Leichtgewichtige Sicherheitsdomänen für spontane Kooperationen

Peter Amthor · Winfried E. Kühnhauser

Technische Universität Ilmenau
[peter.amthor|winfried.kuehnhauser]@tu-ilmenau.de

Zusammenfassung

Thema dieser Arbeit sind Sicherheitsaspekte in sporadisch und spontan gebildeten Kooperationsgemeinschaften, die bei zeitlich begrenzten Interaktionen innerhalb einer Gruppe von IT-Systemen entstehen. Typische Anwendungsszenarien sind Abwicklungen geschäftlicher Transaktionen (elektronisches Bezahlen, Transfer von Geldern) oder Authentisierungen (Einchecken in Flughäfen, Zugang zu Gebäuden), deren elektronische Abwicklung insbesondere durch die wachsende Intelligenz und Leistungsfähigkeit mobiler IT-Geräte wie Smartphones an Bedeutung gewinnt.
Das Papier stellt einen Ansatz vor, sporadisch gebildete Gruppen innerhalb offener Netzwerke durch die spontane Bildung eigener Sicherheitsdomänen zu unterstützen. Bedingt durch den flüchtigen Charakter dieser Gruppenbildung wird dabei ein hohes Gewicht auf Einfachheit bei der Formulierung von Kooperationspolitiken, Autonomie der beteiligten Systeme, geringen Ressourcenbedarf und der Anwendung aktueller Realisierungstechniken gelegt.

Schlüsselbegriffe: Sicherheitsarchitektur, Sicherheitspolitik, Sicherheitsdomäne, spontane Vernetzung, politikkontrollierte Systeme, SELinux, SEAndroid, Android Security Extensions.

1 Einleitung

Einer der wesentlichen Gründe für die schnelle Verbreitung des Internets ist in seiner Offenheit zu sehen, die ursprünglich ausgehend von den Universitäten einer Vielzahl von Wirtschaftsunternehmen die Nutzung des Netzes leicht gemacht hat. Ironischerweise stellt sich heute dieser ehemalige Motor der kommerziellen und privaten Nutzung des Internets zunehmend als Hindernis heraus. Nationale und internationale Gesetze, wirtschaftliche und private Interessen und ethische Grundsätze führen dazu, dass viele Informationen auf IT-Systemen eine sensitive Natur besitzen, so dass ihre Nutzung bestimmten Regeln unterliegt. Zur präzisen Definition von Regeln im Umgang mit sensitiven Informationen versieht daher eine wachsende Zahl von Organisationen ihre IT-Systeme mit rigorosen Sicherheitspolitiken [HM90, HK99, EB04, SYRG07].

Sicherheitspolitiken definieren und kontrollieren Sicherheitsdomänen, innerhalb derer die Regeln der Sicherheitspolitik durch rigorose Methoden und Techniken durchgesetzt werden [HRU76, BL76, SCFY96, ZM04, ZLN05, HRS+10, NR11]. Damit führen Sicherheitspolitiken allerdings gleichzeitig zu einer Inselbildung und konterkarieren so die grundsätzlichen Errungenschaften des Internets: die offene Kommunikation, den globalen Austausch von Informationen, die gemeinsame Nutzung von Ressourcen oder das Potential IT-gestützter Kooperation.

Besonders hinderlich wirkt sich dies bei spontanen und sporadischen Gruppenbildungen aus,

P. Schartner · P. Trommler (Hrsg.) · D•A•CH Security 2013 · syssec (2013) 260-274.

wie sie im Kontext der Abwicklung geschäftlicher Transaktionen auftreten, wie beispielsweise dem elektronischen Bezahlen, dem Kauf von Wertpapieren, oder der Aushandlung von Verträgen. Eine analoge Problematik ist zu beobachten, wenn mobile IT-Geräte wie Laptops und PDAs physisch aus ihrer Heimatdomäne bewegt werden, in einer anderen Domäne benutzt werden und sich dort in die bestehende Kommunikationsinfrastruktur aus Infrarotnetzen, Bluetooth und drahtlosen LANs einbinden. Die Schwergewichtigkeit von Techniken und Methoden zur Aufstellung und Implementierung von Sicherheitspolitiken machen es problematisch, auf so entstehende ad-hoc-Situationen jeweils mit der notwendigen Dynamik zu reagieren.

Seit einer Reihe von Jahren sind Methoden und Technologien verfügbar, mittels derer Sicherheitspolitiken erstellt und präzise spezifiziert werden können; politikgesteuerte Betriebssysteme wie *SELinux* [LS01], *TrustedBSD* [WV03] oder *Solaris* in Verbindung mit den *Trusted Extensions* [Fad06] sind heute unmittelbar in der Lage, Sicherheitspolitiken zu integrieren und durchzusetzen. Diese Technologiebasis steht zunehmend auch für mobile Plattformen wie beispielsweise Smartphones zur Verfügung und öffnet damit die Tür, Sicherheitseigenschaften für die typischen Anwendungsfelder mobiler Systeme mittels politikbasierter Ansätze zu befördern.

Dieses Papier stellt einen Ansatz zur Beförderung sicherer Interaktion in sporadisch und spontan gebildeten Kooperationsgemeinschaften mittels der Bildung leichtgewichtiger Sicherheitsdomänen vor, die regelgesteuert und transparent aus den Sicherheitspolitiken der beteiligten Systeme konstruiert werden. Dazu werden Ansätze der Metapolitiken [Hos92, Küh99, BDCdVS02] aufgegriffen und zu einer Methode weiterentwickelt, durch die auf anwenderfreundliche und transparente Weise mittels präziser algebraischer Spezifikationen automatisch Sicherheitspolitiken für spontane Kooperationsgemeinschaften generiert werden. Eine Implementierungsstudie auf der Grundlage von SEAndroid [SC13] und MOSES [RCCF12] zeigt die Umsetzbarkeit dieser Methode mit heutigen Standardtechnologien.

2 Ein Beispiel

In diesem Abschnitt werden mittels eines Beispielszenarios einige der mit spontanen Kooperationen verbundenen Risiken veranschaulicht. Einzelne Ausschnitte aus diesem Szenario werden im weiteren Verlauf des Papiers mehrfach zur Illustration herangezogen.

Eine der Aufgaben einer Führungskraft im Management eines Versicherungsunternehmens ist die Anbahnung strategischer Kooperationen mit anderen Unternehmen. Zu Planungsgesprächen reist sie regelmäßig zu prospektiven zukünftigen Partnerunternehmen. Auf ihrem Smartphone befinden sich neben den für ihre tägliche Arbeit benötigten sensitiven Daten

- Präsentationsmaterial für ein Referat, das sie im Rahmen der Planungsgespräche hält
- sensitive Informationen über ihr eigenes Unternehmen, von denen sie einige bei erfolgreichem Verlauf des Gesprächs ihren Gesprächspartnern zur Verfügung stellen wird.

Innerhalb ihres eigenen Unternehmens ist ihr Smartphone über lokale drahtlose Netztechnologien ein integrierter Bestandteil der unternehmenseigenen IT-Infrastruktur, und sämtliche Daten auf dem Smartphone sind Teil der Sicherheitsdomäne ihres eigenen Unternehmens. Das Gespräch mit den zukünftigen Kooperationspartnern findet dagegen in auswärtigen Räumen statt, in denen das Smartphone die dortige lokale IT-Infrastruktur z.B. zur Übertragung des

Präsentationsmaterials zu einem Projektor nutzt und somit physisch in die technische Infrastruktur des Fremdunternehmens und dessen Sicherheitsdomäne integriert ist.

Dieses Szenario birgt nun zwei grundsätzliche Risikoarten. Zum Einen ergibt sich eine mögliche Gefährdung der IT-Systeme des Gastgebers, da das Smartphone sich innerhalb einer normalerweise durch eine Firewall von Fremdsystemen isolierten Sicherheitsdomäne befindet und von dort aus aktive Angriffe unternehmen kann. Zum Anderen besteht ein Risiko für das eingebrachte Smartphone, welches nicht länger durch die vertrauenswürdige Umgebung seiner Heimdomäne geschützt ist.

Derartige operationale Risiken für Systeme mit sensitiven Informationen sind wohlbekannt, und zur Risikoreduktion versieht daher eine wachsende Zahl von Organisationen ihre IT-Systeme mit rigorosen Sicherheitspolitiken, die in obigen Beispiel dann das Smartphone einerseits und die Sicherheitsdomäne der Gastgeber-Infrastruktur andererseits kontrollieren. Bei jeder Form der Interaktion zwischen Smartphone und den Systemen des Gastgebers sind folglich zwei Sicherheitspolitiken involviert, die ursprünglich vollkommen unabhängig voneinander aufgestellt worden sind. Dies kann sowohl zu Konfliktsituationen führen (die Sicherheitspolitik des Gastgebersystems akzeptiert z.B. wegen der Gefahr durch Makroviren keine unsignierten Präsentationsdokumente) als auch zu Situationen, in denen für eine konkrete Interaktion keine der Sicherheitspolitiken eine Regel besitzt (die Präsentationssoftware des Gastgebersystems greift auf die Firmendaten des Smartphones zu, dieser Zugreifer ist der Sicherheitspolitik des Smartphones jedoch nicht bekannt). Derzeit sind die Probleme der Kooperation zwischen unterschiedlichen Sicherheitsdomänen entweder schwergewichtig [KvKO95, Küh99] oder werden manuell gehandhabt, beispielsweise dadurch, dass ein zusätzlich mitgebrachter Laptop physisch mit einem Projektor verbunden oder dem Gastgeber ein USB-Stick mit der Präsentation übergeben wird. Tatsächlich beobachten wir hier ein Relikt, welches direkte, effiziente und unkomplizierte Kommunikationsformen behindert und es nicht erlaubt, verfügbare Technologien wie mobile Geräte und allgegenwärtige Kommunikationsinfrastrukturen effizient und effektiv zu nutzen.

Vergleichbaren Risiken und Beschränkungen finden sich in Szenarien, in denen Smartphones z.B. für den Zugriff auf Bankkonten, den Handel mit Wertpapieren, oder dem Check-In auf Flughäfen benutzt werden.

3 Spezifikation von Kooperationspolitiken

Dieses Papier stellte einen Methode zur sicheren Interaktion in sporadisch und spontan gebildeten Kooperationsgemeinschaften vor. Kern der Methode sind Sicherheitspolitiken, die das individuelle Verhalten eines Systems in spontanen Kooperationen beschreiben, in Form algebraischer Spezifikationen in der Sicherheitsarchitektur eines Systems hinterlegt sind und bei Fällen spontaner Kooperation zur automatischen Konstruktion von Kooperationspolitiken herangezogen werden. Dabei ermöglicht die Präzision der algebraischen Spezifikationen eine automatische Generierung der Kooperationspolitiken, die keinerlei Benutzerinteraktion erfordert und transparent im Hintergrund erfolgt.

Dieses Kapitel beschreibt den algebraischen Hintergrund der Komposition von Sicherheitspolitiken und der Bildung der assoziierten Sicherheitsdomänen. Der Entwurf der Kompositionsalgebra wird dabei von drei wesentlichen Eigenschaften des Anwendungsumfelds bestimmt:

der Spontanität in der Bildung kurzlebiger Sicherheitsdomänen, der automatischen und transparenten Generierung der Kooperationspolitiken und der Ressourcenknappheit der beteiligten Systeme; ihre Anwendung muss daher schnell, unsichtbar und ressourcenschonend erfolgen.

Ausgangspunkt des Ansatzes sind Einzelsysteme mit eigener Sicherheitsarchitektur und eigener Sicherheitspolitik, wie dies beispielsweise Android-Smartphones sind. Ziel ist es, Gruppen aus solchen Einzelsystemen zu bilden, die für einen beschränkten Zeitraum eine eigene Sicherheitsdomäne mit einer einzigen Gesamtpolitik bilden, wobei die Gesamtpolitik aus den individuellen Politiken der Beteiligten erzeugt wird. Diese Erzeugung erfolgt automatisiert auf der Grundlage von Kompositionsregeln, die mittels einfacher Terme einer Kompositionsalgebra beschrieben werden. Kompositionsregeln werden – analog zur Sicherheitspolitik – ein einziges Mal beschrieben, in der Sicherheitsarchitektur der Einzelsysteme hinterlegt und kontrollieren im Bedarfsfall die Bildung der Domänenpolitik.

Die aktuelle Politikkomposition bei einer spontanen Gruppenbildung erfolgt transparent im Hintergrund ohne jegliche Interaktion mit dem Benutzer; ebenfalls müssen Benutzer weder den algebraischen Hintergrund der Politikkomposition noch die Kompositionspolitik selbst kennen.

3.1 Kompositionsalgebra

Die Kompositionsalgebra legt die Sprache zur Formulierung von Kompositionsregeln fest. Ihre Elemente sind Sicherheitspolitiken und Verknüpfungsoperatoren, mittels derer präzise Regeln formuliert werden können, die die Semantik der Kombination von Sicherheitspolitiken beschreiben, und nach denen bei einer spontanen Gruppenbildung die einzelnen beteiligten Sicherheitspolitiken automatisch zu einer insgesamt für die Gruppe gültigen Sicherheitspolitik kombiniert werden.

Die hier vorgestellte Algebra ist beschränkt auf die Klasse der Zugriffssteuerungspolitiken, die als gleichzeitiges Mittel zur Wahrung von Vertraulichkeit (Regulierung von lesenden Zugriffen), Integrität (Regulierung von schreibenden Zugriffen) und Verfügbarkeit (Regulierung von Ressourcennutzung) ein hohes Maß an Universalität besitzen.

Zugriffssteuerungspolitiken sind Sicherheitspolitiken, die (u.A.) durch eine Zugriffssteuerungsfunktion f und eine Domäne D (ihren Wirkungsbereich) charakterisiert sind.

Die Zugriffssteuerungsfunktion f_P einer Zugriffssteuerungspolitik P ist eine Abbildung $f : E^n \times A \rightarrow \{true, false\}$ [Lam74] mit der Semantik, dass eine gegebene Aktion $a \in A$ mit n beteiligten Entitäten $e_i \in D_P$ genau dann erlaubt ist, wenn $f(e^n, a) \mapsto true$, wobei D_P die Politikdomäne ist. f_P reflektiert also die Zugriffsentscheidungen einer Sicherheitspolitik und bestimmt damit beispielsweise, ob das Lesen einer bestimmten Datei e_i durch eine bestimmte Applikation e_j erlaubt ist, falls e_i und e_j zur Politikdomäne gehören. In heutigen Standardbetriebs- oder sind Zugriffssteuerungsfunktionen in Form von Zugriffssteuerungslisten oder Capabilitylisten vertreten.

Wenn wir nun als Grundmenge P einer Kompositionsalgebra die Menge aller Zugriffssteuerungspolitiken bezeichnen und jede Politik $P_i \in P$ als eine Zugriffssteuerungsfunktion $f_{P_i} : E_i^{n_i} \times A_i \rightarrow \{true, false\}$ auffassen, dann lässt sich die Semantik einer Politikkomposition mittels elementarer Operatoren auf dieser Grundmenge wie folgt beschreiben:

Konjunktion: Eine Politikkomposition, die eine Aktion genau dann erlaubt, wenn sämtliche

der individuellen Einzelpolitiken zustimmen, erhält man durch den Konjunktionsoperator „$\wedge_P$":
$\wedge_P : P \times P \to P$, wobei $\wedge_P(P_i, P_j) \mapsto P_k$ mit $f_{P_k} = f_{P_i} \wedge f_{P_j}$.

Disjunktion: Eine Politikkomposition, die eine Aktion genau dann erlaubt, wenn mindestens eine der individuellen Einzelpolitiken zustimmt, erhält man durch den Disjunktionsoperator „$\vee_p$":
$\vee_P : P \times P \to P$, wobei $\vee_P(P_i, P_j) \mapsto P_k$ mit $f_{P_k} = f_{P_i} \vee f_{P_j}$.

Negation: Eine Politik, die eine Aktion genau dann erlaubt, wenn eine andere Politik diese verbietet, erhält man durch den Negationsoperator „$\neg_P$":
$\neg_P : P \to P$, wobei $\neg_P(P_i) \mapsto P_j$ mit $f_{P_j} = \neg f_{P_i}$.

Selektion: Eine Politikkomposition, die abhängig vom Ergebnis einer Politikanwendung eine weitere Politik selektiert erhält man durch den Selektionsoperator „$select_P$":
$select_P : P \times P \times P \to P$, wobei $select(P_i, P_j, P_k) \mapsto P_l$ mit $f_{P_l} = \textit{ if } f_{P_i} \textit{ then } f_{P_j} \textit{ else } f_{P_k}$.

$\neg, \wedge$ und $\vee$ sind die bekannten logischen Operatoren in ihrer gebräuchlichen Semantik. Mit den üblichen Eigenschaften der Kommutativität und Assoziativität kann nun die Komposition von Sicherheitspolitiken durch algebraische Ausdrücke beschrieben werden und beispielsweise Regeln der Art „erlaube einen Zugriff genau dann, wenn sämtliche beteiligten Sicherheitspolitiken diesen Zugriff erlauben" durch präzise Terme formuliert werden, mittels derer bei spontan entstehenden Kooperationsgruppen aus den beteiligten Sicherheitspolitiken automatisch eine neue Gruppenpolitik gebildet wird.

Im Allgemeinen lässt sich obige Liste der algebraischen Operatoren zur Politikverknüpfung um beliebige Funktionen ergänzen. Diese Arbeit beschränkt sich auf die vier genannten Elementaroperatoren, da hieraus bei Bedarf weitere Operatoren konstruierbar sind.

3.2 Komposition von Sicherheitspolitiken und Domänen

Wird nun im Zuge der Bildung einer Kooperationsgruppe ihre Sicherheitspolitik erzeugt, so werden hierzu von jedem Gruppenmitglied benötigt

1. Die zur Komposition zu verwendenden individuellen Kompositionsregeln
2. Die zur Komposition zu verwendende individuelle Sicherheitspolitik
3. Der Perimeter des Wirkungsbereichs der Sicherheitspolitik – eine präzise Bestimmung der Akteure und Objekte, die jedes individuelle Gruppenmitglied in Domäne der Gruppenpolitik einbringt.

Das Spektrum möglicher Kooperationsbeziehungen kann hierbei sehr breit sein. Ein Beispiel ist der Erwerb von Wertpapieren von einem Broker durch einen Softwareagenten, wobei der Kauf mittels einer Kreditkarte erfolgt. In diesem Szenario sind Agent, Broker und Kreditkarteninstitut gleichberechtigte Partner mit hoher Autonomie, und jede Partei hat bei der Transaktion ein Vetorecht; auch das eingangs in Kap. 2 genannte Beispiel zeichnet sich durch Gleichberechtigung und Autonomie der Teilnehmer aus. Ein anderes Szenario beispielsweise ist eine hierarchisch strukturierte Gruppe, die einen Gruppenleiter mit dem Recht zur Durchsetzung seiner Entscheidungen besitzt, ohne dass einzelne Gruppenmitglieder Vetorechte besitzen. Andere Szenarien wiederum arbeiten beispielsweise auf der Grundlage von Mehrheitsentscheidungen.

Eine universelle und statische Strategie zur Bildung von Gruppenpolitiken lässt sich somit kaum angeben. Vielmehr geht der Ansatz in dieser Arbeit von einer organisierten Offenheit aus, in der jedes Gruppenmitglied seine eigene Strategie mittels eines Tupels (C,P,D) angeben kann, welches die eigene, individuelle Strategie zur Bildung von Gruppenpolitiken und ihren Domänen beschreibt. Für ein Gruppenmitglied i beschreibt hierbei

1. der Konstruktor C_i die individuellen Regeln des Gruppenmitglieds i zur Bildung der Gruppenpolitik mittels der in Abschnitt 3.1 skizzierten Algebra
2. die Sicherheitspolitik P_i die individuelle Sicherheitspolitik von i, die bei der Bildung der Gruppenpolitik zu verwenden ist
3. die Sicherheitsdomäne D_i diejenigen Datenobjekte, die i der Gruppenpolitik zu unterstellen bereit ist.

3.2.1 Politikkomposition

Bei der Bildung einer Gruppe G mit Mitgliedern $i \in G$ und Tupeln (C_i,P_i,D_i) ergibt sich dem Grundsatz der geringsten notwendigen Rechte folgend die Gruppenpolitik P_g aus

$$P_g = \bigwedge_{i \in G} C_i \text{ mit der Domäne } D_g = \bigcup_{i \in G} D_i \ .$$

Realisiert wird diese Politikbildung durch Austausch der Tupel (C_i,P_i,D_i) (siehe Kap. 4), so dass jedes Gruppenmitglied über alle zur lokalen Bildung der Gruppenpolitik notwendigen Informationen verfügt.

Das Besucher-Szenario aus Kap. 2 ist nun hinsichtlich seiner Sicherheitseigenschaften planbar seitens des Gastes – nennen wir sie Alice – durch ein Konstruktionstupel $(C_{Alice},P_{Alice},D_{Alice})$ mit

(i) $C_{Alice} = P_{Alice} \wedge_P \bigwedge_{i \in G, i \neq Alice} C_i$,

(ii) P_{Alice} mit der (trivialen) Zugriffssteuerungsfunktion $f_{P_{Alice}} \mapsto \{false\}$,

(iii) $D_{Alice} = \{MSPowerPoint\}$,

wodurch ausgedrückt wird, dass lediglich ein einziges Programm *MSPowerPoint* in die Domäne der Kooperationspolitik eingebracht wird (iii), grundsätzlich keinerlei Zugriff auf dieses Programm erlaubt wird (ii), und in der Gruppenpolitik immer mindestens die Regeln der eigenen Politik gelten müssen (Konjunktionsoperator „$\wedge_P$", (i)). Durch eine zu (i) analoge Bildung eines eigenen Konstruktionstupels kann der Gastgeber ebenfalls seine volle Autonomie wahren, wobei er der Domäne der Gruppenpolitik den Projektorserver zugesellt und mittels seiner eigenen eingebrachten Zugriffssteuerungspolitik allen Subjekten der Domäne die Nutzung des Projektor-Servers gestattet (Abb. 1).

Sollen dagegen Gruppen eine gemeinsame, obligatorische Gruppenpolitik P_{MAC} besitzen, so wird dies erreicht durch individuelle Konstruktortupel mit $C_i = P_{MAC}$. Erfolgt dabei die Gruppenbildung zwischen Mitgliedern derselben Organisation, so ist dies auf einfache Weise durch die Sicherheitsadministratoren im Kontext der Konfiguration der Einzelsysteme der Gruppenmitglieder erreichbar; in Szenarien mit autonomen Gruppenmitgliedern wird deren Autonomie dadurch deutlich, dass jedes individuelle Gruppenmitglied durch seinen derart gestalteten individuellen Konstruktor explizit die Akzeptanz der Gruppenpolitik erklären muss.

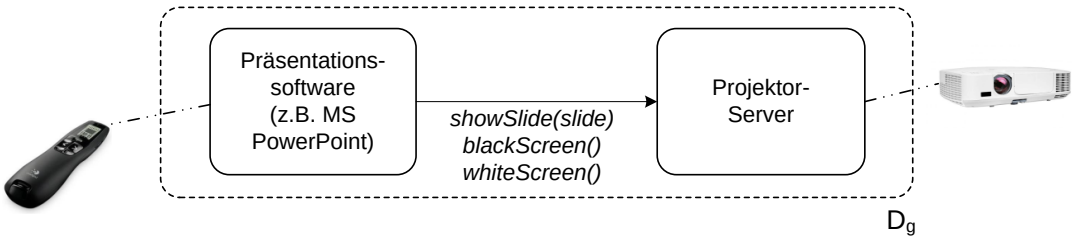

Abb. 1: Sicherheitsdomänenbildung im Besucher-Szenario

Als weiteres Beispiel lassen sich mit diesem Politikkonstruktionsschema demokratische Wahlszenarien aufbauen, in denen ein Zugriff auf eine Ressource auf der Grundlage von Mehrheitsentscheidungen beruht. Beispielsweise durch *leader-election*-Algorithmen [GM82] wird hierbei zunächst ein Wahlleiter bestimmt, der eine das Wahlverfahren implementierende Politik P_{leader} besitzt. Mit $f_{P_{leader}} \mapsto (|\{P_i|f_{P_i} = true\}| \geq \lceil n/2 \rceil)$ implementiert diese Politik z.B. ein Mehrheitswahlverfahren. Ebenfalls lassen sich Quorum-basierte Wahlschemata (eine aus dem Replikationsmanagement bekannte Generalisierung der Mehrheitswahlverfahren) auf analoge Weise erreichen; durch $f_{P_{leader}} \mapsto (|\{P_i|f_{P_i} = true\}| = n)$ ergibt sich ein vollständigen Konsens erforderndes Wahlverfahren, in welchem jedes Gruppenmitglied ein Vetorecht besitzt, während am anderen Ende $f_{P_{leader}} \mapsto (|\{P_i|f_{P_i} = true\}| = 1)$ eine autoritäre Gruppenleitersemantik ergibt.

Die Teilnahme an jedem dieser Wahlverfahren erfolgt mittels eines Konstruktionstupels (P_{leader}, P_i, D_i), wobei D_i die jeweils dem Wahlverfahren unterstellten Entitäten des Teilnehmers i sind.

Analog zur Auswertung der Kompositionsregeln erfolgt die Erzeugung und Durchsetzung der Domäne einer Gruppenpolitik verteilt durch die Referenzmonitore der beteiligten Systeme, die die Gruppenpolitik jeweils für den lokalen Teil der Politikdomänen durchsetzen (vgl. Kap. 4).

4 Implementierungsstudie

Dieser Abschnitt stellt ein Konzept zur Implementierung einer Algebra basierten automatischen Politikkomposition für spontan gebildete Kooperationsgemeinschaften unter Anwendung aktueller Technologien vor. Die Studie baut auf dem Smartphone-Betriebssystem Android auf, das durch seine hohe Verbreitung und Relevanz für das Zielszenario eine Generalisierbarkeit der Ergebnisse erlaubt.

Basierend auf Anforderungen des Anwendungsumfelds (Spontanität, Transparenz, Sparsamkeit; vgl. Kap. 3) berücksichtigt das im Folgenden präsentierte Implementierungskonzept folgende Eigenschaften:

1. Umsetzung der Referenzmonitorprinzipien [And72] zur politikbasierten Zugriffssteuerung
2. Auswertung von algebraischen Termen (lokal) sowie polymorphen Zugriffssteuerungsfunktionen (verteilt)
3. geringe Speicher- und Laufzeitkomplexität
4. Transparenz durch Integration unterhalb der Anwendungsschicht im Android-Software-Stack.

4.1 Verwendete Technologien

In diesem Abschnitt werden die Technologien skizziert, auf denen diese Implementierungsstudie aufbaut. Als Grundlage für die spätere Beschreibung konkreter Mechanismen soll dabei sowohl auf das Android-Betriebssystems als auch die darauf aufbauenden Sicherheitsarchitektur MOSES eingegangen werden.

Da das Anliegen dieser Studie eine Anwendung der Kompositionsalgebra ist, wird zur Herstellung obiger Eigenschaften auf ein bestehendes Android-Derivat aufgebaut. Es existiert eine Reihe von Erweiterungen des Software-Stacks des Betriebssystems, die architekturspezifischen Bedrohungsszenarien begegnen (aus der Klasse der *privilege escalation* Angriffe) und dabei bereits Mechanismen zur hochperformanten Durchsetzung obligatorischer Sicherheitspolitiken bereitstellen. Aus diesem Spektrum haben wir eine Auswahl geeigneter Plattformen identifiziert, darunter SEAndroid [SC13], XManDroid [BDD+12] und MOSES [RCCF12]. Für die vorliegende Implementierungsstudie wurde aufgrund bestehender praktischer Erfahrungen exemplarisch das Framework MOSES ausgewählt.

Android

Das Smartphone-Betriebssystem Android basiert auf einem modifizierten Linux Kern. Dieser übernimmt die unmittelbare Abstraktion der Hardware, also etwa des physischen Speichers, der Peripheriegeräte (wie Kamera, Mikrofon und Bildschirm) und der Netzwerkschnittstellen (sowohl für Mobilfunk- als auch Datennetzwerke). Im Unterschied zu Desktop-Varianten des Linux Betriebssystems setzt auf den Schnittstellen des Kerns nicht direkt die Anwendungsschicht und API auf, sondern eine Middleware-Schicht. Deren Hauptaufgabe besteht in der Interpretation von Bytecode, der als Zwischenrepräsentation ausführbarer Programme aus Java-Quelltext erzeugt werden kann. Die Middleware bildet somit die Laufzeitumgebung für Android-Anwendungen, welche mittels virtueller Host-Prozesse voneinander isoliert werden (Sandboxing-Prinzip).

Um kontrollierte Kommunikation zwischen lokalen Anwendungen zu ermöglichen, biete Android verschiedene IPC-Mechanismen, die sowohl auf Middelware- als auch Kernel-Niveau angesiedelt sind. Diese werden durch Berechtigungen gesteuert, die einer Anwendung nach Zustimmung durch den Nutzer zum Installationszeitpunkt zugewiesen werden (z.B. „SD-Speicher ändern“, „Internetzugriff“, „Kontaktdaten schreiben“ etc.). Diese abstrakte Semantik der Android-Berechtigungen in Kombination mit dem mehrere Abstraktionsschichten überspannenden IPC-Modell haben in der Vergangenheit eine Anzahl an privilege-escalation Angriffen hervorgebracht [SZZ+11, MFv11], welche die existierenden Zugriffssteuerungsmechanismen unterlaufen. Verschiedene Lösungsansätze hierfür basieren auf Sicherheitspolitiken, deren Regeln feingranularer spezifiziert werden können und deren Durchsetzung einer zentralisierten Instanz obliegt.

MOSES

Die Ziele von MOSES liegen in zwei grundlegenden Bereichen: Zum einen sollen Android-typische Bedrohungsszenarien, welche o. g. privlege-escalation Angriffe ermöglichen, ausgeschaltet werden. Hierzu setzt MOSES Sicherheitspolitiken ein, deren Einhaltung systemweit durch eine auf Middleware-Ebene angesiedelte Referenzmonitorarchitektur überwacht wird.

Zum anderen sollen lokale Sicherheitsdomänen geschaffen werden, um verschiedene Arten von Daten und Anwendungen, welche physisch auf einem Smartphone vorliegen, möglichst vollständig voneinander zu isolieren (bspw. eine berufliche von einer privaten Domäne). Hierzu wird eine Erweiterung des grundlegenden Sandboxing von Android-Anwendungen eingeführt, welche Daten und Anwendungen zu Sicherheitsdomänen zusammenfasst. Durch eine Reihe von Mechanismen, welche in die Middleware integriert wurden, können diese Domänen einerseits vollständig voneinander isoliert werden, andererseits können Zugriffssteuerungsmechanismen für Daten und Anwendungen innerhalb einer Domäne feingranular und auf semantisch adäquatem Niveau durch Sicherheitspolitiken gesteuert werden.

Um dies zu erreichen, führt MOSES *Security Profiles* (SP) ein, welche jeweils eine lokale Sicherheitsdomäne repräsentieren. Bei einem SP handelt es sich im Wesentlichem um eine Zusammenfassung bestimmter Anwendungen (wie bspw. Adressverwaltung oder Texteditor), Datenobjekte (wie Telefonkontakte oder Dateien auf einer Speicherkarte) und Sicherheitspolitiken. Letztere legen durch Zugriffsregeln fest, wie Anwendungen und Daten innerhalb eines SPs miteinander interagieren können.

Eine weitere Aufgabe von MOSES, welche der Forderung nach Nutzertransparenz Rechnung trägt, ist der sog. dynamische Kontextwechsel. Dieser Mechanismus ermöglicht es, automatisch zwischen verschiedenen SPs zu wechseln und diese durchzusetzen, und zwar abhängig von externen Parametern („Kontext“) wie Uhrzeit, Aufenthaltsort oder Netzwerkumgebung. Eine Menge von logischen Ausdrücken, die in Abhängigkeit von solchen Parametern einen Kontext beschreiben, sind ebenfalls in jedem SP enthalten. Das sog. *Context Monitoring Module*, welches Teil der Referenzmonitorarchitektur ist, prüft in Form eines periodischen Background-Jobs, welcher Kontext vorliegt und aktiviert das damit assoziierte SP. Ein zuvor aktives SPs wird hierbei deaktiviert, da MOSES keine parallele Gültigkeit mehrerer Kontextdefinitionen vorsieht. Aus diesem Grund werden neue SPs automatisch auf Konflikte mit den Kontextdefinitionen bestehenden SPs überprüft.

4.2 Implementierung der Politikkomposition

In diesem Abschnitt werden die wesentlichen Punkte einer prototypischen Implementierung besprochen. Dabei wird zunächst das verwendete Kommunikations- und Architekturmodell vorgestellt, basierend auf den eingangs genannten Anforderungen sowie der bestehenden MOSES-Architektur. Anschließend werden die Datenstrukturen für die drei Elemente Konstruktor, Sicherheitspolitik und Sicherheitsdomäne jedes Gruppenmitglieds sowie abschließend der Ablauf einer spontanen Gruppenbildung und Domäneninteraktion beschrieben.

Im Folgenden werden Datenstrukturen als Klassen im objektorientierten Programmierparadigma aufgefasst, wie es der Implementierung von Middleware und Anwendungsschicht unter Android sowie von MOSES selbst (mittels Java sowie C++) entspricht. Zur Beschreibung des Kommunikationsmodells wird jedes Gruppenmitglied als Knoten in einer vollständig vernetzten Topologie von Rechnern (hier Android-Smartphones) aufgefasst.

Kommunikationsmodell

Technisch bildet der Ansatz autonomer Teilpolitiken ab auf die Kommunikation und Auswertung von Tupeln der Gestalt (C_i, P_i, D_i). Dabei wird ein Rollenmodell angenommen, bei dem *Klient* einen Prozess bezeichnet, von dem ein Zugriff ausgeht, während der *Server* ein Prozess

(möglicherweise auf einem anderen Knoten) ist, welcher das zugegriffene physische Datenobjekt repräsentiert (bspw. ein FTP-Server). Tatsächlich kommuniziert werden muss zur Gruppenbildung der jeweilige Kompositionsausdruck eines Knotens C_i, eine Menge von Bezeichnern seiner Objekte D_i sowie ein Politik-Interface-Objekt (*policy interface object*, PIO) PIO_i als Repräsentant seiner Sicherheitspolitik P_i. Letzteres nimmt die Rolle eines Stubs ein, durch den vermieden werden kann, dass für eine potentiell kurzfristige spontane Kooperation zwischen Gruppenmitgliedern deren vollständige Politiken kommuniziert werden müssen.

Architekturmodell

Diese Implementierung orientiert sich an einer strikten Trennung von Zugriffsentscheidung (in einem *policy decision point*, PDP) und deren Durchsetzung (in *policy enforcement points*, PEPs). Beide Komponenten sind bereits Teil des MOSES Frameworks und setzen in ihrer Implementierung das Referenzmonitorprinzip der vollständigen Zugriffskontrolle um.

Zusätzlich zur lokalen, politikbasierten Zugriffssteuerung eines jeden Gruppenmitglieds (i. F. bezeichnet als „Tier 1“) wird in dieser Studie eine zweite, verteilte Stufe von Zugriffssteuerungsmechanismen eingeführt, bezeichnet als „Tier 2“. Diese beiden Stufen verhalten sich orthogonal zueinander, ähnlich wie die traditionelle Unix-Zugriffssteuerung zur SELinux-Zugriffssteuerung: Eine Anwendung, welche auf einem der Gruppenmitglieder läuft, muss sowohl für ihren jeweiligen Zugriff durch Tier 1 autorisiert sein als auch, sofern nötig, durch Tier 2. Dieser Ansatz trägt der Motivation des least-privilege-Prinzips Rechnung.

Als Konsequenz dieser Erweiterung werden nun Tier-2-Security-Profiles (T2SP) zusätzlich zu den den bestehenden MOSES Security Profiles (T1SP) eingeführt. Auch hier werden T1SPs weiterhin (unverändert) benutzt, um lokale Isolation von Anwendungen und Daten eines einzelnen Gruppenmitglieds zu erreichen. Darüber hinaus regeln T2SPs sämtliche Zugriffe innerhalb der Kooperationsgruppe, bei denen eine globale Gruppenpolitik P_g involviert ist. Die Idee hierbei ist, für jedes Gruppenmitglied i genau ein Profil T2SP_i zu definieren, von denen (im Gegensatz zu den T1SPs) mehrere parallel aktiv sein können. Ergänzend hierzu kann der dynamische Kontextwechsel unter MOSES ausgenutzt werden, um spontane Gruppen vollständig automatisch und nutzertransparent zur bilden, zu benutzen und aufzulösen. Um dies zu ermöglichen, unterliegen Kontextdefinitionen von T2SPs nicht der Konfliktprüfung, da Konflikte zwischen verschiedenen Sicherheitspolitiken ja nun konstruktiv durch die Kompositionsalgebra vermieden werden. Dies funktioniert natürlich nicht nur bei verteilten, sondern auch bei mehreren lokal aktiven Politiken.

Die originären T1SPs von MOSES beinhalten jeweils eine Sicherheitspolitik sowie eine Menge von Anwendungen und Daten, die ihre jeweilige Domäne beschreiben. Die T2SPs beinhalten, semantisch analog dazu, die drei Tupelelemente der Kompositionsalgebra: Ein T2SP_j auf Knoten $i \neq j$ beinhaltet das Tupel (C_j, PIO_j, D_j), ein T2SP_i das Tupel (C_i, P_i, D_i), wobei P_i die individuelle Sicherheitspolitik von i bezeichnet und PIO_j ein PIO als Kommunikationsschnittstelle für die Politik P_j. Zur Speicherung dieser Tupel wird ein *T2SP Store* hinzugefügt, analog zum MOSES Profile Store (i. F. bezeichnet als *T1SP Store*), welcher weiterhin die lokalen T1SPs vorhält.

Der MOSES Policy Provider, welcher weiterhin die Repräsentation von Tier-1-Sicherheitspolitiken zur Aufgabe hat, wird ebenfalls um ein Gegenstück für Tier 2 ergänzt. Dieser Tier-2-

Policy-Provider (*T2PP*) beinhaltet jedoch nicht die Tier-2-Politiken bzw. PIOs selbst, sondern die Konstruktoren der Gruppenmitglieder.

4.2.1 Konstruktoren

Das Tupelelement C_i wird implementiert als Instanz einer PCO-Klasse (*policy composition object*), welche Regeln der Kompositionsalgebra repräsentiert. Jedes PCO ermöglicht mit einer öffentlichen Methode *eval* lesenden Zugriff auf den bool'schen Wert eines statisch (im Klassenkonstruktor) zusammengesetzten Terms (vgl. Kap. 3). Dessen Grundmenge ist immer P, ihre Mächtigkeit leitet sich also aus der Gruppengröße ab welche ebenfalls im Klassenkonstruktor übergeben wird.

Neben *eval* biete ein PCO eine weitere Schnittstelle an, welche dem T2SP Store während der Gruppenbildungsphase ermöglicht, ein *callback* für die Zugriffsfunktion f_{P_i} jeder Teilpolitik P_i (bzw. des entsprechenden PIO) zu registrieren. So kann die *eval*-Methode zur Laufzeit des verteilten Zugriffssteuerungssystems zu einem Kompositionsergebnis entsprechend der initialisierten Regel des Gruppenmitglieds i gelangen. Dieser Wert wird wiederum vom PDP mit den Ergebnissen aller anderen Kompositionstermen konjunktiv verknüpft, um die Entscheidung der Gruppenpolitik P_g zu konstruieren.

4.2.2 Sicherheitspolitiken

Die individuelle Sicherheitspolitik P_i eines Gruppenmitglieds i kann als binäre Speicherrepräsentation einer maschinenlesbar spezifizierten Zugriffssteuerungspolitik, etwa mittels XACML, verstanden werden. Sie ist Teil mindestens eines T2SP jedes Gruppenmitglieds, welches das eigene Kompositionstupel beschreibt. Alle T2SPs, welche von einem anderen Mitglieder j der Gruppe erhalten wurden, beinhalten stattdessen ein PIO, das wie folgt implementiert wird.

Jedes PIO_j ist Instanz einer spezialisierten PIO-Klasse, abgeleitet von einer abstrakten Oberklasse, die genau eine Schnittstelle nach außen bereitstellt: eine Methode mit bool'schem Rückgabewert, welche die Zugriffsfunktion f_{P_j} repräsentiert. Diese Methode ist polymorph, da ihre genaue Signatur politikabhängig ist, also nur von dem jeweiligen Gruppenmitglied j spezifiziert werden kann. Intern beinhaltet PIO_j private Datenstrukturen, welche die Kommunikation mit dem Gruppenmitglied j ermöglichen, um eigentliche Politikentscheidungen zu erfragen. Dies umfasst, neben den Daten zur korrekten Netzwerkadressierung, insbesondere ein Zertifikat von j (signierter öffentlicher Schlüssel) um die Authentizität dieser kommunizierten Politikentscheidungen sicherzustellen. Darüber hinaus ist es im Sinne der Performanzsteigerung sinnvoll, jedes PIO mit einem Cache für bereits erfragte Politikentscheidungen auszustatten.[1]

Der Aufruf der Zugriffsfunktionen für die individuellen Sicherheitspolitiken der Gruppenmitglieder wird vollständig gekapselt durch die *eval*-Schnittstelle jedes PCOs (s. o.), welches wiederum dem T2PP bekannt ist.

[1] Ein solcher Cache muss mit einer Invalidierungssemantik arbeiten, welche den Eigenschaften des Anwendungsumfelds gerecht wird. Als Vorbild kann hier die Handhabung des *Access Vector Cache* im politikkontrollierten Betriebssystem SELinux [LS01] dienen, welche eine aktive Invalidierung bei jeder Politikaktualisierung vorsieht. Im vorliegenden Fall spontan gebildeter Gruppen wäre dies immer dann, wenn sich die Gruppe (bzw. der MOSES-Kontext) verändert und damit den Neuaustausch der Kompositionstupel erforderlich macht.

4.2.3 Sicherheitsdomänen

Die Implementierung der Sicherheitsdomänen D_i ist konzeptionell geradlinig. Es handelt sich dabei um Instanzen einer Klasse, welche beliebige Mengen von Bezeichnern kapselt. Diese Bezeichner wiederum identifizieren Datenobjekte eines Gruppenmitglieds i lokal eindeutig (bspw. in Form von Dienst- oder Dateinamen), werden in Form des jeweiligen D_i-Objekts kommuniziert und als Teil eines T2SP im T2SP Store jedes Gruppenmitglieds hinterlegt. Bei der Komposition jeder globalen Politikentscheidung im PDP muss nun die korrekte Domänenzugehörigkeit aller Argumente jedes Aufrufs einer Zugriffsfunktion sichergestellt werden, ansonsten handelt es sich um einen Zugriff, welcher die Grenzen der von Gruppenmitglied i festgelegten Sicherheitsdomäne überschreitet und somit durch P_i unter keinen Umständen autorisiert werden kann. Es muss also für jeden Aufruf $f_{P_i}(e_1,\ldots,e_n,a)$ gelten:

$$\exists\, k \in [1\ldots n] : e_k \notin D_i \Rightarrow f_{P_i}(e_1,\ldots,e_n,a) = false$$

Die korrekte Interpretation dieser Semantik obliegt natürlich nicht dem jeweiligen D_i-Objekt, sondern dem PIO_i, welches f_{P_i} kapselt.

4.2.4 Ablaufschema

Die Interaktion zwischen den oben vorgestellten Softwarekomponenten in einem verteilten Szenario wird im Folgenden anhand der beiden Phasen Gruppenbildung und Domäneninteraktion (in Form eines domänenübergreifenden Zugriffs) skizziert.

Gruppenbildungsphase

Jeder Knoten i ...

1. erstellt/wählt eine lokale Kompositionsregel C_i, eine lokale Politik P_i und eine Menge von Bezeichnern D_i für lokale Datenobjekte, welche dieser Politik unterliegen. Dieser Vorgang kann entweder manuell oder für den Nutzer transparent aufgrund eines durch MOSES detektierten Kontextwechsels erfolgen.
2. trägt das Tupel $\text{T2SP}_i = (C_i, P_i, D_i)$ in seinen T2SP Store ein.
3. erstellt PIO_i aus P_i.
4. sendet das Tupel (C_i, PIO_i, D_i) an alle anderen Knoten.

Weiterhin trägt i jedes empfangene Tupel $\text{T2SP}_j = (C_j, PIO_j, D_j)$ in seinen T2SP Store ein und erstellt das PCO_j aus C_j (welches im T2PP hinterlegt wird, wo es durch den Kontextwechsel-Mechanismus von MOSES dynamisch aktiviert werden kann). Dabei wird wie oben beschrieben eine *callback*-Schnittstelle für die Zugriffsfunktion f jeder Teilpolitik P_i eingerichtet, welche die entsprechende Funktion automatisch innerhalb des *eval*-Aufrufs von PCO_j auswertet.

In dieser Implementierungsstudie werden keine weiteren Annahmen zur Authentisierung der Kommunikationspartner während der Gruppenbildungsphase gemacht; grundsätzlich kann hier jede Implementierung einer Publik-Key-Infrastruktur zum Einsatz kommen.

Domäneninteraktion

Sobald nach abgeschlossener Gruppenbildung ein Zugriff über Domänengrenzen hinweg erfolgen soll, sendet der Klient j eine Anfrage an einen Server i, bspw. zum Herunterladen einer

Datei *slides.pptx*. Diese Botschaft wird wie folgt behandelt:

1. Der zugehörige Serverprozess interpretiert die Anfrage und versucht gemäß MOSES-Architektur am jeweiligen PEP (z. B. lokales Dateisystem: PEP_i^{FS}), auf die angefragte physische Ressource zuzugreifen.
2. PEP_i^{FS} übernimmt die semantische Konvertierung der Anfrage in die Argumente der logischen Zugriffssteuerungsfunktion: eine auszuführende Aktion $a \in A$ (z. B. *get*) sowie eine oder mehrere Entitäten aus D_g, etwa *slides.pptx*. Diese Anfrage wird an PDP_i weitergeleitet.
3. PDP_i wertet nun die komponierte globale Politik P_g aus. Dabei wird der Wert $P_g = \bigwedge_{i \in G} C_i$ zurückgegeben, wobei für sämtliche C_i die *eval*-Methode des jeweiligen PCO_i im T2PP aufgerufen wird.
4. *eval* wiederum ruft bei der Auswertung des Kompositionsterms die benötigten Zugriffsfunktionen f_{P_j} aller Gruppenmitglieder j mit den jeweiligen Datenobjekten (aus dem übergebenen Entitätenvektor) auf; diese entscheiden anhand
   - der Zugehörigkeit der Argumente zur Sicherheitsdomäne D_j
   - der Auswertung der Politik P_j, falls diese lokal vorliegt
   - der Kommunikation mit Knoten j durch den PIO_j zur Auswertung von P_j für die Argumente der Zugriffsfunktion.
5. PEP_i^{FS} setzt die vom PDP_i returnierte Entscheidung durch, indem der Zugriff entweder erlaubt oder verboten wird. Dessen weitere Abwicklung erfolgt durch den Serverprozess.

5 Zusammenfassung

Dieses Papier stellte einen Methode zur sicheren Interaktion in sporadisch und spontan gebildeten Kooperationsgemeinschaften vor. Kern der Methode sind Regelsysteme, die das individuelle Verhalten eines Systems in spontanen Kooperationen beschreiben, in Form algebraischer Spezifikationen in der Sicherheitsarchitektur eines Systems hinterlegt sind und bei Fällen spontaner Kooperation zur automatischen Konstruktion von Kooperationspolitiken herangezogen werden.

Die Präzision der algebraischen Spezifikationen ermöglicht eine automatische Generierung der Kooperationspolitiken, die keine weitere Benutzerinteraktion erfordert und transparent im Hintergrund erfolgt. Die Einfachheit der Politikkonstruktoren befördert geringe Laufzeit- und Speicherkomplexität der Politikkonstruktion; ihre Lokalität bewahrt die Autonomie der an der Gruppenbildung beteiligten Systeme und reduziert Kommunikationskosten auf den Austausch eines algebraischen Terms von sehr geringer Größe.

Die Lokalität der Politikentscheidungen im Verlauf einer Kooperation bewahrt ebenfalls die Autonomie der Einzelsysteme und reduziert Kommunikationskosten auf die Kommunikation der Ergebnisse lokal getroffener Politikentscheidungen.

Eine Implementierungsstudie zeigt die Umsetzbarkeit der Methode unter Verwendung heutiger Standardtechnologien.

Literatur

[And72] James P. Anderson. Computer Security Technology Planning Study. Technical Report ESD-TR-73-51, Air Force Electronic Systems Division, Hanscom AFB, Bedford, MA, USA, 1972. Also available as Vol. I, DITCAD-758206. Vol. II DITCAD-772806.

[BDCdVS02] Piero Bonatti, Sabrina De Capitani di Vimercati, and Pierangela Samarati. An Algebra for Composing Access Control Policies. *ACM Transactions on Information and System Security*, 5:1–35, February 2002.

[BDD+12] Sven Bugiel, Lucas Davi, Alexandra Dmitrienko, Thomas Fischer, Ahmad-Reza Sadeghi, and Bhargava Shastry. Towards Taming Privilege-Escalation Attacks on Android. In *19th Annual Network & Distributed System Security Symposium (NDSS)*, February 2012.

[BL76] D.E. Bell and L.J. LaPadula. Secure Computer System: Unified Exposition and Multics Interpretation. Technical Report AD-A023 588, MITRE, March 1976.

[EB04] Mark Evered and Serge Bögeholz. A Case Study in Access Control Requirements for a Health Information System. In *Proc. of the 2nd Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation – Volume 32*, ACSW Frontiers '04, pages 53–61, 2004.

[Fad06] Glenn Faden. Solaris Trusted Extensions - Architectural Overview, April 2006. Sun/Oracle White Paper.

[GM82] H. Garcia-Molina. Elections in a Distributed Computing System. *IEEE Transactions on Computers*, 31(1):48–59, January 1982.

[HK99] Udo Halfmann and Winfried E. Kühnhauser. Embedding Security Policies Into a Distributed Computing Environment. *Operating Systems Review*, 33(2):51–64, April 1999.

[HM90] Russell Housley and Sammy Migues. A Security Policy for Trusted Client-Server Distributed Networks. In *Proceedings of the 13th National Computer Security Conference*. NIST/NCSC, 1990.

[Hos92] Hilary H. Hosmer. The Multipolicy Paradigm. In *Proceedings of the 15th National Computer Security Conference*, pages 409–422. NIST/NCSC, 1992.

[HRS+10] Boniface Hicks, Sandra Rueda, Luke St.Clair, Trent Jaeger, and Patrick McDaniel. A Logical Specification and Analysis for SELinux MLS Policy. *ACM Transactions on Information Systems Security*, 13(3):26:1–26:31, July 2010.

[HRU76] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. Protection in Operating Systems. *Communications of the ACM*, 19(8):461–471, August 1976.

[Küh99] Winfried E. Kühnhauser. Policy Groups. *Computers & Security*, 18(4):351–363, 1999.

[KvKO95] Winfried E. Kühnhauser and Michael von Kopp Ostrowski. A Framework to Support Multiple Security Policies. In *Proceedings of the 7th Canadian Com-*

puter Security Symposium, pages 301–322. Communications Security Establishment Press, May 1995.

[Lam74] Butler W. Lampson. Protection. *Operating Systems Review*, 8(1):18–24, 1974.

[LS01] Peter A. Loscocco and Stephen D. Smalley. Integrating Flexible Support for Security Policies into the Linux Operating System. In Clem Cole, editor, *Proc. 2001 USENIX Annual Technical Conference*, pages 29–42, 2001.

[MFv11] Claudio Marforio, Aurélien Francillon, and Srdjan Čapkun. Application Collusion Attack on the Permission-Based Security Model and its Implications for Modern Smartphone Systems. Technical Report 724, ETH Zurich, April 2011.

[NR11] Prasad Naldurg and Raghavendra K. R. SEAL: A Logic Programming Framework for Specifying and Verifying Access Control Models. In *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies*, SACMAT 2011, pages 83–92. ACM, 2011.

[RCCF12] Giovanni Russello, Mauro Conti, Bruno Crispo, and Earlence Fernandes. MOSES: Supporting Operation Modes on Smartphones. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, SACMAT 2012, pages 3–12, New York, NY, USA, 2012. ACM.

[SC13] Stephen Smalley and Robert Craig. Security Enhanced (SE) Android: Bringing Flexible MAC to Android. In *20th Annual Network & Distributed System Security Symposium (NDSS)*, February 2013.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.

[SYRG07] Scott D. Stoller, Ping Yang, C.R. Ramakrishnan, and Mikhail I. Gofman. PRBAC and ARBAC Policies for a Small Health Care Facility, 2007. Online Documentation, http://www.cs.sunysb.edu/~stoller/ccs2007/healthcare.txt, accessed Feb 19, 2013.

[SZZ$^+$11] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones. In *18th Annual Network & Distributed System Security Symposium (NDSS)*, pages 17–33, San Diego, CA, February 2011.

[WV03] Robert Watson and Chris Vance. The TrustedBSD MAC Framework: Extensible Kernel Access Control for FreeBSD 5.0. In *In USENIX Annual Technical Conference*, pages 285–296, 2003.

[ZLN05] Xinwen Zhang, Yingjiu Li, and Divya Nalla. An Attribute-based Access Matrix Model. In *Proceedings of the 2005 ACM Symposium on Applied Computing*, SAC '05, pages 359–363. ACM, 2005.

[ZM04] Giorgio Zanin and Luigi Vincenzo Mancini. Towards a Formal Model for Security Policies Specification and Validation in the SELinux System. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies (SACMAT '04)*, pages 136–145. ACM, 2004.