Flexible Autorisierung in Web Service-Föderationen

Martin Wimmer, Pia Ehrnlechner und Alfons Kemper

Technische Universität München – Lehrstuhl für Informatik III
D-85748 Garching bei München
{wimmerma|ehrnlech|kemper}@in.tum.de

Abstract: Die Web Service-Technologie stellt die Grundlage für Service Oriented Architectures (SOAs) dar. Dabei ist eine SOA im Wesentlichen ein Zusammenschluss von interagierenden Diensten. Neben dem einfachen Austausch von Daten bedeutet Interaktion insbesondere die Bildung höherwertiger Dienste aus elementareren. In diesem Beitrag stellen wir ein auf XACML basierendes Autorisierungssystem für SOAs vor. Es kann zum einen für die Zugriffskontrolle einzelner Dienste eingesetzt werden. Dienste sind zur Bereitstellung ihrer Funktionalität häufig auf weitere Betriebsmittel wie Datenbanken angewiesen. Da Datenbanksysteme meist eine eigenständige, von der der Dienste unabhängige Autorisierung durchführen, erfolgt dann eine mehrstufige Autorisierung. In diesem Beitrag werden Verfahren für die Abstimmung dieser Autorisierungsschritte vorgestellt. Zum anderen stellt das System Techniken für die organisationsübergreifende Weitergabe von Privilegien bereit. Dieser Delegationsmechanismus bildet die Grundlage sowohl für den Aufbau von schwach wie auch stark gekoppelten Web Service-Föderationen. Schwach gekoppelte Zusammenschlüsse basieren oft auf ad-hoc Interaktionen, wohingegen bei stark gekoppelten in der Regeln eine so genannte *Trusted Third Party* eine Vermittlerrolle einnimmt. Besondere Anforderungen ergeben sich in Bezug auf die Effizienz verteilter Autorisierung. Die Skalierbarkeit des vorgestellten Ansatzes wird durch den Einsatz von rollenbasierter Zugriffskontrolle (RBAC) und Caching-Techniken gewährleistet: Mittels RBAC wird der Verwaltungsaufwand für Rechtezuweisungen verringert. Caching von Autorisierungspfaden reduziert den Kommunikationsaufwand für die verteilte Autorisierungsüberprüfung.

1 Motivation

Web Services, auch Web Dienste oder nur Dienste genannt, entwickeln sich zum defacto Standard für die Integration von (meist betriebswirtschaftlichen) Anwendungen. Dabei geht es nicht nur um die Bereitstellung einzelner Applikationen als Web Services, sondern auch und insbesondere um die kombinierte Ausführung mehrerer Dienste zur Bildung höherwertiger Anwendungsdienste. Die Web Service-Technologie bildet somit die Grundlage für so genannte Service Oriented Architectures (SOAs). Der Zusammenschluss von Diensten, die möglicherweise von verschiedenen Dienstanbietern (Organisationen) bereitgestellt werden, wird auch als Web Service-Föderation bezeichnet. Um den illegitimen Zugriff auf Dienste und Betriebsmittel zu verhindern, muss der Aufbau von Föderationen über ein leis-

tungsfähiges Autorisierungssystem geregelt werden. Man unterscheidet zwischen zwei Arten, wie derartige Zusammenschlüsse gebildet werden können: Einerseits kann eine so genannte *Trusted Third Party*, die allen Beteiligten bekannt ist und von ihnen akzeptiert wird, eine Vermittlerrolle einnehmen. Diese Art von Zusammenschluss wird als *stark gekoppelt* bezeichnet. Auf der anderen Seite gilt es zur Bildung kurzzeitiger, dynamischer Föderationen, bei denen eine entsprechende übergeordnete Einheit nicht vorhanden ist, ad-hoc Interaktionen zu unterstützen. Vertrauen (in Form von Zugriffsrechten) wird vorübergehend zugewiesen, kann aber auch wieder entzogen werden. Man spricht dann von einer *schwachen Kopplung* der beteiligten Dienste, bzw. der zugehörigen Organisationen. Im Gegensatz zu stark gekoppelten Zusammenschlüssen ist hierbei essentiell, dass die beteiligten Organisationen ihre Autorisierungsautonomie behalten: Zugriffsregeln werden dezentral verwaltet und auch ausgewertet.

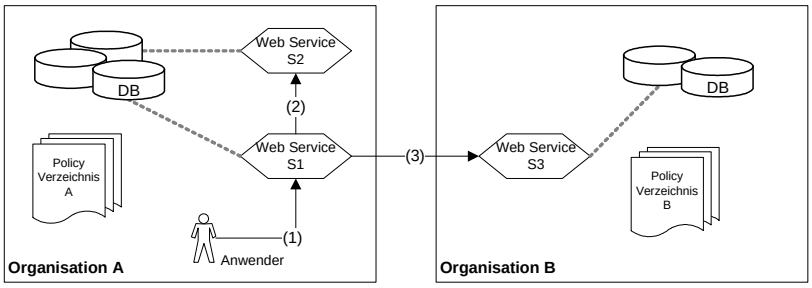

Abbildung 1: Dienstaufrufszenarien mit variierenden Zugriffskontrollen.

Abbildung 1 veranschaulicht unterschiedliche Szenarien für Dienstaufrufe und die damit verbundenen Anforderungen an die Zugriffskontrolle. Betrachtet man die Ausführung von S1 durch einen Benutzer, wie auch den Aufruf von S2 durch S1, welcher stellvertretend für den Anwender erfolgen kann (Schritte (1) und (2) der Abbildung), so handelt es sich hierbei um organisationsinterne Dienstaufrufe. In beiden Fällen erfolgt eine Zugriffskontrolle, die basierend auf lokalen Autorisierungsrichtlinien (Policies) entschieden wird. Die Mehrzahl heutiger Web Dienste greift während der Ausführung auf weitere Ressourcen – insbesondere Datenbanken – zu. Dabei erfolgt in der Regel eine erneute Zugriffskontrolle, die im Allgemeinen von der der Dienste unabhängig ist. Diese mehrstufige Zugriffskontrolle in Einklang zu bringen, ist ein Schwerpunkt des in diesem Beitrag vorgestellten Autorisierungssystems. In Abbildung 1 ist dies anhand konsolidierter Policy-Verzeichnisse für jede Organisation verdeutlicht. Werden Web Service-Föderationen geschlossen, mit dem Ziel komplexere, höherwertigere Funktionalitäten bereitzustellen, so erfolgt eine verteilte Autorisierung. Beispielsweise müssen bei der Ausführung von S3 Policies der Organisationen A und B ausgewertet werden. Im dargestellten Fall müssen dazu auf S1, bzw. dem Benutzer aus A entweder direkt oder indirekt (d.h. über weitere Organisationen, die an einer Föderation beteiligt sind) Rechte für die Ausführung von S3 übertragen werden. Benötigt werden folglich Techniken für die Delegation von Zugriffsrechten.

In [WEEK04] haben wir Techniken für die Abstimmung der Zugriffskontrolle von Web Servcies und von Datenbanksystemen vorgestellt. In diesem Beitrag erweitern wir diesen Ansatz dahingehend, dass Datenbanken und allgemeine Ressourcen nicht nur innerhalb abgeschlossener Vertrauensbereiche, sondern organisationsübergreifend mittels Web Service-Föderationen bereitgestellt werden können. Der Zugriff auf verteilte Datenquellen erfolgt auch in Grid-Anwendungen zunehmend über Web Service-Technologie [DAI, OGS]. [JD93] zeigt Anforderungen und Ansätze für die Zugriffskontrolle verteilter Datenbanksysteme, wie sie auch in unserem Ansatz aufgegriffen und im Grid/Web Service-Kontext behandelt werden. In [PVWKT02] wird ein Autorisierungssystem für Grid Anwendungen vorgestellt. Die Zugriffskontrolle für Zusammenschlüsse zu so genannten Virtual Organizations erfolgt dabei über spezielle Autorisierungsdienste. Dieser zentralisierte Ansatz ist für den Aufbau von Kollaborationsnetzwerken mit wechselnden Vertrauensbeziehungen jedoch nur bedingt anwendbar. Im hier vorgestellten Ansatz bleibt die Autorisierungsautonomie der beteiligten Organisationen erhalten. Zugriffsrechte werden lokal festgelegt und nicht an eine zentrale Stelle übertragen. Auch die Auswertung der Regeln wird dezentral durchgeführt. SDSD [BLP03] leistet die zustandsbasierte Zugriffskontrolle für den Zugriff auf verteilte Objekte und stellt in diesem Zusammenhang Techniken für die Delegation von Zugriffsrechten bereit. Das in diesem Beitrag behandelte Autorisierungsystem unterstützt neben einer direkten Weitergabe von Zugriffsrechten auch die Delegation von Rollen. Die Delegation von Rollen wurde unter anderem auch im dRBAC-System [FPP+02] eingeführt. Abhängig vom Vernetzungsgrad eines Zusammenschlusses ist eine Auswertung der verteilten Rollen- und Rechtezuweisungen kommunikations- und laufzeitintensiv. Ein Caching von Autorisierungsüberprüfungen wirkt dem entgegen. Der dRBAC Ansatz verwendet dazu einen Publish-and-Subscribe-Algorithmus: Eine Invalidierung von Cache-Einträgen wird bei Änderung von Zugriffsrechten angestoßen. Diese Vorgehensweise birgt Gefahren bei Netzwerkausfällen und führt zu verminderter Flexibilität der Rechteverwaltung. Aus diesem Grund verwenden wir in unserem Ansatz eine Validierung von Cache-Einträgen. Auch wenn das in diesem Beitrag vorgestellte Autorisierungssystem generischer Natur ist, so zielt es doch speziell auf die Anwendung im Web Service-Umfeld ab. Ein Austausch von Sicherheitsinformationen und ein verteiltes Identitätsmanagement lassen sich etwa über WS-Security und der darauf aufbauenden Spezifikation WS-Federation [KN+03] realisieren. Mit XLANG und WSFL, deren Ansätze in BPEL4WS [T+03] übernommen wurden, stehen XML-Grammatiken zur Beschreibung der Dienstinteraktion bereit.

Dieser Beitrag ist wie folgt gegliedert: Abschnitt 2 gibt einen Überblick über Autorisierungsmodelle und führt in die verwendete Notation und Rechteverwaltung ein. In Abschnitt 3 wird gezeigt, wie die Autorisierung von Web Services, die auf Datenbanken zugreifen, flexibel und zuverlässlich gestaltet werden kann. Abschnitt 4 beschreibt die lokale und verteilte Autorisierung. Der Einsatz von Caching-Techniken zur Effizienzsteigerung der verteilten Autorsierung wird in Abschnitt 5 gezeigt. Abschnitt 6 liefert eine abschließende Zusammenfassung der vorgestellten Techniken.

2 Grundlagen und Notation

Ein aus dem Datenbankbereich (vgl. [CFMS94]) bekanntes Autorisierungsmodell ist Discretionary Access Control (DAC). Über Zugriffsrechte werden Beziehungen zwischen Benutzern (den Subjekten) und Ressourcen (den Objekten) hergestellt. Ein Tupel

$$(\text{Subjekt},\ \text{Ressource},\ \text{Aktion},\ \text{Bedingungen},\ \text{Wirkung}) \tag{1}$$

legt fest, wer oder was (Subjekt) worauf (Ressource) wie (Aktion) und wann (Bedingungen) zugreifen darf – sofern es sich um ein Privileg handelt. Ob ein Privileg oder ein Verbot deklariert wird, ist durch die Wirkung festgelegt. Bedingungen schränken Privilegien/Verbote weiter ein. Einerseits sind hier globale Parameter wie die Zeit und andererseits auch Aktionsparameter denkbar. Ein Beispiel für letzteres ist, dass Ärzte nur auf Krankendaten von Patienten zugreifen dürfen, die sie behandeln. Rechte können durch den Eigentümer der Ressource auf andere Subjekte übertragen werden. Das Recht zur Weitergabe ist ggf. selbst wiederum übertragbar. Man unterscheidet zwischen positiven und negativen Autorisierungssystemen, abhängig davon, ob jeweils nur Privilegien oder nur Verbote ausgesprochen werden dürfen. Auch gemischte Systeme sind denkbar, allerdings nur dann, wenn Regeln für das Auflösen von Konflikten vorgesehen sind. Ferner muss festgelegt sein, wie sich das System verhält, falls sich keine explizite Genehmigung oder kein Verbot für einen geforderten Zugriff ableiten lässt. Entweder ist der Zugriff dann verboten (abgeschlossenes System) oder erlaubt (offenes System). Positive Systeme lassen sich mit dem ersten Modell vereinen, negative mit dem zweiten. Gemischte Systeme unterstützen beide. Das in diesem Beitrag vorgestellte Autorisierungssystem eignet sich für die Umsetzung dieser Kombinationen. Im Folgenden verwenden wir den Begriff Recht synonym für Privileg und beschränken uns auf positive Autorisierung für abgeschlossene Systeme. Da das in diesem Beitrag beschriebene Autorisierungssystem auch für den Aufbau von ad-hoc Interaktionen vorgesehen ist, bedarf es einer restriktiven Rechtevergabe, was andere Modelle ausschließt.

Rollenbasierte Zugriffskontrolle (RBAC [FSG+01]) reduziert den administrativen Aufwand, indem Privilegien nicht mehr explizit einzelnen Benutzern sondern Rollen zugewiesen werden. Benutzer erhalten Rechte über die Indirektion der Rollen. Hierarchisches RBAC erweitert dieses Konzept um Rollenhierarchien und führt dazu eine partielle Ordnung auf Rollen ein. Rechte- und Rollenzuweisungen werden mittels

$$[\texttt{Subjekt} \rightarrow_P \texttt{Privileg}]_{\texttt{Bedingungen}}\ \ \texttt{Herausgeber} \tag{2}$$

$$[\texttt{Subjekt} \rightarrow_R \texttt{Rolle}]_{\texttt{Bedingungen}}\ \ \texttt{Herausgeber} \tag{3}$$

ausgedrückt. Ein Privileg beschreibt einen Zugriff und fasst dazu Informationen über Ressourcen und zulässige Aktionen zusammen. Privilegien werden der Domäne (Organisation) zugeordnet, die auch die jeweilige Ressource kontrolliert. Herausgeber, Subjekte und Rollen gehören ebenfalls (möglicherweise verschiedenen)

Domänen an. Die Zuordnung zu Domänen wird im Weiteren durch Präfixnotation ausgedrückt. Unter einem Herausgeber versteht man die Entität, die die Rechtezuweisung vornimmt. Falls eine Zuweisung axiomatisch vorgenommen werden soll, d.h. allgemeingültig sein soll, ist der Herausgeber die Domäne selbst.

Die bisher vorgestellte Rechtevergabe erfolgt statisch. Zuweisungen der folgenden Form stellen die Grundlage für ein dynamisches Rechtemanagement, das die Delegation und den Entzug von Rechten und Rollen unterstützt, dar:

$$[\texttt{Subjekt} \rightarrow_{P}^{(\texttt{zuweisen}\,|\,\texttt{entziehen})} \texttt{Privileg}]_{\texttt{Bedingungen}}\ \texttt{Herausgeber} \tag{4}$$

$$[\texttt{Subjekt} \rightarrow_{R}^{(\texttt{zuweisen}\,|\,\texttt{entziehen})} \texttt{Rolle}]_{\texttt{Bedingungen}}\ \texttt{Herausgeber} \tag{5}$$

Subjekten wird dadurch das Recht zugewiesen, ein Privileg (eine Rolle) anderen Subjekten zuzuweisen (oder zu entziehen). Über zusätzliche Bedingungen kann dies eingeschränkt werden. Zum Beispiel kann festgelegt sein, dass ein Privileg (eine Rolle) wiederum nur bestimmten Subjekten zugewiesen (entzogen) werden kann. Die Weitergabe eines Privilegs (einer Rolle) erfordert nicht, dass das weitergebende Subjekt dieses Privileg (diese Rolle) selbst besitzt. Eine unerwünschte Selbstzuweisung kann über entsprechende Bedingungen ausgeschlossen werden.

Zur Repräsentation der Zuweisungen (2) bis (5) verwenden wir XACML [GM+03] – eXtensible Access Control Markup Language. Die Spezifikation legt zudem fest, wie Anfragen gegen Zugriffsrechte (Policies) ausgewertet werden. XACML basiert auf XML und eignet sich daher gut für die regelbasierte Autorisierung im Web Service-Umfeld, da sich die Web Service-Technologie ebenfalls auf XML stützt. Die zentralen Komponenten von XACML sind `Policy` und `Request`. Mit einer `Policy` werden Zugriffsrechte beschrieben. Abbildung 2 zeigt den Aufbau dieses Elements genauer: Eine `Policy` setzt sich aus einem `Target`, einer (möglicherweise) leeren Menge von Zugriffsregeln (`Rules`) und einer optionalen `Obligation` zusammen. Eine `Obligation` legt Operationen fest, die im Anschluss an eine `Policy`-Auswertung ausgeführt werden müssen. Der wichtigste Bestandteil einer Regel (`Rule`) ist ihr `Target`. Das `Target` einer `Rule` wie auch einer `Policy` legt fest, worauf die jeweilige Komponente anwendbar ist. Ein `Target` besteht aus einer Menge von `Subject`-, einer Menge von `Resource`- sowie einer Menge von `Action`-Elementen. Mittels des `Target`s einer Regel lässt sich ein Zugriffsrecht definieren, indem festgelegt wird, wer (`Subject`) worauf (`Resource`) wie (`Action`) zugreifen darf. Falls Teile eines `Target`s nicht weiter eingeschränkt werden sollen, wird `AnySubject`, `AnyResource` bzw. `AnyAction` verwendet. Eine optionale Einschränkung erfolgt über ein `Condition`-Element. Die Wirkung einer `Rule` wird über ihr `Effect` Attribut festgelegt, das entweder den Wert `Permit` oder `Deny` annehmen kann. Über XACML-`Rule` Elemente lassen sich also DAC-Zugriffsrechte gemäß der Tupelrepräsentation (1) ausdrücken. Die Domänenzuordung von `Subject`, `Resource` und `Action` erfolgt über `Issuer`-Attribute. Die Auswertung von Regeln wird über den so genannten `RuleCombiningAlgorithm` einer `Policy` festgelegt. Die Spezifikation sieht unter anderem `permit-overrides` und `deny-overrides` als mögliche Algorithmen vor, wodurch entweder Privilegien oder Verbote priorisiert werden. Da sich die folgenden Betrachtungen auf positive Autorisierung stützen, ist der `Effect` von Regeln

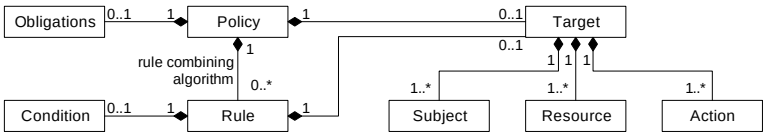

Abbildung 2: Aufbau der XACML-Komponenten (vgl. [GM+03]).

stets `Permit` und der Auswertungsalgorithmus entsprechend `permit-overrides`.

Um eine Zugriffskontrolle durchzuführen, wird ein XACML `Request` erzeugt. Ein `Request` setzt sich aus `Subject`-, `Resource`- und `Action`-Elementen zusammen. `Subject`s können zum Beispiel der Name eines Anwenders oder auch der Bezeichner eines Zertifikats sein. `Resource` und `Action` legen fest, worauf zugegriffen werden soll, und wie dieser Zugriff zu erfolgen hat. Ein `Request` wird gegen die Menge der Zugriffsregeln ausgewertet, die darauf anwendbar sind. Dabei heißt eine `Policy` anwendbar, wenn ihr `Target` auf den `Request` zutrifft. Dazu wird die Menge der `Subject`-Elemente eines `Target`s disjunktiv ausgewertet, d.h. eine Anwendbarkeit liegt dann vor, wenn zumindest eines der Elemente auf den `Request` zutrifft. Ein `Subject`-Element besteht im Allgemeinen aus mehreren Attributen. Diese werde konjunktiv ausgewertet. Falls also die `Subject`s-Menge des `Target`s aus zwei Benutzern besteht, wobei jeder durch einen Namen und eine E-Mail-Adresse identifiziert wird, so ist das `Target` dann zutreffend, wenn der `Request` die vollständige Information (Name und E-Mail) von mindestens einem der Benutzer aufführt. `Resource`- und `Action`-Mengen werden entsprechend ausgewertet. Das Ergebnis einer erfolgreichen Auswertung (`Response`) ist entweder `Permit` oder `Deny`, abhängig vom verwendeten `RuleCombiningAlgorithm` und der Ausprägung der Regeln. Falls keine anwendbare Regel gefunden wird, ist das Ergebnis `NotApplicable`. Da die zuvor angegebenen Formalismen entsprechend in XACML ausgedrückt werden können, werden im Folgenden Begriffe wie Subjekt mit den entsprechenden XACML-Ausdrücken, d.h. `Subject`, gleichgesetzt.

Basierend auf XACML werden drei unterschiedliche Policy-Typen generiert: Permission Policy, Base Policy und Role Assignment Policy. Der Grund für diese Strukturierung liegt in einer Reduktion des Verwaltungsaufwandes für die Autorisierung was die Skalierbarkeit des Ansatzes unterstützt.

Permission Policy Eine Permission Policy ist die XACML-Entsprechung eines Privilegs. Über die Angabe von Ressourcen- und Aktionen-Information wird die Zugriffsart beschrieben. Eine weitere Einschränkung auf spezielle Subjekte oder durch Bedingungen erfolgt im Allgemeinen nicht. Die Zugriffsinformation wird in einer Regel einer derartigen Policy kodiert. Das Target wird nicht weiter spezifiziert, so dass eine Zuweisung des Privilegs zu beliebigen Subjekten möglich ist.

Base Policy Mit Hilfe dieser Policy-Klasse werden Privilegien Subjekten zugewiesen. Das Target enthält dazu die Identität des jeweiligen Subjekts. Auf die Privilegien wird über `PolicyIdReference`-Elemente verwiesen.

```
<Policy PolicyId="ReadMedRecPermission" RuleCombiningAlgId="permit-overrides">
  <Target>
    <Subjects> <AnySubject /> </Subjects>
    <Resources> <AnyResource /> </Resources>
    <Actions> <AnyAction /> </Actions>
  </Target>
  <Rule RuleId="ReadMedRecRule" Effect="Permit">  <Target>
      <Subjects> <AnySubject /> </Subjects>
      <Resources>  <Resource>
          <DBObject>
            <Database>CH-Database</Database>
            <Table>Inpatient</Table>
            <Column>Therapy</Column>
          </DBObject>
      </Resource>  </Resources>
      <Actions>
        <Action>  <DBAction>select</DBAction>  </Action>
      </Actions>
    </Target>  </Rule>
</Policy>
```

Abbildung 3: Permission Policy für den lesenden Zugriff auf Patientendaten (gekürzt).

Role Assignment Policy Dieser Policy-Typ dient dazu, Benutzern Rollen zuzuweisen, bzw. im Sinne von hierarchischem RBAC Rollen Rollen zuzuweisen. In Anlehnung an [And04] fasst eine entsprechende Policy eine Menge von Regeln zusammen, die jede für sich genommen einer Menge von Subjekten eine bestimmte Rolle zuweist. Die Rolle stellt jeweils die Ressource der Regeln dar. Die Aktion ist eine Erweiterung des XACML-Standards, die die Zuweisung ausdrückt. Im Falle einer Delegation, d.h. der Zuweisung durch einen Anwender, wird vermerkt, wer diese Zuweisung ausgesprochen hat.

Der Einsatz der genannten Policy-Typen lässt sich anhand folgender Beispielanwendung nachvollziehen: Dr. Jeffrey Geiger des Chicago Hope Hospitals (mit Domänenname CH) soll das Recht erhalten, die Daten des Patienten Watters einzusehen. Die Abfrage soll dabei durch einen Web Service bereitgestellt werden. Dazu gehen wir wie folgt vor: Zuerst wir das Privileg spezifiziert, welches den Zugriff auf Patientendaten beschreibt. Dieses wird anschließend Dr. Geiger zugewiesen und für den Zugriff auf Watters eingeschränkt.

Wir setzen das Recht, den Dienst ausführen zu dürfen, mit dem Recht, auf die Patientendaten zugreifen zu dürfen, gleich. In der in Abbildung 3 dargestellten Permission Policy ist dies dadurch modelliert, dass die Ressource eine Spalte der Patientendatenbank referenziert und die Aktion ein select-Befehl darauf ist. Auf den Zusammenhang zwischen der Zugriffsverwaltung von Datenbanken und Web Services wird im nachfolgenden Abschnitt näher eingegangen. Dieses Privileg kann Dr. Geiger direkt über eine Base Policy zugewiesen werden. Der Administrationsaufwand für ein allgemeines Krankenhausszenario lässt sich durch die Einführung von Rollen reduzieren. In Abbildung 4 wird das Privileg deshalb der Rolle

```
<PolicySet PolicySetId="AttendingPhysicianPolicy"
                          PolicyCombiningAlgId="permit-overrides">
  <Target>
    <Subjects>  <Subject>
        <Role>CH.AttendingPhysicianRole</Role>
    </Subject>  </Subjects>
    <Resources> <AnyResource /> </Resources>
    <Actions>   <AnyAction />   </Actions>
  </Target>
  <PolicyIdReference> ReadMedRecPermission </PolicyIdReference>
</PolicySet>
```

Abbildung 4: Base Policy für behandelnde Ärzte (gekürzt).

CH.AttendingPhysicianRole zugewiesen. Die zugehörige formale Darstellung ist

$$[\texttt{CH.AttendingPhysicianRole} \rightarrow_P \texttt{CH.ReadMedRecPermission}]\,\texttt{CH}$$

Ärzten soll der lesende Zugriff auf die Daten der betreuten Patienten gestattet sein. Diese Bedingung lässt sich allgemein umsetzen, indem in obige Base Policy eine Abfrage eingefügt wird, ob der Zugriff durch einen behandelnden Arzt erfolgt[1]. Eine für das Beispiel vereinfachte Realisierung ist in Abbildung 5 gezeigt: Die Zuweisung der Rolle zu Dr. Jeffrey Geiger ist nur dann gültig, falls es sich um den Zugriff auf die Daten von Mr. Watters handelt. Formal:

$$[\texttt{CH.JeffreyGeiger} \rightarrow_R \texttt{CH.AttendingPhysicianRole}]_{(\texttt{patientId=CH.MrWatters})}\,\texttt{CH}$$

3 Konsolidierte Zugriffskontrolle

Mit dem beschriebenen Rechtemanagement erfolgt die Gestaltung der Zugriffskontrolle für Web Services in ServiceGlobe. ServiceGlobe [KSSK02] ist eine verteilte und erweiterbare Dienstplattform. Sie ist vollständig in Java 2 implementiert und basiert auf den Standards XML, SOAP, UDDI und WSDL. Dienste greifen während der Ausführung häufig auf weitere Betriebsmittel zu. In vielen Fällen handelt es sich dabei um Datenbanksysteme (DBMSs), wobei jedes DBMS in der Regel ebenfalls eine eigenständige Zugriffskontrolle durchführt. Innerhalb einer Organisation bzw. Domäne, deren Softwarelandschaft heterogen aufgebaut ist, hat man es somit mit einer mehrstufigen Autorisierung zu tun. Zuerst erfolgt eine Autorisierung auf Web Service-Seite und im Anschluss daran eine Zugriffskontrolle durch das DBMS. Dabei können die einzelnen Zugriffskontrollsysteme auf unterschiedlichen Autorisierungsmodellen aufbauen. Mit dem Ziel, die Transparenz der Zugriffskontrolle zu erhöhen und den Administrationsaufwand möglichst gering zu halten, haben wir Techniken für die Konsolidierung der Autorisierungskonzepte von Web Services und

[1] In XACML lässt sich dies mittels eines `AttributeSelector`-Elements repräsentieren.

```
<Policy PolicyId="RoleAssignmentPolicy" RuleCombiningAlgId="permit-overrides">
  <Target>
    <Subjects>  <AnySubject/>  </Subjects>
    <Resources> <AnyResource/> </Resources>
    <Actions>   <AnyAction/>   </Actions>
  </Target>
  <Rule RuleId="AttendingPhysicianAssignment" Effect="Permit">
    <Target>
      <Subjects> <Subject>CH.JeffreyGeiger</Subject> </Subjects>
      <Resources>  <Resource>
          <Role>CH.AttendingPhysicianRole</Role>
      </Resource>  </Resources>
      <Actions> <Action>enable</Action> </Actions>
    </Target>
    <Condition FunctionId="string-equal">
      <Apply FunctionId="string-one-and-only">
        <ResourceAttributeDesignator AttributeId="patientId"
          DataType="XMLSchema#string" Issuer="CH" />
      <AttributeValue DataType="XMLSchema#string">Mr.Watters</AttributeValue>
    </Condition>
  </Rule>
</Policy>
```

Abbildung 5: Beispiel einer Role Assignment Policy (gekürzt).

Datenbanken entwickelt und in ServiceGlobe integriert. Neben einem abgestimmten Identitätsmanagement benötigt man bei regelbasierter Zugriffskontrolle dazu eine Policy-Beschreibungssprache, mit der sich die verwendeten Autorisierungskonzepte ausdrücken lassen. Wie gezeigt, eignet sich XACML zur Umsetzung von DAC- und RBAC-Konzepten. Da zur Ausführung der Web Service-Funktionalität die Interaktion mit (mindestens) einem DBMS notwendig ist, lässt sich das Recht, eine bestimmte Dienstfunktion auszuführen, mit dem Recht, die dazu notwendigen DBMS-Operationen durchzuführen, assoziieren. Ausgehend davon entwickelten wir Vorgehensmodelle für die zuverlässige und flexible Generierung der Zugriffskontrolle für Dienste, die auf Datenbanken zugreifen (vgl. [WEEK04]). Abbildung 6 zeigt, wie die Erstellung der Dienst-Policy weitestgehend automatisiert erfolgt.

Ausgangspunkt ist die Spezifikation des Dienstes. Neben einer Beschreibung der Schnittstellen in Form eines WSDL-Dokuments ist vor allem die Interaktion des Dienstes mit der Datenbank von Bedeutung. Diese wird vom WSDL-Dokument getrennt angegeben. Bezogen auf ein darunter liegendes relationales DBMS bedeutet dies, dass eine Service-Methode mit entsprechenden SQL-Ausdrücken in Beziehung gesetzt wird. Ausgehend davon, werden aus den SQL-Operationen in einem ersten Schritt Ressourcen- und Aktionen-Informationen extrahiert (1). Basierend darauf wird die Menge der Zugriffsrechte formuliert, die zum Ausführen der Dienst-Methode notwendig ist (2). Die Benutzerverwaltung für den Dienst ist von der des DMBS in der Regel unterschiedlich. In Schritt (3) erfolgt deshalb deren Umsetzung, indem über entsprechende Base Policies die zuvor erzeugten Rechte den Dienst-Anwendern zugewiesen werden. Dieser Schritt kann im Gegensatz zu

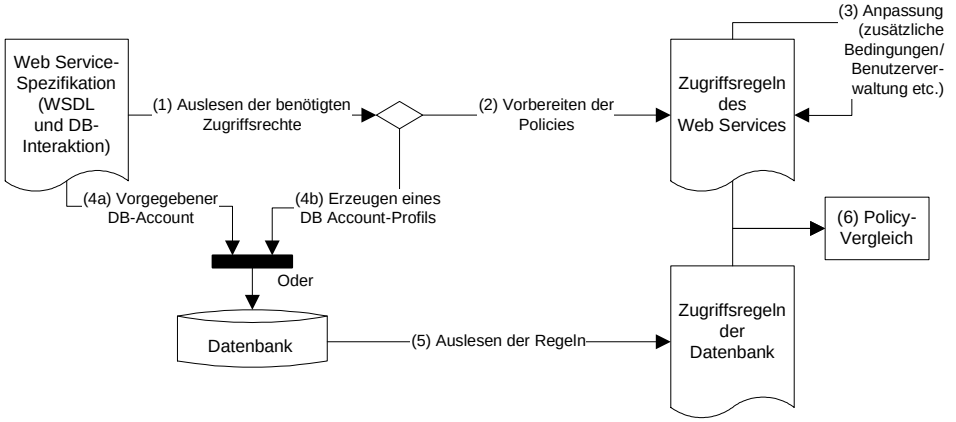

Abbildung 6: Verfahrensmodell für die Erstellung von Service-Policies.

allen anderen meist nicht vollständig automatisiert werden. Der Zugriff auf die Datenbank kann über einen fest vorgegebenen Datenbank-Account erfolgen, wie es in Schritt (4a) dargestellt ist. Ein nicht zu unterschätzendes Risiko heutiger Service-Realisierungen ergibt sich jedoch dadurch, dass hierfür oftmals überprivilegierte Accounts verwendetet werden. Zur Verminderung des Verwaltungsaufwandes werden Accounts oftmals mehrfach, d.h. für unterschiedliche Dienstimplementierungen verwendet, und sind deshalb mit umfangreicheren Zugriffsrechten ausgestattet, als für die tatsächliche Ausführung der jeweiligen Dienste nötig. Für den Fall, dass der Rechner auf dem ein Dienst ausgeführt wird, oder die verwendete Service-Plattform Sicherheitslücken aufweisen, kann dies dazu führen, dass Unautorisierte die Kontrolle über den Dienst erhalten, und damit Zugriff auf die darunter liegende Datenbank. Abhängig von der Fülle an Rechten, mit denen der Account ausgestattet ist, können so gravierende Risiken aus Datenbanksicht entstehen. Diese Risiken werden durch unser System in Schritt (4b) reduziert, indem basierend auf der Ressourcen- und Aktionen-Information ein abgestimmtes Benutzerprofil erzeugt wird. Der Zugriff ist dann nur noch innerhalb eines geeignet eingeengten „Zugriffskorridors“ möglich. In beiden Fällen, können die Rechte des Datenbank-Accounts automatisiert aus der Datenbank ausgelesen und mittels der zuvor eingeführten Policy-Typen in XACML kodiert werden. Diese Extraktion bildet die Basis für eine Kontrollfunktionalität. Denn abschließend werden die erstellten Web Service-Policies mit den extrahierten Datenbank-Policies verglichen. Nur wenn die Zugriffsregeln des Dienstes restriktiver oder höchstens entsprechend umfangreich wie die des Datenbank-Accounts sind, kann eine zuverlässige Autorisierung auf Dienstseite garantiert werden. Restriktiver bedeutet, dass weniger Subjekten eingeschränktere Rechte zugesprochen werden. Dies entspricht also einem geringeren Aktionsumfang auf einer reduzierten Ressourcenmenge. Eine Definition für eine partielle Ordnung auf Policies und Aussagen darüber, unter welchen Bedingungen ein Vergleich berechenbar ist, haben wir in [WEEK04] ausgearbeitet.

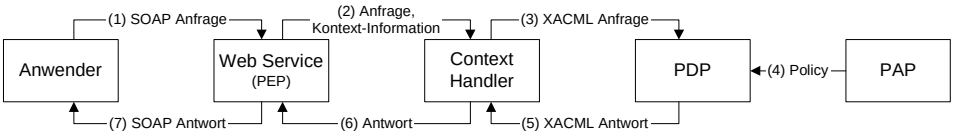

Abbildung 7: Autorisierungs-Workflow (angelehnt an [GM+03]).

4 Zusammenspiel lokaler und verteilter Zugriffskontrolle

Abbildung 7 zeigt die Aufrufschritte der Autorisierung von ServiceGlobe-Web Services. Ein Anwender – ein Benutzer oder ein Web Service – ruft einen Dienst via einen SOAP-Aufruf auf. Im Anschluss an die Authentifizierung des Anwenders erfolgt die vom Web Service initiierte Zugriffskontrolle. Der Web Service wird deshalb auch als Policy Enforcement Point, kurz PEP, bezeichnet. Dazu wird in einem nächsten Schritt ein XACML-Request erzeugt. Dies erfolgt durch einen so genannten Context Handler, der die Zugriffsinformation (Ressource und Aktion), die Identität des Anwenders (Subjekt) sowie ggf. zusätzliches Kontextwissen XACML-konform kodiert. Diese Anfrage wird dann durch den Policy Decision Point, PDP, ausgewertet. Der PDP ist eine eigenständige Komponente, die durch die Web Service-Plattform bereitgestellt wird. Sie wertet XACML-Requests gegen Zugriffsregeln aus, welche vom Policy Administration Point, PAP, bereitgestellt werden. Ein PAP ist eine Komponente, die mit den administrativen Rechten, Policies zu erstellen und zu verwalten, ausgestattet ist. Da wir im Folgendenen näher auf verteilte Autorisierung eingehen, sind PAPs Domänen, d.h. Organisationen, zugeordnet. Obwohl unser System dies nicht verlangt, nehmen wir vereinfachend an, dass jeder Domäne D genau ein PAP zugeordnet ist, der all die Zugriffsinformationen, die Objekte aus der Domäne D betreffen, zur Verfügung stellt. Dies sind Rechtezuweisungen der folgenden Form:

$$[\mathtt{D_S.Subjekt} \rightarrow_P \mathtt{D.Privileg}]_{\mathtt{Bedingungen}} \ \mathtt{D_H.Herausgeber}$$

$$[\mathtt{D_S.Subjekt} \rightarrow_R \mathtt{D.Rolle}]_{\mathtt{Bedingungen}} \ \mathtt{D_H.Herausgeber}$$

Die Subjekte, denen ein Recht bzw. eine Rolle zugewiesen wird, sowie das Entity, welches die Zuweisung erstellt (Herausgeber) können aus unterschiedlichen Domänen stammen. Bezeichnen D und D_S unterschiedliche Organisationen, so entspricht dies einer Weitergabe von Rechten (oder Rollen) über Domänengrenzen hinweg.

4.1 Lokale Zugriffskontrolle

Bei der Evaluierung der Zugriffsrichtlinien wird zwischen einer lokalen und einer verteilten Auswertung unterschieden, abhängig davon, ob eine Autorisierung basierend auf lokal verfügbaren Regeln entschieden werden kann, oder ob eine Anfrage an weitere Domänen notwendig ist. Die lokale Abarbeitung ist in Abbildung 8 dargestellt. Neben der Information über die Art des Zugriffs, d.h. welche Aktion

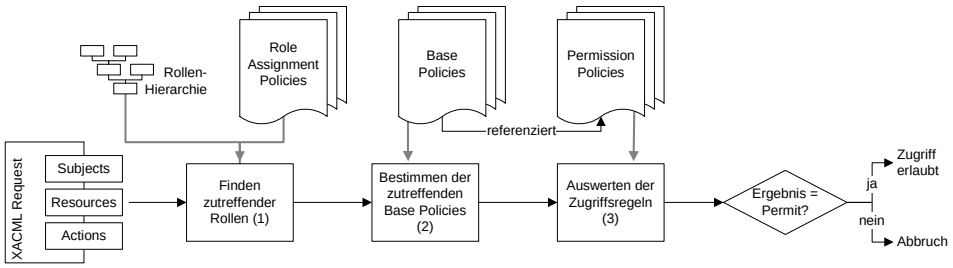

Abbildung 8: Lokale Autorisierung.

auf welcher Ressource ausgeführt werden soll, enthält der vom Context Handler erzeugte XACML-Request die Identität des Anwenders. Privilegien sind Anwendern jedoch häufig nicht ausschließlich über Base Policies zugewiesen, sondern auch über die Indirektion von Rollen. In einem ersten Schritt werden deshalb die Rollen ermittelt, die dem Anwender innerhalb der Domäne zugewiesen sind. Hierzu werden die Role Assignment Policies überprüft, in denen auch die Rollenhierarchie der Domäne abgebildet ist. Das Ermitteln der Rollenzugehörigkeit erfolgt über XACML-Techniken. Es werden dazu entsprechende Requests generiert und gegen die Role Assignment Policies ausgewertet. Die Rolleninformation wird dann in den initialen XACML-Request im Subjects-Abschnitt aufgenommen. Anschließend wird die Menge der Base Policies bestimmt, die für den Anwender direkt oder über Rollenzuweisungen zutreffen. Damit ist die Menge an Rechten festgelegt, die dem Anwender lokal zugewiesen ist. Abschließend wird die Anfrage gemäß der XACML Spezifikation bezüglich dieser Rechtemenge ausgewertet. Für den Fall, dass eine zutreffende Policy ermittelt wird und die Auswertung mit Permit abschließt, wird der Zugriff erlaubt und die Service-Operation ausgeführt. Andernfalls wird ein Zugriff basierend auf den lokalen Richtlinien verwehrt.

Eine Weitergabe von Rechten über Domänengrenzen hinweg kann in ServiceGlobe auf zwei Arten erfolgen. Zum einen ist es möglich, einem Benutzer aus einer Domäne D_s direkt ein Privileg innerhalb der Domäne D zuzuweisen. Dazu wird in D eine entsprechende Base Policy für den Benutzer D_s.s erzeugt. Fordert D_s.s einen Zugriff an, wozu das besagte Privileg nötig ist, so kann die Autorisierung lokal, d.h. in D, erfolgen. Zum anderen können Rechte indirekt über Rollenzuweisungen vergeben werden. Durch domänenübergreifende Rollenzuweisungen ergibt sich ein Zuweisungsgraph, dessen Knoten Rollen und dessen Kanten Delegationen repräsentieren. Für die Zugriffskontrolle gilt es, Autorisierungspfade, d.h. Pfade von Rollenzuweisungen, in diesem Zuweisungsgraphen zu ermitteln.

4.2 Verteilte Zugriffskontrolle

Eine verteilte Autorisierung wird erst dann gestartet, falls die zuvor durchgeführte lokale Auswertung gescheitert ist. Dazu wird in einem ersten Schritt die Menge

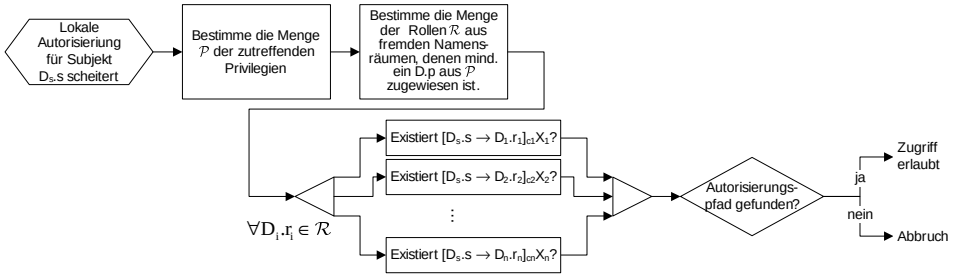

Abbildung 9: Verteilte Autorisierung.

$\mathcal{P}$ der Privilegien[2] bestimmt, die den im XACML-Request formulierten Zugriff gestatten würden. Damit eine positive Autorisierung zustande kommen kann, muss (mindestens) eines dieser Rechte (mindestens) einer Rolle aus einer fremden Domäne zugewiesen sein, und das anfragende Subjekt muss diese Rolle innehaben. Sei $\mathcal{R}$ die Menge der Rollen aus fremden Domänen, denen mindestens ein Recht aus $\mathcal{P}$ zugewiesen ist. Diese Zuweisung kann auch indirekt, d.h. über die Indirektion einer D-lokalen Rollenzuweisung, erfolgt sein. In folgender Definition steht deshalb $\twoheadrightarrow_P$ für die Zuweisung eines Privilegs zu einer Rolle, wobei diese Zuweisung direkt oder über die lokale Rollenhierarchie (also indirekt) erfolgt. X und X′ sind Platzhalter für Herausgeber von Zuweisungen. c_i und c' repräsentieren Bedingungen.

$$\mathcal{R} \stackrel{def}{=} \{ D_i.r_i | D_i.r_i \text{ ist eine Rolle}, D_i \neq D, \exists D.p \in \mathcal{P} : \exists [D_i.r_i \rightarrow_P D.p]_{c_i} X \vee \left(\exists D.r \text{ ist eine Rolle} : \exists [D_i.r_i \rightarrow_R D.r]_{c_i} X \wedge [D.r \twoheadrightarrow_P D.p]_{c'} X'\right)\}$$

Falls das anfragende Subjekt D_s.s eine dieser Rollen innehat, kann der Zugriff gestattet werden. Deshalb wird für jedes Element $D_i.r_i \in \mathcal{R}$ überprüft, ob eine Zuweisung von $D_i.r_i$ zu D_s.s existiert. Dazu wird bei der Autorisierungsstelle der Domäne D_i angefragt. In ServiceGlobe ist dies über *Delegation Services* realisiert, die sowohl PDP- wie auch PAP-Funktionalität bereitstellen. Ein Delegation Service verwaltet die Policies einer Domäne und stellt Methoden für die Delegation von Rechten und die verteilte Autorisierung bereit. Die Verfügbarkeit lässt sich durch Replikation des Dienstes erhöhen. Dabei kann der Zugriff mehrerer Delegation Services auf dieselbe Menge an Policies über bekannte Sperrmechanismen kontrolliert werden. Bei einer verteilten Autorisierungsanfrage überprüft der Delegation Service der Domäne D_i in einem ersten Schritt, ob die Rollenzuweisung (D_i-)lokal belegt werden kann. Dazu werden ähnlich zur lokalen Autorisierung all diejenigen Rollen aus D_i bestimmt, die D_s.s zugewiesen sind. Diese Menge sei $\mathcal{R}_i$. Die folgenden Fälle können auftreten:

1. Falls $D_i.r_i \in \mathcal{R}_i$, so ist die Autorisierung erfolgreich, und die Auswertung wird mit positivem Ergebnis beendet.

[2]Es können Privilegien existieren, die einen umfangreicheren Zugriff erlauben, z.B. den Zugriff auf eine Tabelle, wenn nur der Zugriff auf eine bestimmte Spalte angefordert ist. Diese sind ebenfalls in $\mathcal{P}$ enthalten.

2. Falls $D_i.r_i \notin \mathcal{R}_i$, so kann diese Anfrage nicht allein durch das „Wissen“ von D_i entschieden werden. Über weitere Delegationsstufen, ist es möglich, dass $D_s.s$ die Rolle $D_i.r_i$ innehat. Deshalb erfolgt die weiter unten beschriebene verteilte Abfrage.

3. Falls während der Auswertung ein Fehlerfall auftritt, so erfolgt ein Abbruch. Konnte der gewünschte Zugriff aufgrund fehlender Kontextinformation, die für die Auswertung von Bedingungen erforderlich ist, nicht autorisiert werden, wird dies als Annotation zur Fehlermeldung vermerkt.

In Fall 2 ist nicht ausgeschlossen, dass die Zuweisung von $D_i.r_i$ zu $D_s.s$ über mehrere „Delegationsschritte“ erfolgt, was weitere verteilte Überprüfungen nach sich zieht. Der Begriff Delegationsschritte sagt aus, dass Rollenzuweisungen aus fremden Domänen existieren, die die Rolle $D_i.r_i$ erben. Die Menge $\mathcal{R}_i^{(f)}$ dieser Rollen ist definiert als

$$\mathcal{R}_i^{(f)} \stackrel{\text{def}}{=} \left\{ D_f.r_f \,\middle|\, D_f.r_f \text{ ist eine Rolle}, D_f \neq D_i \wedge \exists \left[D_f.r_f \rightarrow_R D_i.r' \right] \text{ X mit } D_i.r' \succeq D_i.r_i \right\}$$

Die Menge $\mathcal{R}_i^{(f)}$ kann basierend auf den Role Assignment Policies der Domäne D_i ermittelt werden. $\succeq$ ist eine partielle Ordnung auf Rollen, mit $D_i.r' \succeq D_i.r_i$, falls alle Privilegien von $D_i.r_i$ auch Privilegien von $D_i.r'$ sind. $D_i.r'$ wird dann Senior-Rolle von $D_i.r_i$ genannt. Falls $D_s.s$ mindestens eine der Rollen $D_f.r_f \in \mathcal{R}_i^{(f)}$ innehat, kann der Zugriff gewährt werden. Dazu wird ggf. erneut eine verteilte Auswertung gestartet, die im positiven Fall bei einer Rollenzuweisung zu $D_s.s$ endet. Im negativen Fall bricht die Überprüfung für eine Zuweisung von $D_i.r_i$ ab, da entweder $\mathcal{R}_i^{(f)} = \varnothing$ oder alle verteilten Teilanfragen ein negatives Ergebnis liefern. Scheitert die Überprüfung für alle Rollen aus $\mathcal{R}$, wird der Zugriff bzw. die Ausführung des Dienstes verwehrt. Die beschriebene Vorgehensweise entspricht einer Tiefensuche im Zuweisungsgraphen. Dabei bilden Rollen die Knoten des Graphen. Es existiert genau dann eine gerichtete Kante von $D_i.r_i$ nach $D_f.r_f$, falls eine Role Assignment Policy $[D_i.r_i \rightarrow_R D_f.r_f]_c$ X existiert. Die Kanten sind mit den Bedingungen c der Zuweisungen annotiert. Die Tiefensuche kann sequentiell oder parallelisiert ausgeführt werden. Im parallelisierten Fall werden die Domänen D_f, bzw. die zugehörigen Delegation Services, mit $\exists\, D_f.r_f \in \mathcal{R}_i^{(f)}$ asynchron angefragt. Dies führt tendenziell zu einem höheren Nachrichtenaufkommen, da alle Pfade, die potentiell zu einer Autorisierung führen, überprüft werden. Allerdings kann, sobald ein Pfad im Zuweisungsgraphen ein positives Ergebnis liefert, die Ausführung gewährt werden. Die Ausführungszeit ist somit im Allgemeinen niedriger als bei sequentieller Suche, da bei dieser evtl. zuerst Pfade überprüft werden, die keine Autorisierung ermöglichen, ehe ein autorisierender Pfad gefunden wird oder die verteilte Autorisierung scheitert. Detailliertere Kostenbetrachtungen für die verteilte Autorisierung werden in Abschnitt 5 vorgenommen.

Rechte- und Rollendelegation. Web Service-Zusammenschlüsse können in hohem Grade dynamisch sein, d.h. kurzfristig entstehen, in ihrer Art modifiziert und wieder aufgehoben werden. Dies muss durch das zugrunde liegende Autorisierungssystem unterstützt werden. Ein Subjekt erhält durch die auf Seite 5 vorgestellten

Zuweisungen der Art (4) bzw. (5) mit dem Prädikat *zuweisen* das Recht, Privilegien bzw. Rollen zu delegieren. Als Delegation wird die Weitergabe an andere Subjekte bezeichnet. Stellt ein Subjekt $D_{s1}.s1$ an den Delegation Service der Domäne D die Anfrage, ein Recht D.p oder eine Rolle D.r an ein Subjekt $D_{s2}.s2$ weiterzugeben, so erfolgt wie zuvor beschrieben eine Autorisierung dieser Anfrage. Hat das Subjekt tatsächlich das dazu notwendige Privileg, so werden entsprechende Policies zum bestehenden Regelsystem neu hinzugefügt. Handelt es sich um eine Rechtedelegation, so wird eine Base Policy der Form $[D_{s2}.s2 \rightarrow_P D.p]\, D_{s1}.s1$ erzeugt. Im Fall der Delegation einer Rolle entsteht analog eine Role Assignment Policy $[D_{s2}.s2 \rightarrow_R D.r]\, D_{s1}.s1$.

Aufheben von Delegationen. Beim Aufheben von Zuweisungen müssen für das entziehende Subjekt Zusicherungen (4) und (5) mit dem Prädikat *entziehen* existieren (vgl. S. 5). Policies, die die betreffende Rechte- oder Rollenzusicherung ausdrücken, werden dann gelöscht. Man unterscheidet dabei zwischen konservativem und destruktivem Löschen. Entzieht $D_{s1}.s1$ dem Subjekt $D_{s2}.s2$ die zuvor zugewiesene Rolle D.r, so wird bei konservativer Vorgehensweise nur die zugehörige Role Assignment Policy $[D_{s2}.s2 \rightarrow_R D.r]\, D_{s1}.s1$ entfernt. Destruktives Löschen geht darüber hinaus: Verfügt D.r über Privilegien, die Inhaber der Rolle zur Weitergabe von Rechten und/oder Rollen ermächtigt, so werden auch die darauf basierenden Delegationen zurückgenommen. Dazu müssen die Inhaber der Rolle D.r ermittelt werden. Da D.r insbesondere auch Rollen anderer Domänen zugewiesen worden sein kann, erfordert dies im Allgemeinen eine verteilte Abarbeitung. Destruktives Löschen kann kaskadieren, da delegierte Rechte/Rollen selbst wieder zur Delegation von Rechten/Rollen berechtigen können. Das Löschen einer Zuweisung kann dadurch einen Schneeballeffekt auslösen und zum Löschen von Zuweisungen über mehrere Domänen hinweg führen. Destruktives Löschen erfordert nicht nur einen höheren Aufwand sondern ist oftmals auch ausgeschlossen. Wechselt ein leitender Verwaltungsangestellter eines Krankenhauses, der beispielsweise die Einteilung der Angestellten zu Stationen vornahm, seine Stelle, so sollen nach seinem Ausscheiden bisher bestehende Einteilungen nicht aufgehoben werden.

4.3 Beispiele

Anhand zweier Beispiele wollen wir die Funktionsweise der lokalen und verteilten Autorisierung verdeutlichen. In Abschnitt 2 wurden Regeln für den Zugriff von Ärzten auf Patientendaten des Chicago Hope Hospitals (CH) eingeführt. Die Zugriffsregeln speziell für Dr. Jeffrey Geiger waren wie folgt formuliert.

$$[\texttt{CH.AttendingPhysicianRole} \rightarrow_P \texttt{CH.ReadMedRecPermission}]\, \texttt{CH}$$

$$[\texttt{CH.JeffreyGeiger} \rightarrow_R \texttt{CH.AttendingPhysicianRole}]_{(\texttt{patientId=CH.MrWatters})}\, \texttt{CH}$$

Dieses recht einfache Rechtesystem stelle die Autorisierungsgrundlage für einen Web Service dar, welcher die Abfrage von Befunden ermöglicht. Ruft Dr. Geiger diesen Dienst auf, um auf Daten von Watters zuzugreifen, so erfolgt die in Abbildung 8 schematisch dargestellte Zugriffskontrolle. Es wird in einem ersten Schritt

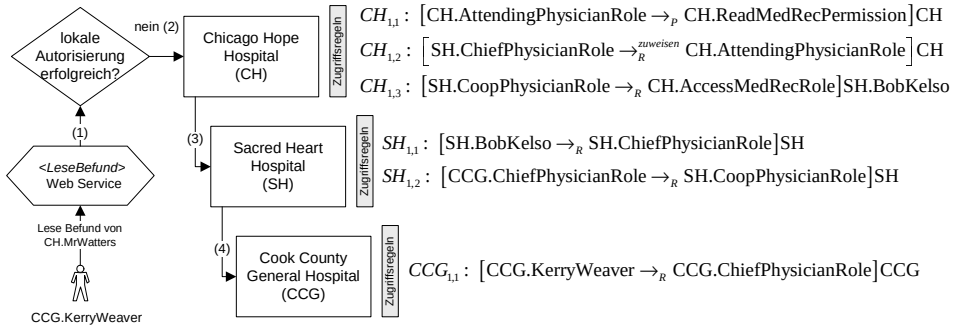

Abbildung 10: Beispielanwendung einer verteilten Autorisierung.

festgestellt (vgl. Policy aus Abbildung 5), dass Dr. Geiger die Rolle CH.AttendingPhysicianRole innehat, denn die einschränkende Bedingung ist erfüllt. Würde Dr. Geiger Daten eines anderen Patienten als Watters anfragen, kommt die Zuweisung der Rolle nicht zustande. Mit dem um die Rolle erweiterten XACML-Request werden anschließend zutreffende Base Policies ermittelt. In unserem Fall ist dies die in Abbildung 4 dargestellte Policy. Da diese auf das Recht verweist, Patientendaten einsehen zu dürfen (vgl. Policy aus Abbildung 3), wird der Zugriff bzw. die Ausführung des Dienstes gestattet.

Um die Funktionsweise der verteilten Autorisierung zu zeigen, wird das bisherige Beispiel erweitert. Dazu wird angenommen, dass mehrere Krankenhäuser zu Forschungszwecken zusammenarbeiten, also ein Kollaborationsnetzwerk bilden. Ärzten aus den beteiligten Krankenhäusern soll es möglich sein, Patientendaten anderer Häuser einzusehen. Zur besseren Übersichtlichkeit wurde auf zusätzliche Bedingungen, wie die Einschränkung auf Patienten, die dieser Weitergabe auch zugestimmt haben, verzichtet. Abbildung 10 zeigt einen exemplarischen Auszug an Zugriffsregeln für die Krankenhäuser Chicago Hope Hospital (CH), Sacred Heart Hospital (SH) und Cook County General Hospital (CCG). Die Regeln des Chicago Hope Hospitals beinhalten eine Weitergabe von Rechten über Domänengrenzen hinweg. Und zwar besagt $CH_{1,2}$, dass Chefärzte des Sacred Heart die Rolle CH.AttendingPhysicianRole anderen Subjekten (uneingeschränkt) zuweisen dürfen, was durch $CH_{1,3}$ angewendet wurde. Ärzte, die mit dem Sacred Heart kooperieren, haben somit diese Rolle inne. Wie zuvor hat diese Rolle das Privileg, Patientendaten innerhalb CH einzusehen. Über $SH_{1,2}$ überträgt sich dieses Recht indirekt auf Chefärzte des Cook County General und mittels $CCG_{1,1}$ insbesondere auf Dr. Kerry Weaver. Stellt Kerry Weaver, wie in der Abbildung angedeutet, die Anfrage, den Befund von Watters zu lesen, so muss dazu der Autorisierungspfad $CCG_{1,1}$, $SH_{1,2}$, $CH_{1,3}$ nach $CH_{1,1}$ in umgekehrter Reihenfolge durchlaufen werden. Zuvor wird jedoch eine lokale Autorisierungsüberprüfung durchgeführt. Da basierend auf den Policies von CH alleine keine Berechtigung für den Zugriff abgeleitet werden kann, werden zunächst die Rollen ermittelt, denen CH.ReadMedRecPermission zugewiesen ist. In unserem Beispiel trifft dies auf die

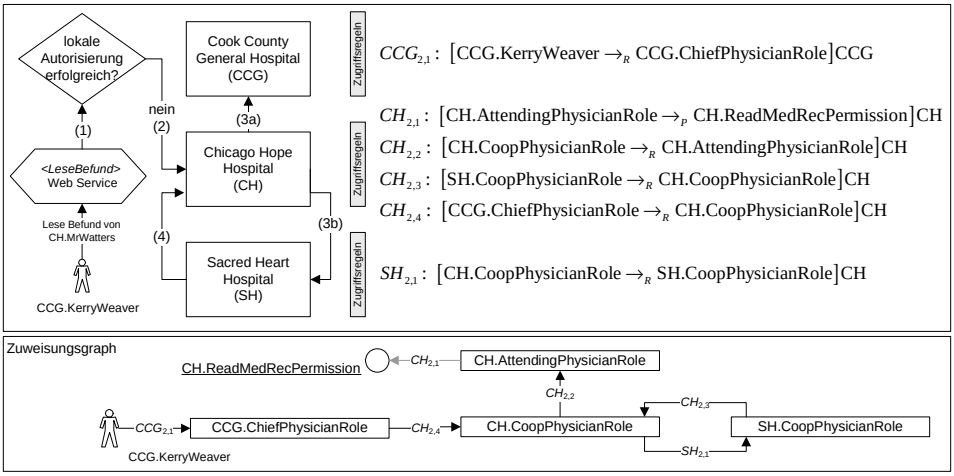

Abbildung 11: Auftreten von Zyklen in Zuweisungsgraphen.

CH-lokale Rolle CH.AttendingPhysicianRole zu. Da Dr. Weaver diese nicht direkt zugewiesen bekommen hat (andernfalls wäre die lokale Auswertung nicht gescheitert), wird eine Abfrage an SH gestartet. Aufgrund von $CH_{1,3}$ wird überprüft, ob Dr. Weaver die Rolle SH.CoopPhysicianRole besitzt. Auch SH kann dies nicht direkt beantworten, sondern initiiert die Anfrage an CCG, ob Dr. Weaver die Rolle CCG.ChiefPhysicianRole besitzt, was aufgrund $CCG_{1,1}$ positiv beantwortet wird.

5 Optimierung der verteilten Autorisierungsauswertung

Bei den bisherigen Betrachtungen wurde nicht ausgeschlossen, dass Rollenzuweisungen zyklisch erfolgen können. Eine Vermeidung von Zyklen ist aufgrund des verteilten Charakters der Zugriffsregeln nur schwer durchsetzbar. Im Allgemeinen ist eine derartige Einschränkung zudem auch zu restriktiv und unerwünscht. Abbildung 11 zeigt ein Beispiel dafür, dass zyklische Abhängigkeiten bewusst aufgebaut werden, um etwa enge multilaterale Zusammenschlüsse zu modellieren. Abgebildet ist eine leicht geänderte Version des vorherigen Beispiels. Ein Zyklus wird hier durch die gegenseitige Zuweisung von CoopPhysicianRole aus CH und SH über die Zuweisungen $CH_{2,3}$ und $SH_{2,1}$ gebildet. Falls Dr. Weaver des Cook County General erneut den Befund von Watters aufruft, laufen die ersten beiden Schritte wie zuvor ab. Im dritten Schritt spaltet sich die Auswertung aufgrund der Zuweisungen $CH_{2,3}$ und $CH_{2,4}$ auf. Während (3a) zu einem positiven Ergebnis führt, würde ab (3b), falls der Zyklus nicht beachtet wird, die Auswertung in eine Endlosschleife geraten. Deshalb werden Auswertungsschritte protokolliert und als Kontext an nachfolgende Stellen übermittelt. Dadurch ist eine Überprüfung hinsichtlich bereits erfolgter Abfragen und somit auch eine Zyklenerkennung möglich.

Die Übermittlung der Rückrichtung, d.h. der Weg einer erfolgreichen Autorisierung, wird dazu verwendet, die verteilte Zugriffskontrolle bezüglich der Ausführungszeit zu optimieren. Betrachtet man ein System, bestehend aus m kollaborierenden Domänen mit einer maximalen Zahl n an Rollendefinitionen je Domäne, so ergibt sich eine obere Schranke von $O\left(m^2 \cdot n^2\right)$ für die Anzahl an zu überprüfenden Rollendelegationen. Diese obere Schranke wird dann angenommen, wenn jede Rolle einer Domäne allen Rollen aus allen anderen Domänen zugeordnet wird. Um die entstehende Laufzeit, Serverbelastung und Netzwerkosten zu verringern, werden Caching-Techniken eingesetzt. Der *Autorisierungs-Cache* der Domäne $\mathrm{D_n}$ enthält Autorisierungspfade der Form:

$$\left\langle \mathtt{D_S.Subjekt} \overset{\mathtt{Bedingungen_1}}{\rightarrow} \mathtt{D_1.Rolle_1} \overset{\mathtt{Bedingungen_2}}{\rightarrow} \mathtt{D_2.Rolle_2} \ \ldots \overset{\mathtt{Bedingungen_n}}{\rightarrow} \mathtt{D_n.Rolle_n} \right\rangle$$

Mit Ausnahme des ersten Knotens, der stets ein anfragendes Subjekt repräsentiert, sind die Knoten des Pfades Rollen. Die Kanten sind mit (möglicherweise leeren) Bedingungen annotiert. Es wird in einem Cache immer die Information bis zur betreffenden Domäne gespeichert. Der Cache der Domäne $\mathrm{D_i}$, die ebenfalls an der dargestellten Autorisierung beteiligt ist, enthält folglich den Pfadausschnitt bis zur Stelle $\mathrm{D_i.Rolle_i}$. Der Delegation Service einer Domäne übernimmt bei einer verteilten Autorisierung sowohl Client- wie auch Server-Funktionalität. Als Client fungiert er, da er Autorisierungspfade zwischenspeichert und als Server, da andere Domänen Autorisierungsanfragen an ihn stellen.

Für das Caching von Autorisierungspfaden gelten hohe Anforderungen an die Cache-Konsistenz. Es kann im Allgemeinen nicht toleriert werden, dass ein Zugriff basierend auf Cache-Einträgen zustande kommt, welche aufgrund eines Rechteentzugs zwischenzeitlich ungültig sind. TTL (*Time-To-Live*)-basierte Verfahren sind somit unzureichend. Selbst serverseitige Invalidierung weist folgende Nachteile auf: Zum einen skaliert eine derartige Vorgehensweise schlecht (vgl. [CO02]), da sich jede Domäne merken muss, welche Anfragen an sie gestellt wurden, um so im Falle einer Policy-Aufhebung benachbarte Stellen zu informieren. Zum anderen dürfen lokale Zugriffsregeln erst dann modifiziert oder gelöscht werden, wenn alle Nachbarstellen darauf basierende Cache-Einträge invalidiert haben. Bei Netzwerkproblemen ist damit die Funktionsweise des Systems beeinträchtigt. Auch server- wie clientseitig zu realisierende Lease-Ansätze, bei denen die Autorisierungsstelle eine Gültigkeitsdauer der Zugriffsregeln vorgibt, sind nur bedingt anwendbar. Die Flexibilität des Rechtemanagements wird eingeschränkt, da ein Rechteentzug frühestens dann stattfinden kann, wenn die einzustellende Zeitspanne verstrichen ist.

Aufgrund der Sensibilität der gehaltenen Daten wird stattdessen eine Validierung durchgeführt. Autorisierungspfade werden als „Wegweiser“ verwendet, um zielgerichtet eine erneute Überprüfung durchzuführen. Betrachtet man das vorherige Beispiel (Abbildung 11), so enthält der Cache von CH nach dem ersten Aufruf des <*Lese Befund*>-Web Services durch Dr. Weaver folgenden Eintrag:

$$\left\langle \mathtt{CCG.KerryWeaver} \overset{\varnothing}{\rightarrow} \mathtt{CCG.ChiefPhysicianRole} \overset{\varnothing}{\rightarrow} \mathtt{CH.CoopPhysicianRole} \right\rangle$$

War beim erstmaligen Ausführen noch eine Verzweigung während der verteilten Autorisierung nötig (3a und 3b in Abbildung 11), so wird diese nun vermieden: Nachdem ein entsprechender Autorisierungspfad vorliegt wird im dritten Schritt explizit nur noch (3a) verfolgt. Auch die Bearbeitung eines Schrittes selbst wird deutlich effizienter. Musste zuvor erst noch bestimmt werden, welche Rollen eine Möglichkeit der Autorisierung bieten, was einer Überprüfung der Role Assignment Policies gleich kam, so entfällt dies hier. Es wird jeweils nur validiert, dass die Zuweisungen, wie sie der Autorisierungspfad angibt, weiterhin gültig sind. Wurde eine der Zuweisungen aufgehoben, schlägt die Überprüfung fehl und der Eintrag wird aus dem Cache entfernt.

Der zu erreichende Effizienzgewinn hängt – neben Cache-spezifischen Parametern wie der Cache-Größe – vom Verzweigungsgrad des Zuweisungsgraphen ab. Ein weiterer Faktor ist die Regelmäßigkeit, mit der sich gleiche oder ähnliche Autorisierungsanfragen wiederholen. Der Aufbau der Einträge ermöglicht auch die Behandlung von Teilanfragen. Wird beispielsweise ermittelt, dass Dr. Kerry Weaver für die Ausführung einer Dienstmethode ein Privileg benötigt, das CH.CoopPhysicianRole zugewiesen ist, so liefert obiger Autorisierungspfad eine zielgerichtete Vorgehensweise. Cache-Einträge werden so verwaltet, dass Autorisierungspfade, die Teilmengen von anderen Pfaden darstellen, durch die Umfassenderen absorbiert werden.

6 Zusammenfassung

Wir haben in diesem Beitrag ein Autorisierungssystem für Service Oriented Architectures (SOAs) vorgestellt. Neben einer flexiblen Zugriffskontrolle für Web Services unterstützt dieses System die Abstimmung der Autorisierung von Web Services mit der von darunter liegenden Datenbanken. Dadurch werden Web Service-Entwickler bei der Erstellung der Zugriffsregeln für Dienste unterstützt. Auch aus Sicht der Datenbank wird die Sicherheit erhöht, indem der Zugriff über abgestimmte Zugriffskorridore erfolgt. Ferner stellt das eingeführte System die Voraussetzungen für den Aufbau von Web Service-Föderationen bereit. Über Web Service-Föderationen werden Ressourcen wie Datenbanken organisationsübergreifend bereitgestellt. Durch einen flexiblen Delegationsmechanismus werden sowohl schwach wie auch stark gekoppelte Zusammenschlüsse unterstützt. An einer Föderation teilnehmende Organisationen behalten dazu stets ihre Autorisierungsautonomie. Das vorgestellte Autorisierungssystem wurde in unserer Web Service-Plattform ServiceGlobe in vollem Funktionsumfang umgesetzt. Die Optimierung der verteilten Autorisierung mittels Caching wird derzeit anhand von Leistungsanalysen validiert. Zudem werden Caching-Techniken auf ihre Anwendbarkeit im Autorisierungskontext hin analysiert. Ein weiterer Forschungsschwerpunkt ist die Abstimmung Organisationen übergreifender Rollendelegationen mit einem verteilten Identitätsmanagement.

Literatur

[And04] A. Anderson. XACML Profile for Role Based Access Control RBAC Profile. working draft Version 2.0, OASIS, Mai 2004.

[BLP03] J. Biskup, T. Leineweber und J. Parthe. Administration Rights in the SDSD-System. In *Proceedings of the Seventeenth Annual Working Conference on Database and Application Security*, Estes Park, Colorado, United States, August 2003.

[CFMS94] S. Castano, M. Fugini, G. Martella und P. Samarati. *Database Security*. Addison-Wesley, 1994.

[CO02] L. Y. Cao und M. T. Özsu. Evaluation of Strong Consistency Web Caching Techniques. *World Wide Web*, 5(2):95–124, 2002.

[DAI] Database Access and Integration Services WG (DAIS-WG). `https://forge.gridforum.org/projects/dais-wg`.

[FPP+02] E. Freudenthal, T. Pesin, L. Port, E. Keenan und V. Karamcheti. dRBAC: Distributed Role-Based Access Control for Dynamic Coalition Environments. In *Proceedings of the Twenty-second IEEE International Conference on Distributed Computing Systems (ICDCS)*, Seiten 411—420, Vienna, Austria, Juli 2002.

[FSG+01] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn und R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, 2001.

[GM+03] S. Godik, T. Moses et al. eXtensible Access Control Markup Language (XACML). `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml`, Februar 2003.

[JD93] D. Jonscher und K. Dittrich. Access Control for Database Federations: a discussion of the state-of-the-art. In *DBTA Workshop on Interoperability of DBSs and DB Applications*, Seiten 156–178, Oktober 1993.

[KN+03] C. Kahler, A. Nadalin et al. Web Services Federation Language (WS-Federation). `http://www-106.ibm.com/developerworks/webservices/library/ws-fed/`, Juli 2003.

[KSSK02] M. Keidl, S. Seltzsam, K. Stocker und A. Kemper. ServiceGlobe: Distributing E-Services across the Internet (Demonstration). In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, Seiten 1047–1050, Hong Kong, China, August 2002.

[OGS] Open Grid Services Architecture – Data Access and Integration (OGSA-DAI). `http://www.ogsadai.org.uk/`.

[PVWKT02] L. Pearlman, I. Foster V. Welch, C. Kesselman und S. Tuecke. A Community Authorization Service for Group Collaboration. In *3rd International Workshop on Policies for Distributed Systems and Networks (POLICY)*, Seiten 50–59, Monterey, CA, USA, Juni 2002. IEEE Computer Society.

[T+03] S. Thatte et al. Business Process Execution Language for Web Services (BPEL4WS 1.1). `http://www-106.ibm.com/developerworks/library/ws-bpel/`, Mai 2003.

[WEEK04] M. Wimmer, D. Eberhardt, P. Ehrnlechner und A. Kemper. Reliable and Adaptable Security Engineering for Database-Web Services. In *Proceedings of the Fourth International Conference on Web Engineering*, Jgg. 3140 of *Lecture Notes in Computer Science (LNCS)*, Seiten 502–515, Munich, Germany, Juli 2004.