This is a post-peer-review, pre-copyedit version of an article published in HMD. The final authenticated version is available online at: http://dx.doi.org/10.1365/s40702-016-0258-1

Security-by-Design in der Cloud-Anwendungsentwicklung

Andreas Schoknecht, Gunther Schiefer, Murat Citak, Andreas Oberweis
Karlsruher Institut für Technologie (KIT), Institut AIFB, Kaiserstr. 89, 76133 Karlsruhe
<vorname>.<nachname>@kit.edu, Tel.: +49 (721) 608 46588, Fax: +49 (721) 608 44548

Schlüsselwörter: Cloud Computing, Sicherheit, Attribute-based Access Control, kontextbasiertes Zugriffsmodell, Security-by-Design

Abstract

Unternehmen erkennen zunehmend die ökonomischen und operationalen Vorteile von Cloud Computing, die es ihnen ermöglichen, sowohl signifikante Kosteneinsparungen zu erzielen als auch den Einsatz neuer Software-Anwendungen zu beschleunigen. Der Einsatz von Cloud Computing erfordert jedoch eine zunehmende Betrachtung neuer Herausforderungen an die Sicherheit von Daten, die eine große Barriere für eine breitere Akzeptanz von Cloud Computing sind. In diesem Artikel werden Erkenntnisse aus dem von der EU geförderten Projekt PaaSword vorgestellt, welches das Ziel verfolgt, das Vertrauen in Cloud Computing zu erhöhen. In diesem Projekt wird ein holistisches Datensicherheits-Framework entwickelt, wobei der Fokus auf Software-Entwicklern liegt, die bei der Entwicklung von sicheren Cloud-Anwendungen und –Diensten unterstützt werden sollen. Dazu wird zunächst das zugrundeliegende Architektur-Konzept für eine sichere Speicherung von Daten vorgestellt, um dann vertieft auf die kontextbasierte Zugriffskomponente einzugehen. Zentraler Aspekt dieser Zugriffskomponente ist ein kontextbasiertes Zugriffsmodell, das von Entwicklern zur Annotation von Data Access Objects verwendet werden kann. Das Zugriffsmodell baut auf einem Attribute-based Access Control Modell auf. Dabei werden Zugriffsrechte gewährt, indem Zugriffsregeln ausgewertet werden, welche Kontextattribute berücksichtigen. Zu diesen Attributen können beispielsweise die IP-Adresse des anfragenden Geräts, die Art des verwendeten Geräts oder die Rolle des Nutzers innerhalb einer Organisation gehören. Im PaaSword-Zugriffsmodell werden Aspekte konzeptualisiert, die bei der Auswahl von Datenzugriffsregeln beachtet werden sollen und mit deren Hilfe das kontextbasierte Zugriffsmodell festlegt, auf welche Daten unter welchen Bedingungen zugegriffen werden darf. Die Formulierung der Regeln baut auf dem XACML-Standard auf, der es ermöglicht, einzelne Regeln mit Kontextbedingungen zu komplexeren Regelwerken zusammenzufassen.

Abstract English

Companies increasingly recognize the economical and operational advantages of Cloud Computing, which enables them to realize significant cost savings and to speed up the setup of software applications. Yet, the usage of Cloud Computing requires the consideration of new challenges regarding data security, which pose a serious threat to the adoption of Cloud Computing. This article presents results from the EU-funded PaaSword project, which aims at increasing the trust in Cloud Computing. A holistic data security framework will be developed during the project, whereby the focus is on software developers, who shall be supported during the development of secure cloud applications and services. Therefore, firstly, the underlying architecture concept for secure storage of data is introduced. The context-based access control component is described in further details afterwards. The central aspect of this access control component is a context-based access control model, which can be used by developers to annotate data access objects. The access control model itself builds upon an attribute-based access control model. Thereby, access rights are granted through the evaluation of access rules, which take context attributes into account. Such attributes might, e.g., be the role of a user within an organization, the IP address or type of the requesting device. The PaaSword access control model conceptualizes aspects which shall be considered during the selection of data access rules and with which the context-based access control model determines under which circumstances an access request on which data is allowed. The formulation of such rules is based on the XACML standard, which allows combining single rules with context conditions to more complex policies.

This is a post-peer-review, pre-copyedit version of an article published in HMD. The final authenticated version is available online at: http://dx.doi.org/10.1365/s40702-016-0258-1

1. Einleitung

Unternehmen erkennen zunehmend die ökonomischen und operationalen Vorteile von Cloud Computing, die es ihnen ermöglichen, sowohl signifikante Kosteneinsparungen zu erzielen als auch den Einsatz neuer Software-Anwendungen zu beschleunigen. Der Einsatz von Cloud Computing erfordert jedoch eine zunehmende Betrachtung neuer Herausforderungen an die Sicherheit von Daten, die laut dem LinkedIn Cloud Security Spotlight Bericht (Schulze 2015) die größte Barriere für eine schnellere Annahme von Cloud Computing sind. 90 % der 1.000 für diesen Bericht befragten Teilnehmer äußerten sich mäßig oder sehr besorgt bezüglich des Sicherheitsniveaus. In vielen Fällen werden jene Daten missbräuchlich verwendet, welche auf Datenträgern gespeichert sind. Im Arbeitsspeicher einer Cloud-Anwendung sind die Daten im Vergleich dazu in der Regel nur eine deutlich kürzere Zeitspanne verfügbar. Um den Schutz der Daten zu erhöhen, muss deshalb verstärkt der Schutz der persistent gespeicherten Daten ins Auge gefasst werden.

In diesem Artikel werden Erkenntnisse aus dem von der EU geförderten Projekt PaaSword[1] vorgestellt, welches das Ziel verfolgt, das Vertrauen in Cloud Computing zu erhöhen. Im Projekt wird ein holistisches Datensicherheits-Framework entwickelt, wobei der Fokus auf Software-Entwicklern liegt, die bei der Entwicklung von sicheren Cloud-Anwendungen und –Diensten unterstützt werden sollen. Diese Entwickler sollen über Annotationen im Code beispielsweise bestimmen können, welche Daten bei der Speicherung in einer Datenbank verschlüsselt werden sollen oder welche Kontextbedingungen bei einem kontextbasierten Zugriff gelten müssen. In dieser Hinsicht möchte das PaaSword-Projekt das Prinzip des Security-by-Design (Waidner et al. 2014) in der Praxis der Software-Entwicklung fester verankern. In diesem Artikel wird zunächst das zugrundeliegende Architektur-Konzept für eine sichere Speicherung von Daten vorgestellt, um dann vertieft auf die kontextbasierte Zugriffskomponente einzugehen.

Zentraler Aspekt dieser Zugriffskomponente ist ein kontextbasiertes Zugriffsmodell, das von Entwicklern zur Annotation von Datenzugriffsobjekten (sog. Data Access Objects, DAO) verwendet werden kann. Das Zugriffsmodell baut auf einem Attribute-based Access Control Model (ABAC) (Hu et al. 2014) auf. Bei ABAC werden Zugriffsrechte gewährt, indem Regeln ausgewertet werden, welche Kontextattribute berücksichtigen. Zu diesen Attributen können beispielsweise die IP-Adresse des anfragenden Computers, die Art des verwendeten Geräts, der momentane Aufenthaltsort oder die Rolle des Nutzers innerhalb einer Organisation gehören. Im PaaSword-Zugriffsmodell werden Aspekte konzeptualisiert, die bei der Auswahl von Datenzugriffsregeln beachtet werden müssen und mit deren Hilfe das kontextbasierte Zugriffsmodell festlegt, auf welche Daten unter welchen Bedingungen zugegriffen werden darf.

Wenn die möglichen Kontextarten und deren Verarbeitung feststehen, können die Entwickler Zugriffsregeln zu Datenzugriffsobjekten hinzufügen. Im Projekt werden dazu die Möglichkeiten von Annotationen in Java genutzt und erweitert. Die Formulierung der Regeln baut auf dem XACML-Standard (Parducci et al. 2013) auf. XACML ermöglicht es, einzelne Regeln mit Kontextbedingungen anhand von Kombinationsalgorithmen zu komplexeren Regelwerken („Policies“) zusammenzufassen. Zudem wird eine ontologische Beschreibung von Regeln eingesetzt, die z. B. eine automatische Plausibilitätsprüfung ermöglicht.

Eine Beschreibung dieses kontextbasierten Zugriffsmodells stellt den Fokus des vorliegenden Artikels dar (Kapitel 3). Dabei wird insbesondere auf die Formulierung von Zugriffsregeln eingegangen. Im 2. Kapitel wird zunächst das PaaSword-Framework zur Entwicklung sicherer Cloud-Anwendungen vorgestellt, um die Hintergründe zur Entwicklung des Zugriffsmodells zu erläutern. Der Artikel endet mit einer Zusammenfassung in Kapitel 4.

2. Entwicklung sicherer Cloud-Anwendungen mit Hilfe des PaaSword-Frameworks

Ziel des PaaSword-Projektes ist die Entwicklung eines Frameworks, um sichere Cloud-Anwendungen entwickeln zu können, um somit die Verbreitung dieser Anwendungen zu unterstützen. Dabei stehen nicht konkrete Sicherheitsmechanismen wie neuartige Verschlüsselungsalgorithmen im Fokus, sondern Anwendungsentwickler sollen bei der Entwicklung durch das Framework unterstützt werden. Abb. 1 zeigt zunächst die konzeptuelle Architektur des PaaSword-Frameworks, mit dessen Hilfe Anwendungsentwickler Cloud-Anwendungen entwickeln können, die auf eine physisch verteilte und vollständig verschlüsselte Datenbank zugreifen. Zu den wesentlichen Bestandteilen des Frameworks zählen (i) der Policy Enforcement Mechanism zur Beschreibung und

[1] www.paasword.eu

Verwaltung von Zugriffsregeln sowie zur Annotation von Data Access Objects, (ii) eine Cloud-Anwendung, die mit Hilfe des PaaSword-Frameworks entwickelt wurde und (iii) eine physisch verteilte und verschlüsselte Datenbank. Im Folgenden werden zunächst die generellen Zusammenhänge innerhalb des PaaSword-Frameworks beschrieben, um dann anschließend in Kapitel 3 vertieft auf einen der Kernaspekte, das kontextbasierte Sicherheitsmodell (Context-aware Policy Access Model) einzugehen.

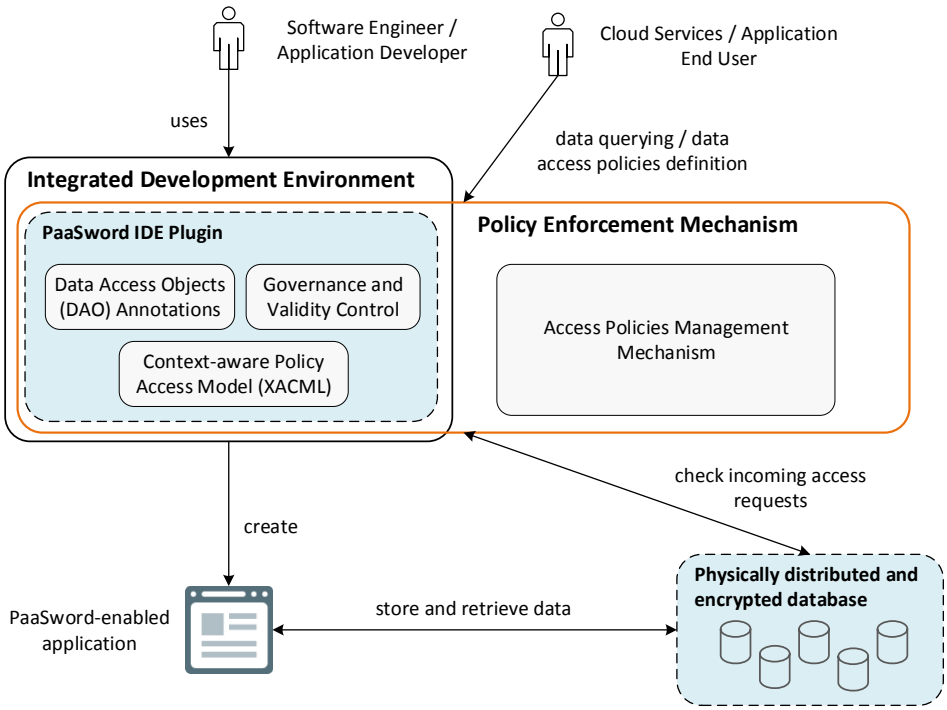

Abb. 1: Übersicht über die konzeptuelle PaaSword-Architektur

Im Rahmen des PaaSword-Projekts werden Bibliotheken entwickelt, die bei der Entwicklung von Java-basierten Cloud-Anwendungen die Annotation von Data Access Objects ermöglichen. Diese Annotationen sind dazu gedacht sensible Daten zu markieren, die geschützt werden sollen; um die physische Verteilung zu beschreiben; oder um die Art der Verschlüsselung und Zugriffsrechte festzulegen. Bei der Kompilierung einer Anwendung werden solche DAO-Annotationen auf ihre Validität hin überprüft und mit dem restlichen Code kompiliert. Der Policy Enforcement Mechanism überprüft dabei die Validität der beschriebenen Zugriffsregeln zur Kompilierzeit (Governance and Validity Control Komponente) und ist ebenso dafür zuständig Zugriffsberechtigungen eingehender Anfragen zur Laufzeit zu bestimmen. Darüber hinaus können zur Laufzeit weitere Zugriffsregeln definiert werden, sodass Regeln auch flexibel während des Betriebs einer Anwendung angepasst werden können. Die letztgenannten Optionen werden über die Access Policies Management Mechanism-Komponente realisiert. Weitere Details zur konzeptuellen Architektur des PaaSword-Frameworks können aus der PaaSword Referenzarchitektur (Gouvas und Michalas 2015) entnommen werden.

3. Kontextbasiertes Sicherheitsmodell

Für die Festlegung von Sicherheitsregelwerken zum Erstellzeitpunkt und zur Laufzeit benötigt der Policy Enforcement Mechanism ein Sicherheitsmodell, mit dem diese Regelwerke definiert werden können. Damit kann ein Entwickler schon beim Entwickeln von Anwendungen festlegen, welche sensiblen Daten einen besonderen Schutz benötigen. Mit Hilfe des Modells kann einerseits ein Sicherheitsprofil für die Speicherung dieser Daten an den Zugriffspunkt gebunden werden, andererseits können kontextabhängige Zugriffsregeln für den Zugriff auf diese Daten festgelegt werden.

3.1. PaaSword-Sicherheitsmodell

Für die Zusammenfassung der genannten Aufgaben in einem Modell berücksichtigt das PaaSword-Sicherheitsmodell die folgenden Aspekte:

- Sicherheitsrelevante Kontextelemente (Security Context Elements)
- Zugriffsberechtigungen (Permissions)

This is a post-peer-review, pre-copyedit version of an article published in HMD. The final authenticated version is available online at: http://dx.doi.org/10.1365/s40702-016-0258-1

- Kontextmuster (Context Pattern)
- Verteilungs- und Verschlüsselungselemente (Data Distribution and Encryption Elements, DDE)

Die ersten drei Aspekte werden für die Formulierung von Zugriffsregeln benötigt. Damit können Data Access Objects annotiert werden. Hierbei werden nicht nur statische Zugriffsregeln definiert, sondern auch kontextabhängige Zugriffsregeln für eine an die Situation dynamisch angepasste Zugriffskontrolle festgelegt. Dazu werden die zur Auswertung relevanten Kontextelemente festgelegt, auf deren Werten die jeweils aktuellen Zugriffsentscheidungen entsprechend der formulierten Zugriffsregeln beruhen. Kontextmuster stehen direkt in Verbindung mit sicherheitsrelevanten Kontextelementen, welche wiederum mit Zugriffsberechtigungen verknüpft sind. Die Kontextmuster werden verwendet, um Zugriffsentscheidungen abhängig von der bisherigen Nutzung zu treffen. Hierdurch können beispielsweise unübliche Datenzugriffe herausgefiltert werden oder es können als Informationsbarriere - im Sinne einer „Chinese Wall“ (Brewer und Nash 1989) - Zugriffe auf konkurrierende Datenteile verhindert werden, welche nicht vermischt werden dürfen.

Die sicherheitsrelevanten Kontextelemente beispielsweise sind in fünf Top-Level-Konzepte eingeteilt (Verginadis et al. 2016):

- Location: Beinhaltet Informationen über den Ort (z. B. physikalischer Standort oder Netzwerk-Standort), an dem die Daten gespeichert werden oder von dem aus auf die Daten zugegriffen wird.
- DateTime: Hierbei werden zeitliche Aspekte wie beispielsweise konkrete Zeitpunkte oder -intervalle betrachtet, die einen Zugriff charakterisieren.
- Connectivity: Dieses Kontextelement enthält Informationen, die in Beziehung zur Konnektivität stehen, welche von Subjekten verwendet werden, um auf sensible Daten zuzugreifen. Die Art des Geräts oder die Art der Verbindung sind Beispiele hierfür.
- Object: Objekte beziehen sich auf jede Art von Artefakten (wie z. B. (nicht) relationale Daten, Dateien oder Software-Artefakte), die gemäß ihrem Sensibilitätsgrad geschützt werden sollen.
- Subject: Subjekte stellen Agenten (wie z. B. Organisation, Personen oder Gruppen) dar, die Zugriff auf bestimmte Artefakte anfordern.

Die Verteilungs- und Verschlüsselungselemente (DDE) legen die Sicherheitsmechanismen für die Speicherung der Daten fest. Hier kann die kryptografische Stärke einer Verschlüsselung ebenso definiert werden wie die geforderte Fragmentierung oder die Verteilung von Datenteilen auf physisch verschiedene Server.

Aus Abb. 2 wird der Zusammenhang der Aspekte im kontextbasierten Sicherheits-Modell von PaaSword ersichtlich. Weitere Details zum kontextbasierten Sicherheitsmodell finden sich in dem PaaSword-Projektbericht Context-aware Security Model (Verginadis et al. 2016). Im folgenden Abschnitt wird näher auf die Formulierung von Zugriffsregeln unter Einbeziehung von Kontextregeln eingegangen.

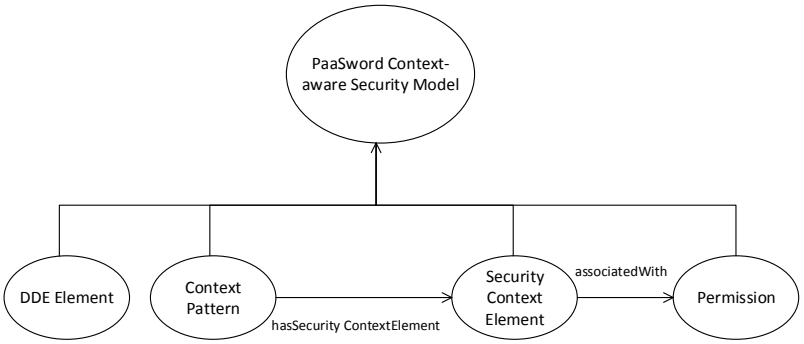

Abb. 2: Kontextbasiertes Sicherheitsmodell von PaaSword

3.2. Formulierung von Zugriffsregeln

Zugriffsmodelle sind dafür zuständig festzulegen, wer (welcher Nutzer) was (welche Operation) mit wem (welchen Objekten) durchführen darf. Objekte können dabei z. B. ein Dienst, ein Server, eine Datenbank oder eine einzelne Datenzeile in einer Datentabelle sein. Übliche Operationen sind schreiben, lesen, aktualisieren und löschen (CRUD[2]) von Daten. Der Nutzer ist das aktive Element und wird als Subjekt bezeichnet. Zugriffsregeln definieren damit die erlaubten Operationen, welche ein Subjekt mit einem Objekt durchführen darf.

Grundlegend können nach Decker (Decker 2011) folgende drei Formen der Zugriffskontrolle unterschieden werden, welche oft miteinander kombiniert werden: Discretionary Access Control (DAC), Mandatory Access Control (MAC) und Role-Based Access Control (RBAC). Beim Einsatz des DAC-Modells wird die Identität des

[2] **C**reate, **R**ead, **U**pdate, **D**elete

Benutzers verwendet, um Zugriffsentscheidungen zu treffen. Die Zugriffe auf Objekte können durch Benutzer gesteuert werden, falls diese Eigentümer dieser Objekte sind, weshalb diese Form auch als benutzerbestimmbare Zugriffskontrolle bezeichnet wird. Bei MAC hingegen liegt die Kontrolle der Zugriffsberechtigungen nicht bei den Benutzern, sondern das System (letztlich der Sicherheitsbeauftragte) legt systemweite Zugriffsstrategien fest, die durch Benutzer nicht mehr beeinflussbar sind. Der zentrale Aspekt für die Kontrolle der Zugriffe bei RBAC ist die Verwendung von Rollen, um Objektzugriffe über die Rollenzugehörigkeit eines Benutzers zu steuern. Eine weitere bekannte Form der Zugriffskontrolle ist das Attribute-based Access Control Model (ABAC), welches die Formulierung von Zugriffsregeln basierend auf Attributen erlaubt. Grundsätzlich wurde gezeigt, dass sich die Zugriffsmodelle DAC, MAC und RBAC mit ABAC abbilden lassen, sodass auch im PaaSword-Framework ABAC verwendet wird (Priebe et al. 2005).

Mit ABAC können beliebige Kontextattribute verwendet werden. Hier kann damit beispielsweise das gerade verwendete Endgerät, der momentane Aufenthaltsort oder die Art des verwendeten Zugangsnetzes (WLAN, Mobilfunk, usw.) als Kontextattribut genutzt werden. Ebenso kann die momentane Rolle eines Nutzers als dynamisches Attribut aufgefasst werden. Entsprechend dem ABAC-Modell werden Regelwerke (policies) als eine Sammlung von Regeln (rules) formuliert. Eine Regel legt fest, was ein Subjekt unter welchen Kontextbedingungen (context expression) für eine Autorisierung besitzt, um eine Aktion mit einem Objekt durchzuführen. In Abb. 3 ist eine ontologische Beschreibung der Zusammensetzung von Regelwerken dargestellt. Kontextbedingungen sind dabei eine beliebige Verknüpfung von Bedingungen der einzelnen Kontextelemente mittels der Booleschen Operatoren AND, OR, XOR und NOT. Diese Regewerke können nach ABAC noch zu Policy-Sets zusammengefasst werden, dies wird jedoch aus Gründen der Verständlichkeit im Weiteren weggelassen.

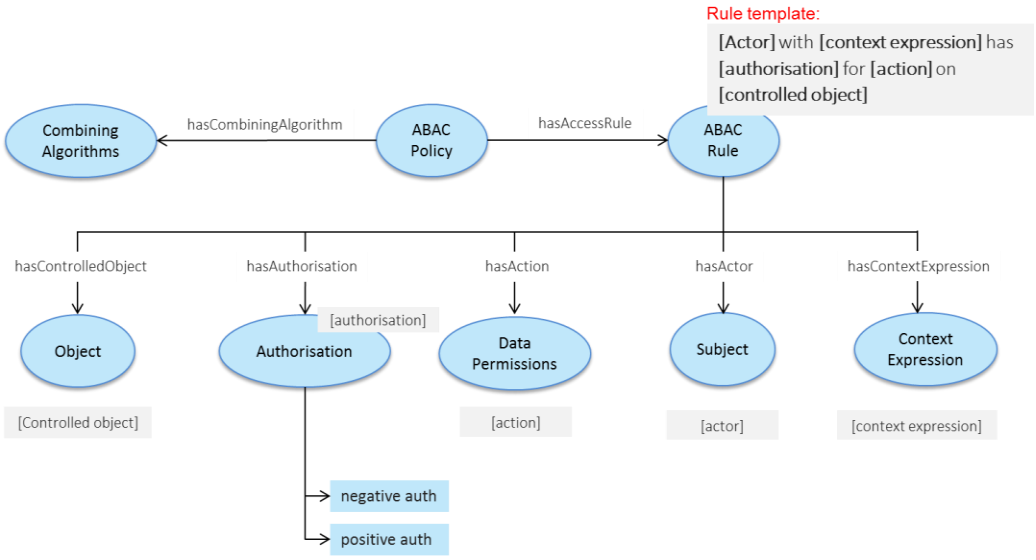

Abb. 3: Ontologische Beschreibung von Regelwerken (Verginadis et al. 2016)

Zudem bietet diese ontologische Repräsentation den Vorteil, die Kompatibilität von Zugriffsregeln durch automatisierte Reasoning-Verfahren überprüfen zu können. Sind z. B. mögliche, zulässige Örtlichkeiten für einen Datenzugriff bekannt, so können implementierte Regeln, die diese Ortsbeschränkungen überprüfen, mit diesem Wissen abgeglichen werden. Sollte ein unbekannter Ort in einer Regel spezifiziert sein, kann diese Regel abgelehnt werden (Veloudis und Paraskakis 2016). Dadurch können letztlich Entwickler bei der Beschreibung und Verwendung von Zugriffsregeln unterstützt werden, indem sie auf Inkonsistenzen oder Widersprüche aufmerksam gemacht werden.

3.3. Anwendung und Auswertung von Zugriffsregeln

Als Beispiel soll hier ein Parkhaus dienen, bei dem Kunden die Parkgebühren unter anderem mit ihrer Kreditkarte bezahlen können. Die Kreditkartendaten sind als schützenswert eingestuft und werden deshalb besonders geschützt. Für die Mitarbeiter im Parkhaus muss die Möglichkeit bestehen, aus den Daten zu entnehmen, ob ein Kunde die Parkgebühren bezahlt hat bzw. manuell Bezahlvorgänge vorzunehmen. Der Zugriff auf die Bezahldaten soll jedoch auf das Parkhaus und die Arbeitszeiten beschränkt werden. Durch diese Einschränkungen wird erreicht, dass die sensiblen Bezahldaten nicht unabsichtlich an öffentlichen Orten mit unbefugten Zuschauern von den Mitarbeitern eingesehen werden können, oder unbefugt von den Mitarbeitern von anderen Orten und außerhalb der Arbeitszeiten geändert werden. Dazu kann die folgende Regel festgelegt werden:

ParkingEmployee with (Location = CarPark AND Time = WorkingHour) has positive auth for Read on PaymentsTable

This is a post-peer-review, pre-copyedit version of an article published in HMD. The final authenticated version is available online at: http://dx.doi.org/10.1365/s40702-016-0258-1

Diese durch das Template aus Abb. 3 beschriebene Regel realisiert einen Teil der oben erläuterten Zugriffskontrolle. Sie kontrolliert das Objekt Bezahldaten (*PaymentsTable*). Sie ist gültig für einen Nutzer (Subjekt), welcher die Rolle Parkhausmitarbeiter hat (*ParkingEmployee*) und für den gleichzeitig die Kontextbedingungen *Location=CarPark* und *Time=WorkingHour* zutreffen. Die Regel gibt eine Erlaubnis (*positive auth*) für das Lesen der Daten (*Read*). Um das oben beschriebene Szenario zu vervollständigen, muss eine zweite Regel definiert werden, bei der die erlaubte Aktion „*Write*" anstelle von „*Read*" lautet. Zusätzlich existiert eine Kombinationsregel für das Regelwerk, welche den Datenzugriff verbietet solange keine gültige Regel eine explizite Erlaubnis gibt. Bei der Auswertung der Regeln können die in Tabelle 1 dargestellten Fälle auftreten:

Subject	Object	Action	Context	Decision
Any person who is not a parking employee	`PaymentsTable`	`Read` or `Write`	Any context	Deny
`ParkingEmployee`	Any object other than `PaymentsTable`	`Read` or `Write`	Any context	Deny
`ParkingEmployee`	`PaymentsTable`	`Read` or `Write`	Location: Any other than `CarPark`	Deny
`ParkingEmployee`	`PaymentsTable`	`Read` or `Write`	Time: Any other than `WorkingHour`	Deny
`ParkingEmployee`	`PaymentsTable`	`Read` or `Write`	Location: `CarPark` Time:`WorkingHour`	Permit

Tabelle 1: Auswertungsfälle der Zugriffsregel

Die jeweilige Entscheidung wird aus folgenden Gründen getroffen: Da es keine Regel gibt, welche anderen Nutzern als den Mitarbeitern des Parkhauses einen Zugriff auf die Bezahldaten gibt, wird der Zugriff verweigert (Zeile 1). Da zudem keine Regel existiert, die den Mitarbeitern des Parkhauses einen Zugriff auf andere Objekte als die Bezahldaten gibt, wird der Zugriff ebenso verweigert (Zeile 2). Die Auswertung der Kontextbedingungen in den Zeilen 3 und 4 ergibt keine gültige Auswertung, sodass die Regel nicht zutrifft und der Zugriff verweigert wird. Erst wenn auch die Auswertung der Kontextbedingungen dazu führt, dass die formulierte Regel anwendbar ist, wird der Zugriff gewährt (Zeile 5).

Ein Entwickler kann dabei etwa für die Kontextbedingung `WorkingHour` die Zeit von 8:00 – 18:00 Uhr angeben. Sollte das Wissen über die Arbeitszeiten der Mitarbeiter des Parkhauses (z. B. zwischen 7:00 und 20:00 Uhr) für die Überprüfung der Regeln vorhanden sein, so kann der Entwickler auf die widersprüchliche Regel automatisiert aufmerksam gemacht werden. Somit wird eine mögliche Fehlerquelle reduziert.

4. Zusammenfassung und Ausblick

Der Artikel gibt einen Einblick in wesentliche Aspekte der kontextbasierten Zugriffskontrolle innerhalb des PaaSword-Projekts, welches Security-by-Design in der Cloud-Anwendungsentwicklung stärker verankern möchte. Dazu benötigen die Anwendungsentwickler Werkzeuge um Sicherheitsregelwerke passend zu den jeweils gewünschten Sicherheitsanforderungen schon bei der Entwicklung umzusetzen. Dazu wurde dargelegt, wie mit Hilfe eines auf ABAC basierenden Modells Regeln zur Zugriffskontrolle beschrieben werden können. Die verwendete ontologische Repräsentation erlaubt zudem ein automatisiertes Schlussfolgern über Zugriffsregeln sowie weiterem Wissen, sodass die Kompatibilität von implementierten Regeln überprüft werden kann.

Die ersten Werkzeuge von PaaSword zur Anwendungsentwicklung werden 2017 zur Verfügung stehen. Dies beinhaltet nicht nur das IDE-Plugin sondern auch den PaaSword-Container in dem der Policy Enforcement Mechanism für die Einhaltung der im Sicherheitsmodell definierten Regeln für die Datenspeicherung und den Datenzugriff sorgt.

Acknowledgement: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644814.

5. Literatur

Brewer D, Nash M (1989) The Chinese Wall Security Policy. Proceedings of IEEE Symposium on Security and Privacy. S. 206–214.

Decker M (2011) Modellierung ortsabhängiger Zugriffskontrolle für mobile Geschäftsprozesse. KIT Scientific Publishing, Karlsruhe

Gouvas P, Michalas A (2015): PaaSword Reference Architecture - Deliverable D1.3. https://www.paasword.eu/deliverables. Gesehen 01.05.2016

Hu V, Ferraiolo D, Kuhn R, Schnitzer A, Sandlin K, Miller R, Scarfone K (2014) Guide to Attribute Based Access Control (ABAC) Definition and Considerations. NIST Special Publication 800-162. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-162.pdf. Gesehen 01.05.2016

Parducci B, Lockhart H, Rissanen E (2013): OASIS eXtensible Access Control Markup Language (XACML) TC http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf. Gesehen 01.05.2016

Priebe T, Dobmeier W, Muschall B, Pernul G (2005) ABAC - Ein Referenzmodell für attributbasierte Zugriffskontrolle. Sicherheit 2005: Sicherheit - Schutz und Zuverlässigkeit, S. 285-296.

Schulze H (2015) LinkedIn Cloud Security spotlight report. http://media.scmagazine.com/documents/114/cloud-security-spotlight-repor_28381.pdf. Gesehen 01.05.2016

Veloudis S, Paraskakis I (2016) Access Policies Model - Deliverable D2.2. https://www.paasword.eu/deliverables. Gesehen 02.05.2016

Verginadis Y, Patiniotakis I, Mentzas G (2016) Context-aware Security Model - Deliverable D2.1. https://www.paasword.eu/deliverables. Gesehen 02.05.2016

Waidner M, Backes M, Müller-Quade J (2014) Development of Secure Software with Security by Design. Fraunhofer Institute for Secure Information Technology, SIT Technical Reports, SIT-TR-2014-03.