technische universität
dortmund

Seminararbeit

Analyzing Security Policies

Sebastian Reitz
30. Januar 2014

Gutachter: Prof. Dr. Jan Jürjens
Dr. Thomas P. Ruhroth

Prof. Dr. Jan Jürjens Lehrstuhl 14 Software Engineering
Fakultät Informatik
Technische Universität Dortmund
Otto-Hahn-Straße 14
44227 Dortmund
http://www-jj.cs.uni-dortmund.de/secse

Sebastian Reitz
sebastian.reitz@tu-dortmund.de
Matrikelnummer: 147739
Studiengang: Bachelor Informatik

Werkzeugunterstützung für sichere Software
Thema: Analyzing Security Policies

Eingereicht: 30. Januar 2014

Betreuer: Prof. Dr. Jan Jürjens

Prof. Dr. Jan Jürjens Lehrstuhl 14 Software Engineering
Fakultät Informatik
Technische Universität Dortmund
Otto-Hahn-Straße 14
44227 Dortmund

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Dortmund, den 30. Januar 2014

Sebastian Reitz

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation und Hintergrund

Ziel dieser Proseminararbeit ist es mit Hilfe von theoretischen Modellen Lösungswege aufzuzeigen, wie Probleme der Nutzerverwaltung und das Modellieren von Zugriffsrechten verwaltet werden können. Als Ausgangspunkt diente dazu das vierte Kapitel des Buches [DJ10].

1.2 Was ist Access Control und welche Probleme entstehen

Bei Access Control, oder auch Zugriffsrechteverwaltung, wird versucht zu kontrollieren, welche Person oder Software die Erlaubnis besitzt auf bestimmte Daten zuzugreifen oder bestimmte Operationen auszuführen, sei es in Programmen oder auf Dateien. Daraus ergeben sich für die Access Control mehrere Probleme, welche in dieser Seminararbeit behandelt werden sollen. Die erste Frage, die sich daraus ergibt, ist, wie das System sich die Rechte der einzelnen Personen im System merkt. Wird einfach für jede Person jedes einzelne Recht abgespeichert oder lassen sich diese Informationen zum Teil bündeln und eventuell Strukturen des Unternehmens darauf abbilden, sodass Nutzer in Gruppen zusammengefasst werden. Dieses Problem wird vor allem in Kapitel 2 und 3 behandelt, in welchem zwei konkrete Modelle vorgestellt werden, welche auf unterschiedliche Weise das Problem lösen wollen. Die zweite Frage ist, wie ein solches System, welches die beschriebenen Probleme löst, überhaupt verwaltet und administriert werden kann. Wie kann dieses dezentralisiert werden und wie können Rechte vergeben und entzogen werden, sodass dabei keine Probleme entstehen können, wie das ungewollte Löschen von zu vielen Rechten. Hierfür wird in Kapitel 4 ein System vorgestellt, welches für das Modell aus Kapitel 3 eine Lösung anbietet. Das dritte und letzte Problem, welches hier behandelt werden soll, ist die Überprüfung der Nutzer, also ob es sich um die Person handelt für die sie sich ausgibt. Hierfür wird ein Ansatz mit Hilfe von Zertifikaten verwendet, welche entlang von hierarchischen Strukturen auf Echtheit überprüft werden. Abschließend wird ein Fazit gezogen, wie weit sich die hier gezeigten formalen Lösungsansätze auf reale Systeme übertragen lassen.

2 Access-Matrix-Based Systems

2.1 Der allgemeine Ansatz

In diesem Kapitel wird ein erster Lösungsansatz für das Problem der Organisation von Zugriffsrechten behandelt. Hierbei handelt es sich um ein System, welches das Prinzip der Zugriffsmatrix bzw. Access-Matrix verwendet. Dieses System ist vereinfacht beschrieben so aufgebaut, dass eine Matrix aus Nutzern auf der einen und Dateien und Programmen auf der anderen Seite gebildet wird. In den Feldern der Matrix wird nun vermerkt, welche Rechte der jeweilige Nutzer für die jeweilige Datei bzw. das jeweilige Programm besitzt.

2.2 Das HRU-Schema

Die vorliegende Quelle [DJ10] verwendet als matrixbasierten Ansatz das HRU-Schema, welches im Folgenden näher erläutert werden soll. Im HRU-Schema existieren in einem aktuellem Zustand y ein Tupel aus Subjects, Objects und der Matrix $S \times O \rightarrow 2^R$. In dieser Matrix werden Subjekt und Objekt einer Menge von Rechten zugewiesen, welche die erlaubten Operationen beschreiben. Dabei müssen die Menge der Subjekte und Objekte in einem Zustand kleiner sein als die Menge der insgesamt möglichen und die der Subjekte immer kleiner als die der Objekte, da jedes Subjekt auch ein Objekt ist. Der Zustandsübergang im HRU-Schema besteht nun aus dem erfolgreichen Ausführen eines Kommandos. Diese Kommandos umfassen das Hinzufügen und Entfernen von Rechten, Subjekten und Objekten. Diese werden in Form von „If. . . then“ Anweisungen ausgeführt, in welchen zunächst die Bedingungen anhand der existierenden Matrix überprüft werden und nur bei erfolgreicher Prüfung ausgeführt werden. Ein Problem des HRU-Schemas ist der Begriff des Leaks. Bei einem Leak wird, während ein Kommando ausgeführt wird, ein Recht in eine Zelle geschrieben, welche dieses nicht unmittelbar vor dem Ausführen des primitiven Kommandos enthielt. Ein Schema ist nun (r)-leak-safe, wenn aus dem Startzustand des Systems es nicht möglich ist, einen Zustand zu erreichen, in dem ein solches Leak auftreten kann. Die Frage, ob ein System (r)-leak unsafe ist, ist im Allgemeinen nicht entscheidbar. Lediglich für Schemata, welche genau eine primitive Operation pro Kommando erlauben, ist dieses Problem NP-vollständig.

2.3 Bewertung

Das HRU-Schema insgesamt besitzt keine große praktische Relevanz. Bei der Verwaltung von Rechten wird der Aufwand sehr schnell groß und wäre bei größeren Systemen praktisch nicht zu warten. Hinzu kommt das Problem der Leak-Safety, welches zeigt, dass ein wichtiger Sicherheitsaspekt nicht, bzw. in einem Spezialfall nur sehr ineffizient zu überprüfen ist.

3 Role Based Access Control anhand von RBAC96

3.1 Der allgemeine Ansatz

Dieses Kapitel beschreibt einen weiteren Ansatz, die Zugriffsrechte zu organisieren. Dabei wird nicht, wie zuvor, jede Erlaubnis einzeln jedem Nutzer zugeordnet, sondern es wird mit Hilfe der Rollen eine Art Zwischenschicht gebaut. Rollen besitzen jeweils bestimmte Rechte, während Nutzer zu einzelnen Rollen gehören. Gleichzeitig können Rollen hierarchisch-transitive Strukturen darstellen, sodass Nutzer automatisch die Rechte der Rollen erhalten, welche unter ihren Rollen liegen. Ziel ist es, den Overhead für die Verwaltung zu vermeiden, da viele Nutzer zusammengefasst werden können und so Änderungen für einzelne Beschäftigungsrollen erheblich übersichtlicher mit geringerem Fehlerpotential durchgeführt werden können.

3.2 Das Modell RBAC96

In diesem Abschnitt wird das RBAC96 (role-based access control '96) Modell beschrieben, welches versucht, diese rollenbasierte Verteilung formal zu beschreiben und in der Ursprungsquelle [DJ10] entsprechend definiert wurde. Das Modell basiert zunächst auf den drei Grundmengen U(Benutzern), R(Rollen) und P(Rechte). Ein Zustand besteht nun aus einem 6-Tupel von Relationen. Dieses gliedert sich in einen Basiszustand, welcher drei Relationen umfasst. UA weist Benutzern Rollen zu, PA weist Rechte Rollen zu und RH beschreibt die Hierarchie der Rollen, welche azyklisch und nicht reflixiv sein muss. Die Rollenhierarchie ist dabei transitiv, sodass eine höhere Rolle alle Rechte der ihr untergeordneten Rollen erhält. Daraus folgt auch, dass ein Benutzer, welcher einer Rolle zugeteilt ist, automatisch den untergeordneten Rollen ebenfalls zugeordnet ist. Zusätzlich besitzt das Schema eine Funktion, welche einem Nutzer im aktuellen Zustand des Systems seine Rollen zuweist. Dazu kommen Funktionen, welche Rollen die höheren und niedrigeren Rollen zuweisen. Der zweite Teil des 6-Tupels umfasst drei Relationen, welche den administrativen Zustand beschreiben. CA beschreibt, welche Benutzer anderen Benutzer bestimmten Rollen zuweisen können und welche Bedingungen diese erfüllen müssen. CR beschreibt, welche Benutzer andere Benutzer von bestimmten Rollen entfernen können. Zuletzt existiert noch CO. Diese Relation beschreibt, welche Rollen sich gegenseitig ausschließen, also nicht vom selben Benutzer belegt werden können. Im Modell existieren nun Aktionen zum Zuweisen und Entfernen von Rollen zu Nutzern. Diese Aktionen werden erfolgreich ausgeführt, wenn der Nutzer zuvor Mitglied, bzw. kein Mitglied der Rolle war, der durchführende Nutzer die nötigen Rechte besitzt

und beim Hinzufügen zusätzlich die Bedingungen aus CO erfüllt werden. Nur dann wird die Operation durchgeführt und das System vollzieht einen Zustandswechsel. Mit dem hier gegebenen Modell ist es nun möglich, die Vergabe von Rechten anhand von hierarchischen Strukturen zu steuern. Dies bietet den Vorteil, dass sich so mögliche Unternehmensstrukturen darauf abbilden lassen und somit ein System entsteht, welches sich wesentlich intuitiver verwalten lässt. Durch das Zusammenfassen in Rollen lässt sich zudem das Anpassen von Berechtigungen erheblich einfacher verwalten.

3.3 Das FUB der Dresdner Bank

Die praktische Relevanz des Modells lässt sich anhand eines Beispiels aus der Praxis verdeutlichen. Das FUB(Funktionale Berechtigungen) System der Dresdner Bank, welches in [SM01] ausführlich besprochen wurde, ist ein Beispiel dafür, wie rollenbasierte Systeme in der Praxis für die Verwaltung von Rechten eingesetzt werden können. Dafür soll die Funktionsweise kurz erläutert werden und Unterschiede zum formalen RBAC96 Modell dargelegt werden. Das System der Dresdner Bank entstand 1990 und verwaltet über 60 Anwendungen und 42000 Benutzerprofile täglich. Rollen sind in diesem System eine Kombination aus der Funktion des Jobs z.B. Finanzanalyst und der Position innerhalb der Hierarchie, z.B. regionaler Vorsitzender oder Gruppenleiter. Die Rollen werden dabei mit der Datenbank der Human Resource Abteilung abgeglichen, sodass die Zuweisung von Rollen immer sehr aktuell ist. Ziel ist, dass jeder Nutzer, wenn möglich, nur eine Rolle belegt, wobei jedoch bis zu vier möglich sind. Dies dient unter anderem dazu, Urlaubsvertretungen zu organisieren. Insgesamt existieren in diesem System ungefähr 1300 Rollen. Bei der Verwaltung des System existiert eine gute Trennung der Aufgaben. Die Human Resource Abteilung definiert Rollen und weist diese Nutzern zu. Die Anwendungsverwaltung definiert Zugriffsrechte und weist diese Anwendungen zu. Zuletzt existieren noch die FUB Administratoren, welche Rollen Anwendungen zuweisen. So entsteht eine gute Trennung der Aufgabe in der Verwaltung. Probleme existieren bei der Einbindung von Freelancern, da dies nur durch FUB Administratoren geschieht und insbesondere nicht durch das Human Resource Management kontrolliert werden kann.
Auch wenn sich das Modell in der Praxis bewährt hat, sind Probleme vorhanden, für welche entsprechende Änderungen am System vorgesehen sind. Zum einen gibt es das Ziel Nutzer zu gruppieren, auch wenn diese nicht in derselben Funktion und Position in der Hierarchie sind. Dies ist beabsichtigt, um Gruppen bestimmten Rollen zuzuweisen, damit dies nicht für jeden einzelnen Nutzer geschehen muss. Dies kann nötig sein, wenn Personen aus unterschiedlichen Bereichen am selben Projekt arbeiten.
Ein weiteres Problem, welches in diesem Modell vorhanden ist, ist die Möglichkeit Rechte einzelnen Nutzern explizit zuzuweisen. Dies widerspricht der Trennung von Nutzern und Rechten des Rollenmodells, ist jedoch für die Bank eine wünschenswerte Möglichkeit. Das dritte Ziel für die Entwicklung des Systems ist ein Modell, welches das Übernehmen von Aufgaben eines anderen Nutzers im Fall von Krankheit oder Urlaub ermöglicht. Dabei stellt sich die Frage, wie Rollen mit einer zeitlichen Beschränkung vergeben werden können.

Das Modell steht in einem zentralen Punkt im starken Gegensatz zum RBAC96 Model. Das FUB Modell besitzt nämlich keine hierarchische Struktur. Zwar wird dies indirekt über die Rollenerstellung gegeben, jedoch existiert keine Relation zwischen den Rollen, sodass auch keine Transitivität existieren kann. Dies führt dazu, dass die Rechte für jede Rolle explizit definiert werden müssen. Durch das Einführen dieser Relationen ließe sich das System wesentlich einfacher verwalten und die Definition von Rollen verkürzen. Insgesamt zeigt dieses Beispiel dennoch, dass rollenbasierte Systeme eine gute Möglichkeit bieten, auch in sehr großen Systemen die Rechte zu verwalten.

3.4 Beispiel einer Relation

In Abbildung 3.1 ist ein einfaches Beispiel für eine transitive Rollenhierarchie dargestellt. Nehmen wir an, D besäße eine bestimmte Menge an Rechten, zum Beispiel die Möglichkeit, auf einen Teil einer Datenbank zuzugreifen. B und C als übergeordnete Rollen besäßen nun ebenfalls dieses Recht. Dazu kommen nun jedoch noch individuelle Rechte, welche voneinander disjunkt sind. B besitzt also andere Rechte als C. A ist nun ein Beispiel für eine Rolle, welche von mehreren Rollen erbt. A besäße nun die Rechte von B und C und allen untergeordneten Rollen von diesen, in diesem Fall also nur D. So lassen sich auch komplexere Rechtestrukturen aufbauen, welche für viele Unternehmens- und Organisationsstrukturen geeignet sind.

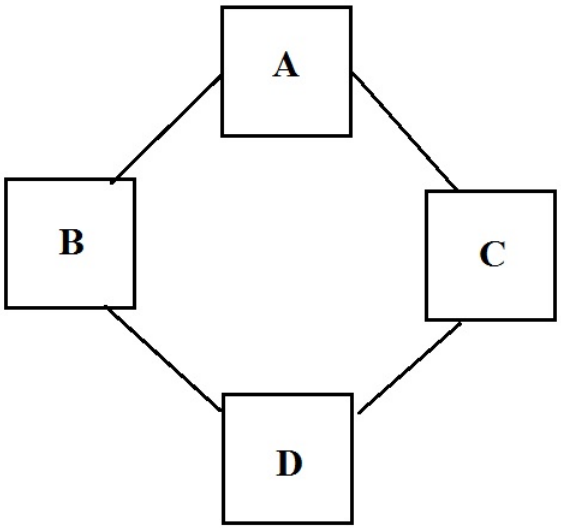

Abbildung 3.1: Beispiel einer transitiven Relation

4 Die Verwaltung von Rollen in RBAC96 mit ARBAC97

4.1 Das Ziel von ARBAC97

In diesem Kapitel soll, aufbauend auf Kapitel 3, die Frage behandelt werden, wie das in Kapitel 3 definierte Modell verwaltet werden soll. Dabei geht es um das Zuweisen von Rechten zu den Rollen, das Zuweisen der Nutzer zu den Rollen und das Zuweisen von Rollen zu Rollen, also den Aufbau der hierarchischen Struktur. Das ARBAC97(administrative RBAC '97) stellt für jedes dieser drei Probleme ein separates Modell bereit, welches für RBAC96 geschaffen wurde und in [SB99] definiert wurde.

4.2 Das Modell URA97

Beim ersten Modell, welches Teil des ARBAC97 Modells ist, handelt es sich um URA97(user-role assignement '97). Dieses Modell beschäftigt sich mit dem Problem des Zuweisens und des Entfernens von Nutzern von Rollen. Dabei soll insbesondere der einfache Fall verhindert werden, dass eine einzelne Person die volle Kontrolle über das Zuordnen und Entfernen von Benutzern zu und aus Rollen hat. Dabei existieren direkt zwei Probleme, nämlich wer darf Nutzern bestimmte Rollen hinzufügen und welche Nutzer hinzugefügt werden dürfen. Diese beiden Probleme werden von dem Modell mit einem einzelnen Hilfsmittel gelöst, den prerequisite conditions. Dabei handelt es sich um einen boolschen Ausdruck, welcher beschreibt, wer einer Rolle zugeteilt werden darf und wer dafür die Rechte besitzt. Die Zuweisung eines Benutzers in eine Rolle wird nun mit Hilfe der can_assign Relation beschrieben, welche überprüft, ob der Benutzer die nötigen Bedingungen erfüllt und ob die Person, welche die Zuweisung durchführen will Mitglied einer Rolle mit den nötigen Rechten, oder einer in der Hierarchie übergeordneten Rolle ist. Somit stellt URA97 sicher, dass das Zuweisen in Rollen den nötigen Beschränkungen unterliegt.
Im Gegensatz zum Zuweisen ist das Entfernen ein komplexeres Problem. Es wird zwischen starkem und schwachem Entfernen unterschieden. Dies liegt an der hierarchischen Struktur des RABC96 Modells und der daraus folgenden Vererbung von Rollen. In URA97 wird das schwache Entfernen definiert. Dies bedeutet, dass wenn eine Operation durch die Rechte des Administrators erlaubt ist, werden dem Nutzer nur, wenn er explizites Mitglied der Rolle ist, die Rechte entzogen. Sollte er implizites Mitglied der Rolle sein und somit Mitglied einer höheren Rolle, behält er die Rechte, welche ihm durch die höhere Rolle gegeben werden und die Operation hat keinen Effekt. Die Idee des starken Entfernens ist hingegen, dass wenn ein Nut-

zer aus einer Rolle entfernt werden soll, so lange in der Hierarchie hinauf gegangen wird, in welcher der Nutzer explizites Mitglied ist. Nun wird mit den Rechten des Administrators ein schwaches Entfernen ab dieser Rolle bis zur eigentlich zu entfernenden Rolle durchgeführt. Sollte davon nur eine Operation fehlschlagen, gibt es keine Änderungen, ansonsten werden alle Operationen durchgeführt. Das URA97 Modell beschreibt jedoch nicht, wie Nutzer erstellt werden. Ein weiteres Problem ist, dass das Hinzufügen und Entfernen nicht symmetrisch ist, da nur beim Hinzufügen zuvor definierte Bedingungen überprüft werden. Bei der Frage, ob ein starkes oder schwaches Entfernen günstiger ist, ist hingegen keine abschließende Beurteilung möglich, da sich hier vor allem die Frage stellt, was in der konkreten Anwendung gewünscht ist.

4.3 Das Modell PRA97

Das zweite Teilmodell ist das PRA97(permission-role assignement '97) Modell, welches das Hinzufügen und Entfernen von Rechten bei Rollen steuert. Der Aufbau ist hierbei praktisch identisch zum URA97 Modell. Es wird geprüft, ob die nötigen Rechte für die Operation vorhanden sind und ob die Rolle anschließend keine zuvor festgelegten Bedingungen verletzt. Beim Entfernen von Rechten wird, wie beim URA97 Modell, das schwache Entfernen implementiert, sodass ein Recht nur entfernt wird, wenn es explizit Teil der Rolle ist. Das starke Entfernen, welches nicht implementiert ist, wird äquivalent zum URA97 definiert.

4.4 Das Modell RRA97

Das letzte Modell aus ARBAC97 ist RRA97(role-role assignement), welches sich mit dem Zuweisen von Rollen zu Rollen befasst. Dabei wird zunächst eine Unterscheidung zwischen unterschiedlichen Rollentypen gemacht. Rollen, welche nur Rechte und andere Rechtesammlungen als Mitglieder haben können, Rollen, welche nur Nutzer und andere Nutzergruppen als Mitglieder haben können und allgemeine Rollen, welche keine Einschränkungen bei den Mitgliedern haben, also sowohl Nutzer als auch Rechte beinhalten können. Für das Zuteilen in die neu eingeführten Rollentypen aus Eigenschaften bzw. Nutzern werden die ARA97 und GRA97 verwendet, welche äquivalent zu URA97 und PRA97 funktionieren. Sämtliche Operationen für das Hinzufügen und Entfernen von Rollen werden von einer Relation überwacht, welche überprüft, ob die Person die Rechte besitzt, Änderungen vorzunehmen. Beim Hinzufügen und Entfernen von Rollen muss unterschieden werden zwischen Rollen am Rand der Hierarchie und Rollen mitten in der Hierarchie. Rollen am Rand können problematisch sein, wenn sie zwei zuvor disjunkte Hierarchien verbinden. In diesem Fall muss sichergestellt werden, dass keine existierenden Bedingungen verletzt werden. Das Entfernen ist in solchen Fällen jedoch ohne Probleme möglich. Bei Operationen innerhalb der Hierarchie muss immer das Einhalten der Bedingungen überprüft werden. Insgesamt stellt das Zuweisen und Entfernen von Rollen in der Hierarchie das wohl komplexeste Problem dar, welches vom ARBAC97 Modell behandelt wird, da durch die hierarchische Ordnung und transitive Relation viele Wechselwirkungen zwischen den einzelnen Rollen entstehen. Das ARBAC97 Mo-

dell stellt jedoch insgesamt eine große Vielfalt an Operationen bereit, um ein auf Rollen basierendes System, verteilt zu verwalten, sodass kein alles überspannender „Superadmin“ entsteht.

4.5 Beispiel für das Entfernen von Nutzern

Abbildung 4.1 zeigt ein sehr einfaches, hierarchisches Schema von Rollen. Das schwache Entfernen von einem Nutzer aus einer Rolle lässt sich daran leicht verdeutlichen. Wenn ein Nutzer explizites Mitglied in Rolle B ist und jemand mit den entsprechenden Rechten den Nutzer aus Rolle C entfernen will, bleibt dies ohne Effekt, da der Nutzer nur indirektes Mitglied der Rolle ist, da er dies als Mitglied von Rolle B geerbt hat. Wenn nun dieser Nutzer aus Rolle B entfernt werden soll, ist dies möglich, da dieser explizites Mitglied ist und nicht Mitglied einer übergeordneten Rolle, aus welcher er diese Mitgliedschaft geerbt hat. Das starke Entfernen würde in diesem Fall anders funktionieren. Sei der Nutzer nun explizites Mitglied in Rolle A. Nehmen wir weiter an, der Nutzer soll aus Rolle C entfernt werden. Dann wird nun entlang der Hierarchie nach oben gegangen, um die Rolle zu finden, in welcher der Nutzer explizites Mitglied ist, in diesem Fall Rolle A. Nun wird dort ein schwaches Entfernen durchgeführt, dann in Rolle B und zuletzt in Rolle C. Wenn die Person nur das Recht hat, Nutzer aus Rolle C zu entfernen, bleibt die Aktion ohne Folgen, da das Entfernen aus A und B fehlschlägt. Besitzt die Person hingegen das Recht aus A, B und C zu entfernen, werden alle drei Operationen erfolgreich durchgeführt und der Nutzer wird aus allen drei Rollen entfernt.

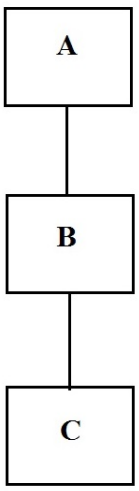

Abbildung 4.1: Beispiel einer transitiven Relation

5 Trust Management durch Zertifikate

5.1 Das Ziel beim Trust Management

Beim Trust Management geht es um die Frage, wie bei geteilten Systemen die Autorisierung der Nutzer erfolgen soll. Dies bedeutet, dass der Nutzer derjenige ist, für den er sich ausgiebt und ob dieser die Rechte für die Operation, die er ausführen will, tatsächlich besitzt.

5.2 Das Modell SPKI/SDSI

SPKI/SDSI ist ein Modell, welches versucht mit zwei zentralen Bestandteilen dieses Problem in den Griff zu bekommen. Dies wurde in der ursprünglichen Quelle definiert[DJ10]. Zum einen gibt es lokale Namensräume, welche jeder Benutzer erstellen kann, die diesem dann untergeordnet werden. Der zweite zentrale Aspekt ist der der Delegation. Hierbei handelt es sich um Zertifikate, welche Autorisierungen vergeben oder die Erlaubnis Autorisierungen zu vergeben erteilen. Die Überprüfung von Autorisierungen erfolgt nun mit zwei Arten von Zertifikaten. Namenszertifikate, welche einen Namen im Namensraum des Ausstellers definieren und Autorisierungszertifikate, welche bestimmte Rechte vergeben oder delegieren. Dabei ist es möglich, das weitere Delegieren der Rechte zu erlauben oder zu verbieten. Dabei wird davon ausgegangen, dass für die Autorisierungszertifikate eine partielle Ordnung existiert, sodass mit einem Zertifikat nicht mehr Rechte vergeben werden können, als durch es selbst vergeben werden. Der zentrale Mechanismus, welcher nun zur Sicherstellung der Rechteverwaltung dient, ist das Werkzeug der Zertifikatketten. Dabei wird, sobald ein Nutzer eine Operation durchgeführt hat, die Delegation der Zertifikate nachvollzogen und als eine Kette zurückgegeben, sollte diese zulässig sein. So soll sichergestellt werden, dass der Nutzer die Rechte tatsächlich besitzt. Dieses Modell lässt sich sehr einfach auf ein Rollenmodell übertragen, da anstelle der einfachen Berechtigungen auch Rollenmitgliedschaften beschrieben werden können. Zertifikatketten sind somit ein einfaches Mittel, um bei rollenbasierten Systemen eine Überprüfung der Nutzer sicherzustellen, damit das Rollenmodell tatsächlich eingehalten wird.

5.3 Beispiel für eine Zertifikatkette

In Abbildung 3 ist eine Relation zwischen Nutzern abgebildet, welche dem Aufbau der Zertifikatketten entspricht. Der Administrator hat an den Gruppenleiter dabei das Recht gegeben, die für das aktuelle Projekt nötigen Daten auf dem Server der

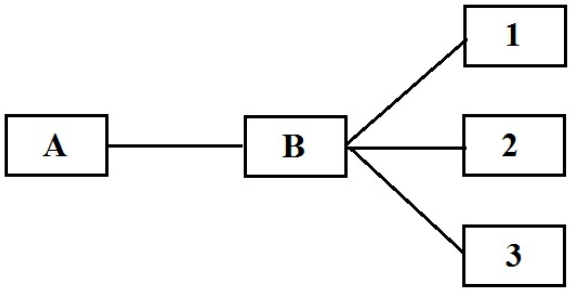

Abbildung 5.1: Beispiel einer Nutzerbeziehung

Firma einzusehen. Außerdem hat der Admin das Recht erteilt, dieses Recht weiter zu vergeben. Die vom Gruppenleiter vergebenen Rechte sind dabei zeitlich begrenzt. Das Recht, diese Daten einzusehen, besitzen nun Nutzer 1,2 und 3. Dabei liegen der Gruppenleiter im Namensraum des Admins und die Nutzer im Namensraum des Gruppenleiters. Wenn nun ein Nutzer mit dem ihm ausgestellten Zertifikat auf die Daten zugreifen will, wird vom Nutzer aus eine Zertifikatkette ermittelt, welche die Delegation der Rechte darlegt, bis zu dem Punkt, an dem die Rechte für die gewünschte Ressource verwaltet werden. Diese Kette übermittelt der Nutzer nun bei seinem Zugriff auf die Daten. Vom Datenhalter wird diese Kette nun auf ihre Gültigkeit geprüft. Diese Prüfung kann zum Beispiel fehlschlagen, weil ein zeitlich begrenzt vergebenes Zertifikat inzwischen abgelaufen ist. Wenn nun in diesem Beispiel Nutzer 1 seine Rechte zeitlich eher als Nutzer 2 und 3 erhalten hat, gibt es im System einen Zeitpunkt, zu welchem für Nutzer 1 der Zugriff verweigert wird, während Nutzer 2 und 3 weiter problemlos auf die Resourcen zugreifen können. In diesem Beispiel ist explizit nur von einzelnen Rechten die Rede, jedoch stellt dies kein Hindernis dar, um das System auf rollenbasierte Systeme zu übertragen. Dafür müsste lediglich anstelle der Weitergabe von Rechten die Weitergabe von Rollen stehen. Insgesamt jedoch zeigt das System einen einfachen Weg, die nötige Kontrolle in ein System zu integrieren.

6 Die vorgestellten Systeme in der Praxis: Ein Fazit

6.1 Die Bewertung von matrixbasierten Systemen

Die matrixbasierten Systeme hinterlassen einen vergleichsweise schwachen Eindruck in der Bewertung der Praxistauglichkeit, da mehrere Probleme entstehen. Zum einen ist die Verwaltung der entsprechenden Systeme nur sehr umständlich zu handhaben, da die Rechte für jeden einzelnen Nutzer explizit vergeben werden müssen, sodass nicht die Rechte einzelner Nutzergruppen gemeinsam in einem Schritt angepasst werden können. Zum anderen stellt sich das Problem der Leak-safety, wodurch das System nicht die vollständige Sicherheit bietet. Insgesamt sind diese Systeme nicht für die Praxis geeignet, da es inzwischen erheblich bessere Lösungen gibt.

6.2 Die Bewertung von rollenbasierten Systemen

Im Gegensatz zu den matrixbasierten Systeme bieten rollenbasierte Systeme eine gute Möglichkeit, Firmenstrukturen abzubilden und so die Nutzerrechte zu modellieren, ohne dies explizit für jeden einzelnen Nutzer machen zu müssen. Dazu kommt, dass es gute Möglichkeiten gibt diese Systeme zu verwalten. Für das Verwalten der Nutzer, der Rechte von Rollen und der Rollenhierarchie gibt es ein Modell, was einen erheblichen Vorteil gegenüber matrixbasierten Systemen darstellt. Zuletzt gibt es bei rollenbasierten Systemen mit Hilfe von Zertifikaten eine gute Möglichkeit, die Delegation von Rollen zu überprüfen, um bei Zugriffen im System die Einhaltung des Modells sicherzustellen. Insgesamt stellt das rollenbasierte System somit eine gute Möglichkeit dar, eine Rechteverwaltung zu implementieren.

Literatur

[DJ10] Anupam Datta und Somesh Jha. *Analysis Techniques for Information Security*. Synthesis Lectures on Information Security, Privacy, And Trust. San Rafael, CA: Morgan und Claypool, 2010.

[SB99] Ravi Sandhu und Venkata Bhamidipati. „The ARBAC97 model for role-based aministration of roles“. In: *ACM Transactions on Information and Systems Security*. ACM Press, 1999, S. 105–135.

[SM01] Andres Schaad und Jonathan Moffett. „The role-based access control system of a European bank: A case study and discussion“. In: *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies*. ACM Press, 2001, S. 3–9.