Heinrich C. Mayr, Martin Pinzger (Hrsg.): INFORMATIK 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 497

Beherrschbarkeit komplexer Firmeninfrastrukturen durch eigenschaftsbasierte Sicherheit [1]

Helmar Hutschenreuter[2] und Dieter Hutter[3]

Abstract: Mit der zunehmenden Vernetzung von IT-Infrastrukturen hat auch die Frage der Sicherheit solcher Infrastrukturen eine zentrale Bedeutung erlangt. Vielfach sind die Betreiber dieser Infrastrukturen bei der Planung und Implementierung ihrer Sicherheitskonzepte auf sich allein gestellt. Umso wichtiger ist es, dass sie geeignete Hilfsmittel an die Hand bekommen, um in dem Spannungsfeld zwischen Compliance- und Sicherheitsanforderungen des Betriebs einerseits und der notwendigen Verfügbarkeit diverser Software andererseits geeignete globale und langfristige Sicherheitspolitiken entwerfen und umsetzen zu können. Dabei ist von zentraler Bedeutung, dass die IT-Verantwortlichen die Konsequenzen ihrer Entscheidungen für die Sicherheit ihrer Infrastruktur jederzeit überschauen und gegebenenfalls adjustieren können. Dieser Beitrag diskutiert mögliche Lösungen insbesondere unter Usability-Gesichtspunkten anhand eines Projekts SaferApps für eine sichere Ausführung von Fremdapplikationen in einer Unternehmensinfrastruktur.

Keywords: Nutzerzentrierte Sicherheit, Identity Management, Eigenschaftszentrierte Sicherheit

1 Einleitung

Es ist inzwischen ein tradierter Gemeinplatz, dass der Mensch das “schwächstes Glied” einer Sicherheitskette bildet [Sc00]. Die Gründe hierfür sind vielfältig. Die Absicherung eines Systems ist komplex und verlangt das Zusammenspiel vielfältiger Sicherheitsmechanismen. Die an das System zu stellenden Sicherheitsanforderungen sind von den Bedürfnissen der Anwender abhängig und können damit nicht fest vorgegeben werden. Die Erfassung seiner Bedürfnisse erfolgt aber in der Regel nicht in Termini seiner Applikationswelt, sondern orientiert sich an den zur Verfügung stehenden einzelnen (technischen) Sicherheitsmechanismen. Die Verantwortung für das geeignete Zusammenspiel der zum Einsatz kommenden Sicherheitsmechanismen wird dem Benutzer aufgebürdet, der mit dieser Aufgabe, die selbst für IT-Sicherheitsspezialisten schwer zu überschauen ist, komplett überfordert ist. Die Folge ist, dass der Benutzer die globalen oder langfristigen Konsequenzen seiner in Termini der technischen Sicherheitsmechanismen vorgenommenen Einstellungen nicht nachvollziehen kann. Die Situation gleicht einem Stellwerksbediensteten bei der Bahn, der für jede Weiche einzeln ihre Stellung festlegt, anstatt komplette kreuzungsfreie Streckenabschnitte für die Züge zu reservieren. Die Folgen sind hinreichend bekannt: Da die Konsequenzen für die Sicherheit eines Systems nicht abgeschätzt werden können, treten ritualisierte Entscheidungsmuster in den Vordergrund. Je nützlicher

[1] Diese Arbeit wurde vom deutschen Bundesministerium für Wirtschaft und Energie im Rahmen des Projektes SaferApps (FKZ KF2013021MS4) gefördert.
[2] DFKI GmbH, Cyber-Physical-Systems, Bibliothekstr.1, 28359 Bremen, helmar.hutschenreuter@dfki.de
[3] DFKI GmbH, Cyber-Physical-Systems, Bibliothekstr.1, 28359 Bremen, hutter@dfki.de

ein System vom Benutzer eingeschätzt wird, desto mehr wird er bereit sein, dem System implizit zu vertrauen und die von ihm angefragten Rechte einzuräumen, da die sicherheitstechnischen Konsequenzen seiner Entscheidung - wenn überhaupt - nur sehr diffus und damit zweitrangig bleiben (siehe z.B. [AC04]). Das ist z.B. auch bei Sozialen Netzwerken der Fall, bei denen Sicherheits- und Privatsphäreneinstellungen für Nutzer oft nur schwer verständlich sind [FNS15]. Der Versuch, den Benutzer lediglich über die Bedeutung einzelner Sicherheitsmaßnahmen aufzuklären, greift zu kurz, da die Konsequenzen aus dem Zusammenspiel der verwendeten Mechanismen für den Benutzer trotzdem nicht begreifbar werden. Bezeichnenderweise ist selbst eine Vielzahl von Systementwicklern oder Administratoren ratlos, wenn sie danach gefragt werden, welche zusätzliche Sicherheitsmaßname für ihr System den besten Zugewinn an Sicherheit leisten könnte.

Eine benutzerzentrierte Sicherheit muss daher von den gewünschten Sicherheitsanforderungen, d.h. von den Sicherheitseigenschaften eines Systems, ausgehen und aus denen die zu ihrer Garantie notwendigen Sicherheitsmechanismen (bevorzugt automatisch) ableiten. Dies umfasst insbesondere eine dem Verständnis des Benutzers angepasste Formulierung bzw. Formalisierung seiner (langfristigen) Sicherheitsbedürfnisse. Deren Umsetzung in technische Sicherheitsanforderungen sowie auch deren Dekomposition in sie garantierende Sicherheitsmechanismen sollte im Idealfall im System selbst erfolgen. Betrachtet man beispielsweise die Konfiguration der Sicherheitseinstellungen eines Smartphones (z.B. Android), so muss der Benutzer seine globalen Sicherheitsbedürfnisse auf entsprechende Zugangsberechtigungen einer App auf die verschiedenen Ressourcen (GPS, Kamera, Netz etc.) herunter brechen. Die Lösung dieser Aufgabe erfordert dabei beim Benutzer ein semantisches Modell sowohl über mögliche Verbindungen zwischen den einzelnen Ressourcen als auch über das Zusammenspiel der verschiedenen Applikationen.

Dieses Papier diskutiert Technologien für eine solche benutzerzentrierte Sicherheit anhand einer ähnlich gelagerten Problematik, in der es um die Einbindung von Drittapplikationen in eine Firmeninfrastruktur geht. Basierend auf einem integrierten Identity- und Infrastrukturmanagementsystem sollen Applikationen von Fremdanbieter sicher in die Firmeninfrastruktur integriert werden.

2 Eigenschaftsbasierte Sicherheit

Die Sicherheit eines Systems definiert sich in vielen Fällen durch die in dem System zum Einsatz kommenden Sicherheitsmechanismen. Beispielsweise macht sich vielfach in der Industrie die (IT-)Sicherheit eines Programms an dem Einsatz einer Firewall und eines Virenerkennungsprogramms fest. Eine Garantie für die Sicherheit des Systems können solche Mechanismen nicht bieten, da sie nur bestimmte Angriffstypen unterbinden können. Analog bedeutet beispielsweise die Garantie der Abwesenheit von Speicherüberläufen in einem Programm nicht, dass das Programm eine von dem Mechanismen unabhängige Sicherheitseigenschaft besitzt. Andere Programmierfehler, die von den eingesetzten Sicherheitsmechanismen nicht erkannt werden, könnten zum Beispiel ähnlich fatale Folgen für die Sicherheit des Systems haben wie die oben erwähnten Speicherüberläufe. Damit ist der Einfluss der Sicherheitsmaßnahmen auf eine Garantie bzw. einen Nachweis der vom

Benutzer geforderten Sicherheitsanforderungen nur indirekt gegeben und der Benutzer mit der Frage, welche Mechanismen wann und wie zum Einsatz kommen sollen, entsprechend überfordert.

Dementsprechend besteht ein großes Interesse an einem Mechanismen-unabhängigen Sicherheitsbegriff, der sich über garantierte (positive) Sicherheitseigenschaften eines Systems definiert. Idealerweise sind solche Sicherheitseigenschaften gesucht, die bezüglich der Komposition sicherer Einzelsysteme oder über verschiedene Verfeinerungsebenen skalieren. Ein klassisches Beispiel eines solchen eigenschaftsbasierten Sicherheitsbegriffs ist die Abwesenheit von Informationsflüssen zwischen bestimmten Ein- und Ausgabegrößen und damit die Unabhängigkeit eines sichtbaren Verhaltens eines Systems von etwaigen vertraulichen Daten in dem System. Speziell im Bereich der Formalen Methoden (und in Deutschland im Rahmen des DFG Schwerpunkts SPP 1496 "Reliably Secure Software Systems) wird an der Entwicklung solcher eigenschaftsbasierten Sicherheitsbegriffen gearbeitet, die mit Hilfe des Einsatzes korrespondierender Sicherheitsmechanismen garantiert werden können.

Eine eigenschaftsbasierte Sicherheit als eine deklarative Spezifikation einer Sicherheitspolitik lässt sich besser einem Benutzer vermitteln als deren Umsetzung im Sinne einer Kollektion von implementierten Sicherheitsmechanismen, da diese Eigenschaften in der Regel nicht an eine spezifische Implementierung gebunden sind sondern auch auf abstrakten Ebenen formulierbar sind. Abhängigkeiten zwischen Ein- und Ausgangsgrößen eines Systems können auch ohne Kenntnis einer Implementierung dem Benutzer vermittelt werden. Eine eigenschaftsbasierte Sicherheit ist nicht an spezielle Mechanismen und damit auf spezielle Abstraktionsebenen, auf denen die Mechanismen definiert sind, gebunden. Sie erlaubt damit die Modifikation der Techniken, die zur Garantie der Eigenschaften beitragen. Will man beispielsweise im Bereich der verschiedenen Applikationen für die Firmeninfrastruktur formalisieren, dass sensitive Firmendaten nicht zu unbefugten Dritten gelangen können, sind die Mechanismen, mit denen dies später erreicht werden kann, erst mal zweitrangig. Damit bilden diese formalen, mathematisch verifizierbaren Sicherheitsbegriffe eine Brücke zwischen den technischen Sicherheitsmechanismen eines Systems und dem intuitiven Sicherheitsverständnis des Benutzers.

3 Fallbeispiel SaferApps

Das Projekt SaferApps beschäftigt sich mit der sicheren Integration von Unternehmensanwendungen in bestehende IT-Infrastrukturen. Die in SaferApps entwickelten Lösungsansätze werden am Beispiel des Serverbetriebssystems *Univention Coporate Server (UCS)*[Un] erprobt, welches von dem Bremer Unternehmen *Univention* entwickelt wird. UCS bietet die Möglichkeit, Unternehmensanwendungen einfach über einen AppStore zu installieren. AppStores sind bereits aus dem Smartphone-Bereich bekannt und erlauben die einfache Installation von Applikationen auf einem System. So können sich Anwendungen (Apps) einfach auf einer UCS-Installation im Unternehmensnetzwerk bereitgestellt werden. Da eine ungeprüfte Integration von Anwendungen aus verschiedenen Quellen aber die Sicherheit des Unternehmensnetzwerkes korrumpieren kann, ist es ein Hauptziel von Sa-

ferApps einen Mechanismus für eine sichere Integration bereitzustellen. Dabei sind die Hauptherausforderungen aus der Sicht der Usability, dass diese Integrationstechnik für die verschiedensten Apps anwendbar sein muss, ohne dass dabei signifikante Mehraufwände für die einzelnen Akteure, d.h. den Hersteller der App, den Betreiber des AppStore bzw. den Administrator der Firmeninfrastruktur anfallen. Dabei treffen die unterschiedlichen Interessen der beteiligten Akteure aufeinander. Der Betreiber des AppStore möchte zur Steigerung seiner Attraktivität eine Vielzahl verschiedener Apps in seinem Store anbieten gleichzeitig aber vermeiden, dass mögliche in einer App verborgene Schadware die Infrastruktur seiner Kunden korrumpieren kann und somit die Vertrauenswürdigkeit des AppStore selbst in Mitleidenschaft zieht. Andererseits möchte er den Aufwand, eine App in seinem Store anbieten zu können, auf ein Minimum begrenzen. Analog zu Smartphones besteht die Lösung dieses Problems in der Bereitstellung von Sandboxen in UCS, in denen einzelne Apps, abgeschottet von anderen Apps und kritischen Basisdiensten des Betriebssystemen, ausgeführt werden.

Abbildung 1 gibt einen Überblick über den Lösungsansatz von SaferApps. Jede App ist in einer eigenen Sandbox gekapselt. Die Grenzen der Sandboxes sind durch eine doppelte Umrandung gekennzeichnet. Das UCS-System stellt Apps verschiedene Betriebsressourcen, insbesondere ein Dateisystem, ein LDAP-Verzeichnis sowie Zugriff auf das Netzwerk, zur Verfügung. Apps nutzen diese Ressourcen zur Kommunikation mit UCS und anderen Apps sowie zur Speicherung von Daten, wie beispielsweise zum persistenten Ablegen von Konfigurationen. Zur Installation neuer Apps verfügt der AppStore über ein SaferApps-spezifisches Installationsprogramm. Dieses setzt eine Zugriffskontrolle an den Grenzen der Sandbox auf, bevor es die neu installierte App innerhalb der Sandbox startet.

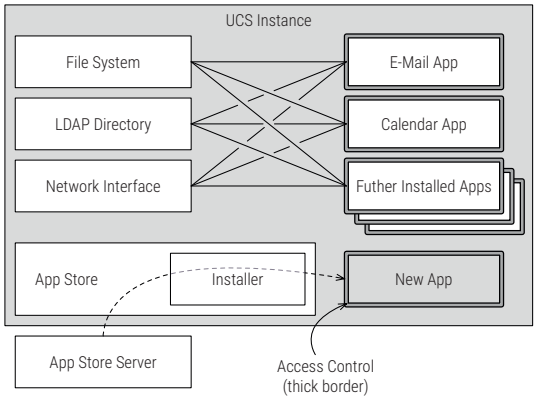

Abb. 1: SaferApps-Ansatz

Diese Sandbox wird mit Hilfe der Linux-Container-Technologie Docker [Do] realisiert. Dafür wird eine Anwendung zunächst als sogenanntes Docker Image bereitgestellt und anschließend auf den UCS-Server heruntergeladen. Ein beliebiger Prozess (bspw. ein Server-Dienst wie ein Webserver) kann auf Basis dieses Image als sogenannter Docker Container gestartet werden und getrennt von anderen Prozessen des Servers laufen. Der Vorteil von Docker ist, dass ein Container nativ im Linux-Kernel des Wirtsystems ausgeführt wird, keine Hardware emuliert werden muss und somit im Vergleich zu bestehenden Virtualisierungslösungen besonders ressourcensparend arbeitet. Docker erlaubt es, externe Ressourcen in den Container einzubinden. So lassen sich Dateien und Verzeichnisse des Hostsystems in das Dateisystem des Gastsystems im Conatiner einbinden oder Netzwerkzugriffe

auf die Netzwerkschnittstelle des Hostsystems an die des Gastsystems weiterleiten. Am Ende entscheidet die Konfiguration des Containers, ob die App einen Zugriff innerhalb oder außerhalb der Sandbox ausführt, sodass die App nicht zwischen Ressourcen des Wirtbetriebssystems und denen ihrer Sandbox unterscheiden muss. Jede Kommunikation eines Docker-Containers nach Außen wird über den typbasierten Zugriffskontrollmechanismus AppAmor [Ap] überwacht. Für die Installation und den Betrieb von Apps, die beispielsweise Groupware-Funktionalitäten anbieten (E-Mail, Kalender, etc.), benötigen diese Apps partiellen Zugriff zu der im LDAP des Hostsystems realisierten Benutzerverwaltung. LDAP-Verzeichnisse sind Datenbanken, die bereits ihre eigene Zugriffskontrolle mitbringen, die sich über *Access Control Lists* (ACLs) konfigurieren lässt. Einträge sind in einem LDAP-Verzeichnis in einer Baumstruktur angeordnet und bestehen aus einer Liste von Key-Value-Paaren, den Attributen. Einträge, für die eine ACL gelten soll, lassen sich über ihre Position im Baum oder über Eigenschaften ihrer Attribute adressieren. Da bekannt ist, wie das Dateisystem oder LDAP-Verzeichnisse Daten intern verarbeiten, kann durch die Analyse von Zugriffen auf diese Ressourcen auf konkrete Informationsflüsse geschlossen werden. Bei Netzwerkdiensten ist nicht bekannt, wie diese Daten intern verarbeiten. Meist befinden sich diese auch außerhalb des Einflussbereichs des UCS-Systems, da sie nicht lokal ausgeführt werden. Daher ist hier eine Abschätzung des Informationsflusses durch die Ressource nicht möglich.

4 Usability in der Praxis

4.1 Ausgangslage

Wie bereits erwähnt, wird zur Kontrolle der Zugriffe einer Sandbox die Zugriffskontrollmechanismen AppArmor und LDAP-ACLs verwendet. Die Konfiguration dieser Zugriffskontrollen wird automatisch aus den Regeln einer Sicherheitspolitik generiert, die gemeinsam mit der App im AppStore hinterlegt ist und bei einem Installationswunsch mit ausgeliefert wird. Im Unterschied zu Smartphones definiert nicht der Endbenutzer sondern der Systemadministrator die (globalen) Sicherheitsbedürfnisse. Damit fallen Ansätze, die Usable Security über die Handlungen des Endnutzers definierten wie z.B. [Ye04, Ro12], aus. Das Konzept von SaferApps wird maßgeblich durch die Sicherheitsanforderungen und Anforderungen der Nutzer beeinflusst. Die zentrale Sicherheitsanforderung ist, dass die automatisiert konfigurierte Zugriffskontrolle zuverlässig arbeitet und ausschließlich der App die Zugriffe genehmigt, die durch ihre Sicherheitspolitik erlaubt sind. Die Anforderungen der Nutzer unterscheiden sich, orientieren sich aber gemeinschaftlich an einer einfachen Bedienung dem intuitiven Verständnis der Funktionsweise von SaferApps.

Um einen AppStore mit einer großen Bandbreite von Anwendungen füllen zu können, ist es wichtig, dass die Barrieren für App-Hersteller möglichst klein gehalten werden. Daher sollte ein Mechanismus, wie SaferApps, keinen signifikanten Mehraufwand für den Hersteller verursachen. SaferApps sieht vor, dass zu jeder App eine Sicherheitspolitik spezifiziert werden muss, die beschreibt, welche Zugriffsrechte die App zur Ausübung ihrer Funktionalität benötigt. Denkbar ist auch eine strukturierte Politik, die in Abhängigkeit

der benötigten Funktionalitäten die angeforderten Rechte reduziert. Inwieweit dies von den Herstellern allerdings unterstützt werden wird, ist fraglich.

Für den Betreiber des AppStor steht die Wartbarkeit des Store im Vordergrund. AppStores, wie sie von Mobilplattformen bekannt sind, lassen neue Anwendungen einen Zulassungsprozess durchlaufen. Hierbei wird geprüft, ob die Anwendung den Anforderungen des AppStore genügt. Dabei steht nicht zwingend die Sicherheit im Vordergrund, sondern auch ihre Inhalte oder ob sie rechtlichen Rahmenbedingungen entsprechen, die vom jeweiligen Betreiber des Stores vorgegeben werden. Unser Ziel war es, dass SaferApps keine weiteren Überprüfungen des AppStore Betreibers erfordert. Insofern wird die vom App-Hersteller formulierte Sicherheitspolitik nicht vom AppStore-Betreiber überprüft, sondern der Administrator einer UCS-Installation soll über eine Benutzeroberfläche die Zugriffsmöglichkeiten in Bezug auf seine individuelle IT-Infrastruktur regeln, denn er muss letztlich die Verantwortung für eine in seinem Unternehmensnetzwerk installierte App tragen und trifft damit die abschließende Installationsscheidung.

Der Administrator ist verpflichtet, die Compliance- und Sicherheitsrichtlinien seines Unternehmens umzusetzen. Er benötigt deshalb ein Werkzeug, welches ihm ermöglicht zu prüfen, ob eine App diesen Richtlinien entspricht. SaferApps soll diese Prüfung insofern unterstützen als es den Administrator informiert, falls eine App den technischen Sicherheitsrichtlinien widerspricht. Das wird dadurch realisiert, indem die Richtlinien auf eine globale Sicherheitspolitik abgebildet werden, die auf der jeweiligen UCS-Installation hinterlegt wird. Die vom Hersteller vorgegebenen Politiken der Apps werden bei der Installation mit dieser Anwenderpolitik abgeglichen. Ein Schwerpunkt bei SaferApps ist umfassende und verständliche Information des Administrators. SaferApps stellt ein Werkzeug zur Verfügung, welches Kommunikationswege mit bereits installierten Anwendungen und dem UCS-System aufzeigt und diese (sofern vorhanden) mit Begründungen des App-Hersteller versieht. Das gibt dem Administrator eine Hilfestellung beim Verständnis von Seiteneffekten und Auswirkungen auf die Vertraulichkeit bestimmter Unternehmensdaten, die durch die Installation einer neuen App entstehen. Mit der zusätzlichen Möglichkeit bestimmte Funktionen von Anwendungen zu deaktivieren (sofern der App-Hersteller dieses unterstützt), kann der Administator die vorgegebene Sicherheitspolitik einer App nachträglich gemäß seinen Anforderungen adjustieren.

4.2 Sicherheitspolitik

Analog zur Smartphone-Welt liefern App-Hersteller ihre Apps mit einer korrespondierenden Sicherheitspolitik aus, die die benötigten Zugriffsrechte der App definiert. Diese Politik kann strukturiert sein, indem sie Rechte einzelnen (optionalen) Funktionalitäten der App zuordnet, und damit es ermöglicht, die App auch mit eingeschränkten Rechten und damit eingeschränkter Funktionalität zu verwenden. Die von einem App-Hersteller bereitgestellte Sicherheitspolitik ist eine sortierte Liste von Zugriffsregeln. Bei der Auswertung der Politik ist die Reihenfolge der Regeln maßgeblich. Analog zu LDAP-ACLs gilt im Allgemeinen, dass für einen Zugriff immer die erste Regel gilt, deren Vorbedingung erfüllt ist. Dieses Konzept regelt damit einerseits den Umgang mit gegenseitig konkurrie-

renden Regeln und erleichtert andererseits die spätere Übersetzung der Zugriffsregeln für die LDAP in die standardmäßige ACL-Syntax.

SaferApps-Sicherheitspolitiken sind Metapolitiken, die nicht zur Laufzeit der App ausgewertet sondern zum Installationszeitpunkt einmalig ausgewertet werden, um die eigentliche Zugriffskontrolle (automatisiert) zu konfigurieren. Dabei wird die Sicherheitspolitik in die benötigten Konfigurationsdateien der verwendeten Zugriffskontrollmechanismen übersetzt. Bei dieser Übersetzung werden insbesondere auch Optimierungen, wie beispielsweise das Entfernen überflüssiger Regeln, vorgenommen.

Während im Falle des Dateisystems und des LDAP-Verzeichnis die Strukturen und der Zugriff der Apps auf die verschiedenen Speicherbereiche offensichtlich ist und dementsprechend disjunkte Bereiche den einzelnen Apps zugeordnet werden können, ist für einen Netzwerkdienst oft keine Struktur (bzw. Abgrenzung von Speicherbereichen) bekannt. Daher wird bei Daten, die von einer App über das Netzwerk versandt werden, immer davon ausgegangen, dass diese von einer anderen App über eine beliebige Netzwerkverbindung wieder empfangen werden können. Somit wird das Netzwerk in seiner Gesamtheit als einzelner (nicht weiter unterteilbarer) Speicherbereich gesehen, und Netzwerkzugriffsregeln dienen dazu die Konfiguration einer Firewall abzubilden.

Per Vergleich der Zugriffsregeln zweier Apps lässt sich herausfinden, ob ein Informationsfluss zwischen den Apps stattfinden kann. Informationsflüsse werden aus sogenannten Regelkollisionen impliziert. Eine Regelkollision entsteht, wenn sich Zugriffsrechte und Speicherbereiche zweier Regeln überschneiden. Die im Folgenden vorgestellte Regelsyntax zeigt, dass Regeln ähnlich wie postalische Adressen aufgebaut sind. Diese geben das Ziel einer Sendung über die Angabe von mehreren Attributen an. Wie die Attribute einer Postadresse den Weg einer Sendung vorgeben, so geben auch die Attribute einer Zugriffsregel die Route zu einem bestimmten Speicherbereich vor. Über einen Vergleich der Attribute zweier Regeln lässt sich feststellen, ob sich ihre Speicherbereiche überschneiden oder nicht. Allerdings ist anzumerken, dass eine Regelkollision lediglich auf einen potentiellen Informationsfluss hindeutet. Ob ein Informationsfluss tatsächlich stattfindet, hängt von den tatsächlichen Zugriffen der Apps ab. Diese werden jedoch von SaferApps nicht berücksichtigt. So ist es möglich, dass Informationsflüsse statisch erkannt werden, die dynamisch nicht auftreten. SaferApps geht von einem Informationsfluss von App1 nach App2 aus, wenn App1 schreibenden und App2 lesenden Zugriff auf einen gemeinsamen Speicherbereich (Dateisystem oder LDAP) hat.

Um den Beteiligten die Formulierung einer Sicherheitspolitik zu erleichtern, ist die Syntax der Regeln sehr einfach gehalten. Eine Regel ist eine Menge von Attribut-Werte-Paaren, welche leicht in gebräuchlichen Datenformaten, wie *XML*, *JSON* oder *YAML* abgebildet werden können. Jede Zugriffsregel muss die Standardattribute `type` und `permission` beinhalten. `type` gibt an, ob die Regel Rechte für Datei-, Netzwerk- oder LDAP-Zugriffe erteilt. `permission` gibt an, welche Zugriffsrechte erteilt werden. Gültige Werte des Attributs `permission` sind `ro` (read only) und `rw` (read/write). Um Netzwerkregeln intuitiver formulieren zu können, existieren zudem die Aliasnamen `listen` (Alias für `ro`) und `connect` (Alias für `rw`). Zudem existiert das optionale Attribut `features`. Dieses gibt an für welche Features der App die Regel benötigt wird. Fehlt dieses Attribut, wird ange-

nommen, dass die Regel essentiell für die Funktion der App ist und nicht an bestimmte Features gebunden ist. Neben den Standardattributen enthält jede Zugriffsregel abhängig von ihrem Typ weitere Attribute (vgl. Tabelle 1).

Attribut	**Beschreibung**
Standardattribute	
`type`	Typ der Zugriffsregel (Datei, Netzwerk, LDAP)
`permission`	Rechte, die diese Regel erteilt
`features`	Features, für die diese Regel erforderlich ist
Attribute für Regeln zum Dateizugriff	
`path`	Datei- oder Verzeichnispfad
Attribute für Regeln zum Netzwerkzugriff	
`host`	Hostname, IP-Adresse
`port`	Portnummer
`protocol`	Verwendung von TCP oder UDP
Attribute für Regeln zum LDAP-Zugriff	
`area`	LDAP-DN und ggf. DN-Selektor
`filters`	LDAP-Filter
`permitted_attrs`	Eingrenzung auf Attribute

Tab. 1: Attribute der verschiedenen Regeltypen

Falls eine App Funktionen besitzt, die optional sind und somit nicht zwingend für ihre Hauptfunktion erforderlich sind, kann der App-Hersteller diese ebenfalls in der Sicherheitspolitik definieren. Eine solche Feature-Definition ist, wie Regeln auch, ebenfalls eine Menge von Attribut-Werte-Paaren. Sie besitzt die Standard-Attribute `id`, `name` und `description`. `id` ist ein eindeutiger Identifier des Feature, der frei gewählt werden kann. Dieser Identifier wird verwendet, um im Regelattribut `features` auf ein Feature zu verweisen. Über die Attribute `name` und `description` soll der App-Hersteller durch die Wahl eines eines prägnaten Names und über einen Beschreibungstext das Feature für den Nutzer bzw. den Administrator einer UCS-Installation der App beschreiben.

4.3 User Interface

In SaferApps ist der Installationsprozess einer neuen App zentral. Hier wird die Zugriffskontrolle konfiguriert, wobei sich ihre Konfiguration aus der Sicherheitspolitik des App-Herstellers und den Konfigurationswünschen des Administrators der UCS-Installation, auf der die neue App installiert wird, ergibt. Um die Informationen aus diesen beiden Quellen zusammenzuführen, verwendet SaferApps eine graphische Benutzeroberfläche (GUI), die dem Administrator Informationen über die Konsequenzen der Installation gibt und ihm auch die Möglichkeit gibt, die Konfiguration der Zugriffskontrolle zu beeinflussen und auf diese Weise die vom App-Hersteller vorgegebene Sicherheitspolitik zu adjustieren.

GUIs vieler Installationsprogramme folgen einem Wizard-Konzept, um mit dem Benutzer zu interagieren. Nach diesem klickt sich der Nutzer durch eine Folge von Dialogfenstern, die aufeinander aufbauen. SaferApps verwendet ebenfalls einen Wizard. Im ersten Schritt

(siehe Abbildung 2) fragt der Wizard vom Administrator ab, welche Funktionalitäten der neuen App tatsächlich genutzt werden sollen. So können die Zugriffe bereits im Vorfeld eingeschränkt werden, indem Zugriffe für nicht benötigte Funktionalitäten automatisch ausgeschlossen werden. Durch die Entscheidung für oder gegen einzelne Funktionalitäten trifft der Administrator indirekt auch die Entscheidung für oder gegen einzelne Zugriffsregeln der Sicherheitspolitik der App. Durch die Verknüpfung mit den Funktionalitäten fällt ihm dieses einfacher, da sich aus den Zugriffsregeln nicht logischerweise bestimmte Funktionalitäten ableiten lassen, während umgekehrt für einzelne Funktionalitäten korrespondierende Zugriffe erforderlich sind.

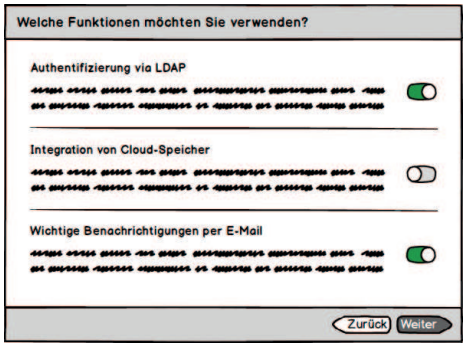

Abb. 2: Einschränkung der App-Funktionalität

Im zweiten Schritt (siehe Abbildung 3) wird der Administrator über Informationsflüsse informiert, die zwischen der neuen App und bestehenden Apps auftreten. Diese werden in verschiedenen Kategorien gruppiert. So gibt es Senden, Empfangen und bidirektionale Zugriffe sowie Netzwerkzugriffe. Falls in einer Kategorie keine Zugriffe vorhanden sind, wird die Kategorie komplett ausgeblendet. Während man auf Smartphone-Betriebssystem Apps Zugriffe auf bestimmte Arten von Daten (Kontakte, Kalender, etc.) erlauben kann, lässt sich aufgrund der individuellen Beschaffenheit der Installationen von Unternehmensservern die dort verarbeiteten Daten nicht eindeutig bestimmten Kategorien zuordnen. Daher werden die Daten nach den Apps gruppiert, die sie verarbeiten, um so dem Administrator eine intuitiv erfassbare Kategorisierung anzubieten.

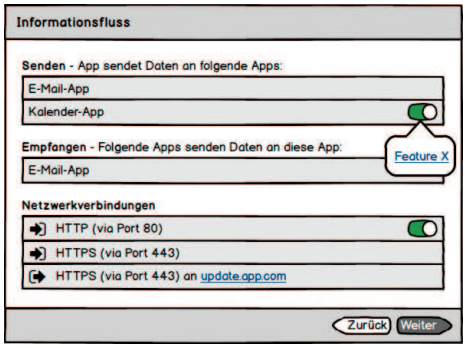

Abb. 3: Information den Informationsfluss

Es ist anzumerken, dass die Menge der Informationen in diesem Schritt von der Anzahl und Art der bereits auf dem System installierten Apps abhängt. Treten nur wenige Regelkollisionen auf, so ist die Anzahl der Informationsflüsse gering. Somit erhält der Administrator gezielt die Zugriffe präsentiert, die tatsächlich einen Einfluss auf den Informationsfluss haben. Alle andere Zugriffe der Sicherheitspolitik muss er nicht prüfen.

Der Administrator ist an vorhandene Compliance- und Sicherheitsrichtlinen seines Unternehmens gebunden und muss diese durchsetzen. Deshalb ist es wichtig, dass er im Fall eines sich ergebenden Informationsflusses, der diesen Richtlinien widersprechen würde, die Sicherheitspolitik korrigieren kann. Eine Adjustierung ist allerdings nur bei Informationsflüssen möglich, die an Funktionalitäten der App gebunden sind, die im ersten

Schritt des Wizard ausgewählt werden konnten (also optional sind). Der Administrator soll so nicht in eine Situation geraten, in der eine installierte App nicht korrekt funktioniert, weil ihre Zugriffe von ihm zu rigide eingeschränkt wurden. Informationsflüsse, die sich verhindern lassen, sind mit einem On-Off-Switch versehen. Bei einem Klick auf diesen wird in einem Pop-Over angezeigt welche Funktionen der App von diesem Informationsfluss abhängen. Mit einen zusätzlichen Klick werden diese Funktionen nachträglich abgewählt und die aus ihnen resultierenden Zugriffe gesperrt.

5 Fazit

SaferApps zeigt, dass die Benutzerfreundlichkeit des Sicherheitskonzeptes wesentlich davon abhängt, dass die Konfiguration der Sicherheitsmechanismen in ihrer Konsequenz auf die Sicherheitseigenschaften des Gesamtsystems illustriert werden. Umgekehrt wäre es auch wünschenswert, dass aufgezeigt würde, welche Modifikation an den Sicherheitsmechanismen welche Effekte auf die Sicherheitseigenschaften haben. SaferApps ist ein erster Schritt in diese Richtung, wobei die angezeigten Informationsflüsse zwischen den Apps bisher die internen Flüsse einer App nicht berücksichtigen. Für (wesentlich kleinere) Smartphone-Apps ist eine solche feinere Informationsflussanalyse bereits heute automatisch machbar (e.g [Ar14]). Eine Analyse der Methodik hinsichtlich eines Security-Usablility-Threat-Modell (z.B. [KFR10]) könnte helfen, um einen Kompromiss für die Granularität einer solchen Informationflussanalyse zu ermitteln.

Literaturverzeichnis

[AC04] Aytes, K.; Connolly, T.: Computer Security and Risky Computing Practices: A Rational Choice Perspective. J. of Organizantional and End User Computing, 16(3):22–40, 2004.

[Ap] AppAmor: , Linux application security system. http://wiki.apparmor.net.

[Ar14] Arzt, S.; Rasthofer, S.; Fritz, C. et al.: FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps. In: Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation. PLDI '14, ACM, New York, NY, USA, 2014.

[Do] Docker: , An open platform for distributed applications. www.docker.com.

[FNS15] Foltz, C.B.; Newkirk, H.E.; Schwager, P.H.: An Empirical Investigation of Factors that Influence Individual Behavior toward Changing Social Networking Security Settings. J. of Theoretical and Applied Electronic Commerce Research, 11(2):1–15, 2015.

[KFR10] Kainda, R.; Flechais, I.; Roscoe, A. W.: Security and Usability: Analysis and Evaluation. In: 10th Int. Conference on Availability, Reliability, and Security (ARES 2010). 2010.

[Ro12] Roesner, F.; Kohno, T.; Moshchuk, A.; Parno, B.; Wang, H. J.; Cowan, C.: User-Driven Access Control: Rethinking Permission Granting in Modern Operating Systems. In: 2012 IEEE Symposium on Security and Privacy. 2012.

[Sc00] Schneier, Bruce: Secrets & Lies - Digital Security in a Networked World. John Wiley & Sons, 2000.

[Un] Univention: , Univention Corporate Server UCS. www.univention.de/produkte/ucs/.

[Ye04] Yee, Ka-Ping: Aligning security and usability. IEEE Security Privacy, 2(5):48–55, 2004.