und $r_{P_G} = P_{RME} \wedge_r P_{OIM} \bigwedge_r P_i \ \forall i \in G$ formuliert. Die Autonomie aller Teilnehmer wird durch die Konjunktion der Zugriffssteuerungsfunktionen gewahrt. In der Ausnahmesituation findet zudem ein Abwägen der Risikofaktoren statt: mittels konjunktiver Verknüpfung der Risikofunktionen der einzelnen Teilnehmer wird das allen mindestens vorliegende Risiko als gruppengemeinsames Risiko anerkannt ($r_{P_G}(access) = \min((0{,}9), 0) = 0$) und in den Zugriffssteuerungsfunktionen abgefragt. Infolgedessen gilt $f_{P_{RME}}(access) \mapsto true$ (da $r_{P_G}(access) \leq 0{,}1$) und $f_{P_{OIM}}(access) \mapsto true$ (da $r_{P_G}(access) = 0$), sodass der Zugriff aufgrund der Ausnahmesituation letztlich gestattet wird.

4 Implementierungsstudie

Dieser Abschnitt präsentiert ein Konzept zur Implementierung der Methode zur Konstruktion risikobasierter Metapolitiken basierend auf aktuellen Technologien politikkontrollierter mobiler Betriebssysteme und zeigt die praktische Umsetzbarkeit.

Um den Anforderungen Spontanität, Sparsamkeit und Transparenz in den angedachten Anwendungsszenarien gerecht zu werden, hat das Implementierungskonzept die folgenden Ziele: (1) die Auswertung von algebraisch formulierten Kompositionstermen zur Laufzeit, welche auf verteilten Sicherheitspolitiken und -domänen basieren, (2) die Umsetzung der für die politikkontrollierte Zugriffssteuerung grundlegenden Referenzmonitorprinzipien [Ande72], (3) die Minimierung des Ressourcenbedarfs durch geringe Kommunikations- sowie Rechenlaufzeitkosten und (4) die Maximierung der Nutzertransparenz und Anwenderfreundlichkeit durch Vollautomatisierung und geringe Nutzerinteraktion (lediglich für die Gruppenbildung).

Der folgende Abschnitt gibt einen Überblick über die in der Implementierungsstudie verwendeten Technologien. Darauf aufbauend wird die Gesamtarchitektur beschrieben, wobei insbesondere das zugrunde liegende Architektur- und Kommunikationsmodell erläutert sowie die Kernelemente einer prototypischen Implementierung (Konstruktoren, Sicherheitspolitiken, Sicherheitsdomänen, Risikoschwellwerte und Gruppenbildung) behandelt werden.

4.1 Verwendete Technologien

Das Implementierungskonzept verwendet das Betriebssystem *Android* für mobile Geräte aufgrund seiner hohen Verbreitung, Quellenoffenheit und guten Dokumentation. Zudem wurden in den letzten Jahren eine Reihe von Ansätzen entwickelt, um Sicherheitspolitiken in Android

zu integrieren und durchzusetzen, wie z. B. MOSES [RCCF12], SEAndroid [SmCr13], FlaskDroid [BuHS13], DROIDFORCE [RALB14] und ANDRABEN [AmCD15]. Aus Gründen wie der guten Verfügbarkeit und bestehender praktischer Erfahrung fällt die Wahl auf *MOSES*.

Android basiert auf einem modifizierten Linux-Kernel. Dieser bietet grundlegende Abstraktionen der Hardwareressourcen (Netzwerkschnittstellen, Speichermanagement, Gerätetreiber etc.). Die Android-API wird durch eine auf dem Kernel aufbauende Abstraktionsschicht bereitgestellt, die Android-Middleware, welche sich aus einem Java-basierten Anwendungsframework, C-basierten nativen Bibliotheken und der Android-Laufzeitumgebung zusammensetzt. Der Hauptzweck der Android-Middleware ist es, DEX-Bytecode der Anwendungen zu interpretieren. Dazu läuft jede Anwendung in einer eigenen Instanz der *Dalvik Virtual Machine* (DVM) als separater Prozess innerhalb eines eigenen virtuellen Adressraumes[1]. Dieser leichtgewichtige Virtualisierungsansatz stellt sicher, dass einzelne Anwendungen zur Laufzeit voneinander isoliert sind.

Für die kontrollierte Kommunikation von Prozessen bietet Android verschiedene Mechanismen zur Interprozesskommunikation, welche vom Kernel bereitgestellt werden. Aus Anwendungssicht werden diese durch die Middleware in Form einer allgemeinen Schnittstelle abstrahiert, welche als *Content Provider* bezeichnet wird. Unter Verwendung eines Content Providers können Anwendungen explizit eine Schnittstelle zu jeglichen Daten innerhalb ihres Adressraums spezifizieren, die dann von anderen Anwendungen verwendet werden kann. Außerdem bietet die Middleware auf Anwendungsebene mittels den *Android Core Libraries* (*libcore*) die Möglichkeit, die Kernel-API für Zugriffe auf das Dateisystem oder die Netzwerkschnittstellen zu verwenden.

Das MOSES-Framework (*MOde-of-uses SEparation for Smartphones*) [RCCF12] ist eine Implementierung politikbasierter Zugriffssteuerungsmechanismen für die Android-Middleware. Das Ziel des Ansatzes ist es, mehrere Domänen für verschiedene Einsatzszenarien, die auf einem einzelnen Gerät koexistieren, kontextbasiert voneinander zu isolieren. Ein geläufiges Anwendungsbeispiel ist die zunehmende Nutzung von privaten mobilen Geräten im Arbeitsumfeld; oftmals mit *Bring your own device* (BYOD) bezeichnet. Dafür ist es essentiell, dass Daten und Anwendungen streng zwischen den Domänen „Privat“ und „Arbeit“ getrennt werden. Innerhalb dieser Domänen werden jegliche Zugriffe auf die Daten und die Kommunikation mit Anwendungen anderer Domänen durch eine benutzerdefinierte Sicherheitspolitik reguliert, die im BYOD-Umfeld durch den IT-Sicherheitsadministrator des Unternehmens verwaltet wird.

In MOSES werden Sicherheitsdomänen als *Security Profiles* (SP) realisiert, welche im Wesentlichen eine Menge von Anwendungen (z. B. Kontakt- und Adressbuch, Texteditor), Ressourcen (z. B. Kontakte, Dateien) und diese kontrollierende Sicherheitspolitiken repräsentieren. Letztere spezifizieren Zugriffssteuerungsregeln für die jeweilige Domäne, vergleichbar mit Zugriffssteuerungsfunktionen. Aufgrund der MOSES zugrunde liegenden Motivation werden lediglich lokale Sicherheitsdomänen und Zugriffssteuerungspolitiken unterstützt. Die Reinterpretation des Domänenkonzeptes von MOSES und eine Reihe an Modifikationen ermöglichen die Komposition verteilter Politiken und Domänen [AmKü15]. Diese angepasste MOSES-Architektur (Arbeitstitel *SpontAndroid*) wird in dieser Arbeit entsprechend erweitert.

4.2 Implementierung der Politikkomposition

Dieser Abschnitt behandelt aufbauend auf den vorgestellten Technologien das Konzept zur Implementierung risikobasierter Metapolitiken und ihrer Konstruktionsmethode auf Architekturniveau.

[1] Mit Android 5.0 wurde Dalvik (*Just-in-Time*-Compiler) durch den *Ahead-of-Time*-Compiler Android Runtime (ART) ersetzt. Da die aktuelle Version von MOSES auf Android 2.3.4_r1 aufbaut, ist dieser nicht weiter relevant.

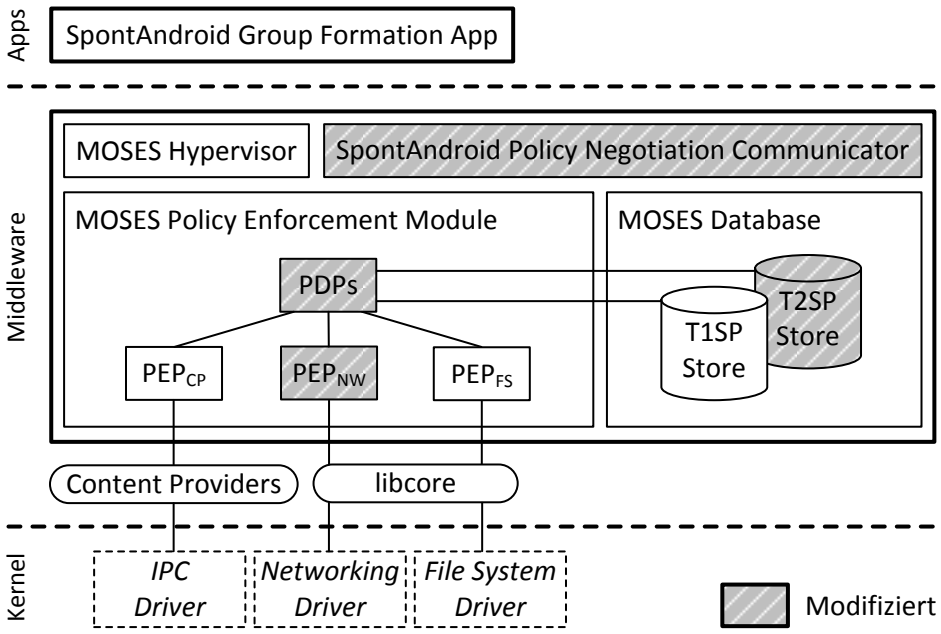

Abb. 1: Angepasste MOSES/SpontAndroid-Architektur

Zunächst werden das Kommunikationsmodell sowie die grundlegende Softwarearchitektur im Kontext des modifizierten MOSES-Frameworks beschrieben. Dabei wird insbesondere herausgestellt, welche Erweiterungen zusätzlich für die entwickelte Methode notwendig sind.

Kommunikationsmodell. Aus technischer Sicht führt der verfolgte Ansatz zu einer vollvermaschten Kommunikationstopologie, denn die Dezentralisierung erfordert die Kommunikation aller Teilnehmer untereinander. Die Kommunikation der beteiligten Teilnehmerknoten (hier Android-Geräte) teilt sich in zwei Phasen: (1) Gruppenbildung und (2) Interaktion. In der Gruppenbildungsphase tauschen alle Knoten Nachrichten der Form $(C_i, P_i, D_i, t_i)\ \forall i \in G$ als Konstruktoren via Broadcast aus. Für die Auswertung der verteilten risikobasierten Metapolitik im Rahmen von Zugriffsanfragen (Interaktionsphase), muss der anfragende Knoten $i \in G$ zunächst alle Risikowerte $r_{P_j}\ \forall j \neq i\ (j \in G)$ einholen. Dies ist notwendig, weil das Ergebnis der Komposition der Risikofunktionen die Grundlage für die Auswertung der Zugriffssteuerungsfunktionen ist. Der gruppengemeinsame Risikowert r_{P_G} wird dann den anderen Kooperationsteilnehmern zur Verfügung gestellt. Anschließend trägt der anfragende Knoten i die Voten aller anderen Teilnehmer f_{P_j} zusammen, um die gruppengesamte Zugriffsentscheidung P_G komponieren zu können. Für nichtlokale Politiken werden Stub-Objekte als Vertreterobjekte eingesetzt, die eine sichere Kommunikationsschnittstelle für Politikanfragen bereitstellen (z. B. mittels eines neu erzeugten, symmetrischen Sitzungsschlüssels). Auf diese Weise liegen die eigentlichen Politiken in der geschützten Umgebung ihres zugehörigen physischen Hostgeräts, das ebenfalls alle notwendigen Schnittstellen und Ressourcen für die lokalen Zugriffsentscheidungsmechanismen bietet.

Architekturmodell. Die Softwarearchitektur orientiert sich eng an der Originalarchitektur von MOSES und den Erweiterungen durch SpontAndroid. Die notwendigen Änderungen für risikobasierte Metapolitiken sind in Abbildung 1 dargestellt. In Anlehnung an [AmKü15] werden die MOSES zugehörigen Komponenten und Mechanismen mit *Tier-1* (T1) und alle zusätzlichen Komponenten und Mechanismen mit *Tier-2* (T2) bezeichnet. Während T1-Zugriffssteuerung lokal implementierte Mechanismen basierend auf MOSES Security Profiles repräsentiert, bezieht

sich T2-Zugriffssteuerung auf die Mechanismen für die verteilte Metapolitikenkomposition und -auswertung. In beiden Fällen erfolgt die Implementierung der Politiken (*Policy Decision Points*, PDPs) und der Mechanismen zu ihrer Durchsetzung (*Policy Enforcement Points*, PEPs) getrennt.

Für die Verwaltung von verteilten Domänen zusätzlich zu den lokalen MOSES-Domänen wurden in [AmKü15] *Tier-2 Security Profiles* (T2SP) eingeführt. Diese werden in der Komponente *T2SP Store* implementiert, die im Wesentlichen die Funktionalität des *T1SP Store* spiegelt: während T1SPs weiterhin verwendet werden, um die Isolation zwischen lokalen Sicherheitsdomänen zu erreichen, enthalten T2SPs zusätzlich Informationen über die verteilten externen Sicherheitsdomänen, welche zur Durchsetzung von Metapolitiken genutzt werden. Die Idee ist es, jedem involvierten Gruppenknoten ein T2SP zuzuweisen. Dieses enthält knotenspezifische Informationen über die Kompositionsregeln (C_{f_i} und C_{r_i}), die (risikobasierte) Zugriffssteuerungspolitik mit einer Zugriffssteuerungsfunktion (f_{P_i}) sowie einer Risikofunktion (r_{P_i}), die Entitäten innerhalb der Sicherheitsdomäne (D_i) und den Schwellwert zur Risikobewertung (t_i). Wie bereits erläutert, werden jeweils für einen Knoten nichtlokale Politiken (jede Politik P_j mit $j \neq i$) durch Stub-ähnliche *Policy Interface Objects* (PIOs) im T2SP ersetzt. Für diesen Zweck wurden die internen MOSES-Klassen zum Datenbankmanagement sowie zum Zugriff auf Politiken modifiziert und neue leichtgewichtige Datenstrukturen für Kompositionsregeln, PIOs und Entitätenmengen wie auch separate Schnittstellenmethoden für T1 und T2 hinzugefügt.

Die genannten Komponenten werden durch die Modifikation der existierenden PDPs zu MOSES hinzugefügt. Dabei haben die PDPs ein T1/T2-Dispatcher-Interface und spezialisierte T2-Methoden zur Politikauswertung erhalten. Da T2-Zugriffe lediglich Socket-basierte Netzwerkkommunikation involvieren, werden lediglich der netzwerkbezogene PEP (PEP_{NW}) erweitert, um die notwendigen Argumente zum PDP zu übergeben.

4.2.1 Konstruktoren

Jeder Konstruktor C_i enthält die Kompositionsregeln des Knotens i, wobei zwei separate Konstruktoren für die Zugriffssteuerungsfunktion (C_{f_i}) und die Risikofunktion (C_{r_i}) der Metapolitik vorliegen. Die Kompositionsregeln beider Teilkonstruktoren werden im Wesentlichen durch boolesche Ausdrücke beschrieben. Dafür liegt zunächst eine allgemeinen Klasse zur Repräsentation logischer Ausdrücke erster Stufe vor. Die Logikoperatoren dieser Klasse werden in SpontAndroid durch sog. *Policy Composition Objects* (PCOs) für die Semantik der Kompositionsoperatoren für Zugriffssteuerungsfunktionen reimplementiert. Für risikobasierte Metapolitiken ist hier die Erweiterung um Kompositionsoperatoren für Risikofunktionen notwendig. Dafür gibt es prinzipiell zwei Möglichkeiten: (1) die Änderung und Erweiterung der bestehenden PCO-Klasse um zusätzliche Datenstrukturen und Methoden oder (2) das Hinzufügen einer weiteren Klasse und Trennung in PCO-ACF (*PCO for a Metapolicy's Access Control Function*) und PCO-RF (*PCO for a Metapolicy's Risk Function*). Aus Gründen der Einfachheit (geringfügigere Änderungen) und der Sparsamkeit (kein Austausch zusätzlicher Objekte, geringerer Speicheroverhead) wird die erste Alternative vorgezogen. Für eine einfache Konfiguration können beide Ausdrücke in einer XML-basierten Syntax (vgl. XACML) spezifiziert werden. Jedes PCO stellt mittels der öffentlichen Methoden *evalACF* und *evalRF* Lesezugriff auf den binären Wert bzw. den $[0,1]$-reellen Wert des entsprechenden privaten Kompositionsterms zur Verfügung, dessen Elemente auf PIOs oder andere PCOs referenzieren; beide werden im T2SP Store gespeichert. Da die Auswertung der Kompositionsterme für die Zugriffssteuerungsfunktion der Metapolitik von der vorherigen Auswertung der Risikofunktionskompositionsterme abhängig ist, wird in Knoten mit risikobasierter Zugriffssteuerungspolitik für die *evalACF*-Methode bereits der

Gruppen-Risikowert r_{P_G} als Ergebnis der Risikofunktionskomposition als Parameter benötigt.

Immer wenn eine T2-Zugriffsanfrage am PEP eingeht, evaluiert der PDP im Anschluss die gespeicherten PCOs basierend auf den Kompositionsregeln der Gruppenpolitik. Diese können beliebig nach den Mitteln der Algebra gestaltet werden oder in einer einfachen Implementierung statisch zu $P_G = (f_{P_G}, r_{P_G}) = (\bigwedge_f C_{f_i}, \bigwedge_r C_{r_i})$ mit $i \in G$ gesetzt werden. Der resultierende Wert ist die finale Zugriffsentscheidung f_{P_G} der Metapolitik P_G und wird zur Laufzeit durch die mit den Kompositionsregeln assoziierten PIOs berechnet. Flexiblere Kompositionsregeln können durch das Hinzufügen eines weiteren Interfaces zum PDP implementiert werden.

4.2.2 Sicherheitspolitiken

Die eigene risikobasierte Zugriffssteuerungspolitik jedes Knotens wird lokal im T2SP Store gespeichert. Für alle anderen Gruppenmitglieder werden PIOs anstatt Politikobjekten gespeichert. In SpontAndroid implementieren Politikobjekte dasselbe abstrakte Interface wie PIOs. Analog zur PCO-Klasse wurde auch hier aus Gründen der Einfachheit und Sparsamkeit die geringfügige Änderung der bestehenden Klassen vorgezogen, anstatt zusätzliche Klassen einzuführen.

Politikobjekte besitzen zur Umsetzung der lokalen Zugriffsfunktion f eine leichtgewichtige Matrixdatenstruktur und zur Repräsentation der lokalen Risikofunktion r zusätzliche Datenstrukturen, die der Risikoanalyse und -bewertung dienen und beispielsweise den Einbezug von Sensordaten ermöglichen. Aufgrund der Abhängigkeit vom zugrundeliegenden Anwendungsszenario werden hier keine konkreten Datenstrukturen vorgegeben. PIOs halten dagegen einen Sitzungsschlüssel für die sichere Kommunikation mit ihrem Ursprungsknoten. Dieser Schlüssel wird von der Komponente *Policy Negotiation Communicator* (PNC) verwendet, um ein SSL-Socket aufzubauen, über das die Politikentscheidungen übertragen werden.

Politikobjekte wie auch PIOs bieten jeweils zwei öffentliche Methoden *acf* und *rf*. Die Methode *acf* dient der Auswertung des zugriffssteuerungsbezogenen Teils und erfordert als Eingabeparameter eine Liste von Entitäten, eine Zugriffsaktion und (falls risikobasiert) den aktuellen gruppengemeinsamen Risikowert r_{P_G}, um die Politikentscheidung auf diesen Argumenten als binären Wert zurückzugeben. Die Ausführung der *acf*-Methode ist also (teilweise) von der vorherigen Ausführung der *rf*-Methode auf allen anderen Gruppenknoten abhängig. Die *rf*-Methode wird mit einer Liste von Entitäten und einer Zugriffsaktion parametrisiert und gibt einen reellen Wert im Einheitsintervall als Risikowert zurück. Es ist es anzumerken, dass für jeden Knoten i exakt ein Politikobjekt (inklusive der Zugriffsmatrix und der Datenstrukturen für die Risikobewertung) mit lokal ausgewerteten Methoden *acf* und *rf* sowie exakt ein PIO für jeden anderen Teilnehmerknoten existiert. Infolgedessen werden alle Zugriffsfunktionen f_{P_j} und Risikofunktionen r_{P_j} (jeweils $j \neq i$) der anderen Teilnehmer durch Stubs innerhalb der entsprechenden PIOs repräsentiert, die ihren Wert mittels desselben *acf*- bzw. *rf*-Aufrufs zurückgeben.

Für den Praxiseinsatz sollten PIOs einen Caching-Mechanismus für Politikentscheidungen implementieren (vgl. SELinux *Access Vector Caches*). Um den Netzwerkkommunikationsaufwand zu verringern und jede nichtlokale Politik nicht bei jeder Zugriffsanfrage *on demand* auswerten zu müssen, würde dann auf vorangegangene und im Cache gespeicherte Politikentscheidungen zurückgegriffen. Aber insbesondere bei der Verwendung auf Risiko basierter Zugriffsentscheidungen muss hier eine für die gesamte Kooperationsgruppe vordefinierte Höchstnutzungsdauer eingehalten werden, weil Risikowerte wegen ihrer hohen Dynamik und Kontextabhängigkeit nur begrenzt gültig sein können (Invalidierungssemantik). Um Aktualität, Authentizität und Integrität des gruppengemeinsamen Risikowerts r_{P_G} sicherzustellen, könnte dieser vom Kom-

positeurknoten mit einem Zeitstempel digital signiert werden und von den Empfängerknoten auf hinreichende Aktualität geprüft werden. Je nach Frequenz der Zugriffsfragen und der Komplexität der Risikofunktion kann es sinnvoll sein, für die Risikobewertung notwendige Teile vorzuverarbeiten; dies ist aber stark szenarioabhängig.

4.2.3 Sicherheitsdomänen

Die Implementierung der lokalen Sicherheitsdomäne jedes Gruppenmitglieds ist in SpontAndroid vergleichsweise einfach vorgenommen. Jede Domäne ist als eine Instanz der Standard-*Set*-Klasse realisiert. Deren Elemente sind die Domänenentitäten mit eindeutigen Bezeichnern, die im Rahmen der Metapolitikenkonstruktion vollständig zwischen allen Gruppenmitglieder ausgetauscht (D_i's) und jeweils lokal in einer Namensraum-Datenbank des T2SP Store gespeichert werden. Daneben ordnet die Datenbank jeder Entität die Knoten-ID des entsprechenden Gruppenmitglieds zu, um die Namensauflösung bei verteilten Zugriffsanfragen zu ermöglichen. Bei jeder Zugriffsanfrage prüft die T1/T2-Dispatchermethode des PDP, ob sich die angefragte Ressource innerhalb der Sicherheitsdomäne D_G befindet, um zu ermitteln, ob zuerst eine T2-Politikentscheidung berechnet werden muss. In diesem Fall werden die PCO-Schnittstellen *evalACF*- bzw. *evalRF* verwendet.

4.2.4 Schwellwert zur Risikobewertung

Die initiale Konstruktionsnachricht jedes Knotens enthält neben den bisher erläuterten Komponenten einen lokalen Schwellwert zur Risikobewertung t_i. Dieser wird den anderen Gruppenmitgliedern für die Bildung des gruppengemeinsamen Risikoschwellwerts t_G zur Verfügung gestellt. Ähnlich wie bei den Konstruktoren kann hier eine bereits im mobilen System fest implementierte Regel für die Bildung des gruppengemeinsamen Schwellwerts zur Risikobewertung t_G zuständig sein oder eine Regel durch den Initiator der Kooperationsgruppe vorgegeben werden. Beispielsweise könnte eine solche Regel durch eine einfache Mittelwert-, Maximum- oder Minimumbildung, aber auch vergleichbar mit einem booleschen Ausdruck, formuliert sein. Alternativ kann der Risikoschwellwert direkt durch den Initiator festgelegt sein, wobei dieser einer initialen Statusnachricht oder dem Konstruktor des Initiatorknotens (Gruppenleiter) entnommen wird.

4.2.5 Gruppenbildung

Um eine Metapolitiken-kontrollierte spontane Interaktion vorzubereiten, muss zunächst eine logische Gruppe von Knoten geformt werden. Dieser Prozess wird durch einen Nutzer (z. B. den Gruppenleiter) initiiert, welcher aktiv Gruppenmitglieder wählen kann. Die Funktion wird durch eine einfache App zur Gruppenbildung (*Group Formation App*) implementiert, die es ermöglicht, eine Menge von Hostnamen für die Gruppenbildung einzugeben. Mitgliederknoten können lokal mit einem WLAN verbunden oder per Internet erreichbar sein. Da jegliche per Socket aufgebaute Netzwerkkommunikation nach dem Interzeptor-Prinzip kontrolliert wird (PEP_{NW}), ist das verwendete Kommunikationsprotokoll nicht von der Implementierung des Frameworks abhängig: TCP-Sockets wie auch Bluetooth-Sockets sind Gegenstand des modifizierten PEP. Folglich müsste für eine Bluetooth-basierte Kooperation nur die App zur Gruppenformation angepasst werden (z. B. Erweiterung um Bluetooth-basierte Geräteerkennung und Pairing-Features).

Zuletzt ist für die Implementierung von Metapolitiken anzumerken, dass die korrekte Komposition und Durchsetzung der verteilten Zugriffssteuerungspolitiken dem gegenseitigen Vertrauen in manipulationssichere Referenzmonitore zwischen allen Gruppenmitgliedern unterliegen. Hier

könnte dies durch den Austausch von Integritätszertifikaten auf der implementierten Sicherheitsarchitektur oder, in hochvertraulichen Anwendungsszenarien, durch die Beschränkung der Gruppenmitgliedschaft auf Geräte, die TPM-artige Zertifikate bereitstellen, erreicht werden.

5 Zusammenfassung

Das vorliegende Papier widmete sich dem Problem, dass heutige Metapolitikkonzepte für die Zugriffssteuerung innerhalb temporär, sporadisch und spontan gebildeten Kooperationsgruppen nicht ausreichend sind, da es nicht möglich ist, Risiken neben Berechtigungen auf Metapolitikebene einfließen zu lassen und gegeneinander abzuwägen.

Für die Lösung des Problems erarbeitete dieses Papier eine Methode, welche auf der automatischen, regelgesteuerten Komposition der risikobasierten Sicherheitspolitiken der einzelnen Gruppenteilnehmer sowie ihrer Sicherheitsdomänen beruht. Die Kernidee der Methode ist es, risikobasierte Politiken in zwei Komponenten aufgeteilt darzustellen: eine Zugriffssteuerungsfunktion und eine Risikofunktion. Diese Auffassung ermöglicht die Konstruktion risikobasierter Metapolitiken mittels zweier separater Kompositionsaktivitäten, welche symmetrisch sowie gleichwertig behandelt werden und je nach Grad gegenseitiger Abhängigkeit der Auswertung nebenläufig ausführbar sind. Die Komposition wird jeweils anhand spezifizierter algebraischer Regelausdrücke vorgenommen. Dafür wurde eine Kompositionsalgebra formuliert, welche mit der booleschen Algebra vergleichbare Operatoren bietet.

Die praktische Umsetzbarkeit der entwickelten Methode wurde anhand einer Implementierungsstudie unter Verwendung aktueller Technologien politikkontrollierter mobiler Systeme (Android, MOSES, SpontAndroid) gezeigt. Zukünftige Forschungsarbeiten können sich der praktischen Implementierung des Konzepts, der Evaluierung bezüglich Laufzeitkomplexität, Laufzeitperformanz und Energiebedarf sowie der Vereinfachung der Spezifikation widmen.

Literatur

[AmCD15] M. Ambrosin, M. Conti, T. Dargahi: On the Feasibility of Attribute-Based Encryption on Smartphone Devices. *In: Proc. 2015 Workshop on IoT Challenges in Mobile and Industrial Systems*, IoT-Sys '15, ACM (2015), 49–54.

[AmKP14] P. Amthor, W. E. Kühnhauser, A. Pölck: WorSE: A Workbench for Model-based Security Engineering. In: *Elsevier Computers & Security*, 42, 0 (2014), 40–55.

[AmKü13] P. Amthor, W. E. Kühnhauser: Leichtgewichtige Sicherheitsdomänen für spontane Kooperationen. *In: Proc. D·A·CH Security 2013*, syssec Verlag (2013), 260–274.

[AmKü15] P. Amthor, W. E. Kühnhauser: Security Policy Synthesis in Mobile Systems. *In: Proc. IEEE 11th World Congress on Services Visionary Track: Security and Privacy Engineering Theme*, SPE '15, IEEE Computer Society (2015), 189–197.

[Ande72] J. P. Anderson: Computer Security Technology Planning Study. Ber. ESD-TR-73-51, Vol. II DITCAD-772806, U.S. Air Force Electronic Systems Division (1972).

[BuHS13] S. Bugiel, S. Heuser, A.-R. Sadeghi: Flexible and Fine-Grained Mandatory Access Control on Android for Diverse Security and Privacy Policies. *In: Proc. 22nd USENIX Security Symposium*, Security '13, USENIX Assoc. (2013), 131–146.

[Chou05] R. Choudhary: A Policy Based Architecture for NSA RAdAC Model. *In: Proc. 6th Ann. Information Assurance Workshop*, IAW '05, IEEE (2005), 294–301.

[Fade06] G. Faden: Solaris Trusted Extensions: Architectural Overview. Sun/Oracle White Paper (2006), URL: https://web.archive.org/web/20081207084104/http://www.opensolaris.org/os/community/security/projects/tx/TrustedExtensionsArch.pdf.

[FlSS96] R. Flin, G. Slaven, K. Stewart: Emergency Decision Making in the Offshore Oil and Gas Industry. In: *Human Factors*, 38, 2 (1996), 262–277.

[JAS04] Horizontal Integration: Broader Access Models for Realizing Information Dominance. Bericht JSR-04-132, JASON Program Office, MITRE Corporation (2004).

[JLTW+08] S. Jha, N. Li, M. Tripunitara, Q. Wang, W. H. Winsborough: Towards Formal Verification of Role-Based Access Control Policies. In: *IEEE Transactions on Dependable and Secure Computing*, 5, 4 (2008), 242–255.

[Lamp74] B. W. Lampson: Protection. In: *ACM SIGOPS Operating Systems Review*, 8, 1 (1974), 18–24.

[LoSm01] P. Loscocco, S. Smalley: Integrating Flexible Support for Security Policies into the Linux Operating System. *In: Proc. FREENIX Track: 2001 USENIX Annual Technical Conference*, ATC '01, USENIX Assoc. (2001), 29–42.

[LuKa11] J. Luo, M. Kang: Risk Based Mobile Access Control (RiBMAC) Policy Framework. *In: Proc. 2011 Military Communications Conference*, MILCOM '11, IEEE Communications Society (2011), 1448–1453.

[RALB14] S. Rasthofer, S. Arzt, E. Lovat, E. Bodden: DroidForce: Enforcing Complex, Datacentric, System-wide Policies in Android. *In: Proc. 9th International Conference on Availability, Reliability and Security*, ARES '14, CPS (2014), 40–49.

[RCCF12] G. Russello, M. Conti, B. Crispo, E. Fernandes: MOSES: Supporting Operation Modes on Smartphones. *In: Proc. 17th ACM Symposium on Access Control Models and Technologies*, SACMAT '12, ACM (2012), 3–12.

[ReLP14] C. Reuter, T. Ludwig, V. Pipek: Ad Hoc Participation in Situation Assessment: Supporting Mobile Collaboration in Emergencies. In: *ACM Transactions on Computer-Human Interaction*, 21, 5 (2014), 26:1–26:26.

[RRFG06] E. E. R. Russo, A. B. Raposo, T. Fernando, M. Gattass: Emergency Environments for the Oil & Gas Exploration and Production Industry. In: *Monografias em Ciência da Computação*, 06, 24 (2006).

[SmCr13] S. Smalley, R. Craig: Security Enhanced (SE) Android: Bringing Flexible MAC to Android. *In: Proc. 20th Annual Network and Distributed System Security Symposium*, NDSS '13, The Internet Society (2013).

[WFMV03] R. Watson, B. Feldman, A. Migus, C. Vance: The TrustedBSD MAC Framework. *In: Proc. DARPA Information Survivability Conference and Exposition*, DISCEX III, IEEE Computer Society (2003), 13–15.

[Wip13] Wipro Limited: Safety and Health Management System in Oil and Gas Industry (2013), URL: https://www.wipro.com/documents/safety-and-health-management-system-in-oil-and-gas-industry.pdf.

[ZaMa04] G. Zanin, L. V. Mancini: Towards a Formal Model for Security Policies Specification and Validation in the SELinux System. *In: Proc. 9th ACM Symposium on Access Control Models and Technologies*, SACMAT '04, ACM (2004), 136–145.