GMD Research Series

GMD –
Forschungszentrum
Informationstechnik
GmbH

Winfried Kühnhauser

Metapolitiken

N° 13/1999

© GMD 1999

GMD – Forschungszentrum Informationstechnik GmbH
Schloß Birlinghoven
D-53754 Sankt Augustin
Germany
Telefon +49 -2241 -14 -0
Telefax +49 -2241 -14 -2618
http://www.gmd.de

In der Reihe GMD Research Series werden Forschungs- und Entwicklungsergebnisse aus der GMD zum wissenschaftlichen, nicht-kommerziellen Gebrauch veröffentlicht. Jegliche Inhaltsänderung des Dokuments sowie die entgeltliche Weitergabe sind verboten.

The purpose of the GMD Research Series is the dissemination of research work for scientific non-commercial use.
The commercial distribution of this document is prohibited, as is any modification of its content.

Anschrift des Verfassers/Address of the author:
Winfried Kühnhauser
Institut für Autonome intelligente Systeme
GMD – Forschungszentrum Informationstechnik GmbH
D-53754 Sankt Augustin
E-mail: Winfried.Kuehnhauser@gmd.de

Die Deutsche Bibliothek - CIP-Einheitsaufnahme:

Kühnhauser, Winfried:
Metapolitiken / Winfried Kühnhauser. GMD – Forschungszentrum Informationstechnik GmbH. - Sankt Augustin :
GMD – Forschungszentrum Informationstechnik, 1999
(GMD Research Series ; 1999, No. 13)
Zugl.: Chemnitz, Univ., Habil.-Schr., 1999
ISBN 3-88457-362-4

ISSN 1435-2699
ISBN 3-88457-362-4

Zusammenfassung

Moderne Unternehmen in Wirtschaft und Verwaltung verwenden heute Organisationsformen, in denen einzelne Unternehmensbereiche und Abteilungen mit einem hohen Maß an Autonomie versehen sind. Diese Eigenständigkeit ermöglicht es, in individuellen Unternehmensstrategien spezifische Besonderheiten der Bereiche zu berücksichtigen und auf diese Weise ihre Effizienz zu befördern.

Dies bleibt nicht ohne Auswirkungen auf die Struktur der IT-Systeme solcher Unternehmen. Wird eine derartige Unternehmensstruktur auf ein IT-System abgebildet, so entsteht ein unternehmensweites verteiltes System mit einer Vielzahl von Domänen, wobei jede dieser Domänen die individuellen unternehmerischen Strategien und Organisationsformen der Bereiche reflektiert.

Im Vordergrund dieser Arbeit stehen die Sicherheitsstrategien oder Sicherheitspolitiken, mit denen die bereichsspezifischen Regeln zum Umgang mit sicherheitsrelevanten Informationen formuliert werden. Charakteristisches Merkmal entsprechender IT-Systeme ist ihre Strukturierung in Sicherheitsdomänen, in denen spezifische, in einer Sicherheitspolitik festgelegte Regeln durchgesetzt werden.

IT-Systeme mit einer Vielzahl von Sicherheitspolitiken werden als Multipolitikensysteme bezeichnet und bilden den thematischen Rahmen dieser Arbeit. Dabei wird insbesondere auf die Konsequenzen einer neuen Referenzarchitektur eingegangen, in der die grundlegenden Eigenschaften sämtlicher zukünftiger IT-Systeme in der amerikanischen Administration formuliert sind. Thema dieser Arbeit ist die Formulierung und Implementierung der Sicherheitseigenschaften von Multipolitikensystemen mittels Metapolitiken. Metapolitiken integrieren die regulären Sicherheitpolitiken eines Multipolitikensystems gemeinsam mit Konfliktpolitiken und Vollständigkeitspolitiken zu einer einzigen Struktur und schaffen auf diese Weise einen eindeutigen Referenzpunkt für die Diskussion und Analyse der Sicherheitseigenschaften eines Multipolitikensystems. Hierzu wird ein formales Modell auf der Grundlage des Zugriffssteuerungskalküls von Harrison/Ruzzo/Ullman entwickelt, in welchem dann allgemeine Sätze über die Entscheidbarkeit des Safety-Problems für Metapolitiken und die Laufzeitkomplexität einer Implementierung formuliert werden. Nichtzuletzt liefert dieses Modell einen detaillierten Konstruktionsplan, der in der Praxis eine automatische Komposition von Metapolitiken aus regulären Sicherheitspolitiken, Konfliktpolitiken und Vollständigkeitspolitiken erlaubt.

Abstract

In large organisations, many branches and departments have their own unique information security requirements that reflect the specific needs of the individual organisational unit. Mapping the structure of such organisations to a computer system results in a large distributed system with a multitude of information security domains, each domain having its individual and unique security policy. Distributed systems supporting a multitude of security policies are called distributed multipolicy systems.

In 1993, the move towards multipolicy systems was taken up by the United States Department of Defense. As part of DoD's *Technical Architecture Framework for Information Management* several new challenges imposed by multipolicy systems were identified.

This work aims at giving answers to these challenges. It describes the design and implementation of multipolicy systems by an approach called policy groups. Policy groups compose the regular security policies of a multipolicy system together with conflict resolution policies and completeness policies into a single structure, hereby providing a single point of reference for the discussion and analysis of a multipolicy system's security properties. To this end, a formal model is developed based on Harrison/Ruzzo/Ullman's access control calculus. The model allows statements about the decidability of the safety problem for policy groups and about the computational complexity of an implementation. Last not least, the model provides a detailed construction plan that in practice can be used for the automatic composition of a policy group model.

Inhaltsverzeichnis

1 Einführung **9**

1.1 Zielsetzung 10

1.2 Gliederung 11

2 Sicherheit in verteilten Systemen **13**

2.1 Grundbegriffe der Informationssicherheit 13

2.1.1 Sicherheitsrisiken 15

2.1.2 Sicherheitsanforderungen 18

2.1.3 Sicherheitspolitiken 21

2.1.4 Sicherheitsmodelle 22

2.1.5 Sicherheitsarchitekturen und Infrastrukturen 23

2.1.6 Sicherheitsmechanismen 25

2.2 Safety und Security 25

2.3 Daten und Informationen 27

2.4 Verteilte Systeme 28

2.5 Zusammenfassung 33

3 Sicherheitspolitiken **37**

3.1 Die Rolle der Sicherheitspolitiken 37

3.2 Modellierung von Sicherheitspolitiken 40

3.3 Modelle für Zugriffssteuerungspolitiken 41

3.3.1 Zugriffssteuerung 42

3.3.2 Kalküle und konkrete Modelle 43

3.3.3 Zusammenfassung 49

3.4 Spezifikation von Sicherheitspolitiken 51

3.4.1 Skippy 51
3.4.2 Vordefinierte Skippy-Elemente 52
3.4.3 Ein Beispiel: Chinese-Wall als Skippy-Deklaration 53
3.4.4 Die Syntax von Skippy 60
3.5 Implementierung von Sicherheitspolitiken 62
3.5.1 Ausgangspunkt 62
3.5.2 Skippy-Spezifikationen als abstrakte Datentypen 64
3.5.3 Eine Sicherheitsarchitektur für Custodians 67
3.5.4 Ein Beispiel: Chinese-Wall als C++ - Custodian 68
3.6 Zusammenfassung 73

4 Metapolitiken 75
4.1 Interaktionen zwischen Politikdomänen 77
4.2 Metapolitiken 82
4.2.1 Unvollständigkeit 82
4.2.2 Konflikte 82
4.2.3 Die Aufgaben von Metapolitiken 83
4.2.4 Definition 85
4.3 Diskussion 88
4.3.1 Metapolitiken über MLS und Chinese-Wall 90
4.3.2 MP-freie Interaktionen 91
4.3.3 Interaktionen im politikfreien Raum 92
4.3.4 Interaktionen im Konfliktraum 95
4.4 Analyse von Metapolitiken 96
4.4.1 HRU-Safety 97
4.4.2 Safety von Metapolitiken 100
4.5 Zusammenfassung 108

5 Spezifikation von Metapolitiken 111
5.1 Elemente einer Metapolitikspezifikation 111
5.2 Skippy$_M$ 112
5.3 Ein Beispiel 114
5.4 Die Syntax von Skippy$_M$ 118

5.5 Zusammenfassung 119

6 Implementierung von Metapolitiken 121

6.1 Spezifikationselemente 122

6.1.1 Metapolitik-Zustand 122

6.1.2 Metapolitikspezifische Operationen 123

6.2 Sicherheitsarchitektur 130

6.3 Zusammenfassung 131

7 Zusammenfassung 133

A Verwendung von Bezeichnern 137

B Eine Metapolitik als C++ – Klasse 139

Literaturverzeichnis 151

1 Einführung

Die Informationstechnologie wird heute in einer Vielzahl von Anwendungen in Wirtschaft, Verwaltung und Forschung eingesetzt. Einer professionellen Nutzung ihrer Möglichkeiten stehen allerdings erhebliche Risiken gegenüber. Insbesondere in sensiblen Anwendungsgebieten der Informationstechnik wie etwa im Rechtswesen, im Bankwesen oder im medizinischen Informationswesen sind gesetzliche Regelungen, ethische Grundsätze, wirtschaftliche und private Interessen zu beachten, die den Umgang mit sicherheitsrelevanten Informationen und Ressourcen regulieren. Innerhalb der Informatik wird dieses Themenfeld mit den Begriffen Datensicherheit, Datenschutz und Informationssicherheit verbunden.

Die Ursachen der Risiken liegen vielfach darin, daß Sicherheitsaspekte informationstechnischer Systeme lange Zeit ignoriert und auch unterschätzt wurden. In den letzten beiden Jahrzehnten hat sich hier jedoch eine deutliche Trendwende vollzogen. Heute kann eine durchaus breite Sensibilisierung der Öffentlichkeit für dieses Thema beobacht werden, nicht zuletzt vorangetrieben durch eine Presse, die stets bereit ist, Nachrichten über Verletzungen der Informationssicherheit ihren Lesern ausführlich zu präsentieren. Eine gesellschaftliche Akzeptanz des Einsatzes informationstechnischer Systeme in sensiblen Bereichen ist heute nicht mehr zu erwarten, wenn nicht gleichzeitig auch Vertrauen in die Sicherheit dieser Systeme vermittelt werden kann.

Die wachsende gesellschaftliche, ökonomische und soziale Relevanz einer verläßlichen Wahrung der Informationssicherheit hat in den letzten Jahren zu deutlichen Fortschritten des Standes der Kunst auf diesem Gebiet geführt. Hierbei spielen zunehmend Sicherheitspolitiken oder Sicherheitsstrategien eine grundsätzliche Rolle. Beide Begriffe werden im deutschsprachigen Raum für den englischen Begriff *security policy* angetroffen und bezeichnen Regelwerke, die den Umgang mit sicherheitsrelevanten Informationen beschreiben. Auf Grund der Nähe zum englischen Begriff wird in dieser Arbeit durchgängig der Begriff Sicherheitspolitik verwendet.

Begründet ist die zentrale Rolle der Sicherheitspolitiken durch die Ganzheitlichkeit und Präzision, mit der individuelle Sicherheitsziele und Strategien formulierbar werden, wodurch sich die Möglichkeit eröffnet, mittels formaler Verfahren Nachweise über Sicherheitseigenschaften informationstechnischer Systeme führen zu können. So befinden sich beispielsweise Sicherheitspolitiken für Klinikinformationssysteme in der Entwicklung, die auf der Grundlage von Informationsflußmodellen präzise Regeln zum Schutz der In-

tegrität, der Vertraulichkeit, der Verfügbarkeit und der Verbindlichkeit medizinischer Daten beschreiben.

1.1 Zielsetzung

Die Organisationsform moderner Unternehmen in Wirtschaft und Verwaltung schafft heute Strukturen, in denen einzelne Unternehmensbereiche und Abteilungen mit einem hohen Maß an Autonomie versehen sind und als eigenständig wirtschaftende Einheiten auftreten, die in der Regel als *Profit Centers* bezeichnet werden. Diese Organisationseinheiten haben vielfach individuelle Sicherheitsanforderungen, welche die spezifischen Besonderheiten der jeweiligen Organisationseinheit berücksichtigen und sich erheblich voneinander unterscheiden können. Wird eine solche Unternehmensstruktur auf ein informationstechnisches System (im folgenden kurz IT-System) abgebildet, so entsteht ein verteiltes System mit einer Vielzahl von Sicherheitsdomänen, in dem jede Domäne eine eigene Sicherheitspolitik mit individuellen Regeln zum Umgang mit sicherheitsrelevanten Informationen innerhalb dieser Domäne besitzt.

Eine ähnliche Domänenstruktur entsteht in Rechnernetzen, in denen unabhängige Organisationen elektronischen Geschäftsverkehr abwickeln oder in Informationsverbunden miteinander kooperieren. Auch hier führen die Autonomie und die individuellen Sicherheitsanforderungen der einzelnen Organisationen zu individuellen Sicherheitspolitiken auf den organisationseigenen IT-Systemen, und es entstehen Sicherheitsdomänen, die typischerweise durch die Zugehörigkeit von IT-Systemen zu Organisationen charakterisiert sind.

IT-Systeme mit einer Vielzahl von Sicherheitspolitiken werden als Multipolitikensysteme bezeichnet. Ihr charakteristisches Merkmal ist ihre Strukturierung in Sicherheitsdomänen, in denen spezifische, in einer Sicherheitspolitik festgelegte Regeln gelten. Hierdurch läßt sich nun auf der einen Seite innerhalb der Domänengrenzen ein qualitativ hochwertiges Niveau der Informationssicherheit erzielen; auf der anderen Seite bringen Multipolitikensysteme jedoch das Problem mit sich, daß jegliche Form der globalen Kooperation die Überwindung von Domänengrenzen erfordert. Multipolitikensysteme konterkarieren somit die Interoperabilität, die als eine der elementaren Errungenschaften verteilter Systeme angesehen wird.

Dieses Spannungsfeld zwischen Interoperabilität und Informationssicherheit in Multipolitikensystemen ist Thema dieser Arbeit. Es wird ein Konzept vorgestellt, mit dem die durch Sicherheitspolitiken errichteten Domänengrenzen auf kontrollierte Weise überwunden und Interaktionen *zwischen* Politikdomänen mit einer analogen Qualität der Sicherheit ausgestattet werden, wie sie das Konzept der Sicherheitspolitiken *innerhalb* einer Domäne erzielt. Ausgangspunkt ist die Idee, Regeln für Interaktionen zwischen Sicherheitspolitikdomänen in übergeordneten Metapolitiken zu formulieren; in der Weise wie Sicherheitspolitiken ein Basiskonzept für sichere Interaktionen innerhalb einer Politikdomäne darstellen, entwickelt die vorliegende Arbeit das Konzept der

Metapolitiken als Grundlage für sichere Interaktionen zwischen Politikdomänen und schafft so eine unabdingbare Voraussetzung für die Interoperabilität in zukünftigen Multipolitikensystemen.

1.2 Gliederung

Die Arbeit beginnt mit einer Einführung in die grundsätzlichen Anliegen, Begriffe und Konzepte der Informationssicherheit. Insbesondere wird hierbei auf die Besonderheiten der Informationssicherheit in verteilten Systemen eingegangen, da solche Systeme das informationstechnische Umfeld der Arbeit bilden und hieraus eine wesentliche Motivation für das Thema erwächst. Zur Veranschaulichung wird ein Beispielszenarium entwickelt, welches sich in seinen Teilaspekten durch die gesamte Arbeit ziehen wird.

Das folgende Kapitel befaßt sich mit Wesen und Aufgaben von Sicherheitspolitiken; es beschreibt ihre Rolle als methodischen Ansatz bei Beschreibung und Implementierung sicherer Systeme und stellt ein durchgängiges Konzept zur Modellierung, Analyse, Spezifikation und Implementierung von Sicherheitspolitiken vor. Ausgehend von einer Betrachtung bekannter Methoden und Kalküle zur Modellierung und Analyse von Sicherheitspolitiken wird eine Spezifikationssprache entwickelt, die den Paradigmenwechsel von Modell zu Implementierung vereinfacht. Schließlich wird ein Konzept zur Implementierung von Sicherheitspolitiken entwickelt, welches anders als klassische listenbasierte Implementierungskonzepte eine algorithmische Formulierung von Sicherheitspolitiken ermöglicht und hierdurch eine erheblich höhere Ausdruckskraft erzielt.

Anschließend wird das Konzept der Metapolitiken entwickelt. Ausgangspunkt ist die *TAFIM* - Referenzarchitektur (*Technical Architecture Framework for Information Management*, [DIS96]), aus der grundlegende Anforderungen und Eigenschaften eines Metapolitikkonzepts abgeleitet werden. Eine präzise Identifikation der Problemfelder der Interoperabilität in Multipolitikensystemen konkretisiert diesen Ansatz und bildet den Ausgangspunkt für die Entwicklung eines durchgängigen Konzepts zur Modellierung, Analyse, Spezifikation und Implementierung von Metapolitiken.

Im Gegensatz zur Modellierung von Sicherheitspolitiken kann bei der Modellierung von Metapolitiken nicht auf bekannte Modelle aufgebaut werden. Daher wird hierzu zunächst ein geeigneter Begriffsapparat geschaffen. Zur Analyse bestimmter Sicherheitseigenschaften so modellierter Metapolitiken wird der von Harrison, Ruzzo und Ullman entwickelte HRU-Kalkül aufgegriffen und mit Ausdrucksmöglichkeiten zur Beschreibung von Metapolitiken versehen. Hierin werden dann Aussagen über die grundsätzlichen Möglichkeiten und Grenzen der Analysierbarkeit von Metapolitiken aufgestellt.

Die Entwicklung einer Spezifikationssprache für Metapolitiken dient anschließend der Umsetzung eines Metapolitikmodells in eine konkrete Implementierung. Die Sprachkonstrukte der Spezifikationssprache werden unmittelbar aus dem Metapolitikmodell

abgeleitet; hierdurch können Metapolitikspezifikationen in modellnahen Paradigmen formulieren werden, die nicht mit problemfremden Eigenschaften allgemeiner Spezifikationssprachen überfrachtet sind. Die Vorzüge dieses Ansatzes liegen in der intuitiven Abbildbarkeit eines Modells auf eine semantisch nahestehende Sprache und in einer weitgehend automatisierbaren Generierung von Programmcode aus einer Spezifikation.

Im letzten Kapitel der Arbeit wird ein Implementierungskonzept für Metapolitiken vorgestellt. Ausgehend von einer Metapolitikspezifikation wird ihre Transformation in ein ausführbares Programm vorgenommen. Die Einbettung so erzeugter Metapolitiken in eine konkrete Sicherheitsarchitektur baut auf dem zuvor entwickelten Paradigma zur Implementierung von Sicherheitspolitiken auf. Metapolitiken werden hierbei als abstrakte Datentypen formuliert, die durch ihre Integration in eine Sicherheitsarchitektur mit Mechanismen ausgestattet werden, die die Metapolitik zu ihrer Durchsetzung benötigt. Auf der Grundlage dieses Konzeptes wird schließlich eine konkrete Implementierung vorgestellt, die einerseits den exemplarischen Nachweis der Machbarkeit führt und andererseits eine konkrete Diskussion der Laufzeitkomplexität elementarer Funktionen zur Realisierung des Metapolitikkonzepts ermöglicht.

2 Sicherheit in verteilten Systemen

Die ursprüngliche Initiative zu einer systematischen Auseinandersetzung mit der Sicherheit informationstechnischer Systeme ging vom amerikanischen Verteidigungsministerium aus. Die in diesem Kontext entwickelten Sicherheitskonzepte waren folglich zunächst auch von den Anforderungen militärischer Einrichtungen geprägt und verfolgten hauptsächlich das einzige Ziel, die Vertraulichkeit von klassifizierten Dokumenten zu wahren.

Das breite Anwendungsspektrum heutiger IT-Systeme läßt eine solch schmalbandige Sichtweise nicht mehr zu. Sensible Anwendungsgebiete der Informationstechnik wie etwa das Rechtswesen, das Bankwesen oder das medizinische Informationswesen haben das Spektrum relevanter Sicherheitsaspekte heute erheblich erweitert. Neben der Vertraulichkeit zählen inzwischen die Wahrung von Integrität, Verbindlichkeit, Originalität und Verfügbarkeit zu den grundsätzlichen Anliegen der Informationssicherheit.

Dieses Kapitel führt in die Thematik sicherer IT-Systeme ein und entwickelt ein Beispielszenarium, an dem grundsätzliche Begriffe, Konzepte und Methoden der Informationssicherheit veranschaulicht werden. Insbesondere wird hierbei auf die charakteristischen Merkmale verteilter Systeme eingegangen, da sich hieraus eine wesentliche Motivation für die vorliegende Arbeit ableitet.

2.1 Grundbegriffe der Informationssicherheit

Als Beispielszenarium dient ein Technologiekonzern, welcher hochentwickelte Produkte für die Luft- und Raumfahrt herstellt; seine Produktpalette umfasse technisch anspruchsvolle Hardware/Softwarekomponenten für Satellitennavigation, für Cockpit-Klimasteuerungen oder für das Motorenmanagement von Fluggeräten. Aus wirtschaftlichen Gründen besitzt der Konzern Standorte in Ländern, in denen er seine Produkte vermarktet; so nehmen örtliche Entwicklungsabteilungen Anpassungen der Produkte an landesspezifische Merkmale wie Sprache, technische Normen oder Gesetze vor, und örtliche Marketingabteilungen beobachten landesspezifische Märkte und pflegen die persönlichen Kontakte zu Kunden. Andere Unternehmensbereiche wie die Forschung und der Einkauf sind dagegen zentral organisiert.

Der Erfolg eines solchen Technologieunternehmens hängt in hohem Maße von der Lei-

stungsfähigkeit und dem Wissen seiner Forschungs- und Entwicklungsabteilungen ab. Der Ausbau des Wissen, die Wahrung des Technologievorsprungs und die schnelle Umsetzung von Forschungsergebnissen in Produkte ist ohne den Einsatz von IT-Systemen heute nicht mehr denkbar. Gleichzeitig stellen diese IT-Systeme für das Unternehmen aber auch ein erheblichen Risiko dar; eine Verletzung der Informationssicherheit, die beispielsweise zu einem ungewollten Bekanntwerden von Forschungsergebnissen führt, kann für das Unternehmen einen existenzbedrohenden Schaden bedeuten.

Am Beispiel der (ausschnittweisen) Entwicklung eines Sicherheitskonzeptes für dieses Szenarium sollen im folgenden grundlegende Ziele und Begriffe der Informationssicherheit erläutert werden. Als ganzheitlicher Ansatz werden hier zunehmend die auch in der Softwareentwicklung verwendeten Prozeßmodelle eingesetzt; solche Modelle stellen Risiken, Sicherheitsanforderungen, Sicherheitspolitiken, Sicherheitsmechanismen, Sicherheitsarchitekturen und Sicherheitsinfrastrukturen in einen kausalen Zusammenhang, in dem letztlich jeder einzelne Sicherheitsmechanismus eines IT-Systems durch ein bestimmtes Risiko begründet ist.

Bild 2.1 zeigt ein Beispiel für ein solches Prozeßmodell[1]. Der Einstieg erfolgt mit einer Analyse möglicher *Bedrohungen* und *Schwachstellen*, leitet hieraus relevante *Sicherheitsrisiken* ab und bewertet sie. Aus dieser Bewertung leitet sich dann ein Katalog von *Sicherheitsanforderungen* ab. Dieser Katalog bildet die Grundlage der Definition strategischer Regeln zum Umgang mit sicherheitsrelevanten Informationen, die in ihrer Gesamtheit als *Sicherheitspolitik* bezeichnet werden. Die präzise Formulierung einer Sicherheitspolitik in der strengen Form eines formalen *Sicherheitsmodells* erlaubt dann eine Analyse der erreichten Sicherheitseigenschaften und deren Vergleich mit den ursprünglichen Sicherheitsanforderungen. Gleichzeitig bildet das Sicherheitsmodell den Ausgangspunkt für die *Spezifikation* und *Implementierung* der Sicherheitspolitik, die in einer konkreten *Sicherheitsarchitektur* unter Verwendung konkreter *Sicherheitsmechanismen* im Kontext einer *Sicherheitsinfrastruktur* erfolgt.

[1] Zyklen des Prozesses sind aus Gründen der Übersichtlichkeit weggelassen worden

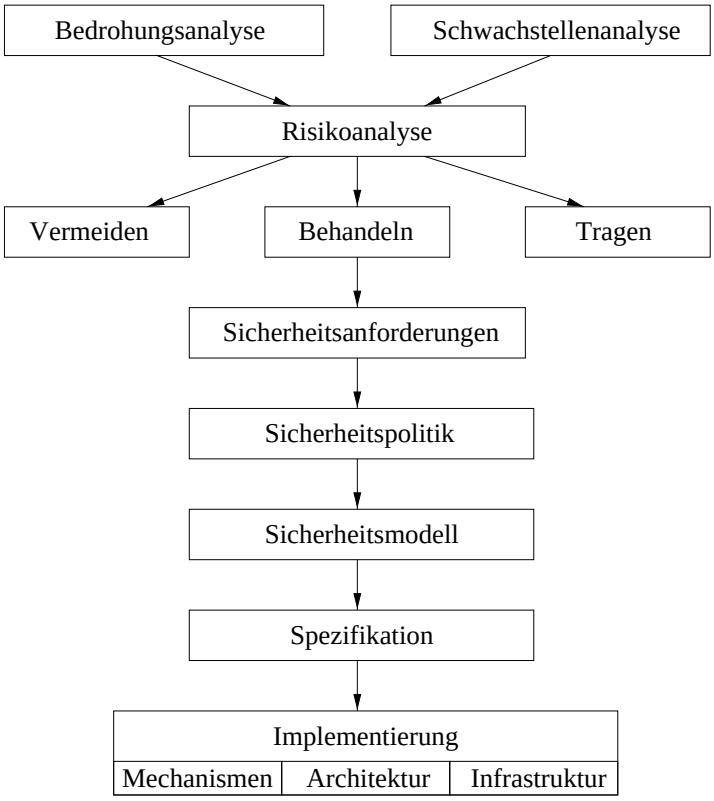


Bild 2.1: Entwicklungsprozeß eines Sicherheitskonzeptes

2.1.1 Sicherheitsrisiken

Der erste Schritt auf dem Wege zu einem sicheren IT-System ist eine sorgfältige Analyse der mit dem Betrieb des Systems verbundenen Risiken und ihre Bewertung. Ausgangspunkt dabei bilden die möglichen Bedrohungen und Schwachstellen, das Ziel ist die Identifikation, die Quantifizierung und die Klassifizierung der Risiken. Die heute bekannten Methoden der Risikoanalyse sind nicht formal und basieren auf Hypothesen, Heuristiken und der Erfahrung von Experten.

Im Szenarium des Technologiekonzerns entstehen Risiken beispielsweise dadurch, daß unbefugte Personen auf gespeicherte Dokumente zugreifen, sie lesen oder ihren Inhalt verändern. Dies kann zum Zweck der Industriespionage geschehen und auf das Auskundschaften von Forschungsergebnissen, produktstrategischen Planungen oder Vermarktungsstrategien abzielen; auch Betrug, Sabotage oder Erpressung durch Manipulationen an Bestellungen, Auftragsbestätigungen, Zahlungsanweisungen, Bilanzen oder Personalakten sind hierdurch möglich.

Dem Schutz vor unbefugten Zugriffen dieser Art dienen Zugriffssteuerungssysteme;

hierunter werden manipulationsgeschützte Softwarekomponenten verstanden, die in der Regel in Betriebssysteme, Datenbanksysteme oder Client/Serversysteme integriert sind und sämtliche Zugriffe innerhalb eines IT-Systems überwachen. Zugriffssteuerungssysteme können allerdings nur solche Zugriffe überwachen, die sie auch bemerken und beeinflussen können; Zugriffe also, die über die Softwareschnittstelle eines Dateisystems, einer Datenbank oder eines Servers erfolgen. Zugriffe über Wege außerhalb des IT-Systems, beispielsweise durch das Abhören von Kommunikationsverbindungen oder das Entfernen eines Datenträgers aus einem IT-System und anschließende Zugriffe durch ein fremdes IT-System stellen ein Risiko dar, welchem durch Zugriffssteuerungssysteme allein nicht begegnet werden kann.

Ein weiteres Sicherheitsrisiko des Beispielszenariums liegt somit in der Architektur des verteilten IT-Systems selbst begründet. Die verteilte Unternehmensstruktur des Beispielkonzerns begründet ein IT-System, in dem die IT-Systeme der verschiedenen Unternehmensbereiche durch lokale Netze oder Weitverkehrsnetze verbunden sind. Die Vernetzung wird für die verschiedensten Formen der Kommunikation oder Kooperation genutzt: zum Übertragen von Dateien, zur Versendung elektronischer Post, zum Verschicken von Produktionsaufträgen, Zahlungsanweisungen oder Software bis hin zur Unterstützung kooperativer Arbeit, beispielsweise bei der Erstellung von Konstruktionsplänen oder bei der Abstimmung zwischen zentraler Entwicklungsabteilung und den länderspezifischen Niederlassungen. Während Daten in unvernetzten Systemen vor physischen Zugriffen durch bauliche und administrative Maßnahmen (Rechenzentren mit Zugangskontrolle) geschützt werden können, sind dagegen Daten in Rechnernetzen vor dem Zugriff Dritter praktisch ungeschützt. Dabei ist es ganz gleich, ob eine verteilte Systemarchitektur durch lokale Netze oder Weitverkehrsnetze realisiert ist; selbst angemietete Standleitungen bieten nur einen minimalen Schutz, da die Gewährleistung ihrer physischen Integrität einen hohen technischen Aufwand erfordert, der wirtschaftlich in der Regel nicht vertretbar ist. In der verteilten Systemarchitektur des Beispielszenariums liegt folglich das grundsätzliche Risiko begründet, daß Daten während einer Kommunikation von Unbefugten mitgelesen oder verändert werden können.

An dieser Stelle liefert das Beispielszenarium bereits hinreichend Substanz zur Motivation und Veranschaulichung fünf grundsätzlicher Aspekte der Informationssicherheit.

Vertraulichkeit. Vertraulichkeit charakterisiert einen Zustand von Daten, in dem ein Informationsgewinn aus den Daten nur berechtigten Personen (respektive den für sie agierenden informationstechnischen Prozessen) möglich ist.

Ein Verlust der Vertraulichkeit führt somit zu einem Informationsgewinn für unbefugte Dritte. Die gewonnenen Informationen können dann zu Industriespionage, Betrug oder Erpressung verwendet werden; bestehen die Informationen aus Paßworten für Zugangsberechtigungen, so eröffnet sich für Dritte die Möglichkeit, durch Vortäuschen einer falschen Identität weitere Schäden durch Manipulationen an Bestellungen, Auftragsbestätigungen, Zahlungsanweisungen, Bilanzen oder Personalakten hervorzurufen.

Integrität. Integrität charakterisiert einen Zustand von Daten, in dem eine unberechtigte Manipulation erkennbar ist. Die Art der Manipulation ist dabei offen; sie kann im einfachen Fall aus der Veränderung des Inhalts eines Dokuments bestehen, sie kann aber auch, beispielsweise durch Verzögerung oder Wiederholen einer Kommunikation, im Verletzen einer zeitlichen Integritätsbedingung liegen.

Die Fähigkeit der Systeme, Manipulationen an Dokumenten und kommunizierten Daten erkennbar zu machen ist eine unabdingbare Voraussetzung für informationstechnische Anwendungen im Rechtswesen, im Bankwesen oder in medizinischen Informationssystemen. Im Beispielszenarium ist die Integrität von Auftragsbestätigungen, Planungsdokumenten oder Konstruktionsplänen ein wichtiger Schutz vor Betrug oder Sabotage.

Verfügbarkeit. Verfügbarkeit charakterisiert einen Zustand von Daten, in welchem sie durch berechtigte Personen (respektive den für sie agierenden informationstechnischen Prozessen) in angemessener Frist nutzbar sind.
Der Verlust der Verfügbarkeit führt somit zu unbefugter Beeinträchtigung der Responsivität eines IT-Systems.

Eine Beeinträchtigung autorisierter Inanspruchnahme von informationstechnischen Dienstleistungen, etwa hervorgerufen durch Unterbrechung der Kommunikation oder Verzögerung zeitkritischer Dienste, stellt einen Angriff auf die Verfügbarkeit dar.

In einer 1995 in den USA durchgeführten Studie zur Abhängigkeit der Unternehmen verschiedener Branchen von der Verfügbarkeit ihrer Informationstechnik wurde als spektakuläres Beispiel die Großbanken genannt. Begründet durch die immensen finanziellen Folgen aus einer unpünktlichen Erfüllung von Verbindlichkeiten gab die Studie für diese Unternehmen eine Überlebenszeit von lediglich zwei Tagen nach dem Gesamtausfall ihrer IT-Systeme an [Neu95]. Im Beispielszenarium führt ein Ausfall der Netzverbindungen zwischen den Niederlassungen unmittelbar zu einem Stillstand gemeinsamer Entwicklungsarbeiten der Entwicklungsabteilungen; ähnlich weitreichende Konsequenzen hat die Beeinträchtigung der Responsivität zentraler Server für sämtliche Geschäftsbereiche des Unternehmens.

Verbindlichkeit. Verbindlichkeit bezieht sich auf Handlungen, die mittels eines IT-Systems vorgenommen werden; sie charakterisiert einen Zustand der die Handlung repräsentierenden Daten, in welchem ein Beweis über ihre Integrität und ihre Authentizität führbar ist.
Auf diese Weise lassen sich Handlungen in IT-Systemen so dokumentieren, daß diese im Nachhinein nicht abgestritten werden können.

Typische Beispiele für verbindliche Handlungen im Beispielszenarium sind die Genehmigung von Entwicklungsvorhaben, Aufträge an die Beschaffungsabteilungen und korrespondierende Auftragsbestätigungen; weitere Beispiele aus anderen Szenarien sind

Einträge in elektronischen Grundbüchern, Medikamentenverordnungen in Klinikinformationssystemen und eine Vielzahl von Vorgängen im elektronischen Handel. Hierbei spielt insbesondere auch die *Rechts*verbindlichkeit eine Rolle; ihre Wahrung durch informationstechnische Maßnahmen setzt allerdings geeignete Rechtsgrundlagen voraus, die zum heutigen Zeitpunkt vielfach noch fehlen.

Originalität. Originalität bezieht sich auf Informationen auf einem IT-System, deren Vervielfältigung besonderen Regeln unterliegt; Beispiele sind Software oder urheberrechtlich geschützte Texte. Originalität charakterisiert einen Zustand der die Informationen repräsentierenden Daten, in dem sie geschützt vor unberechtigter Vervielfältigung sind.

Im Kontext des Technologieunternehmens spielt Originalität beispielsweise dann eine Rolle, wenn die Spezifikation eines Produkts zum Zwecke der Evaluierung einem Kaufinteressenten überlassen wird. Die Wahrung der Originalität verhindert hier, daß der Kaufinteressent unberechtigte Kopien anfertigen und für seine eigenen Interessen mißbrauchen kann. Viele Beispiele finden sich weiterhin in Szenarien aus dem Bereich des elektronischen Handels, wo die Wahrung der Originalität zum Urheberschutz von elektronischen Büchern oder Software unabdingbar ist.

Die in dieser Übersicht genannten Aufgaben sind zum heutigen Zeitpunkt besonders aktuelle Ziele der Informationssicherheit. Die Entwicklung der letzten 30 Jahre hat gezeigt, wie sehr in diesem noch jungen Gebiet die Dinge im Fluß sind; es ist daher zu erwarten, daß zukünftige Anwendungsgebiete einen steten Strom neuer Sicherheitsaspekte mit sich bringen werden.

2.1.2 Sicherheitsanforderungen

Maßnahmen zum Umgang mit Sicherheitsrisiken verursachen Kosten, deren Ausmaß in der Regel eine Abwehr aller denkbaren Bedrohungen unmöglich macht. Die Beurteilung der Höhe eines möglichen Schadens ist daher ein wesentlicher Faktor bei der Bewertung eines konkreten Risikos. So ist beispielsweise das Gefahrenpotential des Verlustes der Vertraulichkeit eines Dokuments abhängig von dessen Inhalt; im Beispielszenarium ist der Vertraulichkeitsverlust einer hausinternen Projektevaluation in der Regel weniger gravierend als der Vertraulichkeitsverlust wesentlicher Forschungsergebnisse gegenüber einer Konkurrenzfirma. Analog dazu besitzen verschiedene Szenarien der Informationstechnik insgesamt unterschiedliche Schadensrisiken; eine Datenbank mit medizinischen Patienteninformationen birgt beispielsweise ein erheblich größeres Risikopotential als die Datenbank eines Fahrplanauskunftsystems.

Eine Bewertung der Risiken berücksichtigt neben den Kosten von Sicherheitsmaßnahmen auch die Wahrscheinlichkeit eines Schadensfalls und die Kosten, die durch den

Eintritt eines Schadensereignisses entstehen. Problematisch ist dabei oft die Quantifizierung der Schadenskosten; als Beispiel möge der Vertrauensverlust einer Firma bei unpünktlichen Lieferungen, fehlerhaften Rechnungen oder Bekanntwerden der Weitergabe von Kundendaten dienen. Das Ergebnis einer Risikobewertung manifestiert sich schließlich in einer Risikoklassifizierung, in der zwischen zu akzeptierenden, zu vermeidenden und solchen Risiken unterschieden wird, denen mit konkreten Maßnahmen begegnet werden muß. Letztgenannte Klasse bildet den Ausgangspunkt für die Aufstellung von *Sicherheitsanforderungen*.

Unter einer Sicherheitsanforderung wird eine Anforderung an ein IT-System verstanden, die zur Abwehr von Sicherheitsrisiken dient. Sicherheitsanforderungen werden mit dem Ziel aufgestellt, die festgestellten Risiken vollständig abzudecken; die Menge aller Sicherheitsanforderungen an ein System bildet daher den vollständigen Anker für seine Sicherheitspolitik und ist die Meßlatte für seine Evaluierung. Sicherheitsanforderungen beschreiben somit, WAS zu tun ist, und Sicherheitspolitiken beschreiben das WIE.

Im Beispielszenarium ist ganz allgemein die Wahrung von Vertraulichkeit, Integrität und Verbindlichkeit von hoher Relevanz. Aus diesem Grund bezieht sich ein wesentlicher Teil der Sicherheitsanforderungen darauf, diese Sicherheitsaspekte zu konkretisieren; die folgenden Sicherheitsanforderungen zeigen Beispiele für Sicherheitsanforderungen zu diesen Aspekten.

Sicherheitsanforderung 1, Wahrung der Vertraulichkeit:
Der Zugriff auf eine Konstruktionszeichnung ist ausschließlich Personen gestattet, die dem zugehörigen Entwicklungsprojekt angehören.

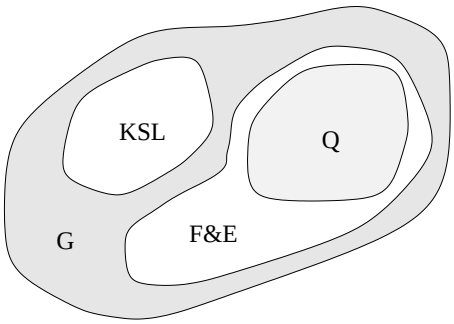
Sicherheitsanforderung 2, Wahrung der Integrität:
Änderungen an einer Konstruktionszeichnung können nur mit der Einwilligung des zuständigen Projektleiters vorgenommen werden.

Sicherheitsanforderung 3, Wahrung der Verbindlichkeit:
Zu jeder an einer Konstruktionszeichnung vorgenommenen Modifikation muß ersichtlich sein, wer die Modifikation vorgenommen hat und wer die Änderung autorisiert hat.

Sicherheitsanforderung 4, Sicherheit der Kommunikation:
Authentizität, Vertraulichkeit, Integrität und Verbindlichkeit müssen auch in einer verteilten Systemarchitektur unter der Nutzung unsicherer Kommunikationsverbindungen gewährleistet sein.

Der Konzern des Beispielszenariums besteht aus vielen Unternehmensbereichen, die wiederum in (eine Hierarchie von) Abteilungen und Projekten gegliedert sind. Konzernweit gültige Sicherheitsanforderungen können daher lediglich von allgemeiner Natur sein und die spezifischen Besonderheiten einzelner Bereiche, Abteilungen oder Projekte nicht berücksichtigen. Aus diesem Grund definieren untergeordnete Organisationseinheiten in der Regel eigene Sicherheitsanforderungen, die sowohl Konkretisierungen

der konzernweiten Sicherheitsanforderungen darstellen können als auch in besonderen Fällen von ihnen disjunkt sein können. Im Falle einer Konkretisierung entsteht hierdurch eine die Unternehmensstruktur reflektierende Hierarchie sich verschärfender Sicherheitsanforderungen, deren äußere Schale durch die konzernweiten Sicherheitsanforderungen gebildet wird und in deren Innersten die Sicherheitsanforderungen eines streng geheimen Projekts der Forschungsabteilung stehen, die beispielsweise ein eigenes, mandatorisches Zugriffskontrollschema oder ein mit dem Rest des Unternehmens unverträgliches Kryptosystem vorschreiben (Bild 2.2).

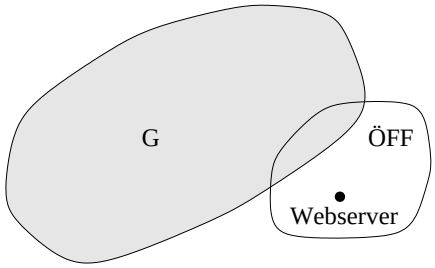


G: Domäne der konzernweiten Sicherheitsanforderungen
KSL: Domäne der Sicherheitsanforderungen des Unternehmensbereichs "Kundenspezifische Lösungen"
F&E: Domäne der Sicherheitsanforderungen des Unternehmensbereichs "Forschung & Entwicklung"
Q: Domäne der Sicherheitsanforderungen der Forschungsabteilung "Q"

Bild 2.2: Hierarchische Anordnung der Domänen von Sicherheitsanforderungen

Neben den hierarchisch angeordneten Domänen lassen sich auch Beispiele anführen, in denen eine teilweise Abschwächung der konzernweiten Sicherheitsanforderungen erfolgt; will beispielsweise die Abteilung für Öffentlichkeitsarbeit das Unternehmen im *Worldwide Web* präsentieren, so werden im Unterschied zu allen anderen Firmenservern und -dokumenten die entsprechenden Server und Seiten für die gesamte Welt zugänglich sein (Bild 2.3).

Teil der Unternehmenspolitik des Beispielkonzerns ist die Förderung der Zusammenarbeit seiner Abteilungen, indem Mitarbeiter regelmäßig für begrenzte Zeiträume ihren Arbeitsplatz in andere Abteilungen verlegen. So werden beispielsweise Experten der Forschungsabteilung im Anschluß an erfolgreiche Forschungsarbeiten für eine bestimmte Zeit in die Entwicklungsabteilung versetzt, um dort die schnelle Umsetzung der Forschungsergebnisse in ein Produkt zu unterstützen. Ebenfalls wechseln Verkaufsmanager frühzeitig vor Abschluß der Serienreife einer neuen Entwicklung zum Kennenlernen des Produkts in die Entwicklungsabteilungen, und Entwickler aus den Zentralllabors unterstützen in den länderspezifischen Entwicklungsabteilungen die örtlichen Mitarbeiter bei der Anpassung neuer Produkte an landesspezifische Merkmale wie Sprache oder technische Normen.



G: Domäne der konzernweiten Sicherheitsanforderungen
ÖFF: Domäne der Sicherheitsanforderungen des Unternehmensbereichs "Öffentlichkeitsarbeit"

Bild 2.3: Nichthierarchische Anordnung der Domänen von Sicherheitsanforderungen

Die zeitweise Versetzung von Mitarbeitern in andere Abteilungen und somit potentiell in Domänen mit anderen Sicherheitsanforderungen erfordert eigene, domänenübergreifende Regeln, unter denen eine Zusammenarbeit zwischen verschiedenen Domänen stattfinden kann. Die folgende Sicherheitsanforderung 5 zeigt ein Beispiel für eine entsprechende Sicherheitsanforderung, die einem versetzten Mitarbeiter sowohl den Zugriff auf seine gewohnte Arbeitsumgebung erlaubt als auch die Möglichkeit der Zusammenarbeit mit den Mitarbeitern der neuen Abteilung eröffnet.

Sicherheitsanforderung 5, Interaktionen zwischen Domänen:
Wird eine Person zeitlich begrenzt zu einer anderen Abteilung abgeordnet (in eine andere Sicherheitsdomäne versetzt), so behält er während dieser Zeit die Rechte aus seiner eigenen Domäne und erhält zusätzlich die Rechte eines Angehörigen der neuen Abteilung.

2.1.3 Sicherheitspolitiken

Eine Sicherheitspolitik ist eine systematische Transformation von Sicherheitsanforderungen in eine Strategie (Algorithmus), die die Erfüllung der Sicherheitsanforderungen zum Ziel hat. Sicherheitspolitiken finden sich in und um IT-Systemen in unterschiedlichen Ausprägungen. Häufig wird differenziert zwischen der Sicherheitspolitik eines Unternehmens und der informationstechnischen Sicherheitspolitik. Unter einer informationstechnischen Sicherheitspolitik wird dabei ein Regelwerk verstanden, welches den Umgang mit sicherheitsrelevanten Informationen *innerhalb* von IT-Systemen festlegt; eine unternehmensweite Sicherheitspolitik umfaßt darüberhinaus Regeln, die außerhalb des Einflußbereichs der Informationstechnik liegen, andererseits zur Umsetzung von Sicherheitsanforderungen aber notwendig sein können. Beispiele hierfür sind die Zugangskontrolle zu Räumen oder die Festlegung von Verantwortungsbereichen von Personen; solche Aspekte der Informationssicherheit werden häufig als ,,allgemeine

Schutzmaßnahmen" bezeichnet [Bun97, Opp97, SB92].

Die informationstechnische Realisierung der ersten Sicherheitsanforderung im Beispielszenarium ruht auf zwei Säulen: der Authentisierung und der Autorisierung. Die Sicherheitsanforderung legt fest, daß ein Zugriff auf ein bestimmtes Dokument ganz bestimmten Regeln unterliegt. Die Gesamtheit aller derartiger Zugriffsregeln bildet das *Autorisierungsschema* innerhalb einer Sicherheitspolitik, welches präzise festlegt, welche Personen wann welche Operationen auf welchen Dokumenten ausführen dürfen. Voraussetzung für ein funktionierendes Autorisierungsschema ist daher insbesondere die Authentisierung der handelnden Personen: eine zweifelsfreie, beweisbare Feststellung ihrer Identität.

Authentisierungs- und Autorisierungsschema sind wesentliche Elemente einer Sicherheitspolitik; sie sind ebenfalls zur Erfüllung der Sicherheitsanforderungen zwei bis vier erforderlich. Sicherheitsanforderung fünf weist ferner darauf hin, daß bedingt durch die Individualität der Sicherheitsanforderungen in verschiedenen Unternehmensbereichen unterschiedliche Sicherheitsdomänen mit individuellen Sicherheitspolitiken entstehen können, in denen voneinander abweichenden Autorisierungs- und Authentisierungsschemata gelten (vgl. Bilder 2.2 und 2.3). So werden Abteilungen mit geringen Sicherheitsanforderungen an die Vertraulichkeit eventuell auf Verschlüsselung der internen Kommunikation verzichten; andere Abteilungen werden hierzu unterschiedliche Kryptosysteme mit unterschiedlichen Kosten und Leistungen einsetzen. In der Konsequenz entstehen hierdurch innerhalb des verteilten IT-Systems des Beispielszenariums Sicherheitsdomänen mit eigenen Sicherheitspolitiken mit der Folge, daß die in der fünften Sicherheitsanforderung implizit geforderte Interoperabilität der Unternehmensbereiche ein erhebliches Problem darstellen kann.

Informationstechnische Sicherheitspolitiken werden ausführlich im folgenden Kapitel 3 besprochen. Die Interoperabilität von Sicherheitsdomänen ist zentrales Thema dieser Arbeit; hiermit werden sich die Kapitel 4, 5 und 6 befassen.

2.1.4 Sicherheitsmodelle

Um den hohen Qualitätsanforderungen bei der Entwicklung von Sicherheitspolitiken gerecht zu werden, werden als Technik zur ihrer präzisen Formulierung Sicherheitsmodelle und deren Spezifikation mittels geeigneter Spezifikationssprachen eingesetzt. Sicherheitsmodelle sind formale Repräsentationen von Sicherheitspolitiken; Ziel ist es dabei, durch Abstraktion, Konkretisierung, Präzisierung und den Einsatz formaler Kalküle eine Sicherheitspolitik mathematischen Verfahren zugänglich zu machen, die das Vertrauen in die Korrektheit der Sicherheitspolitik, beispielsweise hinsichtlich der Vollständigkeit und Widerspruchsfreiheit von Zugriffsregeln, erhöhen [McL94].

Die Komplexität von Sicherheitspolitiken erfordert dabei häufig die Beschränkung formaler Methoden auf besonders kritische Aspekte. Dazu zählen insbesondere das Autorisierungs- und das Authentisierungsschema; Beispiele zu Modellen von Sicher-

heitspolitiken und Authentisierungsverfahren finden sich (unter vielen anderen) in [BL76, Bib77, CW87, BN89, BAN89, San92b].

2.1.5 Sicherheitsarchitekturen und Infrastrukturen

Unter der Architektur eines Systems wird allgemein die Struktur eines Systems auf der Grundlage seiner Komponenten, ihrer Attribute, Funktionen und ihrer Beziehungen untereinander verstanden. In dieser Begriffsbildung wird die Sicherheitsarchitektur eines IT-Systems gebildet von all denjenigen Komponenten der Gesamtarchitektur, die in ihren Funktionen und ihrem Zusammenwirken die Sicherheitseigenschaften des Systems herstellen. Betrachtet man zusätzlich noch das administrative Umfeld einer Architektur, in dem Funktionen wie die Einrichtung von Benutzern oder die Generierung von Schlüsseln für kryptographische Verfahren wahrgenommen werden, so wird das Gesamtgebilde häufig als Sicherheitsinfrastruktur bezeichnet.

Die Sicherheitsarchitektur eines Systems bildet den Rahmen, innerhalb dessen seine Sicherheitspolitiken implementiert sind. Als Beispiel für eine Sicherheitsarchitektur sei die *Kerberos*-Architektur genannt, welche in einem verteilten System mit einem Client/Server - Kommunikationsparadigma ein Authentisierungs- und Autorisierungsschema implementiert [MNSS88]. Die Kerberos-Architektur besteht aus

- einem Authentisierungsserver, welcher das Authentisierungsschema implementiert
- einem Autorisierungsserver, welcher das Autorisierungsschema implementiert
- einem Authentisierungsprotokoll (einer Variante des Needham/Schroeder - Protokolls [NS78]), welches neben der eigentlichen Authentisierung gleichzeitig zukünftige Kommunikationspartner mit Schlüsseln für eine verschlüsselte Kommunikation ausstattet
- einem Kommunikationsprotokoll zur Wahrung von Integrität und Vertraulichkeit, basierend auf einem symmetrischen Kryptosystem (dem *Data Encryption Standard – DES* [Nat77, DH77]).

Das Zusammenspiel dieser Komponenten zeigt Bild 2.4. Will ein Klient einen Dienst eines beliebigen Servers in Anspruch nehmen, so wendet sich der Klient zum Zweck seiner Authentisierung zunächst an den Authentisierungsserver (1). Von diesem authentisiert, erhält der Klient hierüber eine Bestätigung (2), mit der er sich im folgenden gegenüber jedem beliebigen Server ausweisen kann. Als zweites wendet sich der Klient an den Autorisierungsserver und nennt diesem die Dienste, die er in Anspruch nehmen möchte (3). Der Autorisierungsserver prüft dies an Hand seines Autorisierungsschemas und der Authentifikation des Klienten und stellt entsprechende Berechtigungstickets aus

(4). Authentifikation und Tickets ermöglichen es nun dem Klienten, sich gegenüber allen Servern eines Kerberos-Systems auszuweisen und das Recht auf Inanspruchnahme eines Dienstes nachzuweisen (5).

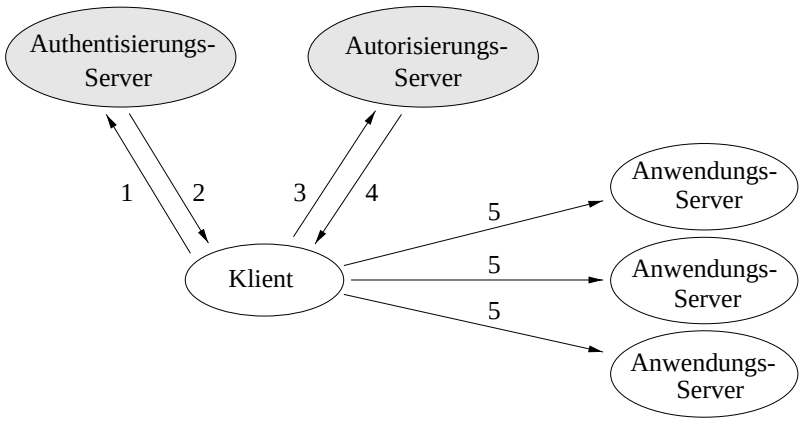


Bild 2.4: Die Kerberos-Architektur

Diese einfache Architektur, bestehend aus zwei Servern, zwei Protokollen und fünf Funktionen ist so allgemein, daß ihre Grundprinzipien in einer Vielzahl sicherer verteilter Systeme anzutreffen sind. Abkömmlinge sind beispielsweise die Sicherheitsarchitektur des *Distributed Computing Environments (DCE)* der *Open Systems Foundation (OSF)* [OSF92] oder das *Secure European System for Applications in a Multi-vendor Environment (SESAME)* der *European Computer Manufacturers Association (ECMA)*. Insbesondere lassen sich mit diesen Prinzipien bereits die zur Umsetzung der ersten beiden Sicherheitsanforderungen des Beispielszenariums notwendigen Authentisierungs- und Autorisierungsschema implementieren; für die Sicherheitsanforderungen drei und vier läßt sich die Architektur um ein drittes Protokoll ergänzen, welches unter Verwendung asymmetrischer Schlüssel und digitaler Signaturen eine informationstechnische Grundlage für die Verbindlichkeit herstellt.

Die Umsetzung der fünften Sicherheitsanforderung dagegen liegt außerhalb der Konzepte von Kerberos. Die zeitweise Versetzung eines Mitarbeiters in eine andere Sicherheitsdomäne mit eigenem Authentisierungs- und Autorisierungsschema erfordert die Formulierbarkeit zusätzlicher, domänenübergreifender Regeln, die beispielsweise die Akzeptanz von Authentifikationen und Autorisierungen fremder Domänen oder die Verwendung kryptographischer Schlüssel für Interaktionen zwischen den Domänen festlegen. Zwar kennt Kerberos den Begriff der „*realm*", welche den Zuständigkeitsbereich eines Authentisierungs- und Autorisierungsserverpaars beschreibt. Jedoch lassen sich Strategien für Interaktionen zwischen *realms* wie in Bild 2.5 gezeigt nicht allein in den Kerberos-Komponenten realisieren: in Bild 2.5 muß entweder der Server in der *realm KSL* die Authentifikation des Klienten durch den (fremden) Authentisierungsserver der *realm Q* akzeptieren, oder der Klient muß sich explizit gegenüber dem Authenti-

sierungsserver der *realm KSL* neu authentisieren. Im ersten Fall liegt also die Strategie für Interaktionen zwischen den *realms* in den Anwendungsservern, im zweiten Fall liegt sie bei den Klienten.

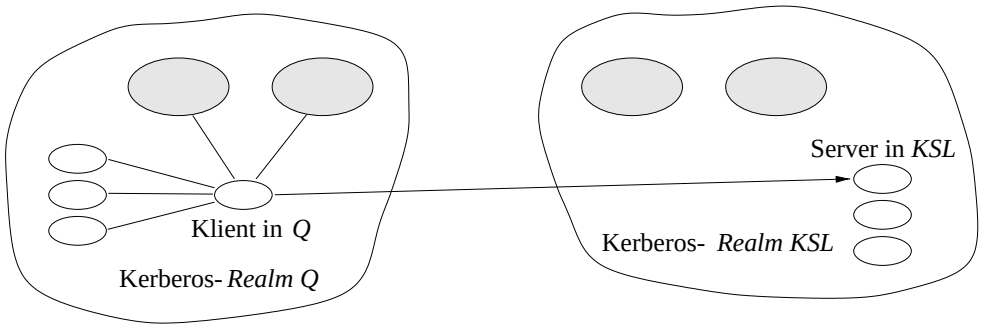


Bild 2.5: Interaktion zwischen Domänen mit kerberosartiger Sicherheitsarchitektur

2.1.6 Sicherheitsmechanismen

Sicherheitsmechanismen sind die Bausteine zur Implementierung von Sicherheitspolitiken, die im Kontext einer Sicherheitsarchitektur konkrete Sicherheitseigenschaften herstellen. Hierzu zählen Authentisierungsprotokolle, Zugriffssteuerungsmechanismen in Betriebssystemen, Verschlüsselungsmechanismen für die vertrauliche Kommunikation in öffentlichen Netzen, Mechanismen zur Wahrung von Integrität und Authentizität von Daten und Mechanismen zur Schlüsselverteilung [Hor85, Sch93, Zim95].

2.2 Safety und Security

Der Begriff „Sicherheit" wird im Zusammenhang mit informationstechnischen Systemen im deutschen Sprachraum zuweilen unterschiedlich interpretiert. Eine Ursache liegt in der Ermangelung eines allgemein akzeptierten präzisen deutschen Äquivalents der beiden englischen Begriffe „*security*" und „*safety*": für beide Begriffe wird im Deutschen das Wort „Sicherheit" verwendet. Eine scharfe Abgrenzung zwischen diesen beiden Begriffen ist kaum zu treffen; selbst die *Encyclopaedia Britannica* beschreibt den einen Begriff als ein mögliches Synonym des anderen. Dennoch soll für den Kontext dieser Arbeit der Versuch einer sinnvollen Abgrenzung gemacht werden und auf nach Meinung des Verfassers bestehende charakteristische Unterschiede hingewiesen werden.

Safety bezeichnet im allgemeinen Sprachgebrauch einen Zustand der (körperlichen) Gefahrlosigkeit; in der Informatik werden unter diesem Begriff Eigenschaften informationstechnischer Systeme betrachtet, welche Schäden durch *unbeabsichtigte* Fehlfunktionen

eines Systems vermeiden oder vermindern. Safety spielt überall dort eine wesentliche Rolle, wo Fehlfunktionen von IT-Systemen zu gesundheits- oder lebensbedrohenden Situationen führen können, beispielsweise in der Fahrzeug- oder Anlagensteuerung. Typische Bedrohungen der *Safety* werden hervorgerufen durch

- technisches Versagen
- menschliches Versagen
- höhere Gewalt.

Zum technischen Versagen werden Fehlfunktionen der Hardware (fehlerhafte elektronische Bauelemente, Alterung von Datenträgern, Strahlungsinterferenzen) und Einflüsse der Umwelt (Stromausfall, mechanische Belastungen der Hardware, Feuer, Schmutz) gezählt. Als charakteristisches Beispiel mag hier der wiederholte Ausfall von zwei der fünf Bordcomputer der Raumfähre Columbia vor dem Wiedereintritt in die Erdatmosphäre und während des Landeanfluges im Jahre 1983 dienen, der durch ein winziges Kügelchen Lötzinn ausgelöst wurde, welches unter dem Einfluß einer Gravitation von 20g durch die Computer irrte [Neu95, SEN84, SEN89]. Nur die Fünffachauslegung der Bordcomputer hielt hier den Schaden durch diese Fehlfunktion in akzeptierbaren Grenzen.

Menschliches Versagen entsteht typischerweise durch mangelnde Ausbildung oder Fahrlässigkeit und äußert sich in Spezifikationsfehlern, Programmfehlern, fehlerhaften Installationen oder Bedienungsfehlern. Das bislang wohl prominenteste Beispiel mit dem größten finanziellen Schaden ist der fehlgeschlagene Start der ersten Ariane-5 - Rakete im Juni 1996, der verursacht wurde durch die unvollständige Spezifikation einer Softwarekomponente zur Kontrolle der Bahnneigung. Diese Komponente war ursprünglich für die Ariane-4 entwickelt worden, bei der die Abweichung von der vertikalen Bahn auf Grund ihrer erheblichen schwächeren Antriebskraft erst sehr viel später nach dem Start erfolgte. Die Softwarekomponente wurde dann für die Ariane-5 übernommen, ohne daß zunächst die Konsequenzen hieraus für die Verantwortlichen klar waren [LLF+96]. Zu höherer Gewalt schließlich werden Ereignisse wie Blitzschlag, Brand oder Erdbeben gerechnet.

In Abgrenzung dazu befaßt sich die Security mit der Vermeidung von Schäden, die durch *beabsichtigte und mit Intelligenz betriebene Angriffe* auf informationstechnische Systeme entstehen. Ziel solcher Angriffe ist es im Allgemeinen, auf illegale Weise Vorteile zu erzielen oder Schäden anzurichten (Wirtschaftsspionage, Betrug, Erpressung durch Androhung der Veröffentlichung von Firmendaten oder medizinischen Informationen über Personen, Zerstörung oder Modifikation von Daten oder Programmen, Verhinderung der Nutzung von IT-Systemen). Die dabei eingesetzten Methoden zum Ausspähen oder Manipulieren von Informationen sind mannigfaltig und teilweise äußerst phantasievoll. Sie umfassen das Vortäuschen fremder Identitäten, das Beobachten und die Manipulation von Kommunikationsverbindungen, den bewußten Einbau verborgener Funktionen in Softwarepakete (Hintertüren), die nachträgliche Modifikation von

Software zum Einbau verborgener Funktionen (trojanische Pferde, Computerviren) bis hin zum Auffangen und der Interpretation elektromagnetischer Abstrahlungen von Bildschirmen.

Diese Arbeit setzt sich in erster Linie mit der Interpretation des Begriffs Sicherheit im Sinne von Security auseinander und macht dieses in der Regel durch Verwendung des Begriffs „Informationssicherheit" deutlich.

2.3 Daten und Informationen

Grundsätzlich werden in IT-Systemen zunächst einmal Daten verarbeitet. Daten werden dann zu einer Information, wenn Wissen zu ihrer Interpretation vorhanden ist: Daten können Programme, Briefe, Krankenbefunde oder Telefongespräche repräsentieren. Dieser grundsätzliche Unterschied soll zunächst an zwei Beispielen veranschaulicht werden; anschließend werden dann die Konsequenzen für die Informationssicherheit – beispielsweise in bezug auf die Wahrung von Vertraulichkeit oder Integrität – betrachtet.

In digitalen Telefonvermittlungsnetzen bestehen normale Punkt-zu-Punkt - Gespräche aus drei Phasen: dem Verbindungsaufbau, der Gesprächsführung und dem Verbindungsabbau. Die in der Aufbauphase an den Vermittlungsrechner übermittelten digitalisierten Wahltöne stellen für den Vermittlungsrechner eine Information dar: mit Hilfe der Wahltöne ist er in der Lage, die gewählte Telefonnummer zu erkennen und so die gewünschte Verbindung herzustellen. Dagegen stellt die digitalisierte Sprache während des Gesprächs für den Vermittlungsrechner keine Information, sondern lediglich Daten dar; selbst wenn diese Daten zufällig digitalisierte Wahltöne enthalten sollten (beispielsweise zur Übermittlung von PINs beim Telefon-Banking), interpretiert sie der Vermittlungsrechner nicht mehr. In dieser Gesprächsphase stellt einzig das Auflegen des Hörers oder das Drücken der Flash-Taste eine Information für den Vermittlungsrechner dar: die Information, daß das Gespräch beendet ist oder ein Dialog mit dem Vermittlungsrechner gewünscht wird.

Am Beispiel des ISO/OSI - Protokollmodells [Int84, CCI88] läßt sich ferner zeigen, daß dieselbe Zeichenfolge für unterschiedliche Komponenten eines IT-Systems einerseits Daten und andererseits Informationen darstellen kann: Im ISO/OSI - Modell besitzt jede der sieben Ebenen der Protokollhierarchie ihre eigene, klar definierte und abgegrenzte Aufgabe. Um diese zu erfüllen, versieht jede Ebene beim Versenden einer Botschaft diese mit einem eigenen, ebenenspezifischen Vorspann. Für jede Ebene der Empfängerseite stellt der von der korrespondierenden Ebene der Absenderseite hinzugefügte Vorspann eine Information dar. Der Rest der Botschaft, einschließlich der Vorspanne aller anderen Ebenen, ist dagegen nicht interpretierbar – es sind lediglich Daten, die uninterpretiert weitergereicht und erst auf einer anderen Ebene zu Informationen werden.

Aus Sicht der Informationssicherheit sind Daten keineswegs gleichzusetzen mit den

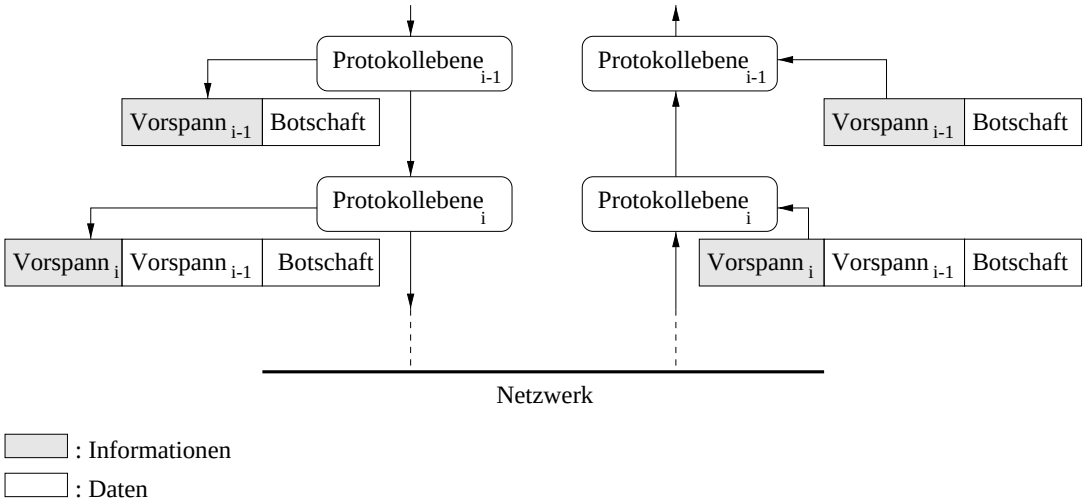


Bild 2.6: Aufbau und Abbau eines ISO/OSI Datenpakets durch die Protokollebenen

verbundenen Informationen. Bei einem Transport von Daten über öffentliche Rechnernetze ist es durch die physische Verletzlichkeit der Übertragungsmedien im allgemeinen unmöglich, ihre Vertraulichkeit zu gewährleisten. Dagegen ist es möglich, Daten durch ihre Verschlüsselung in eine Darstellung zu transformieren, die Unbefugten einen Informationsgewinn hieraus sehr schwer macht. Gleichfalls ist es im allgemeinen unmöglich, eine Modifikation von Daten während ihres Transports über ein Rechnernetz zu verhindern; dagegen lassen sich durch Verwendung digitaler Signaturen Modifikationen der Daten erkennbar (und durch Hinzufügen von Redundanz eventuell korrigierbar) machen; hierdurch kann sichergestellt werden, daß Informationen nur aus Daten abgeleitet werden, deren Integrität gewährleistet ist.

2.4 Verteilte Systeme

Das konzernweite IT-System des Beispielszenariums besteht aus einer Menge autonomer und mittels eines Kommunikationsmediums verbundener Einzelsysteme; solche Systemarchitekturen werden allgemein als vernetzte Systeme bezeichnet. An vernetzte Systeme werden nur geringe Anforderungen an die Integration der Einzelsysteme gestellt; es wird lediglich erwartet, daß sie eine elementare Fähigkeit zur Kommunikation über standardisierte Protokolle (z.B. ISO/OSI) besitzen. Werden dagegen die Einzelsysteme mittels verteilter Software so integriert, daß sie gemeinsam eine verteilte Applikation realisieren, so spricht man von einem verteilten System [CDK94, Tan95]. Verschiedentlich wird darüberhinaus gefordert, daß in verteilten Systemen bestimmte Aspekte der Verteilung kaschiert oder in ihren negativen Auswirkungen abgemildert werden. Ortstransparenz etwa beschreibt die Eigenschaft verteilter Systeme, daß ober-

halb einer bestimmten Betrachtungsebene die physische Verteilung eines Systems dem Betrachter verborgen bleibt und der Eindruck eines einzigen, homogenen Systems vermittelt wird. Weitere derartige Eigenschaften verteilter Systeme sind die Ausfalltransparenz, die Migrationstransparenz oder die Skalierungstransparenz.

Diese Begriffsbestimmung umfaßt ein breites Spektrum an Hardware- und Softwarearchitekturen, angefangen bei busgekoppelten Multiprozessorrechnern bis hin zu autonomen Einzelsystemen, die mittels Weitverkehrsnetzen miteinander kommunizieren. Das konzernweite IT-System des Szenariums ist beispielsweise grundsätzlich ein vernetztes System; in einzelnen Unternehmensabteilungen kann dann hierauf ein verteiltes System aufgebaut werden, in welchem ein verteiltes Dateisystem den ortstransparenten Zugriff auf Dateien ermöglicht oder rechenintensive Simulationsprogramme durch eine Lastverteilungssoftware auf leistungsfähige Vektor- oder Parallelrechner verlagert werden können.

Ein weiteres Beispiel für vernetzte Systeme sind weltumspannende Rechnernetze, die eine Vielzahl autonomer Systeme aus Wirtschaft, Verwaltung und Forschung miteinander verbinden. In solchen Netzen finden sich typischerweise elementare Standarddienste wie Dateitransfer oder elektronische Post; die durch die ATM-Technik heute erzielbare beachtliche Leistungsfähigkeit der Weitverkehrsnetze läßt inzwischen auch Applikationen zu, die über große Entfernungen verteilt sind; Beispiele hierzu sind das Szenarium des verteilten Konzerns, Informationsverbunde von Bibliotheken und Anwendungen aus dem weiten Feld des elektronischen Handels [BKAL97].

Vernetzte und verteilte Systeme haben einen erheblichen Wandel in der Informationssicherheit bewirkt. Ihre charakteristischen Eigenschaften, wie die globale gemeinsame Nutzbarkeit von Ressourcen, ihre Offenheit oder die Autonomie der Einzelsysteme, bringen eine Vielzahl neuer Sicherheitsaspekte mit sich, die in isolierten, geschlossenen Einzelsystemen nicht vorhanden sind. Die physische Verteiltheit und damit die zentrale Rolle der Kommunikation führten weiterhin dazu, daß die ursprünglich getrennten Themenbereiche der *Computersicherheit* und der *Kommunikationssicherheit* miteinander verschmolzen. Diese Verschmelzung führte zu einer deutlichen Erhöhung der Komplexität der Fragestellungen, mit denen sich die Computersicherheit und die Kommunikationssicherheit bisher getrennt auseinandersetzen konnte.

Zur Veranschaulichung soll hier die Nutzung einer einfachen Dienstleistung diskutiert werden, die in einem verteilten IT-System von einem Dienstanbieter (Server) angeboten und durch einen menschlichen Benutzer in Anspruch genommen wird. Dabei wird insbesondere die Komplexität der Lösungen deutlich, die sich allein bei der hierzu notwendigen gegenseitigen Authentisierung von Mensch und Server in dem Szenarium des Beispielkonzerns ergibt.

Beispiel: Authentisierung in verteilten Systemen

In monolithischen (d.h. nichtverteilten) IT-Systemen ist die Identifikation und Authentisierung aller menschlichen Benutzer, aller Server, Ressourcen etc. Bestandteil der lokalen Systemsoftware. Wird ein IT-System von Personen administriert, zu denen die Benutzer ein gewisses Vertrauen besitzen (z.B. dadurch, daß sie derselben Organisationseinheit angehören), so entsteht hierdurch auch Vertrauen in die sorgfältige Administration des IT-Systems und damit seiner Authentisierungskomponenten. Arbeiten diese korrekt, so haben ausschließlich die einer wohlbekannten Benutzergruppe angehörenden und durch eine vertrauenswürdige Systemkomponente authentisierten Menschen Zugang zu diesem System: es ist hinsichtlich der Benutzergruppe geschlossen.

Anders in offenen Systemen, in denen es prinzipiell keine feste Benutzergruppe und keinen Punkt gemeinsamen Vertrauens gibt. Das verteilte IT-System des Beispielkonzerns ist ein offenes System, in dessen Struktur sich der lockere Verbund der weitgehend autonomen Unternehmensbereiche wiederspiegelt. Insbesondere sind hierdurch die differenzierten Sicherheitsanforderungen der unterschiedlichen Unternehmensbereiche umgesetzt, welches bereits in einem früheren Beispiel zu voneinander unabhängigen Sicherheitsdomänen führten (vgl. Bilder 2.2 und 2.3).

Eine der Konsequenzen aus den differenzierten Sicherheitsanforderungen der verschiedenen Unternehmensbereiche ist die Qualität der Benutzerauthentisierung; Bereiche mit weniger strengen Anforderungen werden hier einfache Paßwortverfahren einsetzen, während leistungsfähigere Authentisierungsverfahren innerhalb einer Domäne mit strengen Anforderungen auf Chipkarten basieren. Innerhalb des Gesamtsystems des Konzerns existiert somit keine zentrale Instanz, die sämtliche Benutzer kennt und authentisiert: das Gesamtsystem ist hinsichtlich seiner Benutzergruppe offen.

Betrachtet man in diesem Szenarium die gegenseitige Authentisierung von Mensch und Server, so sind hieran beteiligt

- der Mensch,
- die Authentisierungsinstanz des Menschen, welche den Menschen als Benutzer des Systems kennt und das Verfahren zu seiner Authentisierung implementiert,
- der Server sowie
- die Authentisierungsinstanz des Servers,

die in unterschiedlichen physischen Konstellationen über die Sicherheitsdomänen des IT-Systems verteilt sein können. Ferner wollen wir annehmen, daß die Kommunikation zwischen Mensch[2] und Server zur Wahrung der Vertraulichkeit und Integrität verschlüsselt erfolgt; der dazu notwendige Austausch von Schlüsseln erfolgt wie üblich

[2] respektive des Prozesses, der ihn innerhalb des IT-Systems repräsentiert

im Rahmen eines Authentisierungsprotokolls. Um die Anzahl der Varianten im Beispiel nicht zu groß werden zu lassen, soll ferner angenommen werden, daß der Server nicht mobil ist und mit seiner Authentisierungsinstanz in derselben Sicherheitsdomäne liegt. Benutzer dagegen sind mobil; sie wechseln zeitweise und zeitlich befristet die Abteilungen und damit auch die Sicherheitsdomäne, in der sie sich physisch aufhalten.

Durch die Mobilität der Benutzer ergeben sich nun Varianten bei der Authentisierung, die sich jeweils durch den physischen Aufenthaltsort eines Benutzers und des in Anspruch genommenen Dienst in bezug auf die beteiligten Sicherheitsdomänen unterscheiden.

1. In der ersten Konstellation befindet sich der Benutzer zunächst an seinem gewohnten Arbeitsplatz innerhalb seiner eigenen Abteilung Q und nutzt einen Server innerhalb seiner Heimatdomäne.

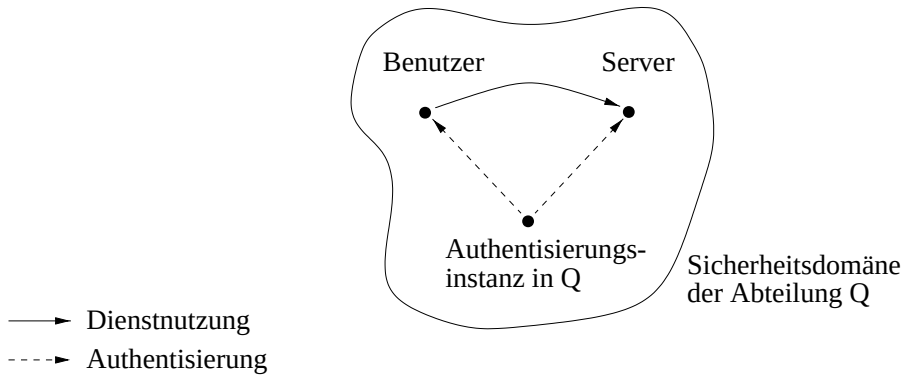


Bild 2.7: Authentisierung und Dienstnutzung bei nicht-versetztem Mitarbeiter

Die Authentisierung sowohl des Benutzers als auch des Servers erfolgt durch eine domänenlokale Authentisierungsinstanz A_Q nach dem vorgeschriebenen Schema der Sicherheitsdomäne Q. Sowohl der Benutzer als auch der Server werden von derselben Instanz authentisiert; da beide dieser Instanz auf Grund ihrer Lokalität zur eigenen Sicherheitsdomäne vertrauen, verlassen sie sich auf die Sicherheit des Authentisierungsschemas und die Wohlgeformtheit der ausgegebenen Schlüssel. Beide Parteien sind ferner in der Lage, die erhaltenen Schlüssel zu benutzen, da sowohl Schlüsselverteilungsinstanz als auch Kommunikationspartner in derselben Sicherheitsdomäne liegen und dasselbe Kryptosystem verwenden.

Die Verhältnisse in dieser Konstellation sind die eines monolithischen IT-Systems; innerhalb der Sicherheitsdomäne existiert eine geschlossene Benutzergruppe mit einer lokalen vertrauenswürdigen Authentisierungsinstanz.

2. Die zweite Konstellation reflektiert die Situation, daß ein Angehöriger der Abteilung Q zeitweise in eine anderen Abteilung P versetzt ist; während dieser Zeit

soll der Mitarbeiter neben den Systemen der Abteilung P auch die Server seiner Heimatdomäne weiternutzen können.

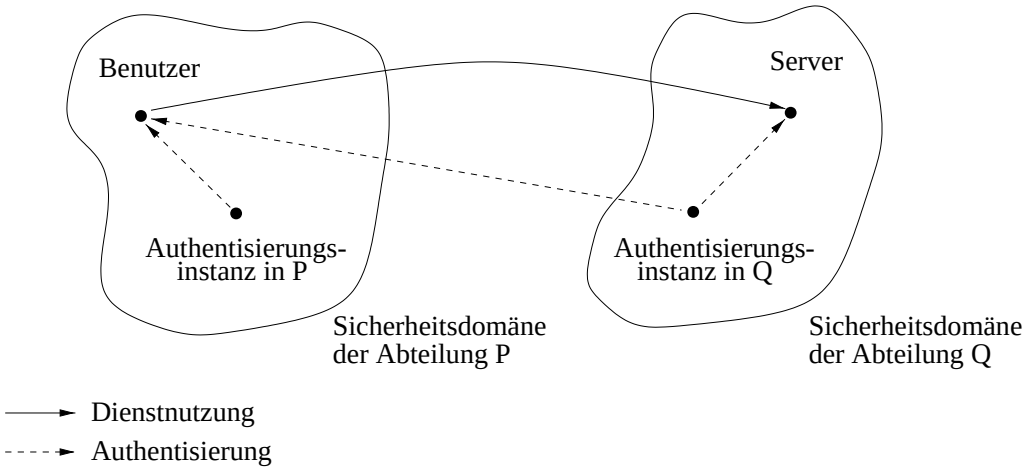


Bild 2.8: Authentisierung und Dienstnutzung bei versetztem Mitarbeiter

In der ersten Variante dieser Konstellation erfolgt während des Versetzungszeitraums die Authentisierung des Mitarbeiters durch die Authentisierungsinstanz der Sicherheitsdomäne P; diese Variante hat die folgenden Merkmale:

- Der Mitarbeiter muß ein fremdes Authentisierungsschema benutzen, beispielsweise ein paßwortbasiertes Verfahren statt seiner gewohnten Chipkarte: insbesondere muß er dabei der fremden Authentisierungsinstanz vertrauen, daß das Authentisierungsschema sicher ist und die zugrundeliegenden Geheimnisse bewahrt.
- Für das Arbeiten mit dem Server seiner Heimatdomäne muß der Server in Q die Authentisierung durch eine fremde Authentisierungsinstanz akzeptieren; ein möglicherweise schwächeres Authentisierungsschema in P stellt dabei für die Sicherheitsdomäne Q ein Risiko dar.
- Die Kryptosysteme in den Sicherheitsdomänen P und Q können durch unterschiedliche Sicherheitsanforderungen inkompatibel sein (symmetrisches/asymmetrisches Verschlüsselungsverfahren, andere Hash-Algorithmen, abweichende Schlüsselllängen); die Authentisierungsinstanz in P muß dann einen geeigneten Schlüssel für die Kommunikation mit dem Server in Q liefern können.

3. Die dritte Konstellation reflektiert gleichfalls die Situation aus Bild 2.8, jedoch erfolgt hier die Authentisierung des Mitarbeiters auch während seiner Versetzung durch die Authentisierungsinstanz seiner Heimatdomäne Q; diese Variante hat die folgenden Merkmale:

- Innerhalb von P muß die notwendige Infrastruktur für das Authentisierungsschema von Q vorhanden sein; verwendet das Authentisierungsschema in Q persönliche Chipkarten, so müssen in P die entsprechenden Lesegeräte vorhanden sein.
- Für das Arbeiten mit den Systemen aus der Domäne P müssen die Server in P die Authentifikation durch die Authentisierungsinstanz von Q akzeptieren; ein eventuell schwächeres Schema in Q stellt dann für P ein Sicherheitsrisiko dar.
- Die Kryptosysteme in P und Q können inkompatibel sein (s.o.); die Authentisierungsinstanz in Q muß dann einen geeigneten Schlüssel für die Kommunikation mit den Servern in P liefern.
- Bei paßwortbasiertem Authentisierungsverfahren in Q wird innerhalb der Domäne P das Paßwort des versetzten Mitarbeiters offenbart.

Aus diesen Betrachtungen wird offensichtlich, daß in verteilten Systemen mit mehreren nebeneinander existierenden Sicherheitsdomänen allein die Authentisierung als Vorbereitung einer Kommunikation erheblich komplexer ist als in Systemen mit einer zentralen, vertrauenswürdigen Authentisierungsinstanz. Betrachtet man darüberhinaus die nachfolgende Kommunikation selbst, so ergibt sich durch die physische Verteiltheit des IT-Systems eine weitere Komplexitätssteigerung, da zur Wahrung der Vertraulichkeit und Integrität bei der Kommunikation kryptographische Verfahren notwendig werden. Diese Komplexität begründet die bereits in Abschnitt 2.1.5 skizzierten Sicherheitsinfrastrukturen, in denen Authentisierungsschema, Autorisierungsschema, Schlüsselmanagement und Kryptosysteme in ihren Funktionen und ihrem Zusammenwirken betrachtet werden.

Zur Realisierung des Beispiels kann auch die in Abschnitt 2.1.5 erwähnte Kerberos-Architektur nur eine partielle Lösung anbieten. Eine Sicherheitsdomäne in unserem Beispielszenarium ist in Kerberos durch eine „Kerberos-*Realm*" ausdrückbar, deren Authentisierungsserver das Authentisierungsschema und deren Autorisierungsserver das Autorisierungsschema implementieren. Ungelöst bleibt hierbei allerdings das Problem der Formulierung eines Autorisierungsschemas für Interaktionen zwischen verschiedenen Sicherheitsdomänen; Kerberos kann hierfür kein Konzept anbieten.

2.5 Zusammenfassung

Kapitel 2 motivierte und veranschaulichte grundlegende Ziele und Begriffe der Informationssicherheit. Als Beispielszenarium diente dazu das konzernweite IT-Systems eines dezentralisierten Unternehmens mit geographisch verteilten Niederlassungen. An diesem Szenarium wurden informationstechnisch bedingte Risiken diskutiert, hieraus dann exemplarisch Sicherheitsanforderungen abgeleitet und die Rolle von Sicherheitspolitiken und -modellen bei der Umsetzung dieser Sicherheitsanforderungen skizziert. Die

strukturelle Gliederung unseres Beispielunternehmens in weitgehend autonome Unternehmensbereiche fand ihren Eingang in die verteilte Architektur des unternehmensweiten IT-Systems; die Sicherheitsarchitektur des IT-Systems gliederte sich in Sicherheitsdomänen, die die individuellen Sicherheitsanforderungen einzelner Unternehmensbereiche reflektieren. Am Beispiel der Nutzung eines Servers durch einen mobilen Benutzer wurde schließlich gezeigt, welche Komplexität allein ein einziger Aspekt der Informationssicherheit, die Authentisierung, in verteilten Multipolitikenszenarien erhält.

Die lokalen Sicherheitsanforderungen in den einzelnen Sicherheitsdomänen und die hieraus resultierenden individuellen Authentisierungs- und Autorisierungschemata führen innerhalb des Gesamtsystems zur Bildung autonomer Inseln, in denen geschlossene Benutzergruppen ausschließlich Handlungen vornehmen können, die mit der domänenspezifischen Sicherheitspolitik konform gehen. Dies garantiert dann zwar ein hohes Qualitätsniveau der Informationssicherheit innerhalb der einzelnen Sicherheitsdomänen; ungelöst dagegen bleibt das Problem der Sicherheit von Interaktionen über Domänengrenzen hinweg. Die im Beispiel benutzte Kerberos-Architektur etwa kann hierfür kein Konzept anbieten.

Dies stellt in der heutigen IT-Landschaft ein gravierendes Defizit dar, da hier ein Konflikt zwischen der Interoperabilität in verteilten Systemen und ihrer Sicherheit offenbar wird. Gleichzeitig hat das Beispiel der domänenübergreifenden Authentisierung deutlich gemacht, daß die durch offene verteilte Systeme eingeführte Komplexität Strategien erfordert, deren Korrektheit, Vollständigkeit und Widerspruchsfreiheit entscheidend ist für die Beherrschbarkeit und Qualität der Informationssicherheit des gesamten verteilten IT-Systems.

Ein qualitativ hohes Niveau der Informationssicherheit in verteilten Systemen unter Wahrung ihrer Interoperabilität erfordert somit eine Lösung des Problems der domänenübergreifenden Interaktion, ohne damit gleichzeitig das durch die Sicherheitspolitiken erreichte Niveau der Informationssicherheit innerhalb der Domänen zu unterminieren. Vielmehr muß es Ziel sein, Interaktionen *zwischen* Politikdomänen mit einer analogen Qualität der Informationssicherheit auszustatten, wie sie das Konzept der Sicherheitspolitiken *innerhalb* einer Domäne erzielt.

Mit diesem Problem setzt sich die vorliegende Arbeit auseinander. Der beschrittene Weg beruht auf der Idee, Autorisierungsschemata für Interaktionen zwischen Sicherheitspolitikdomänen in übergeordneten Metapolitiken zu formulieren. In gleicher Weise, in der Sicherheitspolitiken ein Basiskonzept für sichere Interaktionen innerhalb einer Sicherheitsdomäne darstellen, entwickelt die vorliegende Arbeit das Konzept der Metapolitiken als Basiskonzept für sichere Interaktionen zwischen Sicherheitsdomänen. Die Grundlage hierzu wird durch eine formale Betrachtung von Metapolitiken mit dem Ziel der Analyse ihrer Sicherheitseigenschaften gelegt. Basierend auf dem 1976 von Harrison, Ruzzo und Ullman entwickelten HRU-Kalkül werden Aussagen über die grundsätzlichen Möglichkeiten und Grenzen der Analysierbarkeit von Metapolitiken aufgestellt und bewiesen. Basierend auf diesem Fundament wird eine Spezifikationssprache vorge-

stellt, aus der dann automatisch der zur Laufzeit erfoderliche Code einer Metapolitik generiert wird. Eine Sicherheitsarchitektur zur Implementierung so formulierter Metapolitiken wird schließlich das Gesamtkonzept abrunden. Mit den erzielten Ergebnissen rückt dann der Perspektive weltweiter offener Systeme näher, in denen eine Vielzahl von Sicherheitspolitiken die individuellen Sicherheitsinteressen der Teilnehmer bewahren und eine sichere Interaktion zwischen den Politikdomänen durch Metapolitiken gewährleistet wird.

3 Sicherheitspolitiken

Dieses Kapitel legt die Grundlagen für die Entwicklung eines Metapolitikkonzeptes und befaßt sich dazu mit Wesen und Aufgaben von Sicherheitspolitiken. Es beschreibt ihre Rolle als methodischen Ansatz bei Beschreibung und Implementierung sicherer Systeme und stellt ein durchgängiges Konzept zur Formulierung, Modellierung, Analyse und Implementierung von Sicherheitspolitiken vor. Ausgehend von einer Betrachtung bekannter Methoden und Kalküle zur Modellierung und Analyse von Sicherheitspolitiken wird eine Spezifikationssprache entwickelt, die den Paradigmenwechsel von Modell zu Implementierung vereinfacht. Schließlich wird ein Konzept zur Implementierung von Sicherheitspolitiken entwickelt, welches anders als klassische listenbasierte Implementierungskonzepte eine algorithmische Formulierung von Sicherheitspolitiken ermöglicht und hierdurch eine erheblich höhere Ausdruckskraft erzielt.

3.1 Die Rolle der Sicherheitspolitiken

Seit Beginn einer wissenschaftlichen Auseinandersetzung mit der Sicherheit informationstechnischer Systeme spielen Sicherheitspolitiken bei der Beschreibung und Implementierung von Sicherheitseigenschaften eine grundsätzliche Rolle. Die auch unter dem Namen *Orange Book* bekannten *Trusted Computer System Evaluation Criteria* (TCSEC)[DoD83] definierten bereits im Jahre 1983 eine Sicherheitspolitik als „*Sammlung von Gesetzen, Regeln und Praktiken, die den Umgang, den Schutz und die Verbreitung von sensitiven Informationen in einer Organisation festlegen*". Nach dieser zunächst sehr allgemeinen Formulierung bestimmt eine Sicherheitspolitik also die strategischen Sicherheitsziele einer Organisation und legt die Wege fest, auf denen diese Ziele erreicht werden sollen.

Sicherheitspolitiken finden sich in und um IT-Systemen in unterschiedlichen Ausprägungen. Die geläufige Klassifikation unterscheidet drei verschiedene Gegenstandsbereiche, für die Sicherheitspolitiken existieren, ausgedrückt durch die englischen Begriffe *corporate security policy*, *system security policy* und *technical security policy*.

Als *corporate security policy* wird bezeichnet

> die Menge aller Vorschriften, Regeln und Praktiken, welche die Verwaltung, den

Schutz und die Verteilung aller Werte einschließlich der sensitiven Informationen innerhalb einer Organisation festlegen.

Als *system security policy* wird bezeichnet
die Menge aller Vorschriften, Regeln und Praktiken, welche die Verwaltung, den Schutz und die Verteilung aller Werte einschließlich der sensitiven Informationen innerhalb eines konkreten Systems festlegen.

Als *technical security policy* wird bezeichnet
die Menge aller Vorschriften, Regeln und Praktiken, welche die Verarbeitung sensitiver Informationen und die Nutzung von Ressourcen durch die Hardware und Software eines Systems oder Produkts festlegen.

Die Definition der *corporate security policy* umfaßt insbesondere Maßnahmen, die außerhalb des Einflußbereichs der Informationstechnik liegen, andererseits zur Umsetzung von Sicherheitsanforderungen aber notwendig sein können. Beispiele hierfür sind die Zugangskontrolle zu Räumen oder die Festlegung von Verantwortungsbereichen von Personen; solche Aspekte der Informationssicherheit werden zumeist als „allgemeine Schutzmaßnahmen" bezeichnet und beispielsweise in [Bun97, Opp97, SB92] besprochen. In der vorliegenden Arbeit werden ausschließlich die informationstechnischen Ausprägungen der Sicherheitspolitiken betrachtet.

In ihrem ursprünglichen Kontext waren Sicherheitspolitiken stark beeinflußt von militärischen Zielen; dies erlaubte, mit Konzepten für eng begrenzte Anwendungen in eng begrenzten Szenarien zu arbeiten. Heute spielen Sicherheitsaspekte in einer Vielzahl nichtmilitärischer, sehr viel weniger homogenen und geschlossenen Umgebungen eine Rolle, und die in der Kapitel 2 genannten verschiedenen Aspekte der Sicherheit können in diesen Umgebungen durchaus unterschiedliche Relevanz besitzen. Im skizzierten Szenarium des verteilten Unternehmens spielt die Wahrung der Vertraulichkeit eine wesentliche Rolle, während beispielsweise im Rechtswesen die Integrität und die Verbindlichkeit besondere Relevanz besitzen. Sicherheitspolitiken besitzen daher einen grundsätzlich individuellen Charakter: jede Institution, jede Applikation oder jedes institutionsübergreifende Kooperationsprojekt besitzt eigene Sicherheitsanforderungen, die eine eigene Sicherheitspolitik mit einem wohldefinierten Wirkungskreis definieren. Sicherheitspolitiken und Anwendungssoftware bilden dabei im Idealfall getrennte Einheiten [Küh95a]; diese Trennung ist dann sogar unabdingbar, wenn die Komplexität einer Sicherheitspolitik in Verbindung mit der Notwendigkeit des Einsatzes mathematischer Analyseverfahren eine Beschränkung des Gegenstandes der Analyse auf das minimal Notwendige erfordert [KW97]. Zudem erlaubt die Trennung die Anwendung erprobter softwaretechnischer Entwicklungskonzepte bei der Politikentwicklung und erschließt so eine Fülle von Methoden und Werkzeugen, die die Effizienz und Ökonomie der Politikentwicklung befördern.

Sicherheitspolitiken sind Bestandteil der Sicherheitsarchitektur eines IT-Systems. Sie benötigen zu ihrer Durchsetzung sowohl konkrete Eigenschaften der Sicherheitsarchi-

tektur als auch konkrete Sicherheitsmechanismen. Sicherheitspolitiken zur Autorisierung von Zugriffsoperationen beispielsweise erwarten von der Sicherheitsarchitektur die Einhaltung der Referenzmonitorprinzipien [DoD83]; insbesondere muß eine Sicherheitsarchitektur garantieren können, daß

- Sicherheitspolitiken vor unzulässiger Manipulation geschützt sind
- Sicherheitspolitiken sämtliche Kommunikation innerhalb ihrer Domäne (d.h. zwischen allen ihr untergeordneten Entitäten) beobachten und beeinflussen können
- die Beziehungen der Sicherheitspolitiken zu den ihnen assoziierten Domänen vor unzulässiger Manipulation geschützt sind.

Neben diesen grundsätzlichen Architektureigenschaften benötigen Sicherheitspolitiken vielfach darüberhinaus konkrete Sicherheitsmechanismen; hierzu gehören beispielsweise Protokolle zur Authentisierung in verteilten Systemen [NS87, OR87] oder Funktionen zur Verschlüsselung von Daten zwecks Wahrung ihrer Integrität und Vertraulichkeit bei der Speicherung und Kommunikation [Hor85, Sch93, Zim95]. Dabei ist die Realisierung von Sicherheitspolitiken innerhalb einer konkreten Sicherheitsarchitektur wie z.B. Kerberos und mittels konkreter Sicherheitsmechanismen selten so trivial, daß eine intuitive Auswahl und Komposition von Sicherheitsmechanismen erfolgen kann. Sicherheitsanforderungen wie „*Informationen in elektronischen Patientenakten mit nicht abrechnungsrelevantem Inhalt dürfen nicht von der Pflegestation in die Krankenhausverwaltung fließen*" bedingen ein strategisches Zusammenspiel von Informationsklassifizierung, Authentisierung, Autorisierung und Informationsflußsteuerung, welches einer methodischen und präzisen Formulierung bedarf, und dieses wird durch eine Sicherheitspolitik geleistet.

Zusammenfassend lassen sich als wesentliche Merkmale von Sicherheitspolitiken nennen:

- Sicherheitspolitiken sind systematische Formulierungen von Strategien (Algorithmen), deren Ziel die Erfüllung der individuellen Sicherheitsanforderungen einer Organisation, ihrer Unterstrukturen oder eines Applikationssystems ist.
- Sicherheitspolitiken besitzen einen wohldefinierten Wirkungsbereich, ihre Domäne. Die Festlegung einer Politikdomäne unterliegt dabei keinen festen Regeln; im Beispiel in Kapitel 2 orientierten sich Politikdomänen an den Unternehmensstrukturen.
- Sicherheitspolitiken sind Teil der Sicherheitsarchitektur eines IT-Systems; zu ihrer Implementierung können sowohl Architektureigenschaften als auch konkrete Sicherheitsmechanismen notwendig sein.

3.2 Modellierung von Sicherheitspolitiken

„Security policies are very simple, and should be easy to state in an appropriate formalism [GM82]."

Die Sicherheit eines Systems hängt entscheidend von der Sorgfalt ab, mit der seine Sicherheitspolitiken entwickelt und implementiert werden. Um hier hohen Qualitätsanforderungen zu entsprechen, werden als Technik zur ihrer präzisen Formulierung Sicherheitsmodelle und deren Spezifikation mittels geeigneter Spezifikationssprachen eingesetzt; diese Modelle bilden dann die Grundlage für eine Analyse von Sicherheitseigenschaften, und ihre Spezifikation ist die Ausgangsbasis einer Politikimplementierung. Der Einsatz formaler Sicherheitsmodelle ist in kommerziellen Anwendungen mit hohen Sicherheitsanforderungen in der Regel bereits heute obligatorisch. Gesetzliche Bestimmungen fordern hier die Evaluierung und Zertifizierung der Sicherheitspolitiken nach festgelegten Verfahren, wie sie unter anderem in [DoD83, ITS89, ITS90, Com96, Com98] beschrieben sind. Die Einstufung in höhere Qualitätsklassen setzt dabei die Verwendung von Sicherheitsmodellen und formaler Spezifikation voraus.

Im allgemeinen Sprachgebrauch der Informatik ist ein Modell ein Beschreibungsmittel, mit dem ein Bild der Realität erstellt wird. Ziele dabei sind

- die Vereinfachung der (häufig zu komplexen) Realität
- die Erklärung von beobachteten Phänomenen
- die Vorhersage von Phänomenen.

Die bei der Modellbildung eingesetzten Methoden bestehen aus dem Außerachtlassen nebensächlicher Aspekte und einer präzisen Herausarbeitung der wesentlichen Aspekte des Modellgegenstandes.

Sicherheitsmodelle sind formalisierte Beschreibungen wesentlicher Teile von Sicherheitspolitiken. Die getrennte Betrachtung von Sicherheitseigenschaften und anderen Systemeigenschaften mittels spezialisierter Modelle beruht dabei auf der These, daß formale Methoden selektiv auf kritische, harte Probleme anzuwenden sind, und ein höheres Maß an Formalität an Stellen mit einem höheren Maß an Risiko sein muß [KW97].

Der Begriff eines Sicherheitsmodells findet sich im heutigen Sprachgebrauch in drei Varianten; als Sicherheitsmodelle werden bezeichnet

- allgemeine Kalküle zur formalen Repräsentation konkreter Sicherheitspolitiken
- konkrete formale Repräsentationen konkreter Sicherheitspolitiken

- Spezifikationen von Sicherheitsanforderungen ohne die Beschreibung konkreter Mechanismen zu deren Realisierung.

[KM93] definiert ein Sicherheitsmodell als

> *eine abstrakte Beschreibung der für wesentlich erachteten Aspekte der Sicherheit eines IT-Systems. Ein Sicherheitsmodell dient der (formalen) Validierung und der Vorbereitung der Systemspezifikation und Implementierung.*

In dieser Arbeit wird der Begriff *Sicherheitsmodell* in seiner Interpretation als formale Repräsentation einer Sicherheitspolitik verwendet. Ein Sicherheitsmodell ist somit eine Reformulierung einer Sicherheitspolitik mit dem Ziel, durch eine präzise und in der Regel formale Beschreibung wesentlicher Aspekte einer Sicherheitspolitik ein vertieftes Verständnis zu erreichen und angestrebte Eigenschaften (Zugriffskontrollsicherheit, Informationsflußsicherheit, Ausbreitung von Rechten) formalen Nachweis- und Zertifizierungsverfahren zugänglich zu machen. Wesentlich ist dabei der Aspekt der Einfachheit; ein Sicherheitsmodell ist stets eine *offensichtliche* Repräsentation einer Politik, die den kritischen Paradigmenwechsel zwischen informeller und formaler Repräsentation klar und nachvollziehbar macht.

3.3 Modelle für Zugriffssteuerungspolitiken

Die Ausdruckskraft eines Sicherheitsmodells wird bestimmt durch die Beschreibungsmittel, die dem Entwickler bei der Aufstellung eines Modells zur Verfügung stehen. Ziel dieses Abschnitts ist die Vorstellung der Beschreibungsmittel, mit denen in dieser Arbeit Sicherheitsmodelle formuliert werden; im anschließenden Abschnitt wird dann eine Spezifikationssprache für Sicherheitsmodelle vorgestellt, die mit diesen Beschreibungsmitteln formuliert sind.

Der heutige Stand der Kunst bei der Modellierung von Sicherheitspolitiken erlaubt es noch nicht, von einem breiten Konsens hinsichtlich der verwendeten Modellabstraktionen zu sprechen. Deutlich wird dies an vielen bekannten Modellen von Sicherheitspolitiken (Bell/LaPadula [BL73a, BL73b, BL74, BL76], Biba [Bib77], Clark/Wilson [CW87], *Chinese-Wall* [BN89], *Typed Access Matrix* [San92b], [WWK96] u.v.m.), in denen weitgehend die verwendeten Modellabstraktionen auf die Belange der modellierten Sicherheitspolitik zugeschnitten sind. Mehr noch, die verwendeten Kalküle zur Formulierung der Modelle sind wiederum auf die Belange der Modellabstraktionen zugeschnitten [KM93]. Es besteht also die Situation, daß zwar durchaus für eine gewisse Anzahl von Sicherheitspolitiken Modelle vorliegen, leider jedoch unterschiedlichen Intuitionen folgend, auf unähnlichen Abstraktionen aufbauend und in unterschiedlichen Kalkülen formuliert.

Die fehlende Einigkeit in Beschreibungssprache und insbesondere in den Modellparadigmen macht es folglich schwierig, bei der Verfolgung des Ziels dieser Arbeit – der

Formulierung von Sicherheitspolitiken über domänenübergreifende Interaktionen – auf bestehenden Modellansätzen aufzusetzen. Ein erster Schritt muß also darin bestehen, eine gemeinsame Modellierungsmethode für eine hinreichend signifikante Klasse von Sicherheitspolitiken aufzustellen. Der folgende Diskurs wird sich auf die Klasse der Zugriffssteuerungspolitiken und ihre Modellierung konzentrieren; er begründet die in dieser Arbeit verwendete Modellierungstechnik für Zugriffssteuerungspolitiken und die daraus abgeleitete Spezifikationssprache.

3.3.1 Zugriffssteuerung

Zugriffsberechtigungen werden als organisatorische Maßnahme an Personen vergeben, denen ein entsprechendes Maß an Vertrauen entgegengebracht wird. Wird ein solches Recht fälschlich erteilt, mißbraucht oder unberechtigt angeeignet, so kann sich eine Vielzahl von grundsätzlichen Gefährdungen der Informationssicherheit hieraus ergeben. Hierzu gehören die unmittelbare Gefährdung der Vertraulichkeit, Integrität und Originalität von Daten durch einen direkten Zugriff unberechtigter Benutzer und ebenfalls Gefährdungen, die mittelbar durch unberechtigten Zugriff auf geheime kryptographische Schlüssel oder Paßwortdateien entstehen. Die administrative und informationstechnische Organisation von Zugriffsrechten spielt daher in der Informationssicherheit eine grundlegende Rolle, die sich entsprechend auch in einer Vielzahl von Zugriffssteuerungsmodellen niedergeschlagen hat ([BL76, Bib77, CW87, BN89, San92b, WWK96] u.v.m.).

Grundsätzlich wird zwischen zwei Klassen der Zugriffssteuerung unterschieden: der *wahlfreien* und der *obligatorischen* Zugriffssteuerung (englisch *discretionary* und *mandatory access control*). *Wahlfreie Zugriffssteuerung* bedeutet, daß der Eigentümer eines Objektes frei wählen kann, welche Zugriffsrechte für dieses Objekt vergeben werden. *Obligatorische Zugriffssteuerung* dagegen bedeutet, daß eine übergeordnete Sicherheitspolitik diese Rechte auf der Basis von Subjekt- und Objektattributen festlegt.

Ein bekanntes Beispiel für eine wahlfreie Zugriffssteuerung sind die Mechanismen des Unix-Betriebssystems. In Termen von Lese/Schreib/Ausführungsrechten kann hier der Eigentümer einer Datei bestimmen, welche Rechte er selbst, die Mitglieder einer Gruppe und alle weiteren Systembenutzer erhalten sollen.

Als Folge des Aufkommens trojanischer Pferde gilt heute diese Form der Zugriffssteuerung insbesondere in sensitiven IT-Systemen als zu schwach; Angriffe durch trojanische Pferde sind bei wahlfreier Zugriffskontrolle in der Lage, bestehende Rechte eines Benutzers auszunutzen, um unberechtigten Zugang zu Daten zu erhalten.

Als Konsequenz erfolgt bei der obligatorischen Zugriffssteuerung die Vergabe von Zugriffsrechten durch eine separate und besonders geschützte obligatorische Zugriffssteuerungspolitik, die häufig in einem Sicherheitsmodell formal beschrieben ist. Ein Beispiel hierfür sind die vielerorts verwendeten, auf einer Vertraulichkeitsklassifizierung von Subjekten und Objekten basierenden hierarchischen Sicherheitspolitiken, wie sie unter

anderem im Bell/LaPadula - Modell beschrieben sind.

3.3.2 Kalküle und konkrete Modelle

Im folgenden sollen einige Kalküle und Modelle zur Beschreibung von Sicherheitspolitiken skizziert werden, auf die im weiteren Verlauf der Arbeit vielfach bezug genommen wird.

Die Lampson'sche Zugriffsmatrix

Die klassische Abstraktion zur Formulierung von Zugriffsrechten ist die Lampson'sche Zugriffsmatrix. Zurückgehend auf einen Auftrag des amerikanischen Verteidigungsministeriums aus dem Jahr 1970 schlug Butler Lampson den folgenden Ansatz vor.

Zugriffsrechte in einem IT-Systeme werden modelliert durch

- eine Menge S von handelnden Einheiten (Subjekte), die Operationen auf Objekten ausführen; diese kann beispielsweise Personen, informationstechnische Prozesse oder Arbeitsplatzrechner repräsentieren;
- eine Menge O von passiven Einheiten, den Objekten; dies sind beispielsweise Dokumente oder Dateien;
- eine Menge F von Zugriffsfunktionen, die Subjekte auf Objekten ausführen können;
- eine Funktion $grant : S \times O \times F \rightarrow \{erlaubt, nicht\ erlaubt\}$ zur Prüfung eines Zugriffsrechts.

Das Recht eines Subjekts $s \in S$ zur Ausführung einer Operation $f \in F$ auf einem Objekt $o \in O$ wird nun dadurch beschrieben, daß $grant$ an der Stelle (s, o, f) den Wert „*erlaubt*" besitzt.

Repräsentiert man $grant$ durch eine 2-dimensionale Zugriffsmatrix $ZM : S \times O \rightarrow \wp(F)$, in der jedes Subjekt durch eine Zeile und jedes Objekt durch eine Spalte repräsentiert ist, und die in Zelle $ZM[s, o]$ die für s erlaubten Operationen bezüglich o als Teilmenge von F enthält, so erhält man die Lampson'sche Zugriffsmatrix.

Das Harrison, Ruzzo und Ullman – Modell

Harrison, Ruzzo und Ullman verwenden Lampson's Konzept der Zugriffsmatrix zur Aufstellung eines allgemeinen Kalküls zur Formulierung wahlfreier Zugriffskontrolle, welcher die dynamische Modellierung sicherheitsrelevanter Operationen von Anwendungssystemen einbezieht [HRU76]. Ursprünglich zur Komplexitätsbestimmung der

Analyse von Sicherheitspolitiken gedacht, wurde das *HRU*-Modell auf Grund seiner eingängigen Notation zur Formulierung einer Vielzahl von Zugriffssteuerungspolitiken verwendet. Das Modell vereinigt das klassische Modell des Zustandsautomaten mit der Lampson'schen Zugriffsmatrix auf die folgende Weise.

Analog zu Lampson's Ansatz gibt es eine Subjektmenge S und eine Objektmenge O; Subjekt und Objekt können dabei kurzeitige Rollen sein, die von den Einheiten eines Systems für die Dauer einer Subjekt/Objekt-Interaktion angenommen werden, d.h. $S \subseteq O$. Rechte der Subjekte an den Objekten stammen aus einer endlichen Rechtemenge R und sind formuliert in einer Zugriffsmatrix $ZM : S \times O \rightarrow \wp(R)$.

Das dynamische Verhalten eines HRU-Modells wird mittels eines Zustandsautomaten modelliert, dessen Zustände Tripel der Form (S, O, ZM) sind; dabei sind S und O die Mengen der aktuell in einem Zustand vorhandenen Subjekte und Objekte, und ZM ist die zugehörige Belegung der Zugriffsmatrix.

Jede Eingabe für diesen Automaten modelliert den Aufruf einer konkreten applikationsspezifischen Operation und verändert entsprechend den Zustand des Modells. Eine Eingabe setzt sich zusammen aus der konkreten applikationsspezifischen Operation $f \in F$ (siehe das Lampson'sche Modell) und einem k-dimensionalen Vektor x, der die bei der Operation involvierten k Subjekte und Objekte beschreibt. Die sicherheitsmodellrelevante Semantik einer solchen Operation wird in einem HRU-Modell durch einen Ausdruck der in Bild 3.1 gezeigten allgemeinen Form beschrieben.

$$
\begin{aligned}
f(x) = \text{IF } & r_1 \in ZM(x_{s_1}, x_{o_1}) \text{ AND} \\
& \ldots \\
& r_m \in ZM(x_{s_m}, x_{o_m}) \\
\text{THEN} & \\
& p_1; \ldots; p_n \\
\text{FI} &
\end{aligned}
$$

Bild 3.1: Einheitsform der Beschreibung applikationsspezifischer Operation

Hierbei sind $r_1 \ldots r_m$ Rechte aus R, und $s_1 \ldots s_m$ sowie $o_1 \ldots o_m$ sind Vektorindizes zwischen 1 und k. Die $p_1 \ldots p_n$ stammen aus der Menge der HRU-Primitivoperationen, die in Bild 3.2 aufgelistet ist. Diese Operationen beeinflussen den Zustand eines Modells, sind also auf Subjektmenge, Objektmenge und Zugriffsmatrix definiert. Jede applikationsspezifische Operation wird in einem HRU-Modell somit beschrieben durch einen Bedingungsteil und einem Anweisungsteil; der Bedingungsteil spezifiziert die Rechte, die in der Zugriffsmatrix für die Ausführung des Anweisungsteils vorhanden sein müssen, der Anweisungsteil definiert den Folgezustand des Automaten. Die Gesamtheit aller solcher Regeln zur Veränderung des Modellzustands wird als das *Autorisierungsschema* eines HRU-Modells bezeichnet.

Eine Sicherheitspolitik wird nun modelliert durch Definition der Mengen S, O und R

enter r into ZM[s,o]	trägt das Recht r in die Matrix an der Stelle $[s, o]$ ein
delete r from ZM[s,o]	löscht das Recht r aus der Matrix an der Stelle $[s, o]$
create subject s	erzeugt ein neues Subjekt
create object o	erzeugt ein neues Objekt
destroy subject s	löscht ein existierendes Subjekt
destroy object o	löscht ein existierendes Objekt

Bild 3.2: Die HRU-Primitivoperationen

und durch eine initiale Belegung der Zugriffsmatrix. Die sicherheitsrelevanten Operationen von Anwendungssystemen werden in einem konkreten Autorisierungsschema des obigen Baumusters beschrieben.

Bei der Analyse der Eigenschaften einer so formulierten Sicherheitspolitik ist es nun eine grundsätzliche Fragestellung, ob bei gegebener Anfangsbelegung der Zugriffsmatrix jemals ein bestimmtes Subjekt ein bestimmtes Recht bezüglich eines bestimmten Objektes erlangen kann, mit anderen Worten, ob jemals in eine bestimmte Matrixzelle ein bestimmtes Zugriffsrecht eingetragen werden kann. Eine allgemeine Beantwortung derartiger Fragestellungen erlaubt es dann, konkrete Eigenschaften einer Sicherheitspolitik festzustellen. Ein Beispiel ist die Beantwortung der Frage innerhalb der Analyse einer Sicherheitspolitik für eine Klinik, ob „es jemals geschehen kann, daß bei einer gegebenen Sicherheitspolitik die Klinikapotheke ein Leserecht auf die diagnostischen Informationen einer Patientenakte erhält".

Diese Fragestellung in der allgemeinen Form ist bekannt als das „*safety*-Problem" des HRU-Modells: eine konkrete Belegung der Zugriffsmatrix wird in bezug auf ein Recht $r \in R$ genau dann als **sicher** (*HRU-safe*) bezeichnet, wenn es keine Folge von Operationen des Anwendungssystems gibt, die r in eine Matrixzelle einträgt, welche r nicht bereits enthält.

Eine der wesentlichen Leistungen des HRU-Modells ist der Nachweis, daß das HRU-Safetyproblem nicht allgemein entscheidbar ist. Dieses zunächst entmutigende Ergebnis stieß allerdings neue Arbeiten an, die durch Einschränkungen der Allgemeinheit des Modells neue Modellvarianten hervorbrachten, in denen das HRU-Safetyproblem entscheidbar ist, obschon teils immer noch mit NP-vollständiger Komplexität (monooperationales HRU-Modell, monotones HRU-Modell [HRU76], *Typed Access Matrix* - Modell [San92b]). Dies macht ein Dilemma deutlich, welches heute noch große Schwierigkeiten bei der Modellierung von Sicherheitspolitiken bereitet. Zum einen existieren Kalküle, deren Ausdruckskraft zur Modellierung eines breiten Spektrums an Sicherheitspolitiken hinreicht; für diese Kalküle gibt es aber keine Algorithmen, die automatisch bestimmte Eigenschaften analysieren können. Zum anderen gibt es Kalküle, für die solche Algorithmen existieren; leider besitzen diese Kalküle oft erhebliche Einschränkungen in ihrer Ausdruckskraft.

Lattices

Neben dem HRU-Modell gibt es einen weiteren elementaren Kalkül zur formalen Repräsentation konkreter Zugriffssteuerungspolitiken, die 1976 von Dorothy Denning vorgestellten *lattices* [Den76]. *Lattices* (zu deutsch Verbände) sind algebraische Strukturen bestehend aus einer endlichen partiell geordneten Menge mit wohldefiniertem größten und kleinsten Element und einer transitiven, reflexiven und antisymmetrischen Halbordnung „$\leq$" mit der Interpretation, daß genau dann Informationen von a nach b fließen können, wenn $a \leq b$ gilt.

Warum eignen sich nun Verbände zur Modellierung von Zugriffssteuerungspolitiken? In einer Vielzahl von Anwendungen wird bezüglich der Zugriffsoperationen von Subjekten auf Objekte ledig unterschieden, ob der Zugriff lesender oder schreibender Art ist. Ausgedrückt in Termen des Informationsflusses bedeutet ein Lesezugriff eines Subjekts s auf ein Objekt o, daß Informationen von o nach s fließen; schreibt s dagegen auf o, so kehrt sich die Informationsflußrichtung um. Lese- und Schreibzugriffe lassen sich somit als Informationsflüsse ausdrücken, und genau hierzu sind Verbände besonders geeignet. Die Technik dabei ist die folgende:

In einem ersten Schritt werden die einer Sicherheitspolitik unterworfenen Subjekte und Objekte klassifiziert und mit einem entsprechenden Attribut versehen; es gibt beispielsweise die Klasse der vertrauenswürdigen Subjekte (VWS) und die der weniger vertrauenswürdige Subjekte (WVWS), und es gibt beispielsweise sensitive und weniger sensitive Objekte (SO und WSO). Diese vier Klassen definieren nun die Klassenmenge eines Verbandes.

Im zweiten Schritt werden dann die zulässigen Informationsflüsse zwischen den Klassen definiert; Informationen dürfen beispielsweise von weniger sensitiven Objekten zu weniger vertrauenswürdigen Subjekten fließen, nicht aber von sensitiven Objekten zu weniger vertrauenswürdigen Subjekten. Zur Modellierung von zulässigen Informationsflüssen zwischen den Klassen dient die Halbordnung: Informationen dürfen von einer Klasse X zu einer Klasse Y genau dann fließen, wenn $X \leq Y$ gilt. Durch die Definitionen einer konkreten Halbordnung zwischen *SO, WSO, VWS* und *WVWS* in der Form $WSO \leq WVWS$, $WSO \leq VWS$ und $SO \leq VWS$ läßt sich nun formal beschreiben, daß vertrauenswürdige Subjekte beide Klassen von Objekten lesen dürfen, weniger vertrauenswürdige Subjekte dagegen lediglich die weniger sensitiven Objekte.

Verbände sind auch wegen ihrer intuitiven Darstellbarkeit als Informationsflußgraph beliebt; Bild 3.3 zeigt das Beispiel als Informationsflußgraph.

Verbände wurden viele Jahre lang als eine der grundlegenden Abstraktionen zur Modellierung von informationsflußbasierten Sicherheitspolitiken angesehen. Ende der achtziger Jahre wurden dann Sicherheitspolitiken bekannt, die zunächst scheinbar nicht mittels der Verband-Abstraktion modellierbar waren. Brewer und Nash publizierten in [BN89] ein Modell der Chinese-Wall-Politik und postulierten, dieses sei allein mittels der Verband-Abstraktion nicht ausdrückbar (u.a. auf Grund der in der Sicherheitspoli-

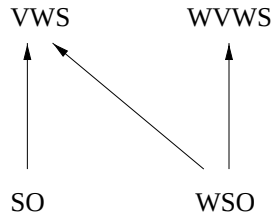


Bild 3.3: Informationsflußgraph des Beispiels

tik enthaltenen wahlfreien Politikanteile). Clark und Wilson stellten eine Sicherheitspolitik vor [CW87], in der Informationsfluß zwischen einem Subjekt S und einem Objekt O niemals direkt stattfindet, sondern ausschließlich über wohldefinierte Transaktionen T erfolgt, wie in Bild 3.4 gezeigt.

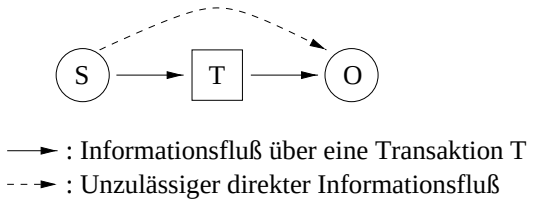


Bild 3.4: Nichttransitiver Informationsfluß

Solche Informationsflüsse sind ebenfalls nicht ohne weiteres durch einen Verband zu beschreiben, da sie Ausnahmen von der Transitivität der Halbordnung des Verbands darstellen; in einem Verband folgt aus $S \leq T$ und $T \leq O$ immer $S \leq O$, und dies steht im Widerspruch zur Clark/Wilson Politik.

Die Suche nach universell verwendbaren Abstraktionen zur Modellierung von Sicherheitspolitiken erhielt durch diese Beispiele zunächst einen Rückschlag. Dann aber gab Sandhu in [San92a] ein verbandbasiertes Modell für Chinese-Wall an, indem er durch eine getrennte Modellierung von menschlichen Benutzern und ihrer Vertretersubjekte im informationstechnischen System das Brewer/Nash-Modell verfeinerte. Foley zeigt in [Fol89] die Modellierung nichttransitiver Informationsflüsse mit Verbänden. Schließlich verallgemeinert [Bry97] Sandhu's Verfahren in [San92a] so, daß auch solche Sicherheitspolitiken mittels Verbänden modelliert werden können, in denen der Informationsfluß in Teilen nichtreflexiv, symmetrisch oder nichttransitiv ist. Bryce gibt einen Algorithmus an, der einen allgemeinen Informationsflußgraphen in einen reflexiven, antisymmetrischen und transitiven Verband transformiert und stellt damit endgültig die Universalität der Verband-Abstraktion unter Beweis.

Hierarchische Zugriffssteuerungsmodelle

Sowohl das HRU-Modell als auch die Verbände sind allgemeine Kalküle zur formalen Repräsentation konkreter Zugriffssteuerungspolitiken. Anwendungen beider Kalküle finden sich entsprechend in konkreten Sicherheitsmodellen, etwa in der Klasse der hierarchischen (multilevel, MLS) Sicherheitsmodelle, von denen das Bell/LaPadula-Modell und das Biba-Modell bekannte Vertreter sind.

Das Bell/LaPadula Sicherheitsmodell

Das Bell/LaPadula-Modell ist ein formales Sicherheitsmodell zum Schutz der Vertraulichkeit. Aus vielerlei Gründen wird dieses Modell häufig als Referenz und Vergleichsobjekt herangezogen. Zum einen war es die erste weitverbreitete Formalisierung einer Sicherheitspolitik, die zudem vom amerikanischen Verteidigungsministerium als eine Standardpolitik propagiert wurde und Eingang in die standardisierten Bewertungskriterien für sichere IT-Systeme fand [DoD83]. Zum anderen ist es als Verband- und Zugriffsmatrix-basiertes Modell bereits sehr konkret und erleichtert so die Implementierung und ihre Verifikation. Schließlich und nicht zuletzt definiert das Modell präzise, was es unter „Sicherheit" versteht und gibt notwendige und hinreichende Bedingungen für die Eigenschaft der Sicherheit eines konkreten Modells an (im Bell/LaPadula *Basic Security Theorem*, [BL73a]).

Kurz skizziert, beruht das Modell auf einer hierarchischen Klassifizierung der Subjekte und Objekte mittels einer bezüglich des Informationsflusses total geordneten Klassenmenge (z.B. $\{\textit{öffentlich} \leq \textit{vertraulich} \leq \textit{geheim} \leq \textit{sehr geheim}\}$); orthogonal hierzu gibt es noch eine Aufteilung in Kategorien (z.B. Marine, Luftstreitkräfte, Heer), die dem *need-to-know* - Prinzip Rechnung trägt.

Die Definition der Bell/LaPadula-Sicherheit setzt nun die beiden Hauptkomponenten des Modells – einen Verband und eine Zugriffsmatrix – zueinander in Beziehung. Dazu wird zunächst angenommen, daß die Informationsflußregeln der Sicherheitspolitik korrekt mittels des Verbands formuliert sind; diese Annahme ist zulässig, da solche Regeln in den typisch avisierten Anwendungsgebieten militärischer Dokumentenverwaltungssysteme extrem einfach sind. Sodann werden diese Regeln auf die Rechtebelegung einer Zugriffsmatrix abgebildet; diese Abbildung kann als Implementierung des Verbands aufgefaßt werden, da Zugriffsmatrizen typischerweise in Form elementarer Sicherheitsmechanismen in Systemplattformen anzutreffen sind. Durch das *Basic Security Theorem* wird nun diese Implementierung mit dem Verband in Beziehung gesetzt und eine Aussage darüber gemacht, unter welchen Bedingungen eine Matrixdarstellung des Verbandes korrekt ist.

Das Biba Sicherheitsmodell

Das Biba-Modell kann als eine Art Gegenpol des Bell/LaPadula - Modells angesehen werden; während die Wahrung der Vertraulichkeit ein vornehmliches Ziel in typischerweise militärischen Umgebungen ist, haben kommerzielle Organisationen häufig ein vorwiegendes Interesse an der Wahrung der Integrität. Kehrt man die Informationsflußrichtung des Bell/LaPadula - Verbands um, so erzielt man die Wirkung, daß nur noch besonders vertrauenswürdige Subjekte die besonders sensitiven Objekte modifizieren dürfen, und man erhält ein hierarchisches Zugriffssteuerungsmodell zur Wahrung der Integrität.

Zumindestens namentlich erwähnt werden sollen als weitere konkrete Modelle noch das Goguen-Meseguer-Modell [GM82] (beruhend auf der gegenseitigen Nichtbeeinflussung von Benutzergruppen durch ihre jeweiligen Aktionsfolgen), das Terry-Wisemann-Modell [TW89] (ein Integritäts- und Vertraulichkeitsmodell beruhend auf endlichen Automaten), das Chinese-Wall-Modell [BN89] (für das spezielle Szenarium englischer Unternehmensberatungen) und das Clark-Wilson-Modell (Wahrung der Integrität durch wohlgeformte Transaktionen); als allgemeiner Kalkül zur Formulierung konkreter Modelle kann noch das *Typed-Access-Matrix*-Modell [San92b] gelten.

3.3.3 Zusammenfassung

Faßt man die Ergebnisse dieses Diskurses über die Modellierung von Zugriffssteuerungspolitiken zusammen, so lassen sich einige signifikante Gemeinsamkeiten der betrachteten Modelle feststellen.

a) Zugriffssteuerungsmodelle modellieren Politikdomänen.

   Anders als in den abgeschlossenen Hochsicherheitssystemen militärischer Anwendungen gilt in offenen verteilten Systemen eine Sicherheitspolitik nicht global für das gesamte System, sondern ist auf eine konkrete Einflußsphäre, ihre Domäne, beschränkt. Politikdomänen können beispielsweise organisationsspezifisch („sämtliche zu einer Organisation gehörenden Applikationssysteme, Datenbanken, etc.") oder etwa applikationsspezifisch („der Mailer und die von ihm verwalteten Mailboxen") festgelegt sein. Konkrete Beispiele für die Modellierung von Politikdomänen sind die Mengen S und O im Lampson'schen Zugriffsmodell und im HRU-Kalkül; analoge Mengen finden sich ebenfalls im Bell/LaPadula-Modell und im Biba-Modell.

b) Zugriffssteuerungsmodelle müssen ein breites Spektrum von Zugriffsregeln ausdrücken können.

   Bei der Breite des Anwendungsspektrums heutiger Sicherheitspolitiken ist es nicht mehr ausreichend, eine einfache Klassifikation der Operationen in lesende und schreibende Operationen vorzunehmen, wie dies beispielsweise bei MLS-

Politiken der Fall ist. Selbst eine professionelle Formulierung einer MLS-Politik enthält bereits neben den Regeln für Lese/Schreiboperationen auch Regeln, die die Reklassifizierung von Dokumenten kontrollieren. Objektbasierte Anwendungssysteme besitzen komplexe Operationen auf Datentypinstanzen, die sich nicht in natürlicher Weise auf eine Lese/Schreibsemantik reduzieren lassen. Innerhalb des Sicherheitsmodells muß somit eine feingranulierte Möglichkeit bestehen, jeder spezifischen Operation auf den Applikationsobjekten eine eigene Regel zu assoziieren, die die Ausführung der Operation kontrolliert. Zugriffsmatrizen beispielsweise ermöglichen dies, wenn als Rechtemenge die Menge aller Anwendungsoperationen gewählt wird; auch der HRU-Kalkül enthält explizite Ausdrucksmittel zur Modellierung anwendungsspezifischer Operationen.
Weiterhin besitzen Sicherheitspolitiken in der Regel einen Zustand, der mindestens aus Domäne, Verband und Zugriffsmatrix besteht. Dieser Zustand ist dynamischen Einflüssen unterworfen. Hierzu gehören beispielsweise im HRU-Kalkül die in Bild 3.2 gezeigten Primitivoperationen; weiterhin ist die Änderung der Klassenzugehörigkeit von Subjekten sowie die Neuklassifikation von Dokumenten für den praktischen Einsatz von MLS-Politiken unabdingbar. Die Regeln für die kontrollierte Veränderung des Politikzustands sind gleichfalls Bestandteil der Sicherheitspolitik und müssen somit im Modell formulierbar sein.

c) Zugriffssteuerungsmodelle müssen Politikzustände ausdrücken können.

   Regeln über die Zulässigkeit von Zugriffen sind in Zugriffssteuerungsmodellen in der Regel von einer dynamisch sich ändernden wahlfreien und obligatorischen Rechtesituation abhängig. Als konkrete Mittel zur Formulierung von Modellzuständen wurden im vorangehenden Diskurs Zugriffsmatrizen, Verbände und Klassifikationsfunktionen vorgestellt.

d) Zugriffssteuerungsmodelle müssen universell gültige Sicherheitseigenschaften ausdrücken können.

   Politikinvarianten dienen zur Spezifikation von globalen Eigenschaften eines Sicherheitsmodells, wie etwa der Safety-Eigenschaft des HRU-Modells oder die BLP-Sicherheit (vgl. Seiten 45 und 48). Hierdurch lassen sich insbesondere Modellrestriktionen formulieren, die bei einer nicht allgemeinen Entscheidbarkeit des Safety-Problems bestimmte Sicherheitseigenschaften explizit herstellen können.
   Weiterhin können Sicherheitspolitiken universell gültige Wahrheiten enthalten, die dann im Modell durch Invariantendefinitionen repräsentiert werden. In der Chinese-Wall-Politik ist es eine solche Wahrheit, daß in der gesamten Zugriffshistorie eines jeden Beraters zu keinem Zeitpunkt Firmen auftauchen dürfen, die miteinander in Konkurrenz stehen.
   Die Formulierung solcher Wahrheiten im Modell als Theoreme und der Beweis ihrer Widerspruchsfreiheit oder sogar ihrer Ableitbarkeit aus den Regeln/Axiomen des Modells ist ein wesentliches Kriterium für die innere Konsistenz des Modells.

e) Zugriffssteuerungsmodelle müssen in wohldefinierte Initialzustände versetzt wer-

den können.

Dies ist eine triviale, aber für die Beweisführung über Sicherheitseigenschaften ebenso fundamentale Eigenschaft. Zudem trägt sie der Praxis Rechnung, da eine Sicherheitspolitik nicht immer seit dem ersten Ablauf eines Applikationssystems diesem assoziiert ist. Wird eine Sicherheitspolitik erst im nachhinein an existierende Applikationssysteme angebunden (beispielsweise im Rahmen eines Austauschs der Sicherheitspolitik), so hatte diese nicht die Möglichkeit, sukzessive mit wachsender Domäne den zugehörigen Politikzustand aufzubauen und muß explizit mit einem der Realität entsprechenden Anfangszustand ausgestattet werden.
Weiterhin ist die Möglichkeit der Definition eines Grundzustands auch praktisch: sie reduziert den Spezifikations- und Beweisaufwand für solche Politiken, die mit statischer Domäne, Matrix und Verband arbeiten: politikspezifische Operationen zur Manipulation des Politikzustands können hier entfallen. Schließlich, und nicht zuletzt, existieren Sicherheitsmodelle, deren HRU-Safety-Entscheidbarkeit nur für statische Domänenmengen gegeben ist; für solche Modelle ist die Möglichkeit der Definition eines Grundzustandes unabdingbar.

Der nächste Schritt auf dem Wege zur Realisierung einer Sicherheitspolitik ist nun die Spezifikation des Sicherheitsmodells (vgl. Bild 2.1). Unter Ausnutzung der festgestellten Gemeinsamkeiten und Anforderungen an Zugriffssteuerungsmodelle wird im nächsten Abschnitt eine Spezifikationssprache vorgestellt, die eine direkte Formulierung eines Zugriffssteuerungsmodells unter den Kriterien a) - e) erlaubt. Der letzte Schritt, die Implementierung der Spezifikation, wird Thema des letzten Abschnitts dieses Kapitels sein.

3.4 Spezifikation von Sicherheitspolitiken

Dieser Abschnitt beschreibt *Skippy*, eine Sprache zur Spezifikation der im vorangehenden Abschnitt beschriebenen Modellkomponenten von Zugriffssteuerungspolitiken. Skippy wird in einer Entwicklungsumgebung für Sicherheitspolitiken [KvKO95] zur Spezifikation von Sicherheitspolitiken eingesetzt; unabhängig von Skippy's Rolle in dieser Arbeit findet sich eine allgemeine Motivation und Beschreibung in [Bry96].

3.4.1 Skippy

Bereits im vorangehenden Abschnitt wurden Beispiele genannt, in denen die zur Beschreibung eines Sicherheitsmodells verwendeten Kalküle auf die jeweiligen Belange der Modellabstraktionen zugeschnitten sind. Brewer und Nash verwenden Prädikatenlogik zur Repräsentation der Chinese-Wall Sicherheitspolitik und zum Beweis von Sicherheitseigenschaften ihres konkreten Modells [BN89], und Terry und Wiseman in [TW89] verwenden *Z* [Spi87, Dil90], eine universelle Spezifikationssprache mit breite-

rem, nicht auf Sicherheitsmodellierung zugeschnittenem Anwendungsspektrum. Universelle Programmiersprachen dagegen wie etwa C++ eignen sich im allgemeinen nicht zur Beschreibung von Sicherheitsmodellen, da sie mit problemfremden Eigenschaften überfrachtet sind und ihre Komplexität Korrektkeitsbeweise erschwert.

Eines der wesentlichen Ziele bei der Entwicklung von Skippy war die Überbrückung der semantischen Lücke zwischen Politikspezifikation und Politikimplementierung und die Vorgabe eines festen sprachlichen Rahmens, innerhalb dessen durch Struktur und Typsystem die Formulierungsmöglichkeiten auf die essentiellen Paradigmen von Zugriffssteuerungspolitiken eingeschränkt sind. Dies führt zu einem homogenen Beschreibungsgerüst, auf dem Werkzeuge wie Visualisierer oder Codegeneratoren aufsetzen können; weiterhin läßt sich durch vordefinierte Typen und die automatische Instanziierung der allen Modellen gemeinsamen Komponenten der Formulierungsaufwand (und die Möglichkeit, Fehler zu machen) reduzieren.

Zur Erläuterung der Rollen der Sprachkomponenten werden zunächst die vordefinierten Typen und Variablen erklärt, anschließend wird eine vollständige Politikspezifikation in einem Beispiel informell erläutert, und schließlich wird die Syntax von Skippy in Backus-Naur Form beschrieben. Da es nicht das Ziel der Arbeit ist, Skippy selbst zu beschreiben, endet die Vorstellung hiernach; für eine ausführliche Definition der Skippy - Semantik wird auf [Bry96] verwiesen. Im weiteren Verlauf der Arbeit werden die Konzepte von Skippy aufgegriffen, um eine Spezifikationssprache für Metapolitikmodelle zu definieren.

3.4.2 Vordefinierte Skippy-Elemente

Skippy enthält die für Zugriffssteuerungspolitiken als wesentlich erachteten oben beschriebenen Modellkomponenten als vordefinierte Typen und Variablen; ihre Instanziierung erfolgt automatisch innerhalb einer Politikdeklaration. Die in Skippy vordefinierten Typen sind

ENT die Menge der Entitäten - Namen; legt in einer Politikspezifikation den Typ (im Sinne einer Programmiersprache) der Entitätenmenge fest.

CLASS die Menge der Klassennamen des Verbands; legt in einer Politikspezifikation den Typ (im Sinne einer Programmiersprache) eines Elements der Verbandmenge fest.

R die Menge der Zugriffsrechte - Namen; legt in einer Politikspezifikation den Typ (im Sinne einer Programmiersprache) der Rechte fest.

OP die Menge der Operationsnamen, zu denen sowohl die applikationsspezifischen Operationen (zu denen die Sicherheitspolitik Zugriffsregeln definiert) als auch die politikspezifischen Operationen (die den Politikzustand

verändern) gehören. Da zu jeder Operation auch ein entsprechendes Zugriffsrecht gehört, gilt $OP \subseteq R$.

ENT, CLASS und R sind generisch in dem Sinne, daß die expliziten Werte undefiniert bleiben, da diese keine Auswirkungen auf die Politiksemantik besitzen. Diese Typen werden lediglich zur Konstruktion der vordefinierten Variablen für die Entitätenmenge, die Zugriffsmatrix und das Klassifikationsschema der Entitäten benutzt. Die in Skippy vordefinierten Variablen sind

entities die Entitätenmenge: eine Teilmenge von ENT, die sämtliche der Sicherheitspolitik unterliegenden Applikationsentitäten enthält und auf diese Weise die Politikdomäne definiert.

m die Zugriffsmatrix, die für jedes Paar $e_1, e_2 \in entities$ die wahlfreien Zugriffsrechte beschreibt: eine Abbildung $ENT \times ENT \rightarrow \wp(R)$.

cl die Verband-Klassifizierung der Entitäten: eine Abbildung $ENT \rightarrow CLASS$, welche die Zuordnung von Verbandklassen zu den Politikentitäten beschreibt.

3.4.3 Ein Beispiel: Chinese-Wall als Skippy-Deklaration

Zur Förderung des intuitiven Verständnisses sollen Syntax und Semantik von Skippy zunächst am Beispiel der in [San92a] vorgestellten Variante eines Modells der Chinese-Wall-Politik illustriert werden. Dieses Beispiel ist weithin bekannt, wird in der Praxis eingesetzt und ist ausführlich in [San92a] diskutiert, so daß eine direkte Vergleichsmöglichkeit von Spezifikation und Modell gegeben ist.

Im Szenarium der Chinese-Wall-Politik hat eine Unternehmensberatung eine Reihe von Klienten, die untereinander durchaus in Konkurrenz stehen können. Bei ihren Beratungstätigkeiten gewinnen die Angestellten der Beratungsfirma natürlich vertrauliche Informationen über die beratenen Unternehmen; die Sicherheitspolitik soll nun verhindern, daß so gewonnenes Insiderwissen bei der gleichzeitigen Beratung konkurrierender Unternehmen ausgenutzt werden kann. Hat ein Berater beispielsweise eine Bank beraten, so soll er hiernach für andere, konkurrierende Banken nicht mehr tätig werden können. Die Chinese-Wall-Politik soll somit verhindern, daß Informationen über ein Unternehmen zu einem anderen, konkurrierenden Unternehmen fließen.

Die Modellierung dieses Szenarios erfolgt im ersten Schritt durch Aufstellung einer Konfliktrelation. Hierzu wird eine Klassenbildung auf der Menge der Unternehmensdaten vorgenommen, die die Daten solcher Unternehmen in einer Konfliktklasse zusammenfaßt, die zueinander in Konkurrenz stehen. Ziel ist die Verhinderung des Informationsflusses zwischen den Elementen derselben Klasse durch eine „Informationsflußmauer" zwischen ihren Elementen – daher der Name der Politik.

Das Beispiel in Bild 3.5 zeigt ein Chinese-Wall-Szenarium, in dem ein Beratungsunternehmen zwei Banken, $Bank_1$ und $Bank_2$, sowie zwei Firmen im Ölgeschäft, $\ddot{O}l_1$ und $\ddot{O}l_2$, als Kunden hat. Sowohl die Banken als auch die Ölfirmen stehen zueinander in Konkurrenz, Banken und Ölfirmen dagegen nicht; die gestrichelte Linie innerhalb der Konfliktklassen deutet die dem Modell namensgebende Mauer an.

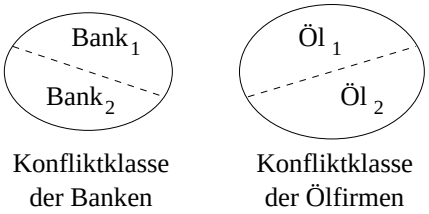


Bild 3.5: Konfliktklassen im Chinese-Wall-Beispiel

Transformiert man diese Konfliktrelation in einen Informationflußgraphen, so sind die legalen Informationsflüsse $Bank_1 \rightarrow \ddot{O}l_1, Bank_1 \rightarrow \ddot{O}l_2, Bank_2 \rightarrow \ddot{O}l_1, Bank_2 \rightarrow \ddot{O}l_2, \ddot{O}l_1 \rightarrow Bank_1, \ddot{O}l_1 \rightarrow Bank_2, \ddot{O}l_2 \rightarrow Bank_1$, und $\ddot{O}l_2 \rightarrow Bank_2$. Dabei entsteht der Informationflußgraph in Bild 3.6.

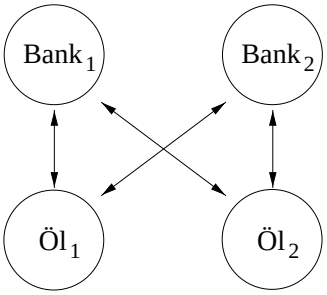


Bild 3.6: Informationsflußgraph im Chinese-Wall-Beispiel

Dieser Informationsflußgraph ist ganz offensichtlich noch kein Verband: er enthält sowohl symmetrischen Informationsfluß ($Bank_1 \rightarrow \ddot{O}l_1, \ddot{O}l_1 \rightarrow Bank_1$) als auch Nicht-Transitivität (aus $\ddot{O}l_1 \rightarrow Bank_1$ und $Bank_1 \rightarrow \ddot{O}l_2$ darf nicht folgen $\ddot{O}l_1 \rightarrow \ddot{O}l_2$). Daher soll nun zunächst aus dem Informationsflußgraphen ein Verband konstruiert werden.

Grundlage jeder Verbandmodellierung ist die Festlegung der Verbandklassen (oder *Labels*) sowie einer Halbordnung, die den zulässigen Informationsfluß zwischen den Verbandklassen beschreibt. Der zulässige Informationsfluß innerhalb einer Applikation wird dann durch die Zuordnung der Verbandklassen zu den Entitäten der Applikation definiert. Wie das obige Beispiel zeigt, sind Label der Art $Bank_1$ oder $\ddot{O}l_2$ offensichtlich nicht ausreichend, um die Informationsflüsse in der Chinese-Wall-Politik

zu modellieren: die Transitivität der Halbordnung in Bild 3.6 führt zu unerwünschten Informationsflüssen, beispielsweise zwischen $Bank_1$ und $Bank_2$ über $\ddot{O}l_1$.

Festlegung der Verbandklassen

Sandhu schlägt in seinem eigenen Chinese-Wall-Modell das folgende Verfahren vor.

Im allgemeinen trifft man bei der Modellierung von Chinese-Wall n Konfliktklassen $C_1, .., C_n$ an, wobei in jeder Konfliktklasse wiederum irgendeine Anzahl von Unternehmensdatenobjekten enthalten sind. Jedes dieser Objekte erhält als Label eine Menge, deren Elemente diejenigen Unternehmen sind, über die das Objekt Informationen enthält. Ein Objekt mit Informationen über $Bank_1$ und $Bank_2$ wird somit mit dem Label $\{Bank_1, Bank_2\}$ versehen. Dieses konkrete Label ist offensichtlich im Sinne der Sicherheitspolitik illegal; da $Bank_1$ und $Bank_2$ zueinander in Konflikt stehen, darf es auch kein Objekt geben, welches Informationen über beide Banken gleichzeitig enthält. Label dieser Bauart werden in Sandhu's Modell konstruktiv vermieden durch Definition des Labels als n-elementigen Vektor $[i_1, i_2, ..., i_n]$ (n ist die Anzahl der Konfliktklassen), wobei jede Stelle des Vektors genau einer Konfliktklasse zugeordnet ist und somit der Vektor maximal ein Element je Konfliktklasse enthalten kann. Hat ein Objekt noch kein Datum aus einer Konfliktklasse gelesen, so wird dies an der entsprechenden Vektorstelle mit ε markiert – dem Symbol für das Infimum der Klassenmenge. Es gilt also für alle $i_k, 1 \leq k \leq n : i_k \in C_k$ oder $i_k = \varepsilon$. Damit ergibt sich als Labelmenge des obigen Chinese-Wall-Beispiels:

$$\textbf{(1)}\ labels = \{[\varepsilon,\varepsilon], [Bank_1,\varepsilon], [Bank_2,\varepsilon], [\varepsilon,\ddot{O}l_1], [\varepsilon,\ddot{O}l_2], \\ [Bank_1,\ddot{O}l_1], [Bank_1,\ddot{O}l_2], [Bank_2,\ddot{O}l_1], [Bank_2,\ddot{O}l_2], SysHi\}.$$

$SysHi$ vervollständigt den Verband und repräsentiert das Supremum; dieses Label charakterisiert ein Objekt, zu dem sämtliche Informationen geflossen sind und welches es gemäß der Sicherheitspolitik nicht geben darf; dieses Label darf somit niemals einem Objekt zugewiesen werden (vgl. auch die Invariante der Spezifikation in Bild 3.8). Die Menge der Unternehmen und die hieraus abgeleitete Labelmenge definiert die erste von sechs Komponenten einer Skippy-Deklaration.

Festlegung der Halbordnung

Als nächsten Schritt wird die Halbordnung „$\sqsubseteq$" über der Labelmenge wie folgt definiert:

$$\textbf{(2)}\ \forall l_1, l_2 \in labels:\ l_1 \sqsubseteq l_2 \ \Leftrightarrow\ \forall 1 \leq k \leq n : l_1[k] = l_2[k] \vee l_1[k] = \varepsilon,$$

wobei $l_1[k]$ das k-te Element des n-elementigen Vektors l_1 bezeichnet.

Diese Halbordnung besagt, daß Informationen von einem Objekt der ersten Klasse (mit Label l_1) genau dann in ein Objekt der zweiten Klasse (mit Label l_2) fließen dürfen,

wenn die Labelvektoren überall dort übereinstimmen, wo in der ersten Klasse eine Vektorstelle $\neq \varepsilon$ ist, oder, mit anderen Worten, zu einem Objekt nur solche Informationen hinzukommen dürfen, die entweder aus den Datensätzen bereits bearbeiteter Firmen stammen oder nicht aus einer Konfliktklasse stammen, aus der bereits Informationen im Objekt vorliegen. Die Anwendung von (2) auf (1) führt dann zur Festlegung der Halbordnung in der graphischen Darstellung in Bild 3.7 und der entsprechenden Definition von $flow$ in der Skippy-Deklaration in Bild 3.8.

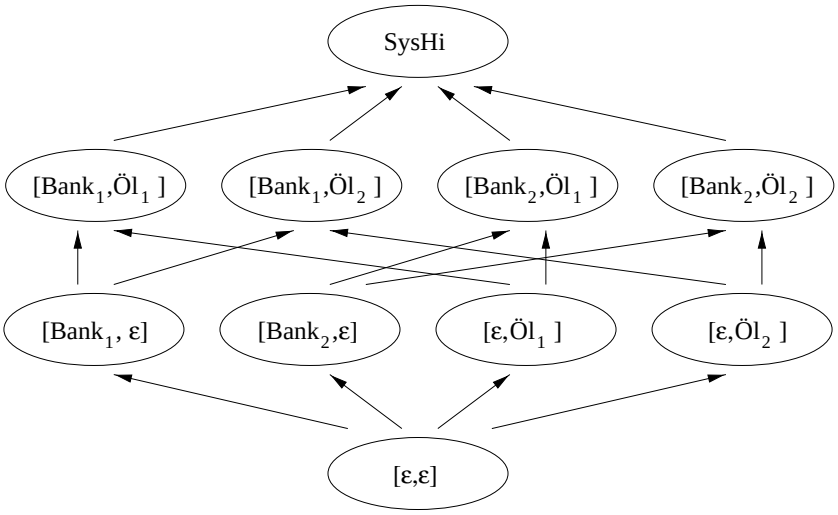


Bild 3.7: Verband der Chinese-Wall-Politik

Zuordnung der Verbandklassen zu den Entitäten der Applikation

Im dritten Schritt wird nun die Zuordnung der Verbandklassen zu den Entitäten der Applikation vorgenommen. Damit der Verband anwendbar wird, werden sämtliche Entitäten der Applikation – die Firmendaten und die Repräsentantenprozesse der Berater – mit Labeln versehen. Gemäß der allgemeinen Regel, daß jede Entität als Label die Menge derjenigen Unternehmen erhält, über die die Entität Daten enthält, erhalten die Firmendaten des Beispiels eines der Label $[Bank_1, \varepsilon], [Bank_2, \varepsilon], [\varepsilon, \ddot{O}l_1]$ oder $[\varepsilon, \ddot{O}l_2]$[1]. Würde im Beispiel jedes Unternehmen genau ein Datenobjekt besitzen, so ergeben sich 4 Datenobjekte $e_1 ... e_4$ mit der Labelzuordnung

$$
\begin{aligned}
cl(e_1) &= [Bank_1, \varepsilon] \\
cl(e_2) &= [Bank_2, \varepsilon] \\
cl(e_3) &= [\varepsilon, \ddot{O}l_1] \\
cl(e_4) &= [\varepsilon, \ddot{O}l_2].
\end{aligned}
$$

[1] wobei hier vorausgesetzt wird, daß ein Firmendatendokument Informationen bzgl. nur eines Unternehmens enthält

Die Repräsentantenprozesse der Berater verfügen zu Beginn über keinerlei Informationen; nimmt man im Beispiel zwei Berater e_5 und e_6 an, so entspricht dies der initialen Zustandsbelegung

$$cl(e_5) = [\varepsilon, \varepsilon]$$
$$cl(e_6) = [\varepsilon, \varepsilon].$$

Im Laufe ihres Lebens lesen die Repräsentantenprozesse dann Firmendaten und erhalten dabei die entsprechenden neuen Label. Hat ein Repräsentantenprozeß beispielsweise das Label $[Bank_1, \varepsilon]$ und greift hiernach auf $\ddot{O}l_1$ zu, so erhält er dabei das neue Label $[Bank_1, \ddot{O}l_1]$. Repräsentantenprozesse wandern also im Laufe ihres Lebens im Verband „nach oben", wodurch ihre Eigenschaft deutlich wird, mit der Zeit immer mehr Informationen zu enthalten. Die Reklassifikation von Repräsentantenprozessen werden in der Politikspezifikation durch die politikspezifische Operation *chgcl* durchgeführt (s.u.).

Entitäten mit dem Label *SysHi* verstoßen gegen die Sicherheitspolitik; sie enthalten beliebige Kombinationen von Firmendaten, unter anderem solche, die gemäß der Konfliktrelation nicht zusammengeführt werden dürfen. Dies gilt global für die gesamte Sicherheitspolitik und ist in der Invarianten formuliert.

Festlegung der Zugriffsrechte-Namen

Die Namen der Zugriffsrechte werden zur Konstruktion der Matrix **m** (vgl. Abschnitt 3.4.2) benötigt. Neben den Rechten, die applikationsspezifischen und politikspezifischen Operationen aufzurufen, benötigt das Beispiel keine weiteren Rechtedefinitionen.

Festlegung der Politikregeln

Die letzte Komponente der Spezifikation ist die Beschreibung der Regeln für die Ausführung der durch die Sicherheitspolitik kontrollierten Operationen. Für applikationsspezifische Operationen besteht diese aus den zwei Komponenten *op* und *pre. op* macht die Festlegung der verbandbasierten Zugriffsregeln explizit: zu jeder Operation (z.B. *read*) gehört eine verbandbasierte Zugriffsregel ($\leq_{read}$), deren Semantik mittels *op* spezifiziert wird. Für politikspezifische Operationen kommt die Definition der Semantik der Operation mittels der *where*-Klausel hinzu.

In dieser Klausel und ebenfalls im Initialisierungsteil einer Spezifikation kommen Ausdrücke vor, die in Skippy in einer Z-artigen Notation formuliert sind [Dil90]. Der Z-Ausdruck

$$\{i : \mathcal{N} | 1 \leq i \leq 6 \bullet e_i\}$$

besteht aus drei Teilen: der Z-Deklaration $i : \mathcal{N}$, der Z-Formel $1 \leq i \leq 6$ und dem Z-Term e_i. Er bezeichnet eine „Menge aller e_i, wobei i eine natürliche Zahl zwischen 1 und

6 ist"; in der gebräuchlichen mathematischen Notation entspricht dies der Schreibweise $\{e_i | i \in \mathcal{N} \ \wedge \ 1 \leq i \leq 6\}$.

„$\oplus$" ist der „*overriding*"-Operator aus Z; ist beispielsweise cl eine Abbildung $ENT \rightarrow CLASS$ und gilt wie in obiger Klassenfestlegung

$$\begin{aligned} cl(e_1) &= [Bank_1, \varepsilon] \\ cl(e_2) &= [Bank_2, \varepsilon] \\ cl(e_3) &= [\varepsilon, \ddot{O}l_1] \\ cl(e_4) &= [\varepsilon, \ddot{O}l_2], \end{aligned}$$

so bewirkt $cl' = cl \oplus \{e_1 \mapsto [\varepsilon, \varepsilon]\}$, daß anschließend

$$\begin{aligned} cl(e_1) &= [\varepsilon, \varepsilon] \\ cl(e_2) &= [Bank_2, \varepsilon] \\ cl(e_3) &= [\varepsilon, \ddot{O}l_1] \\ cl(e_4) &= [\varepsilon, \ddot{O}l_2] \end{aligned}$$

gilt.

Die linke Seite der Formel ist dabei apostrophiert; diese Notation repräsentiert den durch die Formel bewirkten Zustandsübergang, wobei cl die Abbildung vor dem Zustandswechsel und cl' die Abbildung nach dem Zustandswechsel bezeichnet.

Die folgende, vollständige Spezifikation des Chinese-Wall-Beispiels faßt dies alles zusammen. Zur besseren Lesbarkeit sind Schlüsselworte der Sprache, vordefinierte Variablen und Funktionen fett dargestellt.

Policy ChineseWall =

Policy-Sets:

$companies = \{Bank_1, Bank_2, \ddot{O}l_1, \ddot{O}l_2\};$

$labels = \{[\varepsilon,\varepsilon], [Bank_1,\varepsilon], [Bank_2,\varepsilon], [\varepsilon,\ddot{O}l_1], [\varepsilon,\ddot{O}l_2],$
$[Bank_1,\ddot{O}l_1], [Bank_1,\ddot{O}l_2], [Bank_2,\ddot{O}l_1], [Bank_2,\ddot{O}l_2], SysHi\};$

Classification:

$flow : \{[\varepsilon,\varepsilon] \sqsubseteq [Bank_1,\varepsilon], [\varepsilon,\varepsilon] \sqsubseteq [Bank_2,\varepsilon], [\varepsilon,\varepsilon] \sqsubseteq [\varepsilon,\ddot{O}l_1], [\varepsilon,\varepsilon] \sqsubseteq [\varepsilon,\ddot{O}l_2],$
$[Bank_1,\varepsilon] \sqsubseteq [Bank_1,\ddot{O}l_1], [Bank_1,\varepsilon] \sqsubseteq [Bank_1,\ddot{O}l_2],$
$[Bank_2,\varepsilon] \sqsubseteq [Bank_2,\ddot{O}l_1], [Bank_2,\varepsilon] \sqsubseteq [Bank_2,\ddot{O}l_2],$
$[\varepsilon,\ddot{O}l_1] \sqsubseteq [Bank_1,\ddot{O}l_1], [\varepsilon,\ddot{O}l_1] \sqsubseteq [Bank_2,\ddot{O}l_1],$
$[\varepsilon,\ddot{O}l_2] \sqsubseteq [Bank_1,\ddot{O}l_2], [\varepsilon,\ddot{O}l_2] \sqsubseteq [Bank_2,\ddot{O}l_2],$
$[Bank_1,\ddot{O}l_1] \sqsubseteq SysHi, [Bank_1,\ddot{O}l_2] \sqsubseteq SysHi,$
$[Bank_2,\ddot{O}l_1] \sqsubseteq SysHi, [Bank_2,\ddot{O}l_2] \sqsubseteq SysHi\};$

Rights:

$\{read, write, chgcl\};$

Oper:

$read(s,o)$:

op: $a \leq_{read} b = \mathbf{cl}(b) \sqsubseteq \mathbf{cl}(a);$

pre: $s \leq_{read} o \wedge (read \in \mathbf{m}(s,o));$

$write(s,o)$:

op: $a \leq_{write} b = \mathbf{cl}(a) \sqsubseteq \mathbf{cl}(b);$

pre: $s \leq_{write} o \wedge (write \in \mathbf{m}(s,o));$

$chgcl(s,o,c)$:

op: $a \leq_{chgcl} b = \mathbf{cl}(a) \sqsubseteq \mathbf{cl}(b);$

where: $cl' = cl \oplus \{o \mapsto cl(c)\};$

pre: $o \leq_{chgcl} c \wedge (chgcl \in \mathbf{m}(s,o));$

Inv:

$\forall o : \mathbf{entities} \,.\, \mathbf{cl}(o) \sqsubseteq SysHi \wedge \mathbf{cl}(o) \neq SysHi;$

State:

$\mathbf{entities}' = \{i : \mathcal{N} \mid 1 \leq i \leq 6 \bullet e_i\};$

$\mathbf{cl}' = \{i : N \mid 1 \leq i \leq 6 \bullet (\ e_1 \mapsto [Bank_1,\varepsilon],$
$e_2 \mapsto [Bank_2,\varepsilon],$
$e_3 \mapsto [\varepsilon,\ddot{O}l_1],$
$e_4 \mapsto [\varepsilon,\ddot{O}l_2],$
$e_5 \mapsto [\varepsilon,\varepsilon],$
$e_6 \mapsto [\varepsilon,\varepsilon])\};$

$\mathbf{m}' = \{(i,j) : \mathcal{N} \times \mathcal{N} \mid 1 \leq i,j \leq 6 \bullet (e_i,e_j) \mapsto \{read, write, chgcl\}\};$

End ChineseWall.

Bild 3.8: Die Chinese-Wall-Politik als Skippy-Deklaration

3.4.4 Die Syntax von Skippy

Die Definition der Syntax von Skippy erfolgt in der Backus-Naur Form; ≺Ident≻ ist dabei ein wohlgeformter Identifier in der Politik, dessen konkretes Aussehen hier nicht weiter spezifiziert zu werden braucht; das leere Produktionssymbol ist „e".

Identifier, Ausdrücke, etc.

≺Ident_list≻ :== ≺Ident≻ ≺I_list≻ ;

≺I_list≻ :== "," ≺Ident_list≻ | e ;

Politik – Deklaration

≺POL≻ :== "Policy" ≺Ident≻ "=" ≺pol_decl≻ "End" ≺Ident≻ "." ;

≺pol_decl≻ :== "Policy-Sets:" ≺pol_sets_decl≻
"Classification:" ≺class_decl≻
"Rights:" ≺rights_decl≻
"Oper:" ≺oper_decl≻
"Inv:" ≺inv_decl≻
"State:" ≺state_decl≻ ;

Politikmengen – Deklaration

≺pol_sets_decl≻ :== ≺Ident≻ "={" ≺Ident_list≻ "};" ≺ps_decl≻ ;

≺ps_decl≻ :== ≺pol_sets_decl≻ | e ;

Politikklassen – Deklaration

≺class_decl≻ :== ≺Ident≻ ":{" ≺class_list≻ "};" ;

≺class_list≻ :== ≺Ident≻ "⊑" ≺Ident≻ ≺c_list≻ ;

≺c_list≻ :== "," ≺class_list≻ | e ;

Rechte – Deklaration

≺rights_decl≻ :== "{" ≺Ident_list≻ "};" ;

Operationen – Deklaration

≺oper_decl≻ :== ≺operation≻ ";" ≺op_list≻ ;

≺op_list≻ :== ≺oper_decl≻ | e ;
≺operation≻ :== ≺Ident≻ "(" ≺Ident_list≻ "):"
≺op_clause≻
≺where_clause≻
≺pre_clause≻ ;
≺op_clause≻ :== "op:" ≺Ident≻ "≤" ≺Ident≻ ≺Ident≻ "=" ≺B_Exp≻ ";" ;
≺where_clause≻ :== "where:" ≺Z_function_list≻ | e ;
≺Z_function_list≻ :== ≺ent_fn≻ ≺cl_fn≻ ≺m_fn≻ ;
≺ent_fn≻ :== "entities' = " ≺ent_struct≻ ";" | e ;
≺cl_fn≻ :== "cl' = " ≺cl_struct≻ ";" | e ;
≺m_fn≻ :== "m' = " ≺m_struct≻ ";" | e ;
≺pre_clause≻ :== "pre:" ≺B_Exp≻ ";" ;

Invarianten – Deklaration

≺inv_decl≻ :== ≺B_Exp≻ ";" ;

Zustands – Deklaration

≺state_decl≻ :== ≺Z_function_list≻ ;

Das Nichtterminalsymbol ≺B_Exp≻ steht für einen wohlgeformten boole'schen Ausdruck im Prädikatenkalkül erster Ordnung, in dem ausschließlich Variablen der Sicherheitspolitik mittels $\forall, \exists, \wedge, \vee, \Rightarrow$ etc. verknüpft sind. Die Schreibweise wohlgeformter Ausdrücke erfolgt in Anlehnung an [Man74], wobei ebenfalls die dortige verkürzte Form („$\forall x$" statt „$(\forall x)$", „$x \neq y$" statt „$\neg(x = y)$" etc. immer dann verwendet wird, wenn dies zu keinen Mißverständnissen führen kann. Weiterhin sind die Nichtterminalsymbole ≺ent_struct≻, ≺cl_struct≻ und ≺m_struct≻ nicht weiter definiert. ≺ent_struct≻ ist die Menge aller wohlgeformten Mengenausdrücke vom Typ $\wp(ENT)$, ≺m_struct≻ ist die Menge aller Abbildungen vom Typ $ENT \times ENT \rightarrow \wp(R)$, und ≺cl_struct≻ ist die Menge aller Abbildungen vom Typ $ENT \rightarrow CLASS$.

Damit ist die Syntax von Skippy und informell an einem Beispiel auch die Semantik beschrieben. Ziel war die Vorstellung des durch Skippy definierten festen Rahmens, innerhalb dessen durch Struktur und Typsystem die Formulierungsmöglichkeiten auf die essentiellen Modellparadigmen eingeschränkt sind und der eine direkte Formulierung der im vorangehenden Abschnitt beschriebenen Modellabstraktionen ermöglicht.

In diesem Kapitel spielt Skippy die Rolle eines Werkzeugs zur Spezifikation von Sicherheitsmodellen. Kapitel 5 wird die Konzepte von Skippy aufgreifen, um hierauf eine Spezifikationssprache für Metapolitikmodelle aufzubauen.

3.5 Implementierung von Sicherheitspolitiken

Dieser Abschnitt beschreibt den letzten Schritt bei der Entwicklung einer Sicherheitspolitik: die Transformation einer Politikspezifikation in ausführbaren Code und seine Einbettung in eine geeignete Sicherheitsarchitektur. Die hier beschriebenen Prinzipien sind die Grundlage des in Kapitel 6 entwickelten Konzepts zur Implementierung von Metapolitiken.

Bei den nachfolgenden Überlegungen wird das folgende *computational model* zugrunde gelegt: ein IT-System verfügt über eine Reihe von Anwendungssystemen, die nebeneinander koexistieren oder auch miteinander kooperieren können. Jedes Anwendungssystem kann wiederum aus einer Vielzahl einzelner Entitäten bestehen, die im Kontext ihres Anwendungssystems ebenfalls miteinander interagieren können. Informationssicherheit in diesem Szenarium wird durch Sicherheitspolitiken hergestellt, deren Domänen ein gesamtes IT-System, Gruppen von Anwendungssystemen oder auch einzelne Entitäten verschiedener Anwendungssysteme umfassen können. Anwendungssysteme, Sicherheitspolitiken und ihre Durchsetzungsmechanismen sind streng voneinander getrennt [LCC+75, GHF89, Min95, OFSS96, WWK96]; insbesondere sind Sicherheitspolitiken in diesem Kontext gekapselte, manipulationsgeschützte Entitäten, welche durch die Sicherheitsarchitektur ihrer Umgebung in die Lage versetzt werden, sicherheitsrelevante Ereignisse (wie Interaktionen von Entitäten) zu beobachten, zu autorisieren und auf diese Weise ihr Regelwerk durchzusetzen.

Als Konzept zur Implementierung von Sicherheitspolitiken wird in diesem Kapitel das *Custodian*-Paradigma vorgestellt, welches erstmalig in [Küh95a, Küh95b] beschrieben wurde. Dieses Konzept zeichnet sich durch eine hohe Ausdruckskraft hinsichtlich der zu implementierenden Sicherheitspolitik aus: in Skippy spezifizierte und mittels eines Transformationswerkzeugs [KP99] in eine C++ - Klasse übersetzte Sicherheitspolitiken lassen sich als ausführbare Programme mittels einer Custodian-Kapsel in eine Sicherheitsarchitektur integrieren und werden auf diese Weise mit einer geeigneten Ablaufumgebung versehen.

Die Universalität der Custodians wird es im weiteren Verlauf der Arbeit ermöglichen, die Verwandtschaft von Sicherheitspolitik und Metapolitik ein weiteres Mal zu nutzen und Custodians als Ausgangspunkt eines Implementierungsparadigmas für Metapolitiken zu verwenden.

3.5.1 Ausgangspunkt

Ausgangspunkt eines Implementierungskonzepts für Sicherheitspolitiken sind die *Referenzmonitorprinzipien*, ein klassisches und erfolgreiches Architekturkonzept zur Implementierung von Zugriffssteuerungspolitiken, welches bereits 1983 in den *Trusted Computer System Evaluation Criteria* [DoD83] beschrieben wurde. Dieses Architekturkonzept definiert eine Systemarchitektur, in der eine zentrale Komponente, der Refe-

renzmonitor, sämtliche Interaktionen zwischen den Entitäten eines Systems überwacht. Drei Designkriterien werden für einen Referenzmonitor gefordert:

1. Er muß *sämtliche* Zugriffe von Subjekten auf Objekte überwachen und beeinflussen können (vollständige Kommunikationskontrolle).

2. Er muß vor Manipulationen geschützt sein.

3. Er muß klein und wohlstrukturiert sein, so daß Verfahren zur Analyse seiner Korrektheit anwendbar sind.

Die Menge aller Komponenten eines Betriebssystems, welche einen Referenzmonitor implementieren, wird als der *Sicherheitskern* eines Betriebssystems bezeichnet. Sicherheitskerne haben sich als recht erfolgreich innerhalb ihrer abgesteckten Domäne – der Implementierung einer einfachen Zugriffssteuerungspolitik innerhalb eines geschlossenen Systems – erwiesen. Im Szenarium dieser Arbeit werden jedoch Anforderungen an ein Implementierungskonzept für Sicherheitspolitiken gestellt, die zum Zeitpunkt der Aufstellung der Referenzmonitorprinzipien nicht absehbar waren. Weder war es beabsichtigt noch sind klassische Sicherheitskerne in der Lage, Multipolitiken-Szenarien zu implementieren; die gleichzeitige Integration vieler Sicherheitspolitiken in einen einzigen Sicherheitskern erfordert innerhalb des Kerns Mechanismen, die die Manipulationssicherheit zwischen den einzelnen Sicherheitspolitiken selbst gewährleisten; Multipolitiken-Szenarien erfordern also eine Revision des Sicherheitskern-Konzepts.
Weiterhin machen bereits Zugriffssteuerungspolitiken etwas anspruchsvollerer Art weitere Grenzen des Referenzmonitorkonzepts deutlich: die Forderung, daß ein Sicherheitskern *sämtliche* zur Auswertung der Regeln notwendigen Algorithmen und Daten umfassen muß, läßt lediglich ein sehr einfaches Autorisierungsschema zu. Zugriffsregeln mit Kriterien wie der Systemzeit oder dem Vorhandensein einer digitalen Signatur unter einem Vertragsdokument würden dagegen zu einer Sicherheitskerngröße führen, bei der das dritte Designkriterium nicht mehr erfüllbar ist.

Diese Unzulänglichkeiten der Sicherheitskerne als Implementierungskonzept für Multipolitiken-Szenarien waren die Motivation für die Entwicklung der *Custodians* [Küh95a, Küh95b]. Das Custodian-Paradigma beruht auf einer strengen Trennung zwischen einer architekturunabhängigen Sicherheitspolitik auf der einen Seite und einer politikunabhängigen Sicherheitsarchitektur auf der anderen Seite; Sicherheitspolitiken werden einerseits architekturunabhängig als abstrakte Datentypen [CW85] formuliert; andererseits stellt eine politikunabhängige Sicherheitsarchitektur eine geeignete Ablaufumgebung für so formulierte Sicherheitspolitiken bereit. Der folgende Abschnitt erläutert zunächst die Abbildung von Skippy-Spezifikationen auf die Strukturen abstrakter Datentypen; anschließend wird dann exemplarisch eine Sicherheitsarchitektur zur Implementierung so formulierter Sicherheitspolitiken vorgestellt.

3.5.2 Skippy-Spezifikationen als abstrakte Datentypen

Die Implementierung von Skippy-Spezifikationen ist eng mit dem Prinzip der abstrakten Datentypen (ADTs) verbunden, welches in unterschiedlichen Ausprägungen in einer Vielzahl heutiger Programmiersprachen anzutreffen ist; abstrakte Datentypen heißen dort beispielsweise *Klassen* oder *Module*. Allen diesen Ausprägungen gemeinsam ist der Grundgedanke, Daten und hierauf definierte Operationen zu kapseln und den Zugriff auf die Daten ausschließlich über explizit aus der Kapsel exportierte Operationen zu ermöglichen.

Die Abbildung einer Skippy-Spezifikation auf dieses Paradigma ist auf Grund der bereits datentypähnlichen Struktur einer Skippy-Spezifikation sehr geradlinig:

- sämtliche den Zustand einer Sicherheitspolitik beschreibenden Skippy-Elemente – die Entitätenmenge, die Zugriffsmatrix, der Verband und die Klassifikationsfunktion – bilden den Datenteil eines ADTs
- alle anderen Elemente einer Skippy-Spezifikation – Zugriffsregeln, Invarianten, Initialisierung – bilden den Operationsteil eines ADTs.

Die Transformation einer Skippy-Spezifikation in die konkreten Strukturen einer konkreten Programmiersprache ist weitgehend automatisierbar; ein Beispiel dafür ist das im Kontext der in [KvKO95] beschriebenen Politiken-Entwicklungsumgebung entwickelte Werkzeug [KP99], welches eine Skippy-Spezifikation in eine C++ - Klasse transformiert. Das Ergebnis einer solchen automatischen Transformation zeigt Bild 3.9 auf den Seiten 65 und 66; Ausgangspunkt hierfür war die Skippy-Spezifikation in Bild 3.8. Zur größeren Klarheit wurden in Bild 3.9 lediglich einige Details manuell entfernt.

```
class ChineseWall: public GenericDCECustodian {
private:
    /* policy sets: labels
    */
    enum label_t {SysLow, bank1, bank2, oel1, oel2,
                  bank1oel1, bank1oel2, bank2oel1, bank2oel2, SysHigh};

    /* classification: the lattice operator fl, the classification operator cl
    */
    bool fl [SysHigh][SysHigh];
    label_t cl [MaxEntities];

    /* rights: set of rights, access matrix
    */
    enum Rights {none=0, read_r=1, write_r=2, chgcl_r=4};
    Rightset_t m [MaxEntities][MaxEntities];

    /* operations:
    */
    bool read(int s, int o)
    {   CheckInvariant();
        return(fl[cl[o]][cl[s]] && (read_r & m[s][o]));
    }

    bool write(int s, int o)
    {   CheckInvariant();
        return(fl[cl[s]][cl[o]] && (write_r & m[s][o]));
    }

    bool chgcl(int s, int o, int c)
    {   CheckInvariant();
        if (fl[cl[o]][cl[c]] && (chgcl_r & m[s][o]))
        {   cl[o] = cl[c];
            return(true);
        }
        return(false);
    }

    /* invariant
    */
    void CheckInvariant()
    { int i;
        for (i=0;i++;i<6) if (cl[i] == SysHigh) /* raise policy invariant violation exception */;
    }
```

```
    /* initialisations
    */
    void init_m()
    { int i,j;
        for (i=0;i++;i<MaxEntities)
            for (j=0;j++;j<MaxEntities)
                m[i][j] = none;
    }

    void init_cl()
    { int i;
        for (i=0;i++;i<MaxEntities) cl[i] = no_label;
    }

    void init_fl()
    { int i,j;
        for (i=0;i++;i<SysHigh)
            for (j=0;j++;j<SysHigh)
                fl[i][j] = false;
    }

    /* class instance constructor
    */
    ChineseWall()
    {   int s,o;
        init_cl();
        cl[1] = bank1; cl[2] = bank2; cl[3] = oel1; cl[4] = oel2; cl[5] = SysLow; cl[6] = SysLow;

        init_fl();
        fl[SysLow][bank1] = true; fl[SysLow][bank2] = true;
        fl[SysLow][oel1] = true; fl[SysLow][oel2] = true;
        fl[bank1][bank1oel1] = true; fl[bank1][bank1oel2] = true;
        fl[bank2][bank2oel1] = true; fl[bank2][bank2oel1] = true;
        fl[oel1][bank1oel1] = true; fl[oel1][bank2oel1] = true;
        fl[oel2][bank1oel2] = true; fl[oel2][bank2oel2] = true;
        fl[bank1oel1][SysHigh] = true; fl[bank1oel2][SysHigh] = true;
        fl[bank2oel1][SysHigh] = true; fl[bank2oel2][SysHigh] = true;

        init_m();
        for (s=0;s++;s<6)
            for (o=0;o++;o<6)
                m[s][o] = read_r | write_r | chgcl_r;
    }
};
```

Bild 3.9: Die Chinese-Wall-Politik als ausführbare C++ - Klasse

Der gesamte Vorgang der automatischen Abbildung einer Skippy-Spezifikation in eine ausführbare Sicherheitspolitik ist abschließend in Bild 3.10 dargestellt; am Ende dieses Prozesses steht ein Custodian, der in seiner Kapsel eine konkrete Sicherheitspolitik birgt und diese Sicherheitspolitik mit einer zu einer konkreten Sicherheitsarchitektur passenden Hülle versieht. Wie diese Hülle beschaffen ist, zeigt das nächste Kapitel.

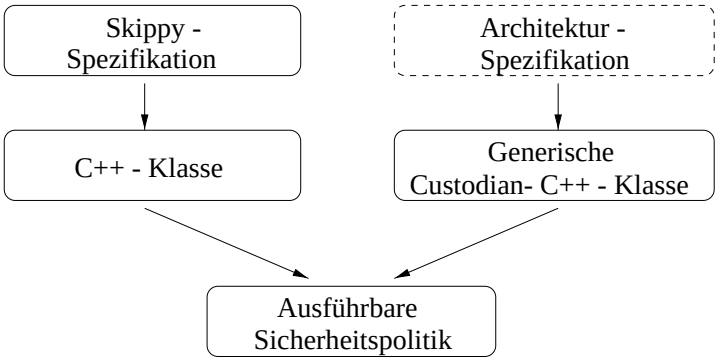


Bild 3.10: Transformation einer Spezifikation in eine ausführbare Sicherheitspolitik

3.5.3 Eine Sicherheitsarchitektur für Custodians

Aufgabe einer Sicherheitsarchitektur ist die Herstellung sämtlicher Voraussetzungen, die zu einer Durchsetzung von Sicherheitspolitiken notwendig sind. Konkret bedeutet dies hier, daß eine Sicherheitsarchitektur zur Durchsetzung des Custodian-Paradigmas die folgenden Eigenschaften herstellen muß:

1. sie muß die vollständige Kommunikationskontrolle gewährleisten
2. sie muß die Manipulationssicherheit der Sicherheitspolitik gewährleisten
3. sie muß die Politikpersistenz gewährleisten
4. sie muß eine unmanipulierbare Zuordnung zwischen den Sicherheitspolitiken und den zugeordneten Applikationsentitäten herstellen.

Eine Sicherheitsarchitektur ist stets Teil einer Systemarchitektur; ohne Bezug auf die konkrete Gesamtarchitektur eines IT-Systems muß eine Beschreibung der Sicherheitsarchitektur vage bleiben. Daher soll in diesem Kapitel beispielhaft eine Sicherheitsarchitektur für das *Distributed Computing Environment – DCE* der *Open Systems Foundation (OSF)* [OSF92] vorgestellt werden [Lux95, Wag96]; ein weiteres Beispiel für eine industriell genutzte Sicherheitsarchitektur für Custodians ist in [BKAL97] beschrieben.

3.5.4 Ein Beispiel: Chinese-Wall als C++ - Custodian

Die *OSF DCE* ist eine Plattform für verteilte Applikationen in heterogenen Rechnernetzen. Ihre Aufgabe ist es, physische Verteilung und Heterogenität eines verteilten Systems weitgehend zu verdecken und ein homogenes, am Klient/Server-Modell orientiertes Programmierparadigma für verteilte Applikationen herzustellen.

Die DCE bietet sich in mehrfacher Hinsicht als Fallstudie an. Zum einen ist sie auf dem Wege zu einem Industriestandard und hat sich somit mit vielen Details und Fußangeln realer Systeme auseinanderzusetzen, von denen ein rein akademischer Prototyp abstrahieren darf. Eine Sicherheitsarchitektur für eine solche Umgebung kann daher auf einer sehr konkreten Ebene mit hohem Detailreichtum studiert werden. Zum anderen sind die Standardisierungsbestrebungen der Industrie ein Hinweis auf eine in Zukunft größere Verbreitung der DCE, so daß Arbeiten auf der Grundlage dieser Plattform eine gewisse Praxisrelevanz erhalten.

Hinzu kommt ein weiterer Aspekt, der es erlaubt, einen Teil der Aufgaben der Sicherheitsarchitektur auf die DCE zu verlagern: die Paradigmen der DCE definieren für alle Applikationen einen festen Rahmen, durch den bereits ein Teil der Aufgaben der Sicherheitsarchitektur erledigt wird. Die DCE verfügt über

- das bereits genannte Klient/Server - Paradigma, welches den DCE-Applikationssystemen bereits eine Subjekt/Objekt - Struktur bei der Kommunikation aufprägt
- ein Kapselungsschema für Klienten und Server
- ein wohldefiniertes Kommunikationsparadigma zwischen Klient und Server.

Im folgenden soll nun beschrieben werden, wie in dieser Umgebung die genannten Sicherheitsarchitektur-Eigenschaften (Gewährleistung der Manipulationssicherheit der Sicherheitspolitik und der vollständigen Kommunikationskontrolle, die Politikpersistenz und die Bindung der Applikationsentitäten an die Politik) implementiert wird. Als Fallbeispiel wird erneut die bereits eingeführte Chinese-Wall-Politik verwendet.

Die Beispielapplikation (Bild 3.11) bestehe aus einem DCE-Server (dem „*Firmendaten-Server*"), welcher die Firmendatenobjekte verwaltet. Die Berater seien als DCE-Klienten implementiert, die über die Server-Operationen *read* und *write* auf Firmendokumente zugreifen. Der Chinese-Wall-Custodian aus Bild 3.9 sei als eigenständige, gekapselte Einheit an den Server angekoppelt und soll die Sicherheitspolitik durchsetzen, indem er eine vollständige Kontrolle der Kommunikation des Servers ausübt.

Manipulationssicherheit der Sicherheitspolitik

Die Manipulationssicherheit von Custodians wird durch direkte Nutzung des DCE-Serverparadigmas erreicht, indem Custodians als DCE-Server realisiert werden. DCE-

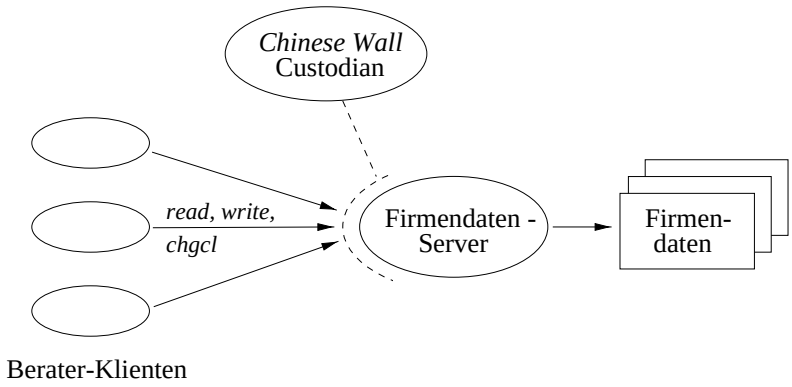


Bild 3.11: Das Chinese-Wall-Szenarium als DCE-Anwendung

Server sind gekapselte Einheiten, die ausschließlich über ihre Schnittstelle angesprochen werden können und deren Manipulationssicherheit durch die Mechanismen des jeweiligen Basisbetriebssystems hergestellt werden. So sind beispielsweise auf Unix-Plattformen DCE-Server als Unix-Prozesse implementiert, deren Unverletzlichkeit direkt auf der Fähigkeit des Betriebssystems basiert, Manipulationen auf Prozessen rigorosen Schutzmechanismen zu unterwerfen.

Durch Implementation einer Sicherheitspolitik als DCE-Server wird somit ein Grad an Manipulationssicherheit erreicht, wie er allgemein für alle DCE-Server erreicht ist. Gelingt es, diesen einen Mechanismus zu umgehen, so brechen gleichzeitig die Integrität der Server und der Sicherheitspolitiken zusammen.

Deutlich wird hiermit, welchen Stellenwert Speicher- und Prozeßverwaltung des Basisbetriebssystems einnehmen; sie bilden einen unabdingbaren Bestandteil der *Trusted Computing Base (TCB)*, worunter die Summe aller Komponenten eines IT-Systems verstanden wird, welche an der Durchsetzung einer Sicherheitspolitik beteiligt sind und denen somit in Hinblick auf ihre Korrektheit Vertrauen entgegen gebracht werden muß. Offensichtlich ist, daß monolithisch strukturierte Betriebssysteme an dieser Stelle große Schwierigkeiten mit der Erfüllung des dritten Referenzmonitorprinzips haben; Chancen diesbezüglicher Technologiefortschritte dürften daher eher bei den mikrokernbasierten Betriebssystemarchitekturen liegen.

Kontrolle der Kommunikation

Die vollständige Kontrolle der Kommunikation der Applikation durch die Sicherheitspolitik ist für DCE-Applikationen dann zu erreichen, wenn DCE-Anwendungen ihre Kommunikation ausschließlich mittels der DCE Fernaufrufmechanismen (DCE-RPC) abwickeln. Dies ist jedoch eine bloße Frage der Einhaltung von Spielregeln: es ist z.B. in einer Unix-Implementierung der DCE nicht verhinderbar, daß ein Klient Zugriffsversu-

che auf die Firmendateien unmittelbar über die Dateisystemschnittstelle des Basisbetriebssystems unternimmt. An dieser Stelle müssen die traditionellen Zugriffsschutzmechanismen des Dateisystems greifen, die somit wie auch die Speicher- und Prozeßverwaltung des Basisbetriebssystems einen unabdingbaren Bestandteil der *TCB* bilden; erneut ergibt sich hieraus die obige Kritik an monolithischen Betriebssystemarchitekturen.

Ebenfalls eine Frage der Einhaltung von Spielregeln ist die Form der Kommunikation zwischen Klient und Server. Es ist in einer Unix-Implementierung der DCE nicht verhinderbar, daß Klient und Server sich der regulären Unix-Systemfunktionen bedienen, um an den Kommunikationsmechanismen der DCE vorbei mittels Dateien, *pipes* oder *sockets* andere Informationsflüsse zu etablieren. Ein hierauf basierendes Umgehen einer Sicherheitspolitik setzt allerdings voraus, daß Klient und Server hier gemeinsame Sache machen. Hält sich ein Server an die Spielregeln und betreibt neben der über DCE-Mechanismen keine weitere Form der Kommunikation, so laufen entsprechende Versuche eines Klienten ins Leere.

Die Annahme eines solchen Wohlverhaltens ist in der Regel zumindest für die Server zulässig, da Server ohnehin die von ihnen verwalteten Objekte in Hinblick auf ihre Verfügbarkeit und, sofern von den Klienten keine eigenen kryptographischen Mechanismen zur Verschlüsselung der Objekte eingesetzt werden, auch in Hinblick auf ihre Vertraulichkeit und Integrität kontrolliert werden. Sie muß nur dann (etwa durch Programmcodeanalyse) untermauert werden, wenn ein á priori nicht vertrauenswürdiger Server mit einer obligatorischen Sicherheitspolitik versehen werden soll.

Unter der Annahme des Wohlverhaltens ist die Technik zum Erreichen vollständiger Kommunikationskontrolle das Aufbrechen des DCE Kommunikationsmechanismusses in einer Weise, daß jegliche Kommunikation zwischen DCE-Klient und DCE-Server über die Sicherheitspolitik umgeleitet wird. Zur näheren Erläuterung wird hierzu die Implementierung eines DCE-RPCs betrachtet.

Neben einer Reihe von Standardservern ist ein wesentlicher Bestandteil der DCE eine Programmbibliothek, welche unter anderem einen Fernaufrufmechanismus zur Implementierung der Klienten/Server - Kommunikation enthält. Automatisch erzeugte und zwischen Klient und Fernaufruf eingebundene Stubmodule verdecken dabei die Details des Fernaufrufmechanismusses, indem sie Botschaftenformate definieren, Parameter in Botschaften ein- und auspacken und Kommunikationsfehler behandeln. Die Abbildung eines Fernaufrufs auf die konkreten Kommunikationsmechanismen einer konkreten Betriebssystemplattform findet schließlich innerhalb der plattformspezifischen DCE-Programmbibliothek statt; eine DCE-Implementierung für Unix etwa verwendet hier die Unix Sockets.

Die Gewährleistung einer vollständigen Kontrolle der Kommunikation erfordert nun den Eingriff in den Kommunikationspfad Klient → Klienten-Stub → DCE-Bibliothek → Betriebssystem → DCE-Bibliothek → Server-Stub → Server. Für die Plazierung des Punktes der Kommunikationsumleitung zur Sicherheitspolitik hat man prinzipiell

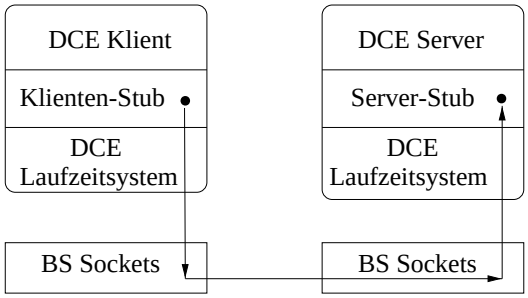


Bild 3.12: Komponenten bei der Kommunikation zwischen Klient und Server

mehrere Alternativen.

a) Plazierung der Kommunikationsumleitung in die Kommunikationsmechanismen innerhalb des Betriebssystems

b) Plazierung im DCE Laufzeitsystem

c) Plazierung im Server-Stub oder Klienten-Stub

d) Plazierung in der Server- oder Klientensoftware selbst.

Wesentliche Kriterien für die Auswahl einer dieser Alternativen sind die Größe der *TCB*, die Verfügbarkeit von Programmquelltexten, die funktionalen Möglichkeiten des Betriebssystems und die Bequemlichkeit für den Anwendungsprogrammierer. Die kleinstmögliche TCB läßt sich durch eine Plazierung des Umleitungspunktes in der Socket-Implementierung innerhalb des Betriebssystems erreichen (Bild 3.13); diese Möglichkeit besteht heute allerdings erst in wenigen Betriebssystemen (Beispiel: Sun Solaris). Im Gegensatz dazu ermöglicht eine Implementierung der Umleitung im Adreßraum des Klienten diesem es prinzipiell, die Umleitung zu umgehen, so daß entweder der Klient zur TCB gerechnet werden muß oder aber von der Sicherheitspolitik signierte Tickets zur Sicherstellung der stattgefundenen Politikkonsultation erforderlich werden. Ausführliche Diskussionen finden sich zu diesem Thema im BirliX-Betriebssystem [SHK+94, HKK93], in DTOS [Min95, OFSS96] und in [Lux95].

Umgeleitete Fernaufrufe laufen schließlich in der Oberklasse (vgl. Bild 3.9, Klasse *GenericDCECustodian*) des Politikcustodians auf. Hier erfolgt nun eine Zuordnung der Klientenoperation zu einer der in der Sicherheitspolitik definierten Zugriffsregeln (im Beispiel *read, write* oder *chgcl*), die anschließende Auswertung der Zugriffsregel unter Berücksichtigung der aus dem Fernaufruf gewonnenen Parameter wie Subjekt, Objekt und Operation, und, abhängig vom Ergebnis, eine Zulassung bzw. Unterdrückung der Kommunikationsoperation.

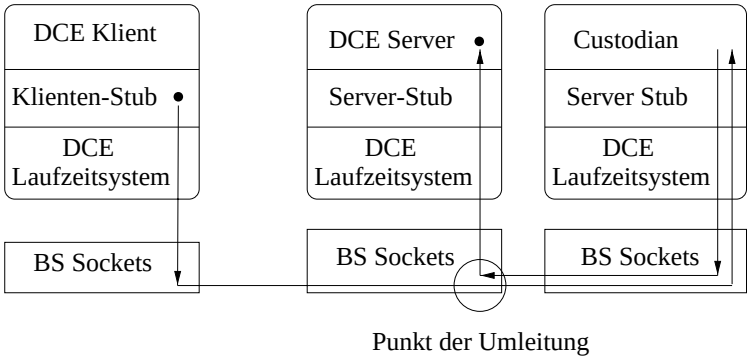


Bild 3.13: Implementierung der vollständigen Kommunikationskontrolle in den Betriebssystem-Kommunikationsmechanismen

Persistenz

Die Persistenz des Politikzustandes muß in der DCE explizit herbeigeführt werden; sie ist keine Eigenschaft eines Servers an sich. Der verwendete Mechanismus ist grundsätzlich unkompliziert: als Teil der Politikinitialisierung wird der Oberklasse *GenericDCECustodian* die Adressen und Größen der Politikzustandsvariablen bekanntgemacht; hierdurch wird die Oberklasse in die Lage versetzt, bei jedem Anhalten der Sicherheitspolitik die Zustandsvariablen auf einen persistenten Speicher (i.d.R. in eine Datei) zu übertragen und beim späteren Wiederanlauf den Politikzustand von dort wieder zu rekonstruieren.

Ganz offensichtlich zeigt diese Form der Persistenzimplementierung lediglich das grundsätzliche Prinzip auf; eine professionelle Implementierung wird dagegen eine Reihe von Verfeinerungen und Ergänzungen besitzen müssen. So wird hier zum einen auf der implementierungstechnischen Ebene vorausgesetzt, daß der übersetzte Programmcode der Sicherheitspolitik statisch gebunden ist und nach einem Wiederanlauf der Sicherheitspolitik der Lader des Betriebssystems das Datensegment der Sicherheitspolitik wieder an dieselbe Adresse des virtuellen Adreßraums lädt. Ist dies nicht der Fall, müssen sämtliche Referenzen zu den Politikvariablen indirekt über eine Referenzvariable erfolgen.

Desweiteren wird bei der Durchsetzung der Kapselungseigenschaft neben dem Mechanismus der Adreßraumseparierung nun auch eine korrekte Implementierung des Dateisystems erforderlich, da während der Passivphasen einer Sicherheitspolitik ihr gesamter Zustand dort abgelegt ist. Dies macht nun wiederum den Stellenwert des Dateisystems für die Garantie der Manipulationssicherheit der Sicherheitspolitik deutlich; das Dateisystem bildet folglich bei dieser Form der Persistenzimplementierung einen unabdingbaren Bestandteil der *TCB*. Die Größe und Komplexität von Dateisystemimplementierungen führen erneut zu Problemen bei der Erfüllung des dritten

Referenzmonitorprinzips, und dies läßt diesen Ansatz zur Persistenzimplementierung zweifelhaft erscheinen. Wie auch bei der Gewährleistung der Manipulationssicherheit während der Aktivphasen einer Sicherheitspolitik erfordert der Stellenwert der Manipulationssicherheit, schlankere Persistenzmechanismen auf einer sehr elementaren Ebene der Betriebssystemarchitektur anzusiedeln.

Schließlich berücksichtigt die beschriebene Implementierung keine Aspekte der Fehlertoleranz. Ein Ausfall des Rechners, etwa hervorgerufen durch einen Stromausfall, führt zum Verlust aller Zustandsveränderungen der Sicherheitspolitik, die seit der letzten Aktualisierung des persistenten Speichers stattgefunden haben. Eine periodische Aktualisierung des Politikzustandes allein ist hier nicht ausreichend: zwischen dem Zustand der Sicherheitspolitik und dem Zustand der Applikation besteht ein enger Zusammenhang, so daß Applikation und Zustand stets gemeinsam eine Recoverylinie bilden müssen.

Anbindung der Sicherheitspolitik an die Applikationsobjekte

Die Anbindung der Sicherheitspolitik an die Applikationsentitäten ist eine Abbildung der realen Welt auf die Modellwelt der Sicherheitspolitik: Applikationsentitäten (Klienten, Server, in der DCE identifiziert durch ihre DCE-UniqueId) werden auf die Politikentitäten (Elemente der Menge *entities*, im C++ - Custodian in Bild 3.9 auf Seite 66 dargestellt durch *Integers*) abgebildet, und die anwendungsspezifischen Operationen der Applikation werden konkreten Politikregeln assoziiert. Die Abbildung wird ermöglicht durch einen Bindevorgang, der Hand in Hand mit der oben beschriebenen Ausübung der Kommunikationskontrolle verläuft: sobald eine Applikationsentität unter eine Sicherheitspolitik gestellt wird, wird einerseits wie oben beschrieben ihre Kommunikation umgeleitet und gleichzeitig der Sicherheitspolitik die Möglichkeit geschaffen, die neue Entität zu identifizieren. Ermöglicht wird dies durch eine einfache Namensdienstfunktionalität, die in der plattformspezifischen Oberklasse eines Custodians generisch implementiert ist. Da diese Oberklasse ebenfalls den Aufrufmechanismus für die Politikoperationen enthält, können die DCE-Klienten- und Serveridentifikationen an dieser Stelle vor Aufruf der Politikregel in die politikinterne Darstellung transformiert werden. Eine genau Beschreibung dieser Mechanismen finden sich in [Wag96].

3.6 Zusammenfassung

Dieses Kapitel beschreibt die Rolle von Sicherheitspolitiken als methodischen Ansatz für die ganzheitliche Formulierung einer Strategie zur Realisierung von Sicherheitsanforderungen. Ein durchgängiges Konzept zur Modellierung, Spezifikation und Implementierung von Sicherheitspolitiken wird beschrieben und anhand von Beispielen veranschaulicht.

Sicherheitspolitiken legen die in einer konkreten Umgebung verfolgten Sicherheitsziele fest und beschreiben die Strategien, mit denen die Ziele erreicht werden sollen. Sicherheitsmodelle stellen die erste Stufe einer Präzisierung dar. Sie werden in diesem Kapitel charakterisiert als abstrakte Beschreibungen von Sicherheitspolitiken, die durch präzise und in der Regel formale Beschreibungen der Sicherheitspolitiken die Grundlage für den Nachweis von Sicherheitseigenschaften bilden. Ziel des Kapitels ist dabei die Entwicklung einer Modellierungs-, Spezifizierungs- und Implementierungsmethode, die ausdrucksstarke Paradigmen zur Formulierung eines signifikanten Spektrums von Sicherheitspolitiken enthält. Die vorgestellte Methode zur Modellierung von Sicherheitspolitiken baut auf den bekannten Zugriffsmatrizen zur Repräsentation von Rechten und den Verbänden zur Repräsentation von Informationsflüssen auf. Die Bandbreite von Sicherheitspolitiken, die mittels der vorgestellten Modellparadigmen formuliert werden können, umfaßt sämtliche Modelle, welche auf Zugriffsmatrizen, Verbänden und Zustandsautomaten beruhen; insbesondere sind sämtliche HRU- und BLP-basierten Sicherheitsmodelle ausdrückbar.

Die Spezifikation der Sicherheitsmodelle erfolgt mittels der Spezifikationssprache Skippy, die eigens zu diesem Zweck entwickelt wurde und eine direkte Repräsentation der Modellparadigmen ermöglicht. Skippy-Spezifikationen bilden den Ausgangspunkt von Politikimplementierungen und legen die Grundlage zum Nachweis ihrer Korrektheit.

Da sich traditionelle Prinzipien zur Implementierung von Sicherheitspolitiken für verteilte Multipolitiken-Szenarien als unzulänglich erwiesen, wird hierfür das Custodian-Paradigma entwickelt. Custodians bilden eine Laufzeitumgebung für algorithmisch formulierte Sicherheitspolitiken, welche die Manipulationssicherheit der Sicherheitspolitik, die vollständige Kontrolle der Kommunikation innerhalb der Sicherheitsdomäne, Politikpersistenz und die Bindung von Sicherheitspolitik und Applikationssystem implementiert.

Diese durchgängige Methodik hat nun die Voraussetzungen geschaffen, Sicherheitsaspekte des Zusammenwirkens von Sicherheitspolitiken in Metapolitiken zu betrachten. Das anschließende Kapitel 4 beschreibt die Rolle der Metapolitiken, und Kapitel 5 und 6 werden sich mit der Spezifikation und der Implementierung von Metapolitiken befassen.

4 Metapolitiken

In großen Organisationen wie dem in Kapitel 2 geschilderten Technologiekonzern haben Unternehmensbereiche, Abteilungen und Projekte vielfach eigene Sicherheitsanforderungen, die die spezifischen Besonderheiten der jeweiligen Organisationseinheit berücksichtigen und sich erheblich voneinander unterscheiden können. Eine Abbildung dieser Strukturen auf ein IT-System resultiert in einem verteilten System mit einer Vielzahl von Sicherheitsdomänen, in dem jede Domäne ihre eigene, individuelle Sicherheitspolitik besitzt. IT-Systeme mit derartigen Domänenstrukturen werden als Multipolitikensysteme bezeichnet [Hos92b]. Der neue *TAFIM* Architekturstandard (*Technical Architecture Framework for Information Management*, [DIS96]) aus dem Jahre 1996 identifiziert grundsätzliche Anforderungen an Multipolitiken-Systeme; er verlangt, daß IT-Systeme

- koexistierende Sicherheitspolitiken jeglicher Komplexität und jeglichen Typs unterstützen, einschließlich Sicherheitspolitiken für sensitive unklassifizierte Daten und Sicherheitspolitiken für hierarchisch klassifizierte Daten
- Interoperabilität zwischen Entitäten mit unterschiedlichen Sicherheitsattributen und Resourcen mit unterschiedlichen Sicherheitspolitiken gewährleisten [FM98].

Aus diesen Forderungen ergeben sich eine Vielzahl neuer Problemstellungen; so ergibt sich im Kontext der Betriebssystemarchitekturen das Problem einer möglichen gegenseitigen Beeinflussung der Sicherheitspolitiken untereinander, welches in den bisherigen Referenzmonitor- und Sicherheitskernarchitekturen ungelöst ist und auch gar nicht gelöst werden mußte (vgl. Kapitel 3). Ein weiteres Problem ist die Realisierung von Politikdomänen, die koexistierende, aber voreinander geschützte Teilbereiche eines IT-Systems darstellen, in denen eine Sicherheitspolitik die Aktionen sämtlicher Entitäten kontrolliert.

Während sich diese Herausforderungen hauptsächlich im Kontext der Betriebssystemarchitekturen und Sicherheitsarchitekturen stellen, befaßt sich dieses Kapitel mit den Herausforderungen an die Sicherheitspolitiken selbst. Sicherheitspolitiken mit ihren Domänen stellen einerseits ein klares Konzept zur Definition und Realisierung unterschiedlicher Sicherheitsanforderungen innerhalb eines IT-Systems dar; andererseits ist die hieraus resultierende Abgrenzung der Politikdomänen kontraproduktiv im Sinne einer systemweiten Interoperabilität: Sicherheitsdomänen tendieren dazu, autonome

Inseln zu etablieren, auf denen eine gegebene Sicherheitspolitik durchgesetzt ist; offen dagegen bleibt, auf welche Weise das Wasser zwischen diesen Inseln auf sichere Weise überquert werden kann [Hos92a].

Jede solche Überquerung einer Domänengrenze ist äußerst sicherheitskritisch; man denke sich als Beispiel zwei Sicherheitspolitiken, eine einfache wahlfreie Zugriffssteuerungspolitik (DAC) wie etwa im Unix-Betriebssystem integriert, und eine obligatorische MLS-Politik. Um in einem solchen Umfeld einem Unix-Subjekt ein Leserecht auf ein MLS-Objekt der Ebene i einzuräumen, muß innerhalb der Unix-Domäne garantiert sein, daß hierdurch erlangte Informationen niemals von einem MLS-Subjekt der Ebenen i-1 und kleiner gelesen werden können; ansonsten entsteht über den Umweg durch ein Objekt der Unix-Domäne ein der MLS-Politik widersprechender Informationsfluß entgegen der Ebenenhierarchie (Bild 4.1).

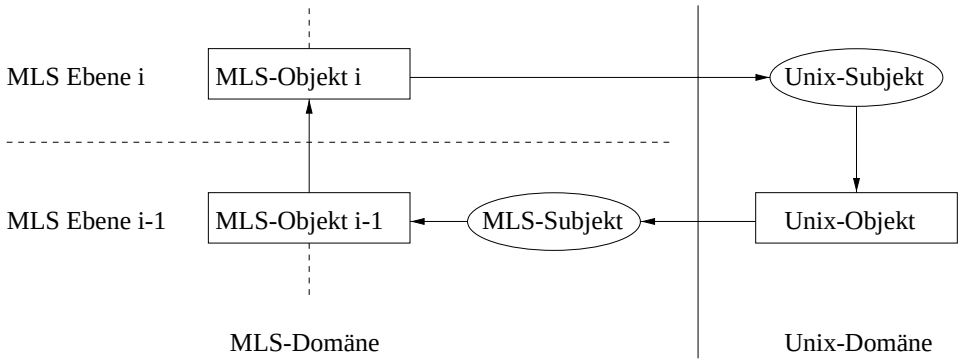


Bild 4.1: Domäneninteraktionen zwischen einer MLS- und einer DAC-Domäne

Eine solche Garantie kann allerdings nur unter ganz besonderen Voraussetzungen gegeben werden; so muß zum Beispiel gewährleistet sein, daß ein Dritter, der Eigentümer des Unix-Objekts, sehr restriktive Regeln bei der Änderung der Zugriffsrechte dieses Objekts befolgt. Im allgemeinen setzt somit jede Interaktion zwischen den Domänen wohldurchdachte Regeln voraus, die präzise die Bedingungen beschreiben, unter denen ein Subjekt der DAC-Domäne auf ein Objekt der MAC-Domäne zugreifen darf – und umgekehrt. Andere Beispiele des allgemeinen Problems domänenübergreifender Aktionen sind das *Propagation Of Local Risk Problem* und das *Cascading Problem* [NCS87].

Die Überwindung der Grenzen der Politikdomänen auf einem Sicherheitsniveau, welches dem Niveau der mittels Sicherheitspolitiken erreichbaren Qualität entspricht, ist Ziel der vorliegenden Arbeit. Der beschrittene Weg beruht auf der Idee, Regeln über Interaktionen zwischen Politikdomänen in übergeordneten Sicherheitspolitiken, den Metapolitiken, zu formulieren. In gleicher Weise, in der Sicherheitspolitiken ein Schlüsselkonzept für die Sicherheit innerhalb einer Politikdomäne darstellen, propagiert die vorliegende Arbeit das Konzept der Metapolitiken als Schlüsselkonzept für die Sicherheit von Interaktionen zwischen verschiedenen Politikdomänen.

Der in dieser Arbeit verfolgte Ansatz versieht Interaktionen zwischen Politikdomänen mit Regeln, die in Metapolitiken formuliert werden. Metapolitikmodelle formalisieren diese Regeln und bilden die Grundlage für die präzise Spezifikation, Analyse und Implementierung von Metapolitiken. Der Weg dorthin beginnt mit einer Klassifikation verschiedener Formen von Interaktionen zwischen Politikdomänen; hieraus werden dann die Aufgaben von Metapolitiken abgeleitet und ein Metapolitikmodell aufgestellt. Basierend auf diesem Modell werden schließlich Sätze entwickelt, die grundsätzliche Möglichkeiten und Grenzen für die Analyse von Metapolitiken aufzeigen. Die anschließenden Kapitel werden sich dann mit der Spezifikation und der Implementierung von Metapolitiken befassen.

4.1 Interaktionen zwischen Politikdomänen

Ziel dieses Abschnitts ist die Identifikation verschiedener Formen von Interaktionen zwischen Politikdomänen. Ausgehend von einer Klassifikation verschiedener Interaktionsformen wird für jede der identifizierten Interaktionsklassen die Rolle einer Metapolitik bei der Beschreibung von Sicherheitseigenschaften diskutiert. Bei den Interaktionsformen liegt die Fokussierung aus den bereits genannten Gründen auf solchen Formen, die durch die Klasse der Zugriffssteuerungspolitiken betrachtet werden; Interaktionen also, bei denen Entitäten einer Subjektmenge Zugriffsoperationen auf Entitäten einer Objektmenge durchführen, und bei denen die Sicherheitspolitiken Regeln über die Zulässigkeit solcher Operationen enthalten.

In Szenarien mit einer einzigen, global in einem IT-System gültigen Sicherheitpolitik enthält diese Sicherheitspolitik Regeln über sämtliche sicherheitsrelevanten Interaktionen zwischen den Entitäten des Systems. In Multipolitiken-Szenarien kann dagegen nicht mehr einfach davon ausgegangen werden, daß bei Interaktionen zwischen Entitäten genau eine Sicherheitspolitik involviert ist. Bereits in dem sehr einfachen Fall, daß ein Subjekt aus einer Domäne P auf ein Objekt aus einer Domäne Q zugreift, sind bereits mindestens zwei Sicherheitspolitiken involviert. (Bild 4.2). Im eingangs in Kapitel 2 geschilderten Technologiekonzern-Szenarium entstehen solche Situationen etwa dann, wenn Mitarbeiter des Unternehmensbereichs „Entwicklung" auf die Server des Unternehmensbereichs „Forschung" zugreifen (vgl. auch Bild 2.2).

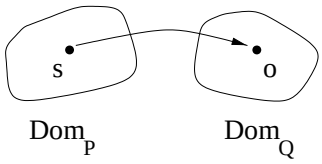


Bild 4.2: Ein Subjekt der Domäne von P greift auf ein Objekt der Domäne von Q zu

Keine der beiden Sicherheitspolitiken ist hier in der Lage, eine Regel für diesen Zugriff anzugeben: Sicherheitspolitik P kennt das Objekt und dessen Sicherheitsattribute nicht, und umgekehrt kennt Sicherheitspolitik Q das Subjekt und dessen Sicherheitsattribute nicht.

In Multipolitiken-Szenarien kann es weiterhin Überlappungen von Sicherheitsdomänen geben; Bild 4.3 zeigt eine solche Überlappung, wobei Subjekt und Objekt des Zugriffs hier innerhalb des Überlappungsbereichs liegen. Solche Domänenüberlappungen sind ein Ansatz zur Lösung des in [FM98] geschilderten Problems der *interdomain transfer objects*; ein solcher Zugriff unterliegt gleichzeitig dem Einfluß zweier Sicherheitspolitiken, woraus sich dann allerdings mögliche Konfliktsituationen ergeben können.

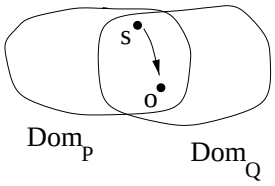


Bild 4.3: Subjekt und Objekt liegen in der Schnittmenge der Domänen von P und Q

Dieser Abschnitt entwickelt im folgenden eine präzise Klassifikation von Domäneninteraktionen. Als Ausgangspunkt wird Lampson's klassisches Modell verwendet, in dem jede Interaktion zwischen den Entitäten eines Systems als Zugriff eines Subjekt s auf ein Objekt o mittels einer Zugriffsfunktion f modelliert wird. Mit $P(s, o, f)$ wird die Anwendung der Sicherheitspolitik P auf einen konkreten Zugriff (s, o, f) bezeichnet; da P eine Zugriffssteuerungspolitik ist, ist $P(s, o, f)$ vom Typ $\{zulässig, unzulässig\}$. Mit Dom_P wird die Domäne einer Sicherheitspolitik P bezeichnet, d.h. diejenige Menge von Entitäten (Subjekten und Objekten), die der Sicherheitspolitik P zugeordnet ist; für jeden Zugriff (s, o, f) enthält eine Sicherheitspolitik P genau dann eine Zugriffsregel, wenn $s, o \in Dom_P$. $\Pi_e = \{P | e \in Dom_P\}$ bezeichnet die Menge aller Sicherheitspolitiken, zu deren Domänen die Entität e gehört. Schließlich wird $|M|$ als gebräuchliche Notation für die Kardinalität einer Menge M verwendet.

Sei nun I eine endliche Indexmenge, $\{P_i\}_{i \in I}$ eine Menge von Sicherheitspolitiken, $s, o \in \bigcup_{i \in I} Dom_{P_i}$ und f eine beliebige Zugriffsfunktion. In einem Multipolitiken-Szenarium mit einer Menge von Sicherheitspolitiken $\{P_i\}_{i \in I}$ läßt sich nun jeder Zugriff (s, o, f) einer der folgenden Zugriffsklassen zuordnen.

Klasse 1: $|\mathbf{\Pi_s}| = |\mathbf{\Pi_o}| = \mathbf{1} \ \wedge \ \mathbf{\Pi_s} = \mathbf{\Pi_o}$

Diese Zugriffsklasse charakterisiert eine Situation, in der Subjekt und Objekt in derselben Sicherheitspolitikdomäne liegen und gleichzeitig in keiner weiteren.

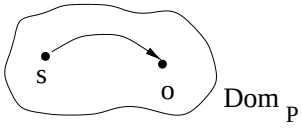


Bild 4.4: Die Zugriffsklasse 1

Zugriffe dieser Klasse überschreiten keine Domänengrenzen; somit ist eine einzige Sicherheitspolitik berechtigt und auch in der Lage, eigenständig über diesen Zugriff zu entscheiden.

Klasse 2: $|\mathbf{\Pi_s} \cap \mathbf{\Pi_o}| = \mathbf{0}$

Diese Zugriffsklasse ist dadurch charakterisiert, daß keine Sicherheitspolitik existiert, in deren Domäne gleichzeitig Subjekt und Objekt liegen. Insbesondere gibt es damit auch keine Sicherheitspolitik, die eine Regel für diesen Zugriff enthält. Da $s, o \in \bigcup_{i \in I} Dom_{P_i}$ gilt, gibt es aber dennoch Sicherheitspolitiken, zu deren Domänen jeweils eine der beteiligten Entitäten gehört; hierbei läßt sich weiter unterscheiden zwischen

a) $|\Pi_s| = 1 \ \wedge \ |\Pi_o| = 1$,

d.h. beide Entitäten liegen in genau einer Politikdomäne (Bild 4.5).

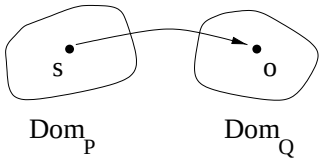


Bild 4.5: Die Klasse 2a

b) $\exists e \in \{s, o\}$ mit $|\Pi_e| > 1$,

d.h. mindestens eine der beteiligten Entitäten liegt in mehr als einer Politikdomäne; Bild 4.6 zeigt ein Beispielszenarium mit vier Politikdomänen.

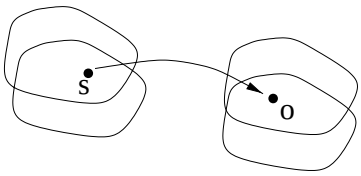


Bild 4.6: Die Klasse 2b

Klasse 3: $|\mathbf{\Pi_s} \cap \mathbf{\Pi_o}| \geq \mathbf{1} \ \wedge \ \exists \mathbf{e} \in \{\mathbf{s}, \mathbf{o}\}$ **mit** $|\mathbf{\Pi_e}| > \mathbf{1}$

Diese Zugriffsklasse ist dadurch charakterisiert, daß zum einen (mindestens) eine Politikregel definiert ist, zum anderen aber zusätzlich (mindestens) eine der beteiligten Entitäten in mehr als einer Politikdomäne liegt.
Weitere Differenzierungen können wie folgt vorgenommen werden:

a) $|\Pi_s \cap \Pi_o| \ = \ 1$,

d.h. es existiert zwar genau eine Regel, dennoch kann diese nicht autonom angewandt werden, da eine der beteiligten Entitäten (o in Bild 4.7) noch zu einer weiteren Politikdomäne gehört.

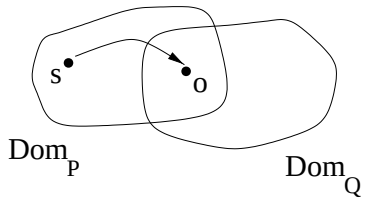


Bild 4.7: Die Klasse 3a

b) $|\Pi_s \cap \Pi_o| \ > \ 1$,

d.h. es sind mehrere, potentiell konfligierende Regeln aus unterschiedlichen Sicherheitspolitiken anwendbar (P und Q in Bild 4.8); hierzu können sich dann weitere Sicherheitspolitiken gesellen, die eine der beteiligten Entitäten in ihrer Domäne haben (R in Bild 4.8).

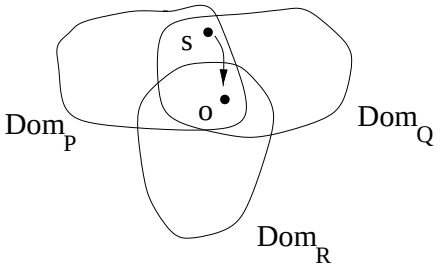


Bild 4.8: Die Klasse 3b

Diese Klassifikation ist sowohl eindeutig als auch vollständig: jeder Zugriff (s, o, f) in einem Multipolitiken-System mit $s, o \in \bigcup_{i \in I} Dom_{P_i}$ kann genau einer der drei Klassen zugeordnet werden. Weiterhin identifiziert jede der Klassen, wie noch erläutert werden wird, notwendige und hinreichende Bedingungen zur Identifikation solcher Zugriffe, die allein mit den existierenden Sicherheitspolitiken P_i nicht handhabbar sind.

Eindeutigkeit

Für den Beweis der Disjunktheit der einzelnen Klasseneigenschaften wird gezeigt, daß aus der Gültigkeit einer der drei Klasseneigenschaften ein Widerspruch zu jeweils den beiden anderen ableitbar ist, d.h. der Beweis folgt dem Schema

a) Annahme, die Klasseneigenschaft 1 gilt $\Rightarrow$ Widerspruch zu den Klasseneigenschaften 2 (Beweisschritt a1) und 3 (Beweisschritt a2)
b) Annahme, die Klasseneigenschaft 2 gilt $\Rightarrow$ Widerspruch zu den Klasseneigenschaften 1 (Beweisschritt b1) und 3 (Beweisschritt b2)
c) Annahme, die Klasseneigenschaft 3 gilt $\Rightarrow$ Widerspruch zu den Klasseneigenschaften 1 (Beweisschritt c1) und 2 (Beweisschritt c2).

a) $|\mathbf{\Pi_s}| = |\mathbf{\Pi_o}| = \mathbf{1} \ \wedge \ \mathbf{\Pi_s} = \mathbf{\Pi_o}$

(a1) Der Widerspruch zu $|\Pi_s \cap \Pi_o| = 0$ ergibt sich unmittelbar aus $\Pi_s = \Pi_o$.

(a2) Die Klasseneigenschaft 3 verlangt die Existenz eines e mit $|\Pi_e| > 1$; da $|\Pi_s| = |\Pi_o| = 1$, kann es kein solches e geben.

b) $|\mathbf{\Pi_s} \cap \mathbf{\Pi_o}| = \mathbf{0}$

(b1) Da jedes Π nichtleer ist, folgt aus der Voraussetzung unmittelbar der Widerspruch zu $\Pi_s = \Pi_o$.

(b2) Direkter Widerspruch der Fallvoraussetzung zu $|\Pi_s \cap \Pi_o| \geq 1$.

c) $|\mathbf{\Pi_s} \cap \mathbf{\Pi_o}| \geq \mathbf{1} \ \wedge \ \exists \mathbf{e} \in \{\mathbf{s}, \mathbf{o}\} : |\mathbf{\Pi_e}| > \mathbf{1}$

(c1) Die Existenz eines e mit $|\Pi_e| > 1$ steht in direktem Widerspruch zu $|\Pi_s| = |\Pi_o| = 1$.

(c2) Direkter Widerspruch der Fallvoraussetzung zu $|\Pi_s \cap \Pi_o| = 0$.

□

Vollständigkeit

Für den Beweis der Vollständigkeit der Zugriffsklassifikation wird eine Fallunterscheidung hinsichtlich der durch einen Zugriff betroffenen Sicherheitspolitiken gemacht.

a) $|\mathbf{\Pi_s} \cap \mathbf{\Pi_o}| = \mathbf{0}$
Sämtliche Zugriffe dieser Art genügen der Definition der Zugriffsklasse 2.

b) $|\mathbf{\Pi_s} \cap \mathbf{\Pi_o}| = \mathbf{1}$
Falls hierbei s und o in genau einer Politikdomäne liegen (d.h. $|\Pi_s| = |\Pi_o| = 1$), so genügen Zugriffe dieser Art der Definition der Zugriffsklasse 1; ansonsten existiert ein $e \in \{s, o\} : |\Pi_e| > 1$, und der Zugriff genügt der Definition der Zugriffsklasse 3.

c) $|\mathbf{\Pi_s} \cap \mathbf{\Pi_o}| > \mathbf{1}$
Hier gilt insbesondere, daß $|\Pi_s| > 1$ und $|\Pi_o| > 1$; das in der Definition der Zugriffsklasse 3 geforderte e existiert somit.

□

4.2 Metapolitiken

Mit der Klassifikation des vorangehenden Abschnitts verfügt man nun über eine vollständige und eindeutige Charakterisierung der Interaktionen zwischen den Entitäten eines Multipolitiken-Szenariums. Dieser Abschnitt identifiziert nun auf dieser Grundlage die Aufgaben einer Metapolitik und leitet hieraus eine Definition ab.

4.2.1 Unvollständigkeit

Interaktionen der Klasse 2 sind dadurch charakterisiert, daß keine der bestehenden Sicherheitspolitiken ($\in \{P_i\}_{i \in I}$) eine Regel über die Zulässigkeit der Interaktion angeben kann. Zwar mag es bei einem Zugriff (s, o, f) mehrere Sicherheitspolitiken geben, in deren Domäne s oder o liegt; dennoch gibt es, da $|\Pi_s \cap \Pi_o| = 0$, keine Sicherheitspolitik, in deren Domäne sowohl s als auch o liegen. Hierdurch entsteht ein politikfreier Raum, in dem das Verhalten des Systems in Hinblick auf die Informationssicherheit unvollständig definiert ist. Folglich ist für diesen Interaktionstyp die Definition neuer Regeln erforderlich; mit anderen Worten, Interaktionen der Klasse 2 erfordern eine neue *Vollständigkeitspolitik* V, die diese Lücke schließt.

Die Domäne von V umfaßt sämtliche Domänen der beteiligten Sicherheitspolitiken $P_{i \in I}$: $Dom_V = \bigcup_{i \in I} Dom_{P_i}$. V definiert dabei Zugriffsregeln für Zugriffe im politikfreien Raum, d.h. für solche Paare (s, o)[1] mit $s, o \in \bigcup_{i \in I} Dom_{P_i}$, für die gilt $|\Pi_s \cap \Pi_o| = 0$.

4.2.2 Konflikte

Interaktionen der Klasse 3 sind dadurch charakterisiert, daß mindestens eine der bestehenden Sicherheitspolitiken ($\in \{P_i\}_{i \in I}$) eine Regel für die Interaktion angeben kann, daß aber mindestens eine der involvierten Entitäten in einer weiteren Politikdomäne liegt. Dies kann zu zwei unterschiedlichen Arten von Konflikten führen: gilt $|\Pi_s \cap \Pi_o| = 1$, so gibt es genau eine anwendbare Sicherheitspolitik, und ein Konflikt zwischen mehreren anwendbaren Regeln ist ausgeschlossen. Dennoch kann die einzige existierende Regel nicht allein angewendet werden, da eine oder beide der Entitäten zu

[1] Für die Klassifikation einer Subjekt/Objekt - Interaktion ist die konkrete Zugriffsfunktion nicht wesentlich. Wenn immer i.f. keine Mißverständnisse möglich sind, wird kurz (s, o) anstelle von (s, o, f) geschrieben.

weiteren Politikdomänen gehören. Diese Konfliktform wird im folgenden als *Domänenkonflikt* bezeichnet. Gilt dagegen $|\Pi_s \cap \Pi_o| > 1$, so gibt es mehr als eine anwendbare Sicherheitspolitik mit entsprechendem Konfliktpotential; diese Konfliktform wird im folgenden als *Regelkonflikt* bezeichnet.

In keiner der genannten Konfliktformen bestehen eindeutige Regeln, die das Verhalten des Systems in Hinblick auf die Informationssicherheit definieren. Folglich ist für diesen Interaktionstyp die Definition neuer Regeln erforderlich; mit anderen Worten, Interaktionen der Klasse 3 erfordern eine neue *Konfliktpolitik* K, die diese Lücke schließt.

Analog zu V umfaßt die Domäne von K sämtliche Domänen der beteiligten Sicherheitspolitiken; K definiert Zugriffsregeln für Zugriffe im Konfliktraum, d.h. für solche Paare (s, o) mit $s, o \in \bigcup_{i \in I} Dom_{P_i}$, für die gilt $|\Pi_s \cap \Pi_o| \geq 1$ und $\exists e \in \{s, o\} : |\Pi_e| > 1$.

4.2.3 Die Aufgaben von Metapolitiken

In unserer Klassifikation existiert lediglich eine einzige Klasse von Interaktionen (Klasse 1), über deren Zulässigkeit unmittelbar und autonom durch eine der bestehenden Sicherheitspolitiken entschieden werden kann; hier besteht weder ein Konflikt noch tut sich ein politikfreier Raum auf. Alle anderen Klassen erfordern zur Konfliktmediation und zur Vervollständigung politikfreier Räume neue Regeln, und genau deren Definition wird als die originäre Aufgabe einer Metapolitik betrachtet.

Die folgenden Definitionen fassen dieses Ergebnis zusammen.

Definition 4.1
Bei einem Zugriff (s, o) besteht ein *Domänenkonflikt* genau dann, wenn mindestens eine der beteiligten Entitäten zu mehr als einer Politikdomäne gehören, d.h.

$$\exists e \in \{s, o\} : |\Pi_e| > 1.$$

Definition 4.2
Bei einem Zugriff (s, o) besteht ein *Regelkonflikt* genau dann, wenn mehr als eine Sicherheitspolitik eine Regel für diesen Zugriff enthält, d.h.

$$|\Pi_s \cap \Pi_o| > 1.$$

Definition 4.3
Ein Zugriff (s, o) heißt *politikfrei* genau dann, wenn es keine Sicherheitspolitik gibt, die für (s, o) eine Zugriffsregel enthält, d.h.

$$|\Pi_s \cap \Pi_o| = 0.$$

Auf der Grundlage dieser Definitionen lassen sich den identifizierten Interaktionsklassen die folgenden Eigenschaften zuordnen.

Zugriffsklasse	Domänenkonflikt	Regelkonflikt	Politikfreiheit
1	nein	nein	nein
2a	nein	nein	ja
2b	ja	nein	ja
3a	ja	nein	nein
3b	ja	ja	nein

Während rein kombinatorisch bei drei Attributen $2^3 = 8$ Klassen möglich sind, sind auf Grund der Definitionen des Domänenkonflikts, des Regelkonflikts und der Politikfreiheit bestimmte Kombinationen ausgeschlossen. Insbesondere folgt aus einem Regelkonflikt stets ein Domänenkonflikt: Definition 4.2 $\Leftrightarrow |\Pi_s \cap \Pi_o| > 1 \Rightarrow |\Pi_s| > 1, |\Pi_o| > 1 \Rightarrow \forall e \in \{s, o\} : |\Pi_e| > 1 \Rightarrow$ Definition 4.1; somit kann es keine Zugriffe der Form (kein Domänenkonflikt, Regelkonflikt, $*$) geben. Schließlich, um einen Regelkonflikt zu erhalten, muß es trivialerweise eben die konfligierenden Regeln geben; dies schließt alle Zugriffe der Form ($*$, Regelkonflikt, Politikfreiheit) aus, so daß mit den oben genannten restlichen Fällen die Klassifikation der Zugriffsoperationen vollständig ist.

Die folgende Definition nennt schließlich die notwendigen und hinreichenden Bedingungen für die Existenz einer Metapolitik in Multipolitiken-Szenarien.

Definition 4.4

Ein Zugriff (s, o) heißt genau dann *MP-frei*, wenn er weder einen Domänenkonflikt noch einen Regelkonflikt impliziert noch im politikfreien Raum liegt.

Eine Metapolitik ist im Sinne dieser Definition genau dann erforderlich, wenn Zugriffe (s, o) existieren, die nicht MP-frei sind. Das folgende Korollar wird den Umgang mit dieser Definition etwas einfacher gestalten.

Korollar 4.5

Ein Zugriff (s, o) ist genau dann MP-frei, wenn gilt

$$|\Pi_s \cap \Pi_o| = 1 \text{ und } |\Pi_s| = |\Pi_o| = 1.$$

Beweis Korollar 4.5

1. von links nach rechts, d.h. z.z.: (s, o) ist MP-frei $\Rightarrow$ Korollar 4.5:
   (s, o) ist MP-frei

$$\begin{array}{ll} \Rightarrow_{Def s.4.1-4.3} & \neg\ (\exists e \in \{s, o\} : |\Pi_e| > 1) \ \wedge \\ & \neg\ (|\Pi_s \cap \Pi_o| > 1) \ \wedge \\ & \neg\ (|\Pi_s \cap \Pi_o| = 0) \\ \Rightarrow & \forall e \in \{s, o\} : |\Pi_e| \leq 1 \ \wedge \ |\Pi_s \cap \Pi_o| \leq 1 \ \wedge \ |\Pi_s \cap \Pi_o| \neq 0 \\ \Rightarrow & \forall e \in \{s, o\} : |\Pi_e| = 1 \ \wedge \ |\Pi_s \cap \Pi_o| = 1 \end{array}$$

2. von rechts nach links, d.h. z.z.: Korollar 4.5 $\Rightarrow$ (s, o) ist MP-frei:

$$\begin{array}{ll} |\Pi_s \cap \Pi_o| = 1 & \Rightarrow \text{ kein Regelkonflikt, nicht politikfrei} \\ |\Pi_s| = |\Pi_o| = 1 & \Rightarrow \text{ kein Domänenkonflikt.} \end{array}$$

□

4.2.4 Definition

Aus diesen Überlegungen und der vorgenommenen Klassifikation von Interaktionen in Multipolitiken-Szenarien leitet sich nun die folgende Definition einer Metapolitik ab.

Definition 4.6

Sei I eine endliche Indexmenge und $\{P_i\}_{i \in I}$ die Menge der Sicherheitspolitiken eines Multipolitiken-Systems. Eine *Metapolitik* $M_{\{P_i\}}$ über der Sicherheitspolitikmenge $\{P_i\}$ ist ein Tupel $M_{\{P_i\}} = (\{P_i\}_{i \in I}, V, K, c)$ bestehend aus

- einer endlichen Menge $\{P_i\}_{i \in I}$ von Sicherheitspolitiken, die die Sicherheitsanforderungen für Klasse-1 Interaktionen implementieren,
- einer Vollständigkeitspolitik V, die die Sicherheitsanforderungen für Klasse-2 Interaktionen implementiert,
- einer Konfliktpolitik K, die die Sicherheitsanforderungen für Klasse-3 Interaktionen implementiert,
- einer Klassifizierungsfunktion c, die für jede Interaktion (s, o), $s, o \in \bigcup_{i \in I} Dom_{P_i}$ die Klasse von (s, o) bestimmt.

V und K sind Sicherheitspolitiken, deren Domäne jede Domäne der P_i umfaßt: $Dom_V = Dom_K = \bigcup_{i \in I} Dom_{P_i}$.

Die Klassifizierungsfunktion c ist vom Typ $\bigcup_{i \in I} Dom_{P_i} \times \bigcup_{i \in I} Dom_{P_i} \rightarrow \{P_i\}_{i \in I} \cup V \cup K$ und ist für jedes $s, o \in \bigcup_{i \in I} Dom_{P_i}$ definiert durch

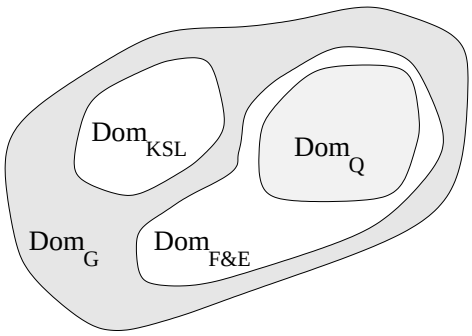
$$c(s,o) = \begin{cases} P_j & \mid |\Pi_s| = |\Pi_o| = 1 \ \wedge \ \Pi_s = \Pi_o = \{P_j\} & \text{(Klasse-1 Interaktion)} \\ V & \mid |\Pi_s \cap \Pi_o| \ = \ 0 & \text{(Klasse-2 Interaktion)} \\ K & \mid |\Pi_s \cap \Pi_o| \geq 1 \ \wedge \ \exists e \in \{s,o\} : |\Pi_e| > 1 & \text{(Klasse-3 Interaktion).} \end{cases}$$

$M_{\{P_i\}}$ läßt sich nun als Sicherheitspolitik des gesamten Multipolitiken-Systems interpretieren, deren Semantik durch

$$M_{\{P_i\}}(s, o, f) = c(s, o)(s, o, f)$$

definiert ist.

Am Beispiel des Technologiekonzerns aus Kapitel 2 soll nun das Spektrum der Möglichkeiten aufgezeigt werden, welches diese Definition aufspannt. Der Beispielkonzern ist in mehrere Unternehmensbereiche gegliedert, die wiederum in einer Hierarchie aus Abteilungen und Projekten organisiert sind. Die individuellen Sicherheitsanforderungen einzelner Organisationseinheiten prägen dem IT-System des Konzerns eine zur Organisationshierarchie symmetrische Struktur auf, in der durch Sicherheitsdomänen den individuellen Sicherheitsanforderungen der Organisationseinheiten Rechnung getragen wird. Bild 4.9 zeigt beispielhaft einen Ausschnitt aus der Domänenstruktur.



G: Globale Sicherheitspolitik
KSL: Sicherheitspolitik des Unternehmensbereichs "Kundenspezifische Lösungen"
F&E: Sicherheitspolitik des Unternehmensbereichs "Forschung & Entwicklung"
Q: Sicherheitspolitik der Forschungsabteilung "Q"

Bild 4.9: Ausschnitt der Domänenstruktur des Beispielszenariums

In diesem Szenarium werden nun die Sicherheitsaspekte der Mobilität von Mitarbeitern betrachtet. Eine Mitarbeiterin der Forschungsabteilung Q, sie heiße Ann, wird für einen begrenzten Zeitraum in den Unternehmensbereich „Kundenspezifische Lösungen", *KSL*, versetzt. Nutzt Ann nun an ihrem neuen Arbeitsplatz das IT-System, so wird sie abhängig vom konkreten Authentisierungsschema entweder als ein Subjekt der Sicherheitsdomäne *KSL* oder als Subjekt ihrer Heimatdomäne Q authentisiert (vgl. die Diskussion in Kapitel 2). Für die Diskussion dieses Beispiels soll hier angenommen werden, daß Ann an ihrem neuen Arbeitsplatz als Subjekt der Domäne *KSL* authentisiert wird; jeder Zugriff von Ann auf ein Dokument in ihrer Heimatdomäne ist dann eine Domäneninteraktion.

Für jeden Zugriff dieser Art ist das Autorisierungsschema des IT-Systems unvollständig: weder das Autorisierungsschema der Sicherheitspolitik in *KSL* noch das Autorisierungsschema der Sicherheitspolitik in Q enthalten eine Regel für einen Zugriff von Ann, Subjekt in Dom_{KSL}, auf Ann's Dokument, Objekt in Dom_Q, da Q Ann nicht kennt und *KSL* Ann's Dokument nicht kennt.

Zum schließen dieser Lücke wird nun eine Metapolitik definiert. Die betroffenen Sicherheitspolitiken sind *KSL* und Q (für einen Augenblick sollen die globale Sicherheitspolitik G und die bereichspezifische Sicherheitspolitik $R\&D$ ignoriert werden), somit ist die zugehörige Metapolitik $(\{KSL, Q\}, V, K, c)$. Π_{Ann} ist $\{KSL\}$, und $\Pi_{Ann's\ Dokument}$ ist $\{Q\}$. Die Klassifizierungsfunktion c identifiziert Ann's Dokumentzugriff als Klasse-2 Interaktion, da $|\Pi_{Ann} \cap \Pi_{Ann's\ Dokument}| = 0$. Folglich wird als zuständige Sicherheitspolitik für diesen Zugriff die Vollständigkeitspolitik V selektiert.

In V sind sämtliche Sicherheitsanforderungen für domänenübergreifende Zugriffe in politikfreien Räumen formuliert, welche das Unternehmen in einer strategischen Ent-

scheidung festgelegt hat. Ein Beispiel für eine solche strategische Entscheidung ist die Regel, daß „jede zeitweise versetzte Person die Rechte aus ihrer eigenen Domäne behält und zusätzlich die Rechte eines Angehörigen der neuen Abteilung erhält". Eine Implementierung dieser Regel durch V kann beispielsweise auf der Grundlage einer in V geführten Liste versetzter Mitarbeiter erfolgen, die im Beispiel Ann als Subjekt der Domäne KSL abbildet auf Ann als Subjekt in Q. Hierbei kann dann auch eine Anpassung an unterschiedliche Sicherheitsstandards erfolgen, indem zum Beispiel von Ann eine erneute Authentisierung verlangt wird, in deren Verlauf dann Ann mit kryptographischen Schlüsseln und Capabilities zur Kommunikation mit den Servern in ihrer Heimatdomäne ausgestattet wird.

Betrachten wir nun eine Konfliktsituation. Bisher wurde ignoriert, daß alle Entitäten in Dom_Q und Dom_{KSL} gleichzeitig noch zur Domäne der globalen Sicherheitspolitik G gehören, und weiterhin alle Entitäten in Dom_Q zu $Dom_{F\&E}$ (siehe Bild 4.9). Eine dies berücksichtigende Metapolitik ist $(\{G, KSL, F\&E, Q\}, V, K, c)$. Arbeitet Ann in ihrer Heimatdomäne, so gilt $\Pi_{Ann} = \{G, F\&E, Q\}$; an ihrem Versetzungsort dagegen ist $\Pi_{Ann} = \{G, KSL\}$. In beiden Fällen ist $|\Pi_{Ann} \cap \Pi_{Ann's\ Dokument}| \geq 1$, und jedes Π enthält mehr als eine Sicherheitspolitik. c klassifiziert daher in diesem Fall Ann's Zugriff auf ihr Dokument als Klasse-3 Interaktion, die durch die Konfliktpolitik K kontrolliert wird.

In K sind nun die Sicherheitsanforderungen formuliert, welche das Unternehmen in einer strategischen Entscheidung über die Behandlung von Konfliktsituationen bei domänenübergreifende Interaktionen festgelegt hat. Eine Option für eine solche Strategie ist die Behandlung sämtlicher Interaktionen zwischen Domänen – sowohl solcher in politikfreien Räumen als auch solcher mit konfligierenden Regeln – durch eine einheitliche Sicherheitspolitik; dies ist erreichbar durch Wahl von $K = V$. Eine feinere Variante ist es, innerhalb der Konfliktpolitik zwischen einem Regelkonflikt ($|\Pi_s \cap \Pi_o| > 1$) und einem Domänenkonflikt ($|\Pi_s \cap \Pi_o| = 1$) zu differenzieren. Ein Regelkonflikt entsteht zwischen den Sicherheitspolitiken G, $F\&E$ und Q, wenn Ann innerhalb ihrer Heimatdomäne auf ihr Dokument zugreift (Bild 4.10).

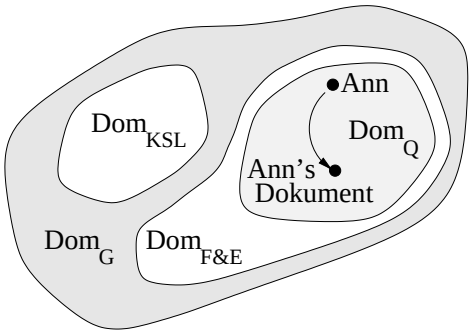


Bild 4.10: Ann arbeitet in ihrer Heimatdomäne und erzeugt einen Regelkonflikt

Unter der Annahme, daß in einer Hierarchie von Sicherheitspolitiken die innersten Sicherheitspolitiken die restriktivsten Regeln definieren, ist es im Falle eines Regelkonflikts eine offensichtliche Strategie, die jeweils restriktivste Sicherheitspolitik auf einen konkreten Zugriff anzuwenden; im Beispiel kann K diese Strategie implementieren, indem K Kenntnis über die Hierarchie aller Sicherheitspolitiken des IT-Systems erhält; stellt K dann einen Regelkonflikt fest, so wählt K aus den konfligierenden Sicherheitspolitiken diejenige mit der höchsten Hierarchieebene aus und wendet diese an. Im Beispiel würde dann Q auf Ann's Zugriff angewendet werden.

Greift Ann dagegen auf ihr Dokument aus ihrer Gastgeberdomäne zu (Bild 4.11), so stellt K einen Domänenkonflikt fest. Die Existenz einer globalen Sicherheitspolitik G

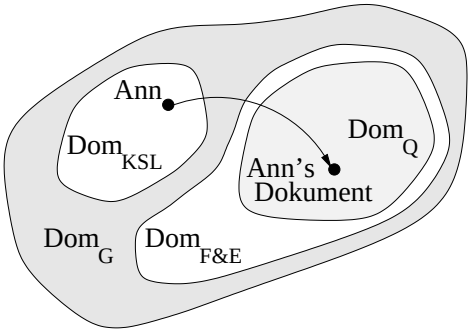


Bild 4.11: Ann arbeitet in ihrer Gastgeberdomäne und erzeugt einen Domänenkonflikt

macht es hier zu einer Option, bei Domänenkonflikten stets diese globale Sicherheitspolitik anzuwenden. Als letztes Beispiel sei die Option genannt, wie bereits oben für V auch für K eine Abbildung zu definieren, die Ann auf ihre Heimatdomäne projiziert und dann wiederum die dortige Sicherheitspolitik Q anwendet.

4.3 Diskussion

Die Definition des Begriffs der Metapolitik im vorangehenden Abschnitt beschreibt ein Konzept, Sicherheitspolitiken für Domäneninteraktionen in verteilten Systemen mit einer Vielzahl von Sicherheitsdomänen zu formulieren. Eine Diskussion dieser Definition im Kontext des Technologiekonzernbeispiels hat bereits die bestehenden großen Freiheitsgrade bei der Formulierung von Konflikt- und Vollständigkeitsstrategien aufgezeigt. In diesem Abschnitt soll nun diskutiert werden, in wie weit ein so definiertes Metapolitikkonzept mit weiteren charakteristischen Merkmalen verteilter Systeme (Wirtschaftlichkeit, Autonomie, Skalierbarkeit) harmonisiert. Im anschließenden Kapitel wird die Definition schließlich als Grundlage von Sätzen verwendet, die Aussagen über wesentliche Sicherheitseigenschaften von Metapolitiken treffen.

Autonomie. Die Organisationsformen moderner Unternehmen in Wirtschaft und Verwaltung versehen heute einzelne Unternehmensbereiche und Abteilungen mit einem hohen Maß an Autonomie; Organisationseinheiten treten vielfach als eigenständig wirtschaftende Einheiten (*profit centers*) auf, die dann auch im Hinblick auf Maßnahmen zur Sicherheit ihrer IT-Systeme eigenverantwortlich und unabhängig von anderen Unternehmensbereichen agieren. Die Durchsetzung dieser Autonomie erfordert die Freiheit für strategische Entscheidungen jeder einzelnen Organisationseinheit, in welchem Grad die eigene Sicherheitspolitik bei Interaktionen mit fremden Politikdomänen aufrecht erhalten werden soll. Denkbar ist hier das gesamte Spektrum zwischen dem Insistieren einer Organisationseinheit auf eine kompromißlose Durchsetzung der eigenen Sicherheitspolitik (mit dem Risiko der Kooperationsunfähigkeit) bis hin zu einer vollständigen Unterordnung der eigenen Sicherheitspolitik unter eine firmenweite, domänenübergreifende Kooperationspolitik.
Ein Metapolitik-Konzept wird in diesem Kontext nur dann akzeptabel sein, wenn es in der Lage ist, diese Freiheitsgrade zu reflektieren und formulieren. Obiges Beispiel machte deutlich, in welcher Form diese Freiheitsgrade bestehen.

Skalierbarkeit. Im Beispiel des Technologiekonzerns wird die Organisationsstruktur des Unternehmens durch die Struktur der Sicherheitsdomänen reflektiert; Metapolitiken für ein solches Szenarium müssen daher den Unternehmensstrukturen folgen können und in Bezug auf die Integration neuer Sicherheitspolitiken, dem Herauslösen bestehender Sicherheitspolitiken sowie dem Wachsen und Schrumpfen von Sicherheitsdomänen skalierbar sein.
Andere Szenarien, beispielsweise Kooperationen zwischen mehreren Institutionen im Kontext von großen Ausschreibungen der öffentlichen Hand oder Forschungsprojekten der Europäischen Kommission haben im allgemeinen eine Lebensdauer von nur wenigen Jahren, teils Monaten. Ihre Vorbereitungszeit ist entsprechend kurz und insbesondere erheblich kürzer, als mit heutigem Stand der Kunst die Entwicklungszeit einer eigenen vollständigen Sicherheitspolitik anzusetzen ist. Ein weiteres, besonderes prägnantes Beispiel für kurzlebige Multipolitikenszenarien sind Expertenkonferenzen, in denen Experten verschiedener Institutionen in stets neuen Konstellationen für einen sehr kurzen Zeitraum zusammenkommen. Derart kurzlebige Multipolitikenszenarien bedingen, daß Skalierung eines Multipolitikenszenariums hinsichtlich der Integration neuer Sicherheitspolitiken, Herauslösen bestehender Sicherheitspolitiken und Wachsen und Schrumpfen von Sicherheitsdomänen ein schneller und unkomplizierter Vorgang sein muß.

Wirtschaftlichkeit. Die Investitionen eines Unternehmens in seine Sicherheitspolitiken sind hoch; ihre Entwicklung ist zwingender als die anderer Softwarekomponenten mit hoher Präzision bei der Anforderungsdefinition, dem kostenintensiven Einsatz formaler Methoden bei der Sicherheitspolitikentwicklung (Sicherheitsmodelle und Spezifikationskalküle) und der Zertifikation der Ergebnisse verbunden. Von daher gibt es bedeutende wirtschaftliche Gründe, existierende Sicherheitspolitiken bei ihrer Integration in eine Metapolitik nicht zu modifizieren, sondern eine Metapolitik durch eine

Komposition der existierenden Sicherheitspolitiken mit Komponenten zur Konfliktmediation und Vervollständigung zu konstruieren.

Ausdruckskraft. Die Bandbreite und Komplexität heutiger Sicherheitsanforderungen ist sehr groß. Entsprechend ausdrucksstark sind heutige Sicherheitsmodelle und Spezifikationskalküle, in denen Sicherheitspolitiken formuliert werden.
Ein Metapolitik-Konzept darf daher in seiner Ausdruckskraft nicht hinter dem heute durch Sicherheitspolitiken ausdrückbaren Spektrum zurückbleiben. Konkret bedeutet dies zum einen, daß das Maß an Einschränkungen zu beurteilen ist, die ein Metapolitikkonzept für die unterstellten Sicherheitspolitiken hinsichtlich der verwendeten Sicherheitsmodelle impliziert. Konkret bedeutet dies zum anderen, daß für die Formulierung der Metapolitik selbst ein Niveau an Ausdruckskraft erreicht werden muß, welches nicht hinter dem des für Sicherheitpolitiken bereits erreichten zurückfällt.

4.3.1 Metapolitiken über MLS und Chinese-Wall

Dieser Abschnitt ist eine weitere Konkretisierung des Technologiekonzern-Szenariums, an der die Eigenschaften des Metapolitik-Konzeptes hinsichtlich der genannten grundsätzlichen Eigenschaften verteilter Systeme diskutiert werden. In dieser Konkretisierung wird nun angenommen, daß Ann's Heimatdomäne eine hierarchische Sicherheitspolitik (MLS) verwendet, ihre Gastgeberdomäne dagegen die Chinese-Wall-Politik einsetzt. Diese Wahl wurde getroffen, da sie eine außergewöhnlich schwierige Situation darstellt: beides sind Sicherheitspolitiken zur obligatorischen Zugriffssteuerung, wobei die Grundlagen der jeweiligen Zugriffsregeln jedoch sowohl syntaktisch als auch semantisch unverträglich sind. Weiterhin finden sich MLS-Politiken vor allem in Organisationen mit sensitiven Aufgabenbereichen, in denen jegliche Schwächung der Informationssicherheit besonders problematisch ist; auf unternehmenspolitischer Ebene spielt neben Wirtschaftlichkeitsaspekten also die Autonomie der Organisationseinheit hinsichtlich ihrer verfolgten Sicherheitspolitik eine große Rolle.

Rekapitulation des Bell/LaPadula Sicherheitsmodells

Das Bell/LaPadula Sicherheitsmodell ist die wohl bekannteste Formalisierung einer hierarchischen Sicherheitspolitik; es hat zum Ziel, die Vertraulichkeit von Informationen zu bewahren und definiert dazu obligatorische Zugriffsregeln über den Zugriff von Subjekten auf Objekte. Grundlage dieser Regeln ist eine Attributierung der Subjekte und Objekte. Objekte werden klassifiziert nach ihrer Vertraulichkeit, z.B. orientiert an den Geheimhaltungsgraden der Bundesrepublik Deutschland *streng geheim, geheim, vertraulich* und *nur für den Dienstgebrauch*. Die Hierarchie der Klassen wird durch eine totale Ordnung „<" zwischen den Klassen in der Form *nur für den Dienstgebrauch* < *vertraulich* < *geheim* < *streng geheim* bestimmt. Subjekte erhalten eine hierzu korrespondierende Klassifikation und Ordnung, die ihre Vertrauenswürdigkeit beschreibt,

z.B. durch die Klassen *Geheimnisträger, Abteilungsleiter, Gruppenleiter* und *einfacher Mitarbeiter*.

Die obligatorische Regel zur Wahrung der Vertraulichkeit ist nun, daß nur solche Subjekte ein Objekt lesen dürfen, deren Vertrauenswürdigkeitsklasse gleich oder höher ist als die korrespondierende Vertraulichkeitsklasse des Objekts. Ebenfalls darf ein Subjekt nur dann Informationen in ein Objekt schreiben, wenn die Vertraulichkeitsklasse des Objektes gleich oder höher als die Vertrauenswürdigkeitsklasse des Subjekts ist. Beide Regeln gemeinsam bewirken, daß Informationen ausschließlich in die ,,aufwärts"-Richtung innerhalb der durch die Ordnung festgelegten Hierarchie wandern.

Rekapitulation des Chinese-Wall Sicherheitsmodells

Chinese-Wall ist eine Sicherheitspolitik, die ein auf britisches Gesetz zur Verhinderung der Ausnutzung von Insiderwissen im Wirtschaftsleben zurückgeht. In seiner Konsequenz besagt dieses Gesetz, daß ein Unternehmensberater einen Klienten nur dann beraten darf, wenn ausgeschlossen ist, daß dieser Berater Insiderwissen über eine in Konkurrenz zu seinem Klienten stehende Firma verwenden kann. Brewer und Nash stellten in [BN89] eine Formalisierung dieser Sicherheitspolitik vor. Wie auch Bell/LaPadula beruht diese auf Subjekt- und Objektattributen, über die dann die Zugriffsregeln formuliert sind. Jedes Subjekt (= Berater) erhält als Attribut seine Zugriffshistorie: einen Vektor aus Objekten, auf denen das Subjekt in seiner Vergangenheit bereits zugegriffen hat. Jedes Objekt repräsentiert im Modell die Informationen über ein bestimmtes Unternehmen (hier ist gegenüber dem Originalmodell etwas vereinfacht worden); es erhält als Attribut eine Konfliktklasse, d.h. die Menge derjenigen anderen Firmenobjekte, zu denen es selbst in Konkurrenz steht.

Zu einem gegebenen Zeitpunkt darf ein Berater s die Daten eines Klienten o nun genau dann lesen, wenn kein Objekt in der Zugriffshistorie von s steht, welches in der Konfliktklasse von o enthalten ist. Firmendaten aktualisieren (d.h. schreiben auf o) darf ein Berater genau dann, wenn er sowohl o lesen darf als auch er niemals in seiner Vergangenheit von *irdendeinem* Objekt $\neq o$ gelesen hat (Vermeidung transitiven Informationsflusses, vgl. Kapitel 3).

4.3.2 MP-freie Interaktionen

In diesem ersten Beispiel soll veranschaulicht werden, daß Metapolitiken in Multipolitiken-Szenarien solche Interaktionen unberührt lassen, die vollständig und ausschließlich im Wirkungsbereich einer einzigen Sicherheitspolitik liegen; Definition 4.4 prägt hierfür den Begriff der MP-Freiheit.

Sei also H eine hierarchische Sicherheitspolitik, C eine Chinese-Wall-Politik. Ist eine

Interaktion (s, o) MP-frei, so gilt gemäß Definition 4.6 für alle Metapolitiken $M_{\{H,C\}}$

$$M_{\{H,C\}}(s, o, f) = \begin{cases} H(s, o, f) & \mid \Pi_s \cap \Pi_o = H \\ C(s, o, f) & \mid \Pi_s \cap \Pi_o = C, \end{cases}$$

da gemäß Korollar 4.5 für MP-freie Interaktionen (s, o) gilt, daß $|\Pi_s \cap \Pi_o| = 1 \wedge |\Pi_s| = |\Pi_o| = 1$; mit anderen Worten, es gilt $\Pi_s = \Pi_o$ und (s, o) ist eine Klasse-1 Interaktion.

Dies besagt nun nichts anderes, als daß für (s, o) genau diejenige Sicherheitspolitik die Zugriffsentscheidung trifft, in deren Domäne s und o gemeinsam liegen. Damit ist unmittelbar klar, daß die Autonomie beider Sicherheitspolitiken in ihren eigenen Domänen durch die Metapolitik gewährleistet ist.

Weiterhin ist es offensichtlich, in welcher Weise der Ansatz skaliert: jede neu hinzukommende Sicherheitspolitik erhöht lediglich die Anzahl der möglichen Alternativen um 1. Im Kapitel über Implementierungskonzepte wird deutlich werden, daß hierzu nicht einmal der Code der Klassifizierungsfunktion c innerhalb der Metapolitikimplementierung verändert werden muß; im Kern besteht diese Funktion aus der Fallunterscheidung

IF $|\Pi_s| = |\Pi_o| = 1$ AND $\Pi_s = \Pi_o$ THEN
  Π_s
ELSIF $|\Pi_s \cap \Pi_o| = 0$ THEN
  V
ELSE
  K
FI .

Unter wirtschaftlichen Aspekten betrachtet ist es gleichfalls offensichtlich, daß in diesem Beispiel durch eine Integration von C und H in eine Metapolitik keine zusätzlichen Kosten entstehen, da keine der existierenden Sicherheitspolitiken hierzu modifiziert werden muß.

4.3.3 Interaktionen im politikfreien Raum

In diesem Beispiel werden die Eigenschaften von Metapolitiken in politikfreien Räumen betrachtet. Interaktionen (s, o) in politikfreien Räumen bilden die Interaktionsklasse 2 und sind dadurch charakterisiert, daß gilt $\Pi_s \cap \Pi_o = \emptyset$. Charakteristisch für die beiden Beispielpolitiken H und C sind also die Zugriffsformen in den Bilder 4.12 und 4.13.

Für $\Pi_s \cap \Pi_o = \emptyset$ und zwei Sicherheitspolitiken H und C definiert Definition 4.6 eine Metapolitik als

$$M_{\{H,C\}}(s, o, f) = V(s, o, f) \text{ mit } Dom_V = Dom_H \cup Dom_C.$$

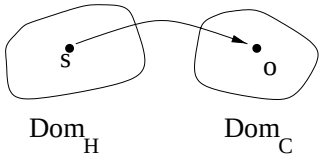


Bild 4.12: Ein Zugriff im politikfreien Raum zwischen H und C

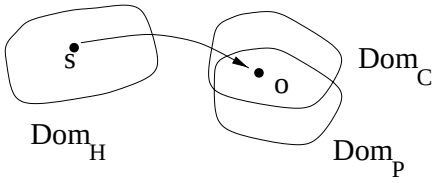


Bild 4.13: Ein Zugriff im politikfreien Raum mit zusätzlichem Domänenkonflikt

Das Verhalten der Metapolitik im politikfreien Raum wird also definiert durch die Vollständigkeitspolitik V. Da V gleichsam wie H und C eine Sicherheitspolitik ist, zu deren Formulierung und Implementierung dieselben Methoden, Paradigmen und Werkzeuge verwendet werden wie für H und C, kann die Formulierung von Regeln über Interaktionen im politikfreien Raum mit derselben Ausdruckskraft erfolgen, die allgemein für die Formulierung von Sicherheitspolitiken zur Verfügung steht.

Es spannt sich daher ein ganzes Spektrum alternativer Strategien auf, angefangen von der Definition einer vollständig neuen und autonomen Sicherheitpolitik für Interaktionen im politikfreien Raum bis hin zu einer Strategie, die – z.B. aus Kostengründen – auf weitestgehender Weiterverwendung der existierenden Sicherheitspolitiken beruht. Wird aus Gründen der Wirtschaftlichkeit die Entscheidung getroffen, die Regeln für Interaktionen zwischen den Domänen mittels der existierenden Sicherheitspolitiken zu beschreiben, so stößt man allerdings auf ein Problem sehr grundsätzlicher Art: eine obligatorische Sicherheitspolitik arbeitet typischerweise mit Entitätenattributen, die charakteristisch sind für die jeweilige Sicherheitspolitik; Chinese-Wall arbeitet mit der Zugriffshistorie eines Subjekts, während Bell/LaPadula - Subjekte mit Vertrauenswürdigkeitsklassen versehen sind. Der unverträgliche Typ dieser Attribute verhindert eine direkte Kombination von Sicherheitspolitiken, wie sie ansonsten durch die Regel ,,beide Sicherheitspolitiken müssen einem Zugriff zustimmen" mittels des Ausdrucks $V(s, o, f) := H(s, o, f) \wedge C(s, o, f)$ formulierbar wäre.

Eine sehr universelle Lösung dieses Problems erhält man durch die Einführung von Entitäten, welche innerhalb einer Sicherheitsdomäne als Repräsentanten von Entitäten fremder Sicherheitsdomänen auftreten. Diese Repräsentanten können von dem Eigentümer einer Politikdomäne so attributiert werden, daß seine Attribute die der re-

präsentierten Entität zugestandenen Rechte exakt repräsentieren. Hiermit lassen sich dann eine Vielzahl von Problemen bei der Interdomänenkommunikation lösen. Sie lösen beispielsweise das in [FM98] geschilderte Problem der *interdomain transfer objects*; gleichfalls sind sie als Software-Firewall auffaßbar, da sie einen wohldefinierten Punkt für die Beschreibung der Rechte bei einer Kommunikation mit Fremddomänen darstellen. Schließlich lösen sie das Problem inkompatibler Attribute und ermöglichen so die gleichzeitige Anwendung mehrerer Sicherheitspolitiken.

Die Zuordnung zwischen Entität und Repräsentant wird durch eine Repräsentantenabbildung rep hergestellt, deren Definition innerhalb der Sicherheitspolitik der Repräsentantenentität erfolgt.

Definition 4.7

Eine *Repräsentantenabbildung* $rep_{P,Q} : Dom_P \rightarrow Dom_Q$ ist eine Abbildung zwischen zwei Politikdomänen Dom_P und Dom_Q, die jeder Entität aus Dom_P eine repräsentierende Entität aus Dom_Q zuordnet.

Im allgemeinen ist rep nicht injektiv; zwei verschiedene Elemente aus Dom_P können durchaus auf denselben Repräsentanten in Dom_Q abgebildet werden.

Unternehmenspolitische Entscheidungen hinsichtlich der Zugriffsregeln in politikfreien Räumen werden nun im Metapolitikmodell umgesetzt durch Definition geeigneter Repräsentantenentitäten und den zugehörigen Repräsentantenabbildungen zwischen den Domänen der beteiligten Sicherheitspolitiken. Hierdurch wird dann die Verwendung existierender Sicherheitspolitiken auch für Zugriffe im politikfreien Raum möglich, eine Eigenschaft, die die Autonomie der beteiligten Institutionen und die Wirtschaftlichkeit des Ansatzes befördert. Das obige Beispiel der Regel „beide Sicherheitspolitiken müssen einem Zugriff im politikfreien Raum zustimmen" läßt sich nun so formulieren:

$$V(s,o,f) := H(s, rep_{C,H}(o), f) \wedge C(rep_{H,C}(s), o, f) \text{ mit } s \in Dom_H, o \in Dom_C \ .$$

Hierdurch wird nun eine Algebra begründet, in der Sicherheitspolitiken mittels der Operatoren $\wedge$ und $\vee$ verknüpft werden. Wie man leicht sieht, handelt es sich hierbei um eine boolesche Algebra, die den drei bekannten elementaren Gesetzen

(1) $P \vee Q = Q \vee P$

(2) $(P \vee Q) \vee R = P \vee (Q \vee R)$

(3) $\neg(\neg P \vee \neg Q) \vee \neg(\neg P \vee Q) = P$

[CR73] genügt. Somit können nun Sicherheitspolitiken durch *Rechenausdrücke* beschrieben werden, und Konfliktpolitiken lassen sich durch Kombination der existierenden Sicherheitspolitiken komponieren. Dies kann unmittelbar für den Umgang mit Konfliktsituationen genutzt werden: Bild 4.13 zeigt eine Situation, in der zusätzlich

zum politikfreien Raum noch ein Domänenkonflikt besteht. Die Repräsentantenabbildung liefert nun einen Mechanismus, mit dessen Hilfe eine Konfliktlösungsstrategie unter Einbeziehung aller betroffenen Sicherheitspolitiken beschrieben werden kann.

Soll beispielsweise für die in Bild 4.13 dargestellte Situation die Konfliktstrategie definiert werden, daß eine Interaktion zwischen s und o genau dann zulässig ist, wenn H und mindestens eine der beiden Sicherheitspolitiken C und P zustimmen, so kann dies unter Verwendung der Politikalgebra durch den Ausdruck

$$V(s,o,f) := \\ (H(s, rep_{C,H}(o), f) \wedge H(s, rep_{P,H}(o), f)) \ \wedge \ (C(rep_{H,C}(s), o, f) \vee P(rep_{H,P}(s), o, f))$$

oder kurz

$$V := H \wedge (C \vee P)$$

beschrieben werden. Zur Reduzierung boole'scher Ausdrücke gibt es eine Vielzahl von Gesetzmäßigkeiten, die zur Minimierung der an einem Zugriff beteiligten Sicherheitspolitiken genutzt werden können – ein wesentliches Merkmal, um die Effizienz dieses Ansatzes zu befördern.

4.3.4 Interaktionen im Konfliktraum

In diesem Beispiel werden die Eigenschaften von Metapolitiken in Konflikträumen betrachtet. Interaktionen (s,o) in Konflikträumen sind dadurch charakterisiert, daß gilt $|\Pi_s \cap \Pi_o| \geq 1$ und $\exists e \in \{s,o\} : |\Pi_e| > 1$; gemäß der Klassifikation ist dies die Interaktionsklasse 3. Durch die globale Sicherheitspolitik G des Beispielszenariums treten Konfliktsituationen immer dann auf, wenn mindestens eine der beteiligten Entitäten in einer weiteren Sicherheitsdomäne liegt.

In bezug auf die beiden Beispielpolitiken H und C ist die Metapolitik im Konfliktraum nach Definition 4.6 beschrieben durch

$$M_{\{H,C\}}(s,o,f) = K(s,o,f) \text{ mit } Dom_K = Dom_H \cup Dom_C.$$

Hinsichtlich einer Diskussion der Eigenschaften dieses Beispiels kann in allen Punkten auf das vorangehende Kapitel verwiesen werden; einerseits kann wiederum im Konfliktfall eine vollständig neue Sicherheitspolitik definiert werden (z.B. die triviale Sicherheitspolitik $K(s,o) = \{unzulässig\}$ für alle s und o aus dem Konfliktraum) oder aber mittels der eingeführten Repräsentantenabbildung und der Politikalgebra ein Ausdruck formuliert werden, der unter Verwendung der existierenden Sicherheitspolitiken die Regeln für Interaktionen im Konfliktraum beschreibt.

4.4 Analyse von Metapolitiken

Die Sicherheitseigenschaften eines IT-Systems hängen in hohem Maße von der Qualität der Spezifikation und Implementierung seiner Sicherheitspolitiken ab. Kapitel 3 stellte als ein Mittel zum Erzielen eines hohen Qualitätsniveaus die Sicherheitsmodelle vor, durch die eine formale Analyse konkreter Sicherheitspolitiken hinsichtlich konkreter Sicherheitseigenschaften ermöglicht wird.

Dieses Kapitel befaß sich nun mit der Analyse von Metapolitiken. Im Mittelpunkt wird dabei ihre *Safety*-Eigenschaft stehen, die bereits in informeller Form im Kontext des HRU-Kalküls in Kapitel 3 erwähnt wurde. Safety-Eigenschaften von Sicherheitspolitiken charakterisieren die Freiheitsgrade, mit denen sich Rechte von Entitäten unter einer gegebenen Sicherheitspolitik ausbreiten können; sie befassen sich also mit Fragestellungen der Art

> *„Kann es jemals geschehen, daß unter einer bestimmten Sicherheitspolitik ein bestimmtes Subjekt ein bestimmtes Recht erhält?"*

Fragestellungen dieser Art sind sehr grundsätzlicher Natur, und ihre Beantwortung steht daher an prominenter Stelle bei der Analyse einer Sicherheitspolitik. In Szenarien wie etwa einem Krankenhausinformationssystem streben Safety-Analysen dann die Beantwortung von Fragen der Art

> *„Kann es jemals unter einem gegebenen konkreten Sicherheitsmodell geschehen, daß Mitarbeiter der Klinikapotheke ein Schreibrecht auf eine Partientenakte erhalten?"*

an.

Die Durchführbarkeit von Safety-Analysen war eine der wesentlichen Motivationen für die Definition des HRU-Kalküls. In anderen formalen Sicherheitsmodellen, etwa im Bell/LaPadula-Modell, findet sich dieses Problem in verwandter Form; es wird dort als *Bell/ LaPadula-Security* bezeichnet und ist Gegenstand des *Basic Security Theorems*.

Im Kontext der Metapolitiken ist das Safety-Problem von gleichsam grundsätzlicher Natur. In Multipolitiken-Szenarien kontrollieren Metapolitiken sämtliche Interaktionen zwischen den Politikdomänen und können somit stark in die Autonomie dieser Sicherheitspolitiken eingreifen. Eine typische Safety-Fragestellung ist hier beispielsweise

> *„Kann es jemals geschehen, daß während eines Gastaufenthaltes von Ann in der Entwicklungsabteilung andere Mitglieder dieser Abteilung Zugriff auf Ann's Dokumente in ihrer Heimatdomäne erhalten?"*

Der Eingriff in die Autonomie der beteiligten Sicherheitspolitiken und das hiermit verbundene Risiko ist nur dann tragbar, wenn die Ausbreitung von Rechten unter

einer Metapolitik ein berechenbares Problem darstellt. Hierfür ist aber nur dann eine Lösung möglich, wenn es gelingt, das Safety-Problem für Metapolitiken zu lösen.

Thema dieses Kapitels sind daher die Safety-Eigenschaften von Metapolitiken. Hierzu wird zunächst die Definition der Safety-Eigenschaft für Sicherheitspolitiken rekapituliert. Hierauf aufbauend wird anschließend die Safety einer Metapolitik definiert. In einem dritten Teil werden dann Möglichkeiten und Grenzen des Nachweises von Safety-Eigenschaften betrachtet. Der dabei verfolgte Ansatz geht davon aus, daß Kenntnisse der Safety-Eigenschaften über die einzelnen Sicherheitspolitiken eines Multipolitiken-Szenarium vorliegen. Die entwickelten Sätze zeigen dann, unter welchen Voraussetzungen und in welcher Form aus den Eigenschaften der einzelnen Sicherheitspolitiken auf die Eigenschaften der gesamten Metapolitik geschlossen werden kann.

Die Beweisführung über Safety-Eigenschaften wird im konkreten HRU-Kalkül geführt; dieser Kalkül wurde explizit mit dem Ziel einer Safety-Analyse entwickelt und hat einen weithin prägenden Einfluß auf spätere Sicherheitsmodelle ausgeübt.

4.4.1 HRU-Safety

Der HRU-Kalkül ist ein einfacher Formalismus zur Formulierung von Zugriffssteuerungsmodellen und zur Analyse ihrer Safety-Eigenschaften. Der Formalismus basiert auf Lampson's Zugriffssteuerungsmodell (vgl. Abschnitt 3.3.1) und verwendet Zustandsautomaten zur Modellierung dynamischen Verhaltens. Jeder Zustand ist ein Tripel (S, O, ZM), wobei S eine Subjektmenge, O eine Objektmenge und ZM ein Zugriffsmatrix $ZM : S \times O \rightarrow \wp(R)$ ist, die in jeder Zelle $ZM(s, o)$ die Rechte von s bezüglich o in Form einer Teilmenge der endlichen Rechtemenge R enthält.

Definition 4.8 (HRU-Modell einer Sicherheitspolitik)
Ein *HRU-Modell einer Sicherheitspolitik* ist ein Zustandsautomat (Z, E, δ, z^0) mit

$Z = (S, O, ZM)$ als Zustandsmenge mit
S als Subjektmenge,
O als Objektmenge,
$ZM : S \times O \rightarrow \wp(R)$ als Zugriffsmatrix über einer endlichen Rechtemenge R;
$E = (F, X)$ als Eingabemenge mit $X = (S \cup O)^k$;
$\delta : Z \times E \rightarrow Z$ als Zustandsübergangsfunktion, und
z^0 als Initialzustand[2].

Jede Eingabe für diesen Automaten modelliert den Aufruf einer konkreten applikationsspezifischen Operation, bei der k Subjekte und Objekte involviert sind. δ modelliert nun die sicherheitsrelevante Semantik einer solchen Operation durch einen Ausdruck der allgemeinen Form

[2] z^0 bezeichnet hier einen konkreten Zustand in Z, nicht etwa einen 0-dimensionalen Vektor!

$$\begin{aligned}\delta(z,(f,x)) := \text{IF } & r_1 \in ZM(x_{s_1}, x_{o_1}) \text{ AND} \\ & \ldots \\ & r_m \in ZM(x_{s_m}, x_{o_m}) \\ \text{THEN} & \\ & p_1; \ldots; p_n \\ \text{FI}, & \end{aligned}$$

wobei z ein Zustand aus Z ist, f eine applikationsspezifische Operation aus F, x ein k-dimensionaler Vektor von Subjekten und Objekten aus $S \cup O$ und $r_1 \ldots r_m$ Rechte aus R sind; $s_1 \ldots s_m$ und $o_1 \ldots o_m$ sind Vektorindizes zwischen 1 und k. Die $p_1 \ldots p_n$ stammen aus der Menge H der HRU-Primitiven

$$\begin{aligned}H = \ & \{\textit{enter } r \textit{ into } M(x_s, x_o),\ \textit{delete } r \textit{ from } M(x_s, x_o), \\ & \textit{create subject } x_s,\ \textit{create object } x_o, \\ & \textit{destroy subject } x_s,\ \textit{destroy object } x_o\}\end{aligned}$$

und überführen den Automaten in einen neuen Zustand; s und o sind hierbei wieder Vektorindizes zwischen 1 und k.

Die Safety eines Zustandes wird nun definiert über sämtliche möglichen Folgen von Zustandswechseln, ausgehend von diesem Zustand.

Definition 4.9 (HRU-Safety einer Sicherheitspolitik)
Sei $\alpha \circ e$ eine Eingabefolge aus E^*, die aus einer einzelnen Eingabe $\alpha \in E \cup \{\varepsilon\}$ (ε ist die leere Folge) und einer Folge $e \in E^*$ besteht. Dann sei $\delta^* : Z \times E^* \to Z$ definiert durch

$$\begin{aligned}&\delta^*(z, \varepsilon) = z \\ &\delta^*(z, \alpha \circ e) = \delta^*(\delta(z, \alpha), e).\end{aligned}$$

Ein konkreter Zustand $z^0 = (S^0, O^0, ZM^0)$ in einem gegebenen HRU-Modell wird genau dann als *HRU-safe* bezüglich eines Rechtes $r \in R$ bezeichnet, wenn es ausgehend von z^0 keine Folge von Eingaben aus E^* gibt, die r in eine Matrixzelle einträgt, welche r nicht bereits in z^0 enthält, d.h. für alle $s \in S$ und $o \in O$ und alle $e \in E^*$ gilt

$$r \notin ZM^0(s, o) \Rightarrow r \notin ZM^i \text{ mit } z^i = \delta^*(z^0, e).$$

Eine der wesentlichen Leistungen des HRU-Modells ist es, daß es Aussagen zur Komplexität der Entscheidung trifft, ob bei gegebenem Modell ein Anfangszustand bezüglich eines bestimmten Rechtes HRU-safe ist; hierzu stellten Harrison, Ruzzo und Ullman 1976 zwei Sätze auf:

Satz 4.10 (Entscheidbarkeit des Safety-Problems für allgemeine HRU-Modelle [HRU76])
Das Safety-Problem für allgemeine HRU-Modelle ist nicht entscheidbar.

Dieses Ergebnis ist zunächst einmal ein herber Rückschlag, obwohl es im Grunde zu erwarten war: da Subjekte im allgemeinen autorisiert sind, neue Subjekte zu erzeugen, ist das allgemeine HRU-Modell unbeschränkt, und es ist nicht ohne weiteres zu erwarten, daß starke Aussagen über die Safety-Eigenschaften ohne ein Opfer an Ausdruckskraft möglich sind. Schränkt man jedoch die Allgemeinheit des Modells geeignet ein, so erhält man für bestimmte Einschränkungen doch die Entscheidbarkeit:

Satz 4.11 (Entscheidbarkeit des Safety-Problems für monooperationale HRU-Modelle [HRU76])
Es gibt einen Algorithmus, der entscheidet, ob ein gegebenes monooperationales HRU-Modell für einen Zustand z^0 und ein Recht r HRU-safe ist oder nicht.

Dabei bedeutet Monooperationalität, daß der Anweisungsteil einer HRU-Operation (vgl. Bild 3.1) lediglich eine einzige Primitivoperation enthalten darf. Der Beweis dieser Sätze und eine Interpretation dieser Ergebnisse findet sich in [HRU76].

Die Sätze 4.10 und 4.11 machen einen Konflikt zwischen der Ausdruckskraft des Kalküls und der Analysefähigkeit der in ihm formulierten Modelle deutlich: auf der einen Seite existiert ein ausdrucksstarker Kalkül, mit dem ein breites Spektrum von Zugriffssteuerungspolitiken modelliert werden kann; für damit formulierte Modelle läßt sich aber kein allgemeiner Algorithmus angeben, der das Safety-Problem entscheiden kann. Auf der anderen Seite verlangt Satz 4.11 in der Monooperationalität eine Einschränkung des allgemeinen Kalküls, die zwar die Safety-Eigenschaft nun entscheidbar macht, jedoch zu einer erheblichen Einschränkung der Ausdruckskraft des Kalküls führt: nicht einmal etwas scheinbar so einfaches wie die Schutzmechanismen des Unix-Betriebssystems lassen sich mit monooperationalen HRU-Modellen ausdrücken; das Erzeugen einer Datei ist in Unix stets auch mit einer initialen Rechtevergabe für das erzeugende Subjekt verbunden, und in einer HRU-Modellierung würde dies bedeuten, daß eine entsprechende HRU-Operation mindestens aus den zwei Primitiven *create object o* und *enter r into ZM(s,o)* besteht.

Aus diesem Dilemma sind in den Folgejahren weitere HRU-Varianten und Ableger entstanden, in denen unter bestimmten Randbedingungen das Safety-Problem entscheidbar ist; diese werden unter anderem in [HR78, LS78, AS91, San92b] diskutiert. Die folgenden Sätze 4.12 und 4.13 fassen einige signifikante Ergebnisse zusammen.

Satz 4.12 (Entscheidbarkeit des Safety-Problems für statische HRU-Modelle [HRU76])
Es gibt einen Algorithmus, der entscheidet, ob ein gegebenes statisches HRU-Modell für einen Zustand z^0 und ein Recht r HRU-safe ist oder nicht.

Statische HRU-Modelle sind dadurch charakterisiert, daß nach ihrer Initialisierung keine weiteren Subjekte und Objekte mehr erzeugt werden; ihr Autorisierungsschema enthält keinerlei *create subject* und *create object* - Primitiven. Dieser Modelltyp ist

insbesondere bei Anwendungen relevant, die von vorn herein mit statischen Entitätenmengen operieren; Anwendungen also, deren dynamisches Verhalten bekannt ist und die häufig in Realzeit-Szenarien anzutreffen sind.

Satz 4.13 (Entscheidbarkeit des Safety-Problems für monotone monokonditionale HRU-Modelle [HR78])
Es gibt einen Algorithmus, der entscheidet, ob ein gegebenes monotones und monokonditionales HRU-Modell für einen Zustand z^0 und ein Recht r HRU-safe ist oder nicht.

Monotone HRU-Modelle sind dadurch charakterisiert, daß weder Subjekte noch Objekte gelöscht noch vorhandene Rechte entzogen werden können, daß also keine *destroy*- oder *delete*-Primitiven im Autorisierungsschema vorkommen. Monokonditionale HRU-Modelle haben ausschließlich Regeln im Autorisierungsschema, deren Bedingungsteil aus lediglich einer einzigen Bedingung besteht. Wie nahe man sich hier bereits an der Grenze der Entscheidbarkeit befindet zeigt die Tatsache, daß monotone, aber bikonditionale Modelle bereits nicht mehr entscheidbar sind [HR78].

Monotone monokonditionale HRU-Modelle werden im strikten Umfeld der klassischen Bell/LaPadula-artigen Anwendungssysteme mit „stiller" Klassifikationsfunktion angetroffen; die individuellen Vertrauensklassen der Entitäten ändern sich hierbei während der Laufzeit der Sicherheitspolitik nicht, und somit bleibt auch die Zugriffsmatrix unverändert (siehe das Bell/LaPadula *Basic Security Satz*). Typischerweise sind dies Systeme zur Dokumentenverwahrung, in denen Dokumente bis in alle Ewigkeiten aufbewahrt werden und in denen seltene administrative Vorgänge wie eine Reklassifikation von Dokumenten durch besonders vertrauenswürdige Personen am Sicherheitsmodell vorbei vorgenommen werden.

Als Ergebnis dieses Abschnitts existiert nun eine präzise Definition eines HRU-Modells einer Sicherheitspolitik und des Begriffs der Safety im Kontext der HRU-Modelle. Es wurden Sätze aufgeführt, die hinreichende Bedingungen für die Entscheidbarkeit des Safety-Problems nennen; dabei wurde deutlich, daß Aussagen über die Entscheidbarkeit des Safety-Problems nur unter teilweise bedeutenden Einschränkungen der Ausdruckskraft des Kalküls möglich werden.

4.4.2 Safety von Metapolitiken

Dieser Abschnitt setzt sich mit den Safety-Eigenschaften von Metapolitiken auseinander. Ziel dabei ist es, für Metapolitikmodelle Aussagen über ihre HRU-Safety zu gewinnen, die sich auf den HRU-Safetyeigenschaften der kontrollierten Sicherheitspolitiken abstützen. In Analogie zur Vorgehensweise im vorangehenden Abschnitt wird zunächst definiert, was ein HRU-Modell einer Metapolitik ist. Hierauf aufbauend wird anschließend dann die Safety eines solchen Modells definiert. In einem dritten Teil

werden dann Möglichkeiten und Grenzen des Nachweises von Safety-Eigenschaften betrachtet; zu diesem Zweck werden Sätze über die Entscheidbarkeit des Safety-Problems für HRU-Metapolitikmodelle aufgestellt und bewiesen.

Im folgenden wird vorausgesetzt, daß alle beteiligten Sicherheitspolitiken – die P_i, die Vollständigkeitspolitik V und die Konfliktpolitik K HRU-modelliert sind. Dann existiert für jedes P_i sowie für V und K der in Abschnitt 4.4.1 beschriebene Zustandsautomat, und aus den Definitionen 4.6 und 4.8 läßt sich die folgende Definition herleiten.

Definition 4.14 (HRU-Modell einer Metapolitik)

Sei I eine endliche Indexmenge und jedes $P \in \{P_i\}_{i \in I}$ eine im HRU-Kalkül modellierte Sicherheitspolitik gemäß Definition 4.8; seien ferner V und K ebenfalls so modellierte Sicherheitspolitiken. Das *HRU-Modell einer Metapolitik* M[3], $M = (\{P_i\}_{i \in I}, V, K, c)$ ist ein Zustandsautomat $(Z_M, E_M, \delta_M, z_M^0)$ mit

$Z_M = (S_M, O_M, ZM_M)$ mit
$S_M = \bigcup_{i \in I} S_{P_i}$ als Subjektmenge,
$O_M = \bigcup_{i \in I} O_{P_i}$ als Objektmenge,
$R_M = \bigcup_{i \in I} R_{P_i}$ als endlicher Rechtemenge,
$ZM_M : S_M \times O_M \to \wp(R_M)$ als Zugriffsmatrix definiert durch

$$ZM_M(s,o) = \begin{cases} ZM_{P_j}(s,o) & \mid c(s,o) = P_j \\ ZM_V(s,o) & \mid c(s,o) = V \\ ZM_K(s,o) & \mid c(s,o) = K, \end{cases}$$

$E_M = \bigcup_{i \in I} F_{P_i} \times (S_M \cup O_M)^k$ als Eingabemenge,

$\delta_M : Z_M \times E_M \to Z_M$ als Transitionsfunktion definiert durch

$$\delta_M(z_M, (f,x)) = \begin{cases} \delta_{P_j}(z_j, (f,x)) & \mid c_k(x) = P_j \\ \delta_V(z_V, (f,x)) & \mid c_k(x) = V \\ \delta_K(z_K, (f,x)) & \mid c_k(x) = K \end{cases}$$

$z_M^0 = (S_M^0, O_M^0, ZM_M^0)$ als initialer Zustand definiert durch
$S_M^0 = \bigcup_{i \in I} S_{P_i}^0$
$O_M^0 = \bigcup_{i \in I} O_{P_i}^0$

$$ZM_M^0(s,o) = \begin{cases} ZM_{P_j}^0(s,o) & \mid c(s,o) = P_j \\ ZM_V^0(s,o) & \mid c(s,o) = V \\ ZM_K^0(s,o) & \mid c(s,o) = K. \end{cases}$$

Obwohl diese Definition auf den ersten Blick kompliziert erscheinen mag, ist sie es nicht wirklich: tatsächlich beschreibt sie eine geradlinige Komposition der HRU-Modelle der beteiligten Sicherheitspolitiken, wobei Unvollständigkeiten und mögliche Konflikte durch die bekannte Klassifizierungsfunktion c identifiziert werden und mittels der HRU-Modelle von V und K behandelt werden.

[3] Wenn immer es zu keinen Mißverständnissen führen kann, wird im folgenden kurz M anstelle von $M_{\{P_i\}}$ geschrieben

Da im HRU-Kalkül bei einer anwendungsspezifischen Operation k Subjekte und Objekte beteiligt sein können, ist es notwendig, die Klassifizierungsfunktion c auf k Dimensionen zu erweitern. Sei also I eine endliche Indexmenge, $\{P_i\}_{i \in I}$ eine Menge von Sicherheitspolitiken, $x \in (\bigcup_{i \in I} Dom_{P_i})^k$. Die Klassifizierungsfunktion c_k wird für einen k-dimensionalen Vektor x, bestehend aus Subjekten und Objekten, definiert durch

$$c_k(x) = \begin{cases} P_j & | \ \forall 1 \leq l \leq k : |\Pi_{x_l}| = 1 \ \wedge \ \Pi_{x_l} = \{P_j\} \\ V & | \ |\bigcap_{1 \leq l \leq k} \Pi_{x_l}| \ = \ 0 \\ K & | \ |\bigcap_{1 \leq l \leq k} \Pi_{x_l}| \geq 1 \ \wedge \ \exists x_l, 1 \leq l \leq k : |\Pi_{x_l}| > 1. \end{cases}$$

Die bisherige Definition von c (Definition 4.6) wird mit $k = 2$ zu einem Spezialfall dieser allgemeinen Fassung.

Als Beispiel für die Anwendung von Definition 4.14 betrachte man zwei Sicherheitspolitiken P und Q mit $S_P = \{s_{p_1}, ..., s_{p_k}, s_x\}, O_P = \{o_{p_1}, ..., o_{p_l}, o_x\}, S_Q = \{s_{q_1}, ..., s_{q_m}, s_x\}$ und $O_Q = \{o_{q_1}, ..., o_{q_n}, o_x\}$. Bild 4.14 zeigt die Zugriffsmatrix der zugehörigen Metapolitik ZM_M, konstruiert nach Definition 4.14.

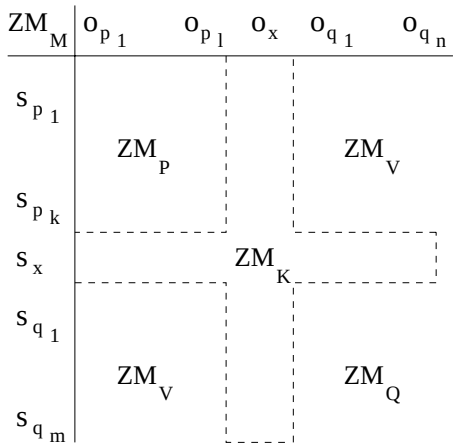


Bild 4.14: Die Zugriffsmatrix einer Metapolitik über P und Q

Man beachte, daß Definition 4.14 bei gegebenen HRU-Modellen für V, K und die P_i einen präzisen Bauplan für das HRU-Modell der zugehörigen Metapolitik angibt; dies ist von besonderer praktischer Bedeutung, da der Bauplan zu einer automatisierten Konstruktion einer Metapolitik genutzt werden kann, wenn erst einmal P_i, V und K in Form eines HRU-Modells vorliegen.

Im allgemeinen ist die Zustandsübergangsfunktion eines HRU-Modells keine partielle Funktion; eine Sicherheitspolitik ist definiert für jede sicherheitsrelevante Operation innerhalb ihrer Domäne. Allerdings impliziert $Dom_V = Dom_K = \bigcup_{i \in I} Dom_{P_i}$, daß

δ_V und δ_K definiert sind für jegliche Interaktion innerhalb der Vereinigung sämtlicher Politikdomänen, was tatsächlich aber nicht der diesen Funktionen zugedachten Rolle entspricht. Während einerseits zwar die Konstruktion von δ_M in Definition 4.14 die Anwendung der Funktionen auf „falsche" Interaktionen verhindert und auf diese Weise die Autonomie der P_i für Klasse-1 - Interaktionen innerhalb deren originärer Domäne herstellt, so wird eine Diskussion der Safety-Eigenschaften von Metapolitiken einfacher, wenn eine entsprechende Beschränkung unmittelbar den Definitions- und Wertebereichen der δ's von V und K auferlegt wird. Ohne Beschränkung der Allgemeinheit soll daher der Begriff der P_i-*Autonomie* eingeführt werden.

Definition 4.15 (P_i-Autonomie)

Ein HRU-Modell (Z, E, δ, z^0) mit $Z = (S, O, ZM)$ heißt *eingeschränkt* auf eine Entitätenmenge $X \subseteq S \cup O$, wenn $\delta : Z \times E \to Z$

a) im Definitionsbereich eingeschränkt ist auf solche Entitäten, die in X liegen, d.h. $E = (F, X^k)$ gilt.

b) im Wertebereich eingeschränkt ist durch $Z = (S_{|X}, O_{|X}, ZM_{|X})$, wobei $ZM_{|X}$ die Einschränkung der Matrix ZM auf solche Zellen (s, o) bezeichnet, für die s und o in X enthalten sind.

Ein HRU-Modell einer Metapolitik $M = (\{P_i\}_{i \in I}, V, K, c)$ heißt dann P_i-*autonom*, wenn V eingeschränkt ist auf $\{x|\ x \in (S_M \cup O_M)^k \ \wedge\ c_k(x) = V\}$ und K eingeschränkt ist auf $\{x|\ x \in (S_M \cup O_M)^k \ \wedge\ c_k(x) = K\}$.

P_i-Autonomie garantiert, daß Vollständigkeits- und Konfliktpolitiken niemals mit solchen Interaktionen interferieren, die irgendein P_i autonom bearbeiten kann. Weiterhin ist hiermit klar, welche Freiheiten noch für die V und K P_i-autonomer Metapolitiken übrig bleiben. Da δ nur noch auf der eingeschränkten Zugriffsmatrix arbeitet, können die Autorisierungsschemata von V und K weder Subjekte noch Objekte erzeugen oder löschen noch können Rechte betrachtet oder modifiziert werden, die im Autonomiebereich einer der P_i liegen. Konkret bedeutet dies, daß die Autorisierungsschemata der Vollständigkeits- und Konfliktpolitiken P_i-autonomer Metapolitiken lediglich die HRU-Primitiven *enter r into ZM(s,o)* und *remove r from ZM(s,o)* aus Bild 3.2 enthalten, wobei s und o in der einschränkenden Menge X der Definition 4.15 enthalten sein müssen.

Zur Veranschaulichung sollen nun zwei Beispiele betrachtet werden, bei denen die Metapolitiken nach dem Bauplan der Definition 4.14 erstellt sind. Als Beispiel sollen wiederum einzelne Szenarien aus dem Technologiekonzern verwendet werden.

Beispiel 1

Bild 4.15 zeigt Ann und Joe, beide Angestellte des Konzerns, in ihren jeweiligen Domänen *KSL* und *Q*.

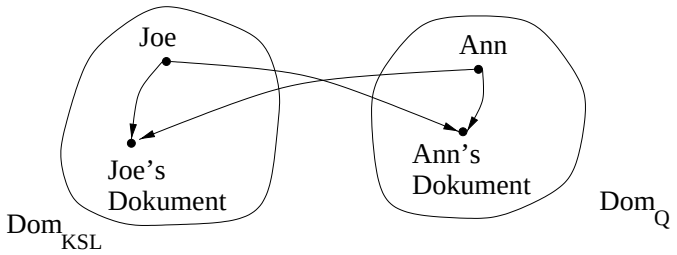


Bild 4.15: Ein Zweipolitiken-Szenarium mit Interaktionen zwischen den Domänen

Alle Zugriffe in diesem Szenarium fallen in die Interaktionsklassen 1 und 2: (*Ann, Ann's Dokument*) und (*Joe, Joe's Dokument*) sind Klasse-1 Interaktionen, und (*Ann, Joe's Dokument*) und (*Joe, Ann's Dokument*) gehören zur Klasse 2.

Die Metapolitik für die Sicherheitspolitiken *KSL* und Q lautet hier $(\{KSL, Q\}, V, K, c)$, ihr HRU-Modell ist $((\{Joe,\ Ann\}, \{Joe's\ Dokument,\ Ann's\ Dokument\}, ZM), E, \delta, q^0)$. Bild 4.16 zeigt *ZM*, wobei in den Zellen die jeweils gemäß der Klassifizierungsfunktion c für den Zugriff zuständige Sicherheitspolitik aufgeführt ist; *ZM* wird konstruiert, indem dann jeweils die politikspezifischen Zugriffsmatrizen ZM_{KSL} und ZM_Q dort eingesetzt werden.

ZM	*Joe's Dokument*	*Ann's Dokument*
Joe	*KSL*	*V*
Ann	*V*	*Q*

Bild 4.16: Die Zugriffsmatrix der Metapolitik für das Zweipolitiken-Szenarium

Beispiel 2

Das zweite Beispiel berücksichtigt, daß Q's Domäne eingebettet ist in die Domäne des F&E-Unternehmensbereichs (vgl. auch Bild 4.10). In dieser arbeitet ein dritter Angestellter, Jerry.

Immer wenn entweder Ann oder Ann's Dokument an einem Zugriff beteiligt ist, entsteht nun ein möglicher Konflikt zwischen Q und *KSL*. Bild 4.18 zeigt die Zugriffsmatrix der resultierenden Metapolitik.

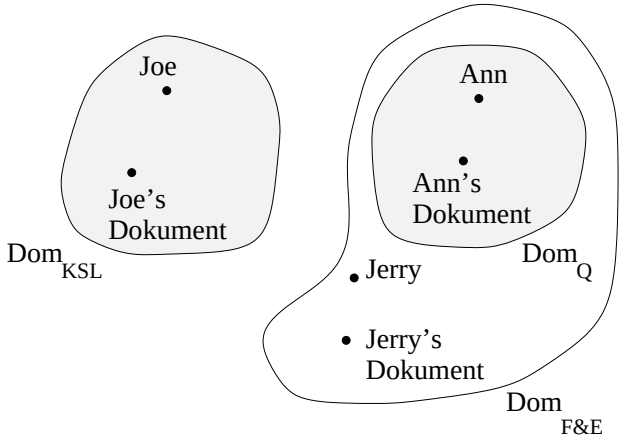


Bild 4.17: Ein Dreipolitiken-Szenarium mit Interaktionen zwischen den Domänen

ZM	*Joe's Dokument*	*Ann's Dokument*	*Jerry's Dokument*
Jerry	*V*	*K*	*R&D*
Joe	*Q*	*K*	*V*
Ann	*K*	*K*	*K*

Bild 4.18: Die Zugriffsmatrix der Metapolitik für das Dreipolitiken-Szenarium

Definition 4.14 hat gezeigt, auf welche Weise zu einer Menge von HRU-modellierten Sicherheitspolitiken P_i ein HRU-Metapolitikmodell konstruiert werden kann. Im folgenden werden nun Sätze genannt, die es erlauben, bekannte HRU-Safety-Eigenschaften der P_i auf die Ebene der Metapolitiken über die P_i zu „liften" und auf diese Weise Aussagen über die Analysierbarkeit von Metapolitiken zu gewinnen.

Satz 4.16 (Entscheidbarkeit des Safety-Problems für allgemeine HRU-Metapolitikmodelle)
Das Safety-Problem für allgemeine HRU-Metapolitikmodelle ist nicht entscheidbar.

Da jedes allgemeine HRU-Metapolitikmodell nach Definition 4.14 insbesondere ein allgemeines HRU-Modell ist, folgt dies unmittelbar aus Satz 4.10. Wie der folgende Satz zeigt, reicht tatsächlich bereits eine schwächere Annahme aus, um das Safety-Problem für HRU-Metapolitikmodelle unentscheidbar zu machen.

Satz 4.17 (Entscheidbarkeit des Safety-Problems für Metapolitik- modelle über nicht entscheidbare P_i)
Sei $\{P_i\}_{i \in I}$ eine Menge von Sicherheitspolitiken und $M = (Z_M, E_M, \delta_M, z_M^0)$ ein P_i-autonomes HRU-Modell einer Metapolitik über den P_i. Existiert mindestens ein $P_j : Dom_{P_j} \cap Dom_{P_i} = \emptyset$ für alle $i, j \in I, i \neq j$, für das das HRU-Safety-Problem nicht allgemein entscheidbar ist, so ist auch das Safety-Problem für M nicht allgemein entscheidbar.

Beweis

Der Beweis folgt dem Schema, die lokale Nichtentscheidbarkeit des Safety-Problems in P_j auf die Ebene der Metapolitik hochzuziehen.

Sei zunächst eine Filterfunktion *purge* definiert, die aus einer Folge $e \in {E_M}^*$ von Eingaben alle solche Elemente entfernt, die nicht von einer gegebenen Sicherheitspolitik P autonom behandelt werden können:

$$purge(\varepsilon, P) = \varepsilon$$

$$purge(\alpha \circ e, P) = \begin{cases} \alpha \circ purge(e, P) & \mid \alpha = (f, x_1 \ldots x_k) \text{ mit} \\ & \quad \Pi_{x_l} = \{P\} \text{ für alle } 1 \leq l \leq k \\ purge(e, P) & \mid \text{sonst.} \end{cases}$$

Hierbei bezeichnet „$\circ$" den Sequenzoperator, ε ist die leere Eingabefolge und $\alpha \in E_M$ eine einzelne Eingabe.

Sei nun z ein beliebiger Zustand von P_j. Mit e als beliebiger Aktionsfolge aus ${E_M}^*$ und δ^* definiert durch Definition 4.9 sei dann

a) $(S'_{P_j}, O'_{P_j}, ZM'_{P_j})$ der Zustand, den P_j beginnend mit z durch die Eingabefolge $purge(e, P_j)$ erreicht, d.h. ${\delta_{P_j}}^*(z, purge(e, P_j)) = (S'_{P_j}, O'_{P_j}, ZM'_{P_j})$;

b) (S'_M, O'_M, ZM'_M) der Zustand, den M beginnend mit z durch die Eingabefolge e erreicht, d.h. ${\delta_M}^*(z, e) = (S'_M, O'_M, ZM'_M)$.

Da $Dom_{P_j} \cap Dom_{P_i} = \emptyset$ für alle $i, j \in I, i \neq j$ gilt, gilt für alle Eingaben $e = (f, x)$ mit $x \in {Dom_{P_j}}^k$, daß für alle $1 \leq l \leq k : |\Pi_{x_l}| = 1 \ \wedge \ \Pi_{x_l} = \{P_j\}$; mit anderen Worten es gilt $c_k(x) = P_j$.

Da M P_i-autonom ist, wird dann $ZM_{M\ |Dom_{P_j}}$ ausschließlich durch P_j kontrolliert, so daß gilt $ZM'_{M\ |Dom_{P_j}} = ZM_{P_j}$.

Dies bedeutet, daß alle Eingabefolgen, die unter der Kontrolle der Metapolitik M ausgeführt werden, zu genau der Zugriffsmatrix (eingeschränkt auf die Domäne von P_j) führen, die entsteht, wenn nur die P_j-lokalen Teile der Eingabefolgen in P_j ausgeführt

werden. Mit anderen Worten, M verhält sich lokal in P_j exakt wie P_j selbst. Die Nichtentscheidbarkeit des Safety-Problems in M folgt dann aus der lokalen Nichtentscheidbarkeit in P_j.

□

Als Anmerkung sei erwähnt, daß eine Aussage der Art: für alle P_i ist die Safety nicht entscheidbar, daraus folgt, daß auch die Safety für M nicht entscheidbar ist, im allgemeinen nicht richtig ist: mit $I = \{1, 2\}$ und $Dom_{P_1} = Dom_{P_2}$ liegt bei jeder Interaktion ein Konflikt vor, so daß für die Safety-Entscheidbarkeit von M ausschließlich die Safety-Entscheidbarkeit von K relevant ist; ist K dann beispielsweise monooperational (s.u.), so ist M Safety-entscheidbar auch dann, wenn die P_i dies nicht sind.

Die folgenden Sätze treffen eine Aussage über hinreichende Bedingungen für die Entscheidbarkeit des Safety-Problems für HRU-modellierte Metapolitiken. Da Definition 4.14 das HRU-Modell einer Metapolitik als Komposition der Zustandsautomaten der $\{P_i\}_{i \in I}$, V und K beschreibt, lassen sich die Sätze direkt aus den entsprechenden Sätzen über HRU-modellierte Sicherheitspolitiken ableiten.

Satz 4.18 (Entscheidbarkeit des Safety-Problems für Metapolitiken über monooperationale HRU-Modelle)
Sei $\{P_i\}_{i \in I}$ eine Menge von Sicherheitspolitiken und $M = (Z_M, E_M, \delta_M, z_M^0)$ ein P_i-autonomes HRU-Modell einer Metapolitik über den P_i; seien ferner alle P_i sowie V und K monooperational. Dann gibt es einen Algorithmus, der entscheidet, ob M für einen Zustand z_M^0 und ein Recht r_M HRU-safe ist oder nicht.

Beweis

Die Art der Konstruktion der Metapolitik in Definition 4.14 erlaubt es, auf eine sehr geradlinige Weise die Eigenschaften der P_i auf die Ebene der Metapolitik zu liften. Da sämtliche P_i sowie V und K monooperational sind, ist gemäß der Konstruktion von δ_M und Definition 4.14 auch M ein monooperationales HRU-Modell; die Behauptung folgt dann unmittelbar durch Anwendung von Satz 4.11.

□

Satz 4.19 (Entscheidbarkeit des Safety-Problems für Metapolitiken über statische HRU-Modelle)
Sei $\{P_i\}_{i \in I}$ eine Menge von Sicherheitspolitiken und $M = (Z_M, E_M, \delta_M, z_M^0)$ ein P_i-autonomes HRU-Modell einer Metapolitik über den P_i; seien ferner alle P_i sowie V und K statisch. Dann gibt es einen Algorithmus, der entscheidet, ob M für einen Zustand z_M^0 und ein Recht r_M HRU-safe ist oder nicht.

Beweis

Da sämtliche P_i sowie V und K statisch sind, ist gemäß der Konstruktion von δ_M und Definition 4.14 auch M ein statisches HRU-Modell; die Behauptung folgt dann unmittelbar durch Anwendung von Satz 4.12.

□

Satz 4.20 (Entscheidbarkeit des Safety-Problems für Metapolitiken über monotone monokonditionale HRU-Modelle)
Sei $\{P_i\}_{i \in I}$ eine Menge von Sicherheitspolitiken und $M = (Z_M, E_M, \delta_M, z_M^0)$ ein P_i-autonomes HRU-Modell einer Metapolitik über den P_i; seien ferner alle P_i sowie V und K monoton und monokonditional. Dann gibt es einen Algorithmus, der entscheidet, ob M für einen Zustand z_M^0 und ein Recht r_M HRU-safe ist oder nicht.

Beweis

Da sämtliche P_i sowie V und K monoton und monokonditional sind, ist gemäß der Konstruktion von δ_M und Definition 4.14 auch M ein monotones und monokonditionales HRU-Modell; die Behauptung folgt dann unmittelbar durch Anwendung von Satz 4.13.

□

4.5 Zusammenfassung

Dieses Kapitel stellt Metapolitiken als ein Konzept zur Formulierung von Regeln für den Umgang mit sicherheitsrelevanten Informationen in Multipolitikensystemen vor, in denen eine Vielzahl von Sicherheitspolitiken und ihre Domänen nebeneinander existieren. Metapolitiken bieten einen Ansatz, Interoperabilität zwischen Entitäten aus unterschiedlichen Sicherheitsdomänen mit unterschiedlichen Sicherheitsattributen und unterschiedlichen Sicherheitspolitiken herzustellen.

Eine Klassifizierung der Interaktionen zwischen Politikdomänen identifiziert hierbei zwei Problembereiche: Interaktionen, welche Konflikte zwischen den beteiligten Sicherheitspolitiken hervorrufen und Interaktionen, auf die keine der beteiligten Sicherheitspolitiken anwendbar ist. Es wird argumentiert, daß solche Interaktionen neue Sicherheitsanforderungen erfordern, welche in Form von Vollständigkeits- und Konfliktpolitiken in eine Metapolitik einfließen. Schließlich werden auf der Grundlage der Klassifizierung notwendige und hinreichende Bedingungen aufgestellt, die die Anwendung der Vollständigkeits- und Konfliktpolitiken im Kontext einer Metapolitik kontrollieren.

Als Grundlage einer formalen Betrachtung von Metapolitiken wurde ein Metapolitikmodell vorgestellt, welches durch eine präzise Formulierung der Beziehungen zwischen der Menge der Sicherheitspolitiken, der Vollständigkeits- und der Konfliktpolitik die Grundlage für die Analyse der Sicherheitseigenschaften von Metapolitiken legt. Basierend auf dem HRU-Kalkül wurde die Klasse der HRU-Metapolitikmodelle definiert, und es wurden Sätze über die grundsätzlichen Möglichkeiten und Grenzen der Analysierbarkeit aufgestellt.

Als praktisches Ergebnis sei festgehalten, daß das Kapitel eine konkrete Bauanleitung für eine Metapolitik über HRU-modellierte Sicherheitspolitiken liefert. Ausgehend von

einer gegebenen Menge von Sicherheitspolitiken werden zugehörige Vollständigkeits- und Konfliktpolitiken definiert und in einem HRU-Modell gemäß Definition 4.8. formalisiert. Sodann wird die in Definition 4.14 aufgezeigte Konstruktionsvorschrift angewendet, um hieraus ein HRU-Modell einer Metapolitik zu erhalten. Hierauf lassen sich dann die Sätze 4.10 bis 4.13 sowie 4.16 bis 4.20 anwenden, um Aussagen über die Entscheidbarkeit von Sicherheitseigenschaften der so konstruierten Metapolitik zu erhalten.

5 Spezifikation von Metapolitiken

Kapitel 4 entwickelte die Grundlagen für eine formale Betrachtung von Metapolitiken und stellte mit Definition 4.6 ein allgemeines Modell für die Beschreibung von Metapolitiken auf. In diesem Kapitel soll nun der nächste Schritt auf dem Weg zu einer Implementierung von Metapolitiken betrachtet und eine Spezifikationssprache für das Metapolitikmodell vorgestellt werden. Wie auch bei der Entwicklung der Spezifikationssprache für Sicherheitspolitikmodelle in Kapitel 3 wird auf eine ballastfreie Formulierbarkeit des Modells geachtet, die durch eine hohe Spezialisierung der Sprache in bezug auf das zu implementierende Modell angestrebt wird. Hierdurch wird eine Überfrachtung mit problemfremden Eigenschaften allgemeiner Spezifikationssprachen vermieden, die Komplexität der Darstellung reduziert und der Paradigmenwechsel vom Modell zur Spezifikation erleichtert. Ausgangspunkt für die Entwicklung der Spezifikationssprache bildet somit die Metapolitikdefinition 4.6.

5.1 Elemente einer Metapolitikspezifikation

Gemäß Definition 4.6 ist eine Metapolitikspezifikation durch die folgenden Elemente beschrieben:

- die *Sicherheitspolitikmenge* $\{P_i\}_{i \in I}$ bestimmt, über welche Sicherheitspolitiken eine Metapolitik M definiert ist; aus dieser Menge lassen sich ableiten
  - die Domäne der Metapolitik: $Dom_M = \bigcup_{i \in I} Dom_{P_i}$;
  - die für die Klassifizierung der Entitäteninteraktionen notwendigen $\Pi_e, e \in Dom_M$;
  - die Regeln für Entitäteninteraktionen der Klasse 1;
- die *Vollständigkeitspolitik* V, aus der sich die Regeln für Entitäteninteraktionen der Klasse 2 ableiten lassen;
- die *Konfliktpolitik* K, aus der sich die Regeln für Entitäteninteraktionen der Klasse 3 ableiten lassen;

- die Klassifizierungsfunktion $c_k : {Dom_M}^k \rightarrow \{P_i\}_{i \in I} \cup V \cup K$ in der allgemeinen Variante für einen k-dimensionalen Entitätenvektor.

Da die Spezifikation einer Metapolitik gegenüber der abstrakten Definition 4.6 einen konkretisierenden Schritt in Richtung auf eine Implementierung darstellt, enthält eine Metapolitikspezifikation neben diesen sich unmittelbar aus der Definition ergebenden Komponenten die folgenden weiteren Elemente.

- *Politikspezifische Operationen* spezifizieren Operationen auf der Metapolitik selbst, die den Zustand der Metapolitik dynamisch zu ihrer Laufzeit beeinflussen. Beispiele hierfür sind die Änderung der Sicherheitspolitikmenge $\{P_i\}_{i \in I}$ durch Hinzunahme/Herausnahme von Sicherheitspolitiken, die Änderung der Domäne der Metapolitik (bedingt durch die Veränderung der Domäne einer der P_i) oder ein Austausch von Konflikt- oder Vollständigkeitspolitik.
  Operationen zur Beeinflussung des Metapolitikzustands sind Bestandteil der Metapolitik selbst und bilden folglich einen Teil ihrer Spezifikation. Die Rechte zu ihrem Gebrauch werden durch eine *Rechtematrix* modelliert, welche damit gleichfalls zu einem Bestandteil einer Metapolitikspezifikation wird.

- Die *Invarianten* spezifizieren die universell über die gesamte Metapolitiklaufzeit gültigen Fakten; ein Beispiel ist die P_i-Autonomie einer HRU-modellierten Metapolitik (vgl. Definition 4.15).

- Der *Grundzustand*; eine Metapolitik ist nicht immer seit dem ersten Ablauf der von ihr kontrollierten Sicherheitspolitiken dabei (vgl. das oben genannte Beispiel einer Expertenkonferenz). Wird eine Metapolitik in nachhinein über eine Menge existierender Sicherheitspolitiken gelegt, so hat sie keine Möglichkeit, einen dem Zustand der Sicherheitspolitiken entsprechenden Metapolitikzustand sukzessive aufzubauen.
  Hieraus ergibt sich die Notwendigkeit, den Grundzustand einer Metapolitik festlegen zu können, in dem initial die kontrollierten Sicherheitspolitiken, deren Domänen und die Zugriffsmatrix definiert werden. Daneben ist die Möglichkeit der Definition eines Grundzustandes praktisch: sie reduziert den Spezifikations- und Beweisaufwand für solche Metapolitiken, die mit statischem Zustand arbeiten; politikspezifische Operationen zur Manipulation des Metapolitikzustands können dann entfallen.

5.2 Skippy$_M$

Dieser Abschnitt beschreibt wesentliche Sprachelemente einer Spezifikationssprache für Metapolitiken. Die in Kapitel 3 getroffenen Vorbereitungen ermöglichen eine enge Anlehnung an die Spezifikationssprache für Sicherheitspolitiken, Skippy; zur Differenzie-

rung wird die hier vorgestellte Metapolitiken-Spezifikationssprache mit *Skippy*$_M$ bezeichnet. Wie auch bei Skippy wird auf eine ballastfreie Formulierbarkeit der Sicherheitspolitik Wert gelegt; dies wurde durch eine sorgfältige Trennung von stets vorhandenen, generischen Spezifikationskomponenten einerseits und andererseits den jeweils für eine spezielle Sicherheitspolitik im Einzelfall zu spezifizierenden Komponenten angestrebt.

Skippy$_M$ enthält somit die oben beschriebenen Spezifikationselemente als vordefinierte Typen beziehungsweise Operationen; ihre Instanziierung erfolgt innerhalb einer Metapolitikdeklaration automatisch. Im einzelnen sind in jeder Skippy$_M$-Spezifikation folgende Typen vordefiniert:

POL ist eine Menge von Sicherheitspolitikbezeichnern; *POL* legt in der Metapolitikspezifikation den Typ (im Sinne einer Programmiersprache) eines Sicherheitspolitikbezeichners fest.

ENT ist eine Menge von Entitätenbezeichnern; *ENT* legt in der Metapolitikspezifikation den Typ (im Sinne einer Programmiersprache) eines Entitätenbezeichners fest.

R ist die Menge der Zugriffsrechte - Namen; R legt in einer Metapolitikspezifikation den Typ (im Sinne einer Programmiersprache) der Rechte fest.

OP ist die Menge der Namen der sicherheitspolitikspezifischen Operationen; da zu jeder Operation auch ein entsprechendes Zugriffsrecht gehört, gilt $OP \subseteq R$.

POL, ENT und R sind generisch in dem Sinne, daß die expliziten Werte undefiniert bleiben, da diese keine Auswirkungen auf die Metapolitiksemantik besitzen. Diese Typen werden lediglich zur Konstruktion der folgenden, gleichfalls vordefinierten und automatisch instanziierten Sicherheitspolitikvariablen verwendet.

policies ist vom Typ $\wp(POL)$ und bezeichnet die Menge der Sicherheitspolitiken, die der Metapolitik unterliegen: $policies \subset \wp(POL)$ implementiert $\{P_i\}_{i \in I}$.

v,k sind Variablen vom Typ POL und bezeichnen die Vollständigkeits- und die Konfliktpolitik.

rm ist vom Typ $Dom_M \rightarrow \wp(OP)$ und bezeichnet die Rechtematrix, die zu jeder Entität aus den Domänen der kontrollierten Sicherheitspolitiken ihr Recht zum Aufruf metapolitikspezifischer Operationen aus OP definiert.

classify ist eine Abbildung vom Typ $ENT^k \rightarrow POL$ und implementiert die Klassifizierungsfunktion c_k. *classify* tritt in der Regel in einer Spezifikation nicht explizit in Erscheinung, sondern wird bei jeder Interaktion zwischen Entitäten zur Selektion einer Sicherheitspolitik aus $\{\{P_i\}_{i \in I}, V, K\}$ verwendet (vgl. Definition 4.6 und Kapitel 6).

5.3 Ein Beispiel

Zur Förderung einer intuitiven Vorstellung sollen Syntax und Semantik von Skippy$_M$ zunächst anschaulich an einem Beispiel illustriert werden.

Das Beispiel greift erneut das Szenarium des Technologiekonzerns auf und ist eine weitere Konkretisierung des Ausschnittszenariums aus Abschnitt 4.3.1, in dem angenommen wird, daß Ann's Heimatdomäne Dom_Q eine hierarchische Sicherheitspolitik (BLP) verwendet, ihre Gastgeberdomäne Dom_{KSL} dagegen die *Chinese Wall* - Sicherheitspolitik einsetzt, wie sie bereits in Kapitel 3 als Beispielpolitik spezifiziert wurde.

Beide Sicherheitspolitiken mögen in unserem Beispiel disjunkte Domänen besitzen; hierdurch wird das Beispiel übersichtlich gehalten, da sich dann keine Notwendigkeit für eine Konfliktpolitik ergibt. Für Interaktionen zwischen den Politikdomänen benötigen wir allerdings eine Vollständigkeitspolitik, da die beteiligten Subjekte und Objekte aus unterschiedlichen Domänen stammen und somit weder KSL noch Q die Zugriffsentscheidung autonom treffen können.

Bei der Definition einer Vollständigkeitspolitik lassen sich zwei grundsätzliche Ansätze unterscheiden. Zum einen lassen sich die existierenden Sicherheitspolitiken nutzen, um hieraus mittels einer Politikenalgebra eine neue Sicherheitspolitik abzuleiten; dieser Ansatz wurde bereits in Abschnitt 4.3.3 auf Seite 92 diskutiert. Mögliche Varianten sind hier beispielsweise „$Q \wedge KSL$" (beide Sicherheitspolitiken müssen bei einem Zugriff zustimmen), „$Q \vee KSL$" (Zustimmung einer der beiden Sicherheitspolitiken ist ausreichend) oder auch einfach „Q" (ausschließlich die Sicherheitspolitik Q in Ann's Heimatdomäne entscheidet). In allen genannten Fällen sind Repräsentantenabbildungen gemäß Definition 4.7 erforderlich, die die beteiligten Entitäten mit den notwendigen Attributierungen versehen.

Stehen der Wiederverwendung der existierenden Sicherheitspolitiken für die Definition einer Vollständigkeitspolitik strategische, rechtliche oder wirtschaftliche Gründe entgegen, so wird man eine vollständig neue Sicherheitspolitik definieren; dies soll nun anhand dieses Beispiels veranschaulicht werden.

Hierzu möge sowohl die Domäne Q als auch die Domäne KSL ein elektronisches Postsystem besitzen, welches innerhalb der jeweiligen Domäne den Austausch elektronischer Post organisiert. Das Postsystem bestehe – für das Beispiel gegenüber der Realität etwas vereinfacht – aus zwei Komponenten: einem *Mailer* zur Entgegennahme und Verschickung elektronischer Briefe und einem *Mailbox*-Server, der Briefe vom Mailer entgegennimmt und sie zur Abholung durch die Empfänger in empfängerspezifische Postfächer einsortiert (Bild 5.1).

Soll nun Post auch über Domänengrenzen hinweg zugestellt werden können, so muß hierzu der *Mailer* aus Dom_Q auf die *Mailbox* in Dom_{KSL} Schreibzugriff erhalten, und analog der *Mailer* aus Dom_{KSL} auf die *Mailbox* in Dom_Q. Dazu wird eine sehr einfache, recht starre Vollständigkeitspolitik gewählt, die bei Interdomänenzugriffen nur Subjekten aus dem diskret festgelegten elitären Kreis der *Mailer* den Zugriff auf Objekte

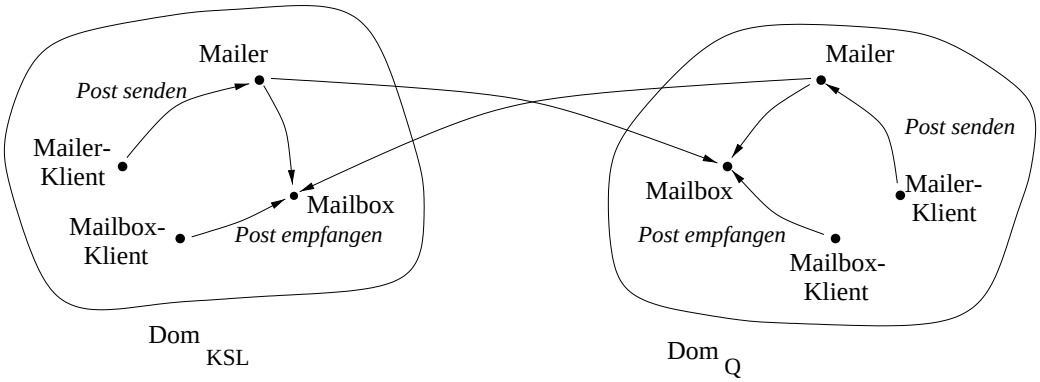


Bild 5.1: Ein elektronisches, domänenübergreifendes Postsystem

einer ebenfalls diskret festgelegten Objektmenge der *Mailboxen* gestattet; die resultierende Sicherheitspolitik läßt sich dann auf anschauliche Weise mittels einer statischen Zugriffsmatrix beschreiben. Bild 5.2 zeigt die zugehörige Skippy-Spezifikation dieser Vollständigkeitspolitik, in der lediglich die beiden Mailer der jeweiligen Domänen direkten Schreibzugriff auf die Mailboxen in der jeweils anderen Domäne erhalten.

Policy Completeness =
  Policy-Sets:
    $TheLables = \{\}$; /* no labels */
  Classification:
    $TheLattice : \{SysLo \sqsubseteq SysHi\}$; /* no lattice */
  Rights:
    $\{write\}$;
  Oper:
    $write(s, o)$:
      op: $a \leq_{write} b = true$; /* no lattice restriction */
      pre: $s \leq_{write} o \wedge$
        $(s \in \mathbf{entities}) \wedge (o \in \mathbf{entities}) \wedge (write \in \mathbf{m}(s, o))$;
  Inv:
    /* no invars */
  State:
    entities' = {*Q-Mailer, KSL-Mailer, Q-Mailbox, KSL-Mailbox*};
    m' = { *(Q-Mailer,KSL-Mailbox)* $\mapsto \{write\}$,
      (KSL-Mailer,Q-Mailbox) $\mapsto \{write\}$};
END Completeness.

Bild 5.2: Beispiel einer Skippy-Spezifikation einer Vollständigkeitspolitik

Für die Metapolitik sind damit die Spezifikationselemente **policies** = {*Q,KSL*}, **v** =

Completeness und $\mathbf{k} = \mathit{no_policy}$ sowie die Rechtemenge bestimmt. Der nächste Schritt bestimmt nun die politikspezifischen Operationen, Operationen also, die auf der Sicherheitspolitikmenge, den Variablen für die Vollständigkeits- und Konfliktpolitik und der Rechtematrix definiert sind. Da zwischen den einzelnen Operationen keine grundsätzlichen Unterschiede bestehen, wird sich das Beispiel auf die exemplarische Betrachtung der Operationen zur Manipulation der Sicherheitspolitikmenge beschränken.

Die Sicherheitspolitikmenge einer Metapolitik muß immer dann verändert werden, wenn bedingt durch beispielsweise der Änderung der Organisationsstruktur neue Sicherheitspolitiken unter eine Metapolitik gestellt werden sollen oder aber bestehende Sicherheitspolitiken aus der Sicherheitspolitikmenge ausgeschlossen werden sollen. Die Operationen zur Manipulation der Sicherheitspolitikmenge erlauben es, strategische Entscheidungen in algorithmischer Form zu repräsentieren und in eine Metapolitik einzubringen. Eine solche Strategie kann es z.B. sein, für die Aufnahme einer neuen Sicherheitspolitik in die Sicherheitspolitikmenge das Einverständnis aller existierenden Sicherheitspolitiken einzuholen; dies kann in Skippy_M ausgedrückt werden durch

$$\begin{aligned} &JoinPolicySet(s, NewP) : \\ &\qquad \textbf{pre}: \quad \forall P : policies.\ JoinPolicy_P(s, NewP) = granted \end{aligned}$$

Bild 5.3 zeigt ein Beispiel, in welchem mittels der metapolitikinternen Rechtematrix geprüft wird, ob das aufrufende Subjekt das Recht zur Modifikation der Sicherheitspolitikmenge besitzt. Die hier spezifizierte Strategie vergibt in der Initialisierungsphase der Metapolitik ein unwiderrufbares Recht zur Modifikation der Sicherheitspolitikmenge an ein einziges, ausgezeichnetes Subjekt namens ,,adm". Dieses Subjekt agiert in der Rolle eines Metapolitikadministrators und ist als einziges in der Lage, auch ohne Mitwirkung Dritter Sicherheitspolitiken aus der Politikmenge zu entfernen und neue Sicherheitspolitiken aufzunehmen. In der Spezifikation wird dies durch die Initialisierung der Rechtematrix

$$\mathbf{rm}' = \{i : N \mid i = adm \bullet adm \mapsto \{JoinPolicySet, LeavePolicySet\}\}$$

im *State*-Teil beschrieben. Bei der Aufnahme einer neuen Sicherheitspolitik wird weiterhin jeweils ein Administrator bestimmt, der für diese eine Sicherheitspolitik das Recht erhält, diese wieder aus der Metapolitik zu entfernen. Diese Strategie wird im Beispiel durch einen Teil der *where* - Klausel für die *JoinPolicySet*-Operation spezifiziert:

$$\mathbf{rm}' = \mathbf{rm} \oplus \{(PolAdm, PName) \mapsto \{LeavePolicySet\}\}.$$

Die Assoziierung des ,,adm"-Subjekts mit einem realen Subjekt erfolgt ebenfalls wie die Assoziierung der Namen ,,KSL" und ,,Q" zum Zeitpunkt der Initialisierung der Metapolitik im Rahmen ihrer Integration in die Sicherheitsarchitektur einer realen Systemplattform; das Prinzip dieser Anbindung wurde bereits im Kapitel 3 über die Implementierung von Sicherheitspolitiken skizziert.

Da das Beispiel auf Invarianten verzichtet, verbleibt die Spezifikation des Grundzustands der Metapolitik; hier werden die initiale Sicherheitspolitikmenge sowie die Vollständigkeits- und Konfliktpolitik mittels der Variablen *policies, v* und k definiert und das Recht an die Entität „adm" vergeben, die Sicherheitspolitikmenge der Metapolitik zu modifizieren. Hiermit ist die Spezifikation abgeschlossen; Bild 5.3 zeigt die vollständige Spezifikation. Zur Schreibweise gelten die entsprechenden Anmerkungen zur Spezifikation der *Chinese Wall* - Sicherheitspolitik auf Seite 58.

Metapolicy ChineseBell =
  Rights:
    /* only two operations on policy set of metapolicy
    */
    $\{JoinPolicySet, LeavePolicySet\}$;
  Oper:
    /* new policy joins metapolicy domain
    */
    $JoinPolicySet(s, PName, PolAdm)$:
      pre: $s = adm \ \wedge \ PName \notin \mathbf{policies}$;
      where: $\mathbf{policies}' = \mathbf{policies} \cup PName$;
        $\mathbf{rm}' = \ \mathbf{rm} \oplus$
          $\{(PolAdm, PName) \mapsto \{LeavePolicySet\}\}$;

    /* policy leaves metapolicy domain
    */
    $LeavePolicySet(s, PName)$:
      pre: $(s = adm \ \vee$
        $LeavePolicySet \in \mathbf{rm}(s, PName)) \ \wedge$
        $PName \in \mathbf{policies}$;
      where: $\mathbf{policies}' = \mathbf{policies} \setminus PName$;
        $\mathbf{rm}' = \mathbf{rm} \oplus \{(s, PName) \mapsto \{\}\}$;
  Inv:
    /* no invariant
    */
    true;
  State:
    $\mathbf{policies}' = \{KSL, Q\}$;
    $\mathbf{v}' = Completeness$;
    $\mathbf{k}' = no_policy$;
    $\mathbf{rm}' = \{i : \mathcal{N} \mid i = adm \bullet adm \mapsto \{JoinPolicySet, LeavePolicySet\}\}$;
End ChineseBell.

Bild 5.3: Spezifikation einer Metapolitik über *KSL* und *Q*

5.4 Die Syntax von Skippy$_M$

Zur Abrundung des Kapitels soll abschließend die Syntax von Skippy$_M$ angegeben werden; diese Definition erfolgt in Anlehnung an die Syntaxdefinition von Skippy in Kapitel 3. ≺Ident≻ ist wiederum ein wohlgeformter Identifier in der Metapolitik, dessen konkretes Aussehen hier nicht weiter spezifiziert werden soll; das leere Produktionssymbol ist „e".

Identifier, Ausdrücke, etc.

≺Ident_list≻ :== ≺Ident≻ ≺I_list≻ ;

≺I_list≻ :== "," ≺Ident_list≻ | e ;

Metapolitik – Deklaration

≺METAPOL≻ :== "Metapolicy" ≺Ident≻ "=" ≺mpol_decl≻
"End" ≺Ident≻ "." ;

≺mpol_decl≻ :== "Rights:" ≺rights_decl≻
"Oper:" ≺oper_decl≻
"Inv:" ≺inv_decl≻
"State:" ≺state_decl≻ ;

Rechte – Deklaration

≺rights_decl≻ :== "{" ≺Ident_list≻ "};" ;

Operationen – Deklaration

≺oper_decl≻ :== ≺operation≻ ";" ≺op_list≻ ;

≺op_list≻ :== ≺oper_decl≻ | e ;

≺operation≻ :== ≺Ident≻ "(" ≺Ident_list≻ ");"
≺pre_clause≻
≺where_clause≻ ;

≺pre_clause≻ :== "pre:" ≺B_Exp≻ ";" ;

≺where_clause≻ :== "where:" ≺Z_function_list≻ | e ;

≺Z_function_list≻ :== ≺pol_fn≻ ≺v_fn≻ ≺k_fn≻ ≺rm_fn≻ ;

≺pol_fn≻ :== "policies' = " ≺pol_struct≻ ";" | e ;

≺v_fn≻ :== "v' = " ≺Ident≻ ";" | e ;

≺k_fn≻ :== "k' = " ≺Ident≻ ";" | e ;

≺rm_fn≻ :== "rm' = " ≺rm_struct≻ ";" | e ;

Invarianten – Deklaration

≺inv_decl≻ :== ≺B_Exp≻ ";" ;

Zustands – Deklaration

≺state_decl≻ :== ≺Z_function_list≻ ;

Analog zu Kapitel 3 steht das Nichtterminalsymbol ≺B_Exp≻ für einen wohlgeformten boole'schen Ausdruck im Prädikatenkalkül erster Ordnung; die Bemerkungen auf Seite 61 gelten entsprechend.

Weiterhin sind hier die Nichtterminalsymbole ≺pol_struct≻ und ≺rm_struct≻ nicht weiter definiert. ≺pol_struct≻ ist die Menge aller wohlgeformten Mengenausdrücke vom Typ $\wp(POL)$, und ≺rm_struct≻ ist die Menge aller Abbildungen vom Typ $Dom_M \rightarrow \wp(OP)$.

Mit diesen Definitionen lehnt sich Skippy$_M$ an die in Kapitel 3 beschriebene Spezifikationssprache für Sicherheitspolitiken an. Eine Fortschreibung des in [KP99] entwickelten Werkzeugs für Skippy$_M$ wird es dann gleichfalls ermöglichen, Skippy$_M$-Spezifikationen weitgehend automatisiert in ein übersetzbares C++ - Programm zu transformieren.

5.5 Zusammenfassung

In diesem Kapitel wurde eine Sprache zur Spezifikation von Metapolitiken vorgestellt. Sie dient dem Ziel, aufbauend auf dem in Kapitel 4 entwickelten Metapolitik-Modell ein Werkzeug zu erhalten, welches die Umsetzung eines formalen Modells in eine konkrete Implementierung fördert. Die Spezifikationssprache basiert auf der in Kapitel 3 entwickelten Spezifikationssprache Skippy und folgt den bereits dort angewandten Prinzipien der Ballastfreiheit und Problemorientiertheit. Dieser Ansatz führt zu einem kleinen, handhabbaren Sprachumfang und, wie an Beispielen gezeigt, zu lesbaren und übersichtlich strukturierten Spezifikationen, die weitgehend automatisiert in ein übersetzbares C++ - Programm transformiert werden können.

Allerdings sind auch noch ungelöste Probleme vorhanden, die auf dem bisher betrachteten Abstraktionsniveau nicht sichbar geworden sind; diese offenbaren sich dann, wenn eine Spezifikation mehr als eine akademische Übung werden und die Grundlage der Im-

plementierung einer Metapolitik bilden soll, welche in die Sicherheitsarchitektur einer realen Systemumgebung integriert werden soll. Hierzu wird das anschließende Kapitel einige Fußangeln aufzeigen, die sich insbesondere bei der Implementierung der Π_e und damit der *classify*-Funktion zeigen werden.

6 Implementierung von Metapolitiken

Thema dieses Kapitels ist der letzte Schritt des Weges vom Sicherheitsmodell einer Metapolitik bis zu ihrer Implementierung. Begründet durch die Erfahrung, daß Hintertüren und Fußangeln vielfach erst auf einer konkreten Realisierungsebene sichtbar werden, werden hier verschiedene Aspekte der Implementierung des Metapolitikmodells diskutiert. Dies betrifft zum einen die Frage nach der Implementierung der Elemente einer Metapolitikspezifikation: die Sicherheitspolitikmenge $\{P_i\}_{i \in I}$, ihre Domänen Dom_{P_i}, die Π_e und die metapolitikspezifischen Operationen, die auf diesen Elementen definiert sind; diese werden in diesem Kapitel in einer realen Programmiersprache formuliert, und Sicherheitsaspekte bei der Implementierung werden diskutiert. Zum anderen betrifft dies die Frage nach den Eigenschaften, die eine Sicherheitsarchitektur zur Durchsetzung des Metapolitikmodells besitzen muß.

Bei den Überlegungen wird erneut das *computational model* aus Kapitel 3 zugrunde gelegt: jedes IT-System verfügt über eine Reihe von Anwendungssystemen, die nebeneinander existieren oder auch miteinander interagieren können; jedes Anwendungssystem besteht wiederum aus Entitäten, die im Kontext ihres Anwendungssystems ebenfalls miteinander interagieren können. Innerhalb ihrer Domäne stellen Sicherheitspolitiken die domänenspezifischen Sicherheitseigenschaften her; Regeln über die Sicherheitseigenschaften domänenübergreifender Interaktionen sind in Metapolitiken formuliert.

Ein Leitgedanke bei der Implementierung von Metapolitiken ist die scharfe Trennung von Anwendungssystemen, Sicherheitspolitiken und Metapolitiken; insbesondere sind Metapolitiken wie Sicherheitspolitiken gekapselte, manipulationsgeschützte Entitäten, die durch die Sicherheitsarchitektur ihrer Umgebung in die Lage versetzt werden, sicherheitsrelevante Interaktionen der Entitäten zu beobachten, zu autorisieren und auf diese Weise die Sicherheitseigenschaften herzustellen.

Das Konzept zur Implementierung von Metapolitiken baut auf dem in Kapitel 3 vorgestellten Custodian-Paradigma auf. In diesem Paradigma werden Metapolitiken als abstrakte Datentypen formuliert, die in eine Sicherheitsarchitektur integriert werden und hierdurch mit Mechanismen ausgestattet werden, die die Metapolitik zu ihrer Durchsetzung benötigt. Dieses Kapitel hat somit zwei Abschnitte: der erste erläutert die Abbildung einer Skippy_M – Spezifikation in einen abstrakten Datentyp einer konkreten Programmiersprache, der zweite erläutert die Einbettung dieses Datentyps in eine Sicherheitsarchitektur.

6.1 Spezifikationselemente

Dieser Abschnitt entwickelt aus den Bausteinen einer Skippy$_M$ – Spezifikation die zugehörigen konkreten Repräsentationen in der Programmiersprache C++ und fast diese zu einer C++ – Klassendefinition zusammen. Die Verwendung von C++ hat in erster Linie pragmatische Gründe, da auf diese Weise weite Teile des in [KP99] beschriebenen Transformationswerkzeugs wiederverwendet werden können.

Zur Illustration wird in Anhang B eine vollständige C++ – Implementierung der Skippy$_M$-Spezifikation aus Bild 5.3 angegeben, die neben der dort spezifizierten politikspezifischen Operation *LeavePolicySet* zusätzlich sämtliche weiteren politikspezifischen Operationen einer Metapolitik zeigt.

6.1.1 Metapolitik-Zustand

In einer Skippy$_M$ – Spezifikation wird der Zustand einer Metapolitik beschrieben durch eine Sicherheitspolitikmenge $\{P_i\}_{i \in I}$, den Vollständigkeits- und Konfliktpolitiken V und K und der Rechtematrix rm. Zu jedem P_i gehört weiterhin eine Politikdomäne Dom_{P_i}; aus der Vereinigung aller Politikdomänen leitet sich die Entitätenmenge der Metapolitik mittels $Dom_M = \bigcup_{i \in I} Dom_{P_i}$ ab.

Zur Beschreibung dieses Zustands definiert eine C++ – Metapolitikklasse die folgenden Typen.

POL ist der Typ der Sicherheitspolitiknamen.

ENT ist der Typ der Entitätennamen.

RIGHT ist der Typ der Rechte an metapolitikspezifischen Operationen; da zu jeder metapolitikspezifischen Operation genau ein Recht zu ihrem Aufruf gehört, ist der Rechtetyp identisch mit der Menge der Operationsnamen.

DOM ist der Typ einer Politikdomäne; Politikdomänen werden als Mengen von Entitäten dargestellt.

Mit diesen Typen werden die folgenden Klassenvariablen definiert.

Policies repräsentiert $\{P_i\}_{i \in I}$ und ist ein Mengentyp, dessen Elemente vom Typ *POL* sind. Der Mengentyp und die hierauf definierten Operationen sind in der Hilfsklasse „*template <class t> class set*" definiert (siehe Anhang B).

V, K sind vom Typ *POL* und repräsentieren die Vollständigkeits- und Konfliktpolitik.

rm ist eine Abbildung vom Typ $Dom_M \times \{P_i\}_{i \in I} \rightarrow \wp(OP)$ und repräsentiert die Rechtematrix; OP repräsentiert die Menge der metapolitikspezifischen Operationen und wird durch einen Mengentyp realisiert, dessen Elemente vom Typ *RIGHT* sind.

PolicyDomains repräsentiert $\{Dom_{P_i}\}_{i \in I}$ in Form einer Abbildung $\{P_i\}_{i \in I} \rightarrow Dom_{P_i}$, die jedem P_i seine Politikdomäne Dom_{P_i} zuordnet.

Entities repräsentiert die Metapolitikdomäne Dom_M, wird abgeleitet aus $\bigcup_{i \in I} Dom_{P_i}$ und ist ein Mengentyp mit Elementen vom Typ *ENT*.

Pi repräsentiert die Mengen $\Pi_e, e \in Dom_M$ in Form einer Abbildung $Dom_M \rightarrow \wp(\{P_i\}_{i \in I})$, die jeder Entität in Dom_M die Menge aller Sicherheitspolitiken zuordnet, in deren Domäne die Entität liegt.

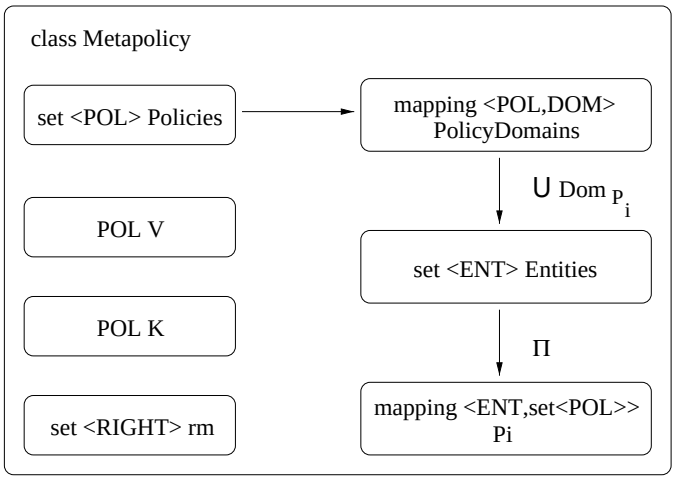


Bild 6.1: Komponenten und ihre Beziehungen einer C++ – Metapolitikklasse

6.1.2 Metapolitikspezifische Operationen

Die metapolitikspezifischen Operationen sind auf den Zustandsvariablen einer Metapolitikinstanz definiert und können diese zur Laufzeit der Metapolitik verändern. Eine Metapolitik hat insgesamt sieben politikspezifische Operationen; hinzu kommt der Klassenkonstruktor, der genau einmal zur Initialisierung einer Metapolitik-Klasseninstanz ausgeführt wird.

Die Rechte zum Aufruf der metapolitikspezifischen Operationen werden aus den Vorbedingungen der einzelnen Operationen aus der Spezifikation abgeleitet; sind sie dort nicht explizit definiert, so werden die Rechte nach folgendem Schema organisiert:

- Bei der Klasseninstanzkonstruktion wird als Ankerpunkt für die weitere Rechtevergabe eine Entität spezifiziert, die als Metapolitik-Verantwortliche agiert und die
  - als einzige Entität das unveräusserbare Recht erhält, die Sicherheitspolitikmenge *Policies* zu erweitern
  - das unwiderrufbare Recht erhält, Sicherheitspolitiken auch ohne Mitwirkung Dritter aus einer Metapolitik zu entfernen.
- Neue Sicherheitspolitiken können ausschließlich durch die Metapolitik-Verantwortliche in eine Metapolitik eingebracht werden; hierbei wird dann für jede eingebrachte Sicherheitspolitik eine für die neue Sicherheitspolitik verantwortliche Entität bestimmt, welche anfänglich alle Rechte – auch die zur Rechteweitergabe und zum Rechteentzug – bezüglich der neuen Sicherheitspolitik erhält.
- das Entfernen einer Sicherheitspolitik aus der Metapolitik kann durch die Metapolitik-Verantwortliche sowie durch jede Entität erfolgen, die über das entsprechende Recht verfügt; hat die Politikverantwortliche dieses Recht nicht weitergegeben oder aufgegeben, so ist sie neben der Metapolitik-Verantwortlichen die einzige Entität, die dies darf.
- Der Austausch von Vollständigkeits- oder Konfliktpolitik ist in erheblichem Maße mit Sicherheitsrisiken verbunden und stellt somit einen schweren Eingriff in die Metapolitik dar; das Recht hierzu ist daher unveräußerbar und ausschließlich der Metapolitik-Verantwortlichen vorbehalten.
- Die Rechte zur Benutzung aller anderen metapolitikspezifischen Operationen richten sich nach der Rechtematrix.

Metapolitikspezifische Operationen auf $\{P_i\}_{i \in I}$

Die Sicherheitspolitikmenge $\{P_i\}_{i \in I}$ wird in der Metapolitik-Variablen *Policies* repräsentiert; auf dieser Variablen operieren die beiden Operationen *JoinPolicySet* und *LeavePolicySet*.

```
bool JoinPolicySet (ENT s,
                    POL PolId,
                    DOM PolDom,
                    ENT PolAdmin);
```

JoinPolicySet fügt der Sicherheitspolitikmenge ein neues Element hinzu und bestimmt, welche Entität verantwortlich für die neue Sicherheitspolitik agiert.

Parameter

s identifiziert, wie bei allen metapolitikspezifischen Operationen, die ausführende Entität.

PolId ist der eindeutige Name der neuen Sicherheitspolitik, der der Menge $\{P_i\}_{i \in I}$ hinzugefügt wird. Die Namen der Sicherheitspolitiken sind in der vorliegenden Metapolitikimplementierung als *Integers* formuliert; die Abbildung zwischen dieser Repräsentation und den Namen von Sicherheitspolitiken in realen Systemumgebungen (Pfadnamen, DCE-UniqueIds) erfolgt durch einen Bindevorgang in Zusammenarbeit mit der Sicherheitsarchitektur (vgl. Kapitel 3).

PolDom ist die Domäne der neuen Sicherheitspolitik, ausgedrückt als Entitätenmenge.

PolAdmin identifiziert eine Entität, die initial alle Rechte zur Manipulation der neuen Sicherheitspolitik in der Metapolitik erhält und so als Politikverantwortliche agieren kann.

Rückgabewert

Die Operation terminiert mit dem Wert *true* genau dann, wenn s das Recht zur Ausführung besitzt, *PolAdmin* selbst in der Domäne der neuen Politik liegt und die neue Politik nicht bereits in der Sicherheitspolitikenmenge der Metapolitik enthalten ist.

Vorbedingungen

Zur Ausführung der Operation muß s die verantwortliche Entität für die Metapolitik sein. Die Authentizität von s ist dazu unabdingbar: Maskerade kann hier durch Einschleusung „trojanischer" Sicherheitspolitiken zur Beeinflussung der Metapolitik führen, wenn

a) die Domäne der neuen Sicherheitspolitik die einer anderen überschneidet oder

b) M nicht P_i-autonom ist und V oder K durch einen Politikenalgebra-Ausdruck konstruiert sind (Mehrheitsentscheidungen, Veto-Recht).

Aus dem gleichen Grund muß die Integrität der Parameter entweder durch die Kommunikationsmechanismen der Systemplattform gewährleistet werden oder explizit, beispielsweise durch Verwendung digitaler Signaturen, überprüfbar sein.

Laufzeitkomplexität

Die Zuordnung einer neuen Sicherheitspolitik zu einer Metapolitik bedingt, daß die zur Klassifikation einer Interaktion notwendigen Mengen Π_e sich ändern.

Dies kann zum einen dadurch erreicht werden, daß die Π_e bei jeder Interaktionsklassifikation neu berechnet werden oder alternativ dadurch, daß bei jeder Änderung an den Politikdomänen die Π_e unmittelbar neu berechnet werden. Da Änderungen an den Domänen der Sicherheitspolitiken im Vergleich zu einer Interaktionsklassifikation im allgemeinen wesentlich seltener auftreten, wurden die Datenstrukturen so gestaltet, daß Domänenänderungen eine vergleichsweise hohe, Klassifikationsoperationen dagegen eine geringere Laufzeit erfordern.

In der hier vorliegenden Implementierung ist die Laufzeit von *JoinPolicySet* proportional zu $|\{P_i\}| \cdot (|Dom_M| + 1)$, wobei von einer konstanten Zeit für Mengenoperationen ($\in$, Einfügen oder Löschen eines Elementes) ausgegangen wird. In einer Umgebung mit obligatorischen Sicherheitspolitiken dagegen beruhen die Zugriffsrechte generell auf Entitätenattributen; dies kann hier genutztz werden indem Π_e zu einem weiteren Attribute der Entität e wird. Bei Diesem Ansatz entfällt folglich die Berechnung der Mengen Π, so daß dann die Laufzeit proportional zur Mächtigkeit der Domänenmenge Dom_{PolId} ist.

bool LeavePolicySet (ENT s,
POL PolId);

LeavePolicySet entfernt eine Sicherheitspolitik aus der Sicherheitspolitikmenge einer Metapolitik; die zugehörige Domäne wird aus *PolicyDomains* entfernt, Dom_M sowie die Π_e werden neu berechnet und sämtliche Rechte bezüglich der entfernten Sicherheitspolitik gelöscht.

Parameter

s identifiziert die aufrufende Entität.

PolId identifiziert die zu entfernende Sicherheitspolitik.

Rückgabewert

Die Operation terminiert mit dem Wert *true* genau dann, wenn s das Recht zur Ausführung besitzt und die zu entfernende Politik in der Sicherheitspolitikenmenge der Metapolitik enthalten ist.

Vorbedingungen

Die ausführende Entität s muß das entsprechende Recht in der Rechtematrix besitzen; s muß authentisch sein, eine Maskerade kann bei unberechtigter Herausnahme einer Sicherheitspolitik

- *denial-of-service* – Situationen hervorrufen

- bei Konstruktion von V oder K mittels eines Ausdrucks aus der Politikalgebra können Mehrheitsentscheidungen beeinflußt oder Vetos verhindert werden.

Aus dem gleichen Grund muß die Integrität der Parameter entweder durch die Kommunikationsmechanismen der Systemplattform gewährleistet werden oder explizit, beispielsweise durch Verwendung digitaler Signaturen, überprüfbar sein.

Laufzeitkomplexität

Mit den Bemerkungen zur Laufzeit von *JoinPolicySet* ist die Laufzeit von *LeavePolicySet* proportional zu $|Dom_M| \cdot |\{P_i\}| + |Dom_M| + |\{P_i\}|$ respektive zu $|Dom_{PolId}|$.

Metapolitikspezifische Operation zum Ändern einer Politikdomäne

bool SetPolicyDomain (ENT s,
POL PolId,
DOM PolDom);

SetPolicyDomain modifiziert die Domäne einer Sicherheitspolitik; die Operation wird stets aufgerufen, wenn innerhalb der Domäne einer Sicherheitspolitik eine Entität erzeugt oder gelöscht wird.

Parameter

s	identifiziert die ausführende Entität.
PolId	ist der Name der Sicherheitspolitik, deren Domäne modifiziert wird.
PolDom	ist die neue Domäne, ausgedrückt als Entitätenmenge.

Rückgabewert

Die Operation terminiert mit dem Wert *true* genau dann, wenn s das Recht zur Ausführung in der Rechtematrix besitzt.

Vorbedingungen

Die ausführende Entität s muß das entsprechende Recht besitzen; s muß authentisch sein, eine Maskerade kann

- durch Verkleinern einer Domäne zum Herausfallen von Entitäten aus dem Schutzbereich einer Sicherheitspolitik führen
- durch Einschleusen einer Entität in eine Domäne kann ein trojanisches Pferd eingebracht werden, welches dann allgemeine Rechte der Entitäten dieser Domäne mitnutzen kann.

Aus dem gleichen Grund muß die Integrität der Parameter gewährleistet oder überprüfbar sein.

Laufzeitkomplexität

Entsprechend der Bemerkungen zu *JoinPolicySet* ist die Laufzeit proportional zu $|\{P_i\}| \cdot (|Dom_M| + 1)$ respektive zu $|Dom_{PolId}|$.

Metapolitikspezifische Operationen zum Ändern von Rechten

```
bool GrantRight (ENT s,
                 ENT Subject,
                 POL PolId,
                 RIGHTSET r);
bool DenyRight ( ENT s,
                 ENT Subject,
                 POL PolId,
                 RIGHTSET r);
```

Operationen zur Vergabe bzw. Rücknahme von Rechten; diese Operationen verändern die Zugriffsmatrix.

Parameter

s	identifiziert die ausführende Entität.
Subject	identifiziert die Entität, für die Rechte vergeben/gelöscht werden sollen.
PolId	ist der Name der Sicherheitspolitik, auf die sich die Rechte beziehen.
r	ist die Rechtemenge.

Rückgabewert

Die Operation terminiert mit dem Wert *true* genau dann, wenn s das Recht zur Ausführung in der Rechtematrix besitzt.

Vorbedingungen

Die ausführende Entität s muß das entsprechende Recht zur Rechtemanipulation besitzen; s muß authentisch sein, eine Maskerade kann jede Form der unzulässigen Manipulation an der Sicherheitspolitik zur Folge haben. Aus dem gleichen Grund muß die Integrität der Parameter gewährleistet oder überprüfbar sein.

Laufzeitkomplexität

Bei der vorliegenden Implementierung der Rechtematrix als 2-dimensionales Feld ist die Laufzeit der Operationen eine Konstante, wobei wieder eine konstante Zeit für die Mengenoperationen angenommen wird. Bei wirklich großen Entitäten- und Sicherheitspolitikmengen und kleinem virtuellem Adreßraum der Basismaschine kann es notwendig werden, Implementierungsformen für dünn besetzte Matrizen zu wählen, so daß die Zugriffsoperationen sich dann möglicherweise verteuern.

Metapolitikspezifische Operationen zum Ändern von V und K

bool SetV (ENT s,
POL V_Pol);
bool SetK (ENT s,
POL K_Pol);

Diese Operationen tauschen die Vollständigkeits- bzw. Konfliktpolitik der Metapolitik aus. Sie sind dem Metapolitikadministrator vorbehalten und besitzen eine konstante Laufzeit.

Die Interaktionsklassifizierung

POL classify (ENT s, ENT o);

Diese Operation bildet das Herz der Metapolitik; sie entscheidet bei jeder Interaktion zwischen den Entitäten in der Metapolitikdomäne, zu welcher Interaktionsklasse diese gehört und wählt auf der Grundlage dieser Klassifizierung die zuständige Sicherheitspolitik aus. Sie implementiert unmittelbar die in Definition 4.6 genannte Fallunterscheidung.

$$classify(s,o) = \begin{cases} P_j & \mid |\Pi_s| = |\Pi_o| = 1 \ \wedge \ \Pi_s = \Pi_o = \{P_j\} \\ V & \mid |\Pi_s \cap \Pi_o| \ = \ 0 \\ K & \mid |\Pi_s \cap \Pi_o| \geq 1 \ \wedge \ \exists e \in \{s,o\} : |\Pi_e| > 1 \end{cases}$$

Für HRU-modellierte Metapolitiken ist diese zweidimensionale Variante der Klassifizierungsfunktion im allgemeinen nicht ausreichend; in diesem Fall kommt dann die in Definition 4.14 beschriebene k-dimensionale Variante zur Anwendung.

Laufzeitkomplexität

Hier können nun die Früchte des bei domänenmanipulierenden Operationen getrieben Aufwands geerntet werden; die Laufzeit der Operation ist unter den bereits genannten Voraussetzungen hinsichtlich der Mengenoperationen proportional zu $|\{P_i\}|$; bei weiterer Optimierung der Datenstrukturen (z.B. durch

Verwendung einer Hash-Funktion für die Implementierung der Mengen) ist dieses Ergebnis weiter auf eine konstante Laufzeit verbesserbar.

6.2 Sicherheitsarchitektur

Metapolitiken sind gekapselte, manipulationsgeschützte Einheiten, die sämtliche Interaktionen zwischen den Entitäten der Domänen der unterstellten Sicherheitspolitiken klassifizieren, aus dieser Klassifikation die zuständige Sicherheitspolitik identifizieren und diese dann die Interaktion autorisieren lassen. Die Analogie zu den Aufgaben einer Sicherheitspolitik führt zu äquivalenten Anforderungen an ein Implementierungskonzept für Metapolitiken:

a) Es muß die vollständige Kommunikationskontrolle gewährleisten.

b) Es muß die Manipulationssicherheit der Metapolitik gewährleisten.

c) Es muß die Metapolitikpersistenz gewährleisten.

d) Es muß eine unmanipulierbare Zuordnung zwischen einer Metapolitik und den zugeordneten Sicherheitspolitiken herstellen.

Dieses sind exakt die Anforderungen, die bereits bei der Diskussion eines Implementierungskonzepts für Sicherheitspolitiken in Kapitel 3 aufgestellt wurden und die dort zur Entwicklung des Custodian-Paradigmas geführt hatten. Für die Implementierung von Metapolitiken können daher in vollem Umfang die Ergebnisse aus Kapitel 3 verwendet werden.

Ein Beispiel für die resultierende Architektur eines Systems mit zwei Sicherheitspolitiken und einer Metapolitik zeigt Bild 6.2. Wie bereits in Bild 3.11 gezeigt sind Sicherheitspolitiken durch einen Bindevorgang an Applikationsentitäten angebunden; durch diesen Bindevorgang werden sämtliche Interaktionen der Applikationsentitäten über die Sicherheitspolitik umgeleitet und können so von ihr autorisiert werden. Bei Metapolitiken findet ein äquivalenter Bindervorgang statt, durch welchen nun Sicherheitspolitiken mit Metapolitiken assoziiert werden.

Der Bindevorgang, die vollständige Kontrolle der Interaktionen, die Gewährleistung der Manipulationssicherheit und die Persistenz werden mit denselben Mechanismen hergestellt, wie sie bei der Implementierung der Sicherheitspolitiken in Kapitel 3 zur Anwendung kommen. Lediglich die Assoziierung zwischen anwendungsspezifischen Operationen und Sicherheitspolitikregeln entfällt hier: eine Metapolitik klassifiziert eine Interaktion ausschließlich auf der Grundlage der beteiligten Entitäten und bestimmt allein damit die zugehörige Sicherheitspolitik; erst hier findet dann die Zuordnung der Interaktion zu einer konkreten Sicherheitspolitikregel statt.

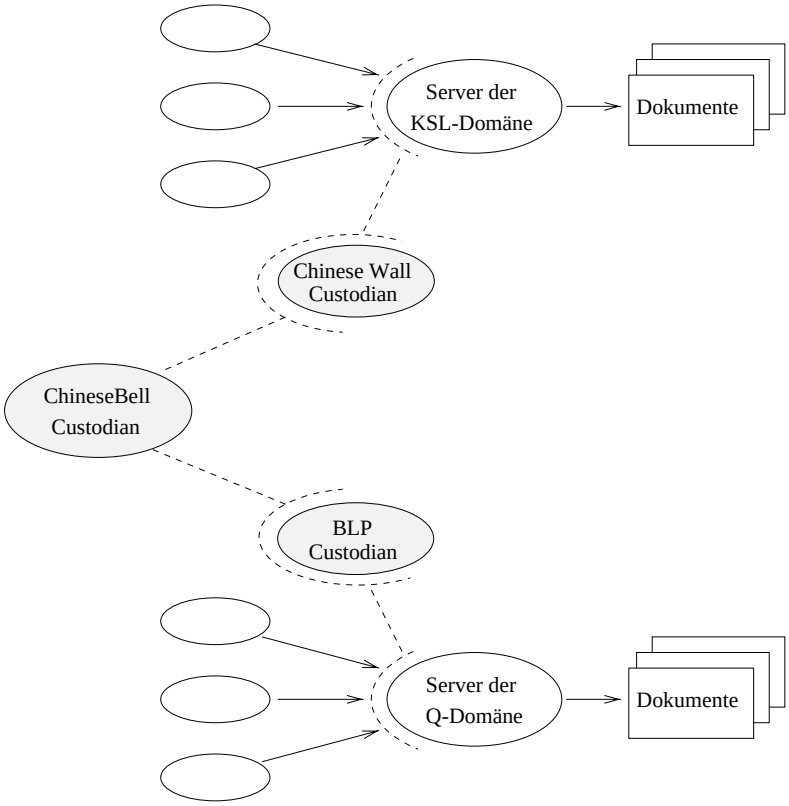


Bild 6.2: Architektur eines Systems mit zwei Sicherheitspolitiken und einer Metapolitik

6.3 Zusammenfassung

In diesem Kapitel wird ein Implementierungskonzept für Metapolitiken vorgestellt. Dieses Konzept beruht auf der Formulierung einer Metapolitik als abstrakter Datentyp und seiner Integration in eine Sicherheitsarchitektur. Ausgehend von einer Metapolitikspezifikation in Skippy$_M$ erfolgt in zwei Schritten eine Umsetzung der Spezifikation in eine ablauffähige Metapolitik als Bestandteil einer Sicherheitsarchitektur.

Im ersten Schritt wird dazu eine Skippy$_M$ – Spezifikation in weitgehend automatisierbarer Form in eine C++ – Klasse transformiert. Aus dieser Klasse wird in einem zweiten Schritt ein Custodian erzeugt [HK99], der für die Metapolitik sämtliche zur Laufzeit benötigten Eigenschaften herstellt (Manipulationssicherheit, Kommunikationskontrolle, Persistenz und Anbindung der Sicherheitspolitikmenge $\{P_i\}_{i \in I}$) und ihre Integration in die Sicherheitsarchitektur einer konkreten Systemplattform vornimmt [Küh95a].

Auf der Grundlage dieses Konzeptes wird dann eine konkrete Implementierung vorgestellt, die einerseits den exemplarischen Nachweis der Machbarkeit führt und andererseits eine konkrete Diskussion der Laufzeitkomplexitäten der Metapolitikoperationen

ermöglicht. Die in Anhang B gezeigte Implementierung verlagert einen großen Teil der Laufzeitkosten auf die relativ seltenen Operationen zur Manipulation der Politikdomänen; die Laufzeiten dieser Operationen sind proportional zu $|Dom_M| \cdot |\{P_i\}_{i \in I}|$, während die bei jeder Entitäteninteraktion aufgerufene *classify*-Operation in konstanter Zeit abläuft. Durch eine aufwendigere Implementierung der Entitätenmenge unter Verwendung von Referenzzählern und eine leicht abgewandelte Schnittstelle läßt sich die Laufzeitkomplexität der Operationen zur Manipulation der Politikdomänen noch weiter auf eine Zeit proportional zur Anzahl der zu einer Domäne hinzukommenden oder zu entfernenden Entitäten verbessern.

Aus diesen Ergebnissen lassen sich Konsequenzen für die Architektur von Multipolitikensystemen ableiten. Zum einen stellt sich heraus, daß dieselben Architekturprinzipien, die eine Implementierung von Sicherheitspolitiken in Multipolitikensystemen ermöglichen, gleichfalls zur Implementierung von Metapolitiken notwendig und hinreichend sind; die Anwendbarkeit des Custodian-Paradigmas sowohl auf Sicherheitspolitiken als auch auf Metapolitiken demonstriert dies praktisch. Zum anderen erlauben es Laufzeit und Umfang der Metapolitik-Mechanismen, sie auf den elementarsten Ebenen einer Betriebssystemarchitektur zu plazieren, so daß hierdurch eine minimale Größe der *Trusted Computing Base* erreichbar ist.

7 Zusammenfassung

Das Szenarium dieser Arbeit sind große Organisationen in Wirtschaft und Verwaltung, deren Unternehmensbereiche, Abteilungen und Projekte vielfach eigene Sicherheitsanforderungen besitzen, die die spezifischen Besonderheiten der jeweiligen Organisationseinheit berücksichtigen. Eine Abbildung dieser Strukturen auf ein unternehmensweites IT-System führt zu einem verteilten System mit einer Vielzahl von Sicherheitsdomänen, in dem jede Domäne ihre eigene, individuelle Sicherheitspolitik besitzt. Die neue Referenzarchitektur des *Technical Architecture Framework for Information Management* [DIS96] begreift solche Multipolitikensysteme als eine der zentralen zukünftigen Herausforderungen und stellt die grundsätzlichen Forderungen auf, daß zukünftige IT-Systeme koexistierende Sicherheitspolitiken jeglicher Komplexität unterstützen und die Interoperabilität zwischen Entitäten aus unterschiedlichen Sicherheitsdomänen gewährleisten müssen.

Das Spannungsfeld zwischen Interoperabilität und Informationssicherheit in diesem Umfeld ist das Thema dieser Arbeit. In Multipolitikensystemen führen die individuellen Sicherheitsanforderungen der einzelnen Sicherheitsdomänen zur Bildung autonomer Inseln, auf denen strikt die Regeln der lokalen Sicherheitspolitik durchgesetzt werden. Dies garantiert dann zwar ein hohes Qualitätsniveau der Informationssicherheit innerhalb einer Sicherheitsdomäne, durchkreuzt aber die Interoperabilität innerhalb des Gesamtsystems und konterkariert damit eine der elementarsten Errungenschaften verteilter Systeme.

Dies stellt in der heutigen IT-Landschaft ein gravierendes Defizit dar, da hier ein Konflikt zwischen der Interoperabilität in verteilten Systemen und ihrer Sicherheit offenbar wird. Eine Beseitigung dieses Defizits darf dabei nicht hinter dem bereits erreichten zurückfallen; gesucht ist ein Konzept, welches eine auf hohem Niveau stehende Qualität der Informationssicherheit für Interaktionen zwischen den Sicherheitsdomänen erzielt, ohne dazu das durch Sicherheitspolitiken erzielbare Niveau der Informationssicherheit innerhalb von Domänen negativ zu beeinflussen.

Mit diesem Problem setzt sich die vorliegende Arbeit auseinander. Der beschrittene Weg beruht auf der Idee, Regeln über Interaktionen zwischen den Sicherheitsdomänen eines Multipolitikensystems in übergeordneten Metapolitiken zu formulieren. In gleicher Weise, in der Sicherheitspolitiken die Interaktionen innerhalb einer Sicherheitsdomäne kontrollieren, kontrollieren Metapolitiken die Interaktionen zwischen den Si-

cherheitsdomänen.

Die Arbeit beginnt mit einer allgemeinen Einführung in die Thematik und Begrifflichkeit. Anschließend wird ein Szenarium entwickelt, an dem die eingeführten Begriffe veranschaulicht werden und aus dem gleichfalls die gesamte Arbeit ihre Beispiele bezieht. Als Beispielszenarium dient das konzernweite IT-System eines dezentralisierten Unternehmens mit geographisch verteilten Niederlassungen. Die strukturelle Gliederung des Unternehmens in weitgehend autonome Unternehmensbereiche spiegelt sich in der Strukturierung des unternehmensweiten IT-Systems in Sicherheitsdomänen wieder, welche die individuellen Sicherheitsanforderungen einzelner Unternehmensbereiche reflektieren. Am Beispiel mobiler Benutzer wird unter anderem gezeigt, welche Komplexität allein ein einziger Aspekt der Informationssicherheit, die Authentisierung, in verteilten Multipolitikenszenarien hierdurch erhält.

Die Grundlagen für das Metapolitikkonzept werden durch die Diskussion von Sicherheitspolitiken in Kapitel 3 gelegt. Sicherheitspolitiken werden als methodischer Ansatz charakterisiert, auf ganzheitliche Weise eine Strategie zur Realisierung einer konkreten Menge von Sicherheitsanforderungen zu entwickeln und zu formulieren. Ziel des Kapitels ist dabei die Entwicklung einer Modellierungs-, Spezifizierungs- und Implementierungsmethode, die ausdrucksstarke Paradigmen zur Formulierung eines signifikanten Spektrums von Sicherheitspolitiken enthält. Die vorgestellte Methode zur Modellierung von Sicherheitspolitiken baut auf den bekannten Zugriffsmatrizen zur Repräsentation von Rechten und den Verbänden zur Repräsentation von Informationsflüssen auf. Besondere Berücksichtigung fand weiterhin der von Harrison, Ruzzo und Ullman entwickelte HRU-Kalkül, der durch seine Fähigkeit zur Modellierung dynamischen Verhaltens für die Beschreibung einer Vielzahl wahlfreier und obligatorischer Sicherheitspolitiken zur Zugriffssteuerung verwendet wird, und der Aussagen über die grundsätzlichen Möglichkeiten und Grenzen der Analysierbarkeit von Zugriffssteuerungspolitiken ermöglicht. Es wird argumentiert, daß mit einer Kombination dieser Grundelemente – den Zugriffsmatrizen, den Verbänden und dem Zustandsautomatenmodell des HRU-Kalküls – ein Modellierungsgerüst entsteht, mit dem ein breites Spektrum aus der Klasse der Zugriffssteuerungspolitiken modelliert werden kann.

Für dieses Modellierungsgerüst wird anschließend eine Spezifikationssprache entwickelt, in der unter anderem HRU-Automatenmodell, Zugriffsmatrizen, Verbände und Domänendefinitionen fester Bestandteil der Sprache sind. Dieser Ansatz führte zu kleinen, lesbaren und übersichtlich strukturierten Spezifikationen, die den kritischen Paradigmenwechsel zwischen Modell und Implementierung durch eine kleine semantische Lücke vereinfachen.

Da sich traditionelle Prinzipien zur Implementierung so spezifizierter Sicherheitspolitiken als unzulänglich erwiesen, wird hierfür das Custodian-Paradigma entwickelt. Custodians bilden eine Laufzeitumgebung für algorithmisch formulierte Sicherheitspolitiken, welche die Manipulationssicherheit der Sicherheitspolitik, die vollständige Kontrolle der Kommunikation innerhalb der Sicherheitsdomäne, Politikpersistenz und

die Bindung einer Sicherheitspolitik an ihre Sicherheitsdomäne implementiert.

Als Ergebnis liegt somit ein methodischer Ansatz zur Formulierung und Implementierung von Sicherheitspolitiken vor, bestehend aus einem Modellierungsgerüst, einer Spezifikationssprache und einem Implementierungsparadigma. Ein analoger Ansatz wird anschließend in Kapitel 4 für Metapolitiken verfolgt. Im Gegensatz zu den Sicherheitspolitiken kann hier allerdings nicht mehr auf bekannte Modellierungstechniken aufgebaut werden; in der Schaffung entsprechender Grundlagen liegt der theoretische Schwerpunkt der Arbeit.

Ausgangspunkt des Metapolitikkonzepts ist eine Klassifizierung der Interaktionen zwischen Politikdomänen, die zwei Problembereiche identifiziert: Interaktionen, welche Konflikte zwischen den beteiligten Sicherheitspolitiken hervorrufen und Interaktionen, auf die keine der beteiligten Sicherheitspolitiken anwendbar ist. Es wird argumentiert, daß solche Interaktionen neue Sicherheitsanforderungen erfordern, welche in Form von Vollständigkeits- und Konfliktpolitiken die existierenden Sicherheitspolitiken ergänzen und in dieser Form in eine Metapolitik einfließen. Schließlich werden auf der Grundlage der Klassifizierung notwendige und hinreichende Bedingungen aufgestellt, die die Anwendung der Vollständigkeits- und Konfliktpolitiken im Kontext einer Metapolitik kontrollieren.

Zur formalisierten Beschreibung einer Metapolitik wird ein Modell entwickelt, welches eine präzise Formulierung der Beziehungen zwischen Sicherheitspolitikmenge, Vollständigkeits- und Konfliktpolitik ermöglicht und den Ausgangspunkt für die Analyse der Sicherheitseigenschaften von Metapolitiken bildet. Basierend auf dem HRU-Kalkül wird die Klasse der HRU-Metapolitikmodelle definiert, und es werden Aussagen über grundsätzliche Möglichkeiten und Grenzen der Analysierbarkeit gemacht. Die konstruktive Form der Definition eines HRU-Metapolitikmodells erlaubt es dabei, direkt von den Eigenschaften der Sicherheitspolitiken auf die Eigenschaften der entsprechenden Metapolitik zu schließen.

Praktisches Ergebnis dieses Kapitels ist eine konkrete Bauanleitung für eine Metapolitik über HRU-modellierte Sicherheitspolitiken. Ausgehend von einer gegebenen Menge von Sicherheitspolitiken, über die eine Metapolitik formuliert werden soll, wird für diese Politikmenge eine Vollständigkeits- und Konfliktpolitik definiert und diese in einem HRU-Modell formalisiert. Sodann wird die aufgezeigte Konstruktionsvorschrift angewendet, um hieraus ein HRU-Modell einer Metapolitik zu erhalten. Hierauf lassen sich dann Sätze anwenden, um Aussagen über die Entscheidbarkeit bestimmter Sicherheitseigenschaften der so konstruierten Metapolitik zu erhalten.

Aufbauend auf dem entwickelten Metapolitikmodell wird dann eine Sprache zur Spezifikation von Metapolitiken vorgestellt, welche die Umsetzung eines formalen Modells in eine konkrete Implementierung einleitet. Die Spezifikationssprache basiert auf der in Kapitel 3 entwickelten Spezifikationssprache für Sicherheitspolitiken und folgt dem bereits dort angewandten Prinzip, den Paradigmenwechsel zwischen Modell und Implementierung zu vereinfachen. Dieser Ansatz führt zu einem kleinen, handhabbaren

Sprachumfang und, wie an Beispielen gezeigt, zu lesbaren und übersichtlich strukturierten Spezifikationen.

Das folgende Kapitel rundet das Metapolitikkonzept durch eine Implementierungsmethode für Metapolitiken ab. Die Methode baut auf dem in Kapitel 3 vorgestellten Custodian-Paradigma auf und transformiert eine Metapolitikspezifikation in zwei Schritten in eine ablauffähige Metapolitik als Bestandteil einer Sicherheitsarchitektur.

Im ersten Schritt wird dazu eine Spezifikation mittels eines Programmgenerators in weitgehend automatisierter Form in eine C++ – Klasse transformiert. Aus dieser Klasse wird in einem zweiten Schritt ein Custodian erzeugt, der für die Metapolitik sämtliche zur Laufzeit benötigten Eigenschaften herstellt und ihre Integration in die Sicherheitsarchitektur einer konkreten Systemplattform vornimmt. Auf der Grundlage dieses Konzeptes wird dann eine konkrete Implementierung einer Metapolitik vorgestellt, die einerseits den exemplarischen Nachweis der Machbarkeit führt und andererseits eine konkrete Diskussion der Laufzeitkomplexität der Metapolitikoperationen ermöglicht. Die im Anhang dieser Arbeit gezeigte Implementierung verlagert einen großen Teil der Laufzeitkosten auf die relativ seltenen Operationen zur Manipulation der Politikdomänen; die Laufzeit dieser Operationen ist dann proportional zum Produkt aus der Mächtigkeit der Domänenmenge und der Mächtigkeit der Sicherheitspolitikmenge einer Metapolitik, während die bei jeder Entitäteninteraktion aufgerufenen Klassifizierungsfunktion in konstanter Zeit ausgeführt wird. Durch eine aufwendigere Implementierung der Entitätenmenge unter Verwendung von Entitätenattributen und Referenzzählern und eine leicht abgewandelte Schnittstelle läßt sich die Laufzeit der Operationen zur Manipulation der Politikdomänen noch weiter auf eine Zeit proportional zur Anzahl der zu einer Domäne hinzukommenden oder zu entfernenden Entitäten verbessern.

Mit den vorliegenden Ergebnissen wird ein durchgängiges Konzept zur Modellierung, Analyse, Spezifikation und Implementierung von Metapolitiken vorgelegt. Metapolitiken formulieren strategische Regeln über die Interaktionen zwischen den Sicherheitsdomänen eines Multipolitikensystems, setzen diese gemeinsam mit den in Multipolitikensystemen existierenden Sicherheitspolitiken durch und leisten auf diese Weise einen Beitrag, den Konflikt zwischen Informationssicherheit und Interoperabilität in Multipolitikensystemen zu überwinden.

Damit rückt die Perspektive der *TAFIM* Referenzarchitektur näher, in der eine Vielzahl von Sicherheitspolitiken die individuellen Sicherheitsinteressen aller Teilnehmer bewahren und in der sichere Interaktionen zwischen den Politikdomänen gewährleisten sind. Gleichzeitig ist das erreichte ein Baustein für die nächste *TAFIM*-Generation. Metapolitiken sind die in dieser Arbeit gegebene Antwort auf die Forderung der heutigen Architekturgeneration nach Multipolitikensystemen und ihrer Interoperabilität; diese Antwort hat in Form des Custodian-Implementierungskonzepts Rückwirkungen auf die Architekturprinzipien für Multipolitikensysteme, die dadurch wiederum zum Gegenstand einer neuen Referenzarchitektur werden.

A Verwendung von Bezeichnern

Bezeichner	**Verwendung**
Dom_P	Domäne der Politik P
$rep_{P,Q}$	Repräsentantenabbildung von Dom_P nach Dom_Q
δ	Zustandsübergangsfunktion eines Zustandsautomaten
E	Eingabemenge eines Zustandsautomaten
ε	leere Folge (von Eingaben eines Zustandsautomaten)
F	Menge der Operationen von Subjekten auf Objekten
$grant$	Funktion zur Prüfung von Zugriffsrechten im Lampson'schen Zugriffssteuerungsmodell
H	die Menge der HRU-Primitiven
I	endliche Indexmenge
K	Konfliktkomponente einer Metapolitik
k	Dimension des Entitätenvektors der Eingabe für ein HRU-Modell
M	Metapolitik
$\mathcal{N}$	Menge der natürlichen Zahlen
O, o	Objektmenge und Objekt bei einer Zugriffsoperation
O^n	n-dimensoinaler Vektorraum über einer Objektmenge O
P, Q	Sicherheitspolitiken
Π_e	Menge aller Sicherheitspolitiken, in deren Domäne e liegt
R, r	Rechtemenge und Recht
$\wp(..)$	Potenzmenge
S, s	Subjektmenge und Subjekt bei einer Zugriffsoperation
S^n	n-dimensoinaler Vektorraum über einer Subjektmenge S
(s, o)	Zugriff eines Subjekts s auf ein Objekt o
V	Vollständigkeitskomponente einer Metapolitik
Z, z	Zustandsmenge und Zustand eines Zustandsautomaten
ZM	Zugriffsmatrix

B Eine Metapolitik als C++ – Klasse

```
#include "mysets.h"

#define MaxEntities 1000
#define MaxPolicies 100
#define NoPol 0
#define NoEnt 0
#define MPAdminId 42

class Metapolicy
{
public:
    typedef int ENT;
    typedef int POL;
    typedef set<ENT> DOM;
    enum RIGHT {JoinPolicySet_r, LeavePolicySet_r, SetPolicyDomain_r,
                GrantRight_r, DenyRight_r};
    typedef set<RIGHT> RIGHTSET;


    /* submit new policy to metapolicy
    */
    bool JoinPolicySet(ENT s,
                       POL PolId,
                       DOM PolDom,
                       ENT PolAdmin);


    /* remove policy from metapolicy
    */
    bool LeavePolicySet(ENT s,
                        POL PolId);
```

```
    /* update the domain of a policy
    */
    bool SetPolicyDomain(ENT s,
                         POL PolId,
                         DOM PolDom);

    /* grant some rights
    */
    bool GrantRight(ENT s,
                    ENT Subject,
                    POL PolId,
                    RIGHTSET r);


    /* withdraw some rights
    */
    bool DenyRight(ENT s,
                   ENT Subject,
                   POL PolId,
                   RIGHTSET r);


    /* alter the completeness and conflict resolution policies
    */
    bool SetV(ENT s,
              POL V_Pol);

    bool SetK(ENT s,
              POL K_Pol);


    /* class instance constructor
    */
    Metapolicy(ENT Admin);

private:
    set<POL> Policies;
    POL V,K;
    RIGHTSET rm[MaxEntities][MaxPolicies];

    mapping<POL,DOM> PolicyDomains;
    mapping<ENT, set<POL> > Pi;
    set<ENT> Entities;
```

```
    ENT MetapolAdmin;


    /* compute all Pi's;
    ** called whenever any policy domain changes or when policy
    ** enters or leaves metapolicy
    */
    void ComputePi();


    /* compute metapolicy domain;
    ** called whenever any policy domain changes or when policy
    ** enters or leaves metapolicy
    */
    void ComputeEntities();


    /* select the policy to be used for a specific entity
    ** interaction
    */
    POL classify(ENT s, ENT o);
};
```

```
/* Metapolicy Implementations
*/


/* submit new policy to metapolicy
*/
bool Metapolicy::JoinPolicySet(ENT s,
                               POL PolId,
                               DOM PolDom,
                               ENT PolAdmin)
{  if ((s == MetapolAdmin) &&
       (PolDom.in(PolAdmin)) &&
       !(Policies.in(PolId)))
   {
      Policies.set_include(PolId);
      PolicyDomains.define(PolId,PolDom);
      ComputeEntities();
      ComputePi();
      rm[PolAdmin][PolId].set_include(LeavePolicySet_r);
      rm[PolAdmin][PolId].set_include(SetPolicyDomain_r);
      rm[PolAdmin][PolId].set_include(GrantRight_r);
      rm[PolAdmin][PolId].set_include(DenyRight_r);
      return true;
   }
   else
    return false;
}


/* remove new policy from metapolicy
*/
bool Metapolicy::LeavePolicySet(ENT s,
                                POL PolId)
{  if (((s == MetapolAdmin) && (Policies.in(PolId))) ||
       ((Entities.in(s)) && (Policies.in(PolId)) &&
        rm[s][PolId].in(LeavePolicySet_r)))
   {
      Policies.set_exclude(PolId);
      PolicyDomains.undefine(PolId);
      for (e=Entities.set_iterate(NoEnt);
           e=Entities.set_iterate(e); e!=NoEnt)
        rm[e][PolId].set_clear();
      ComputeEntities();
```

```
      ComputePi();
      return true;
   }
   else
      return false;
}


/* update the domain of a policy
*/
bool Metapolicy::SetPolicyDomain(ENT s,
                                 POL PolId,
                                 DOM PolDom)
{  if (Entities.in(s) &&
       Policies.in(PolId) &&
       rm[s][PolId].in(SetPolicyDomain_r))
   {
      PolicyDomains.define(PolId,PolDom);
      ComputeEntities();
      ComputePi();
      return true;
   }
   else
      return false;
}


/* grant some rights
*/
bool Metapolicy::GrantRight(ENT s,
                            ENT Subject,
                            POL PolId,
                            RIGHTSET r)
{  if (Entities.in(s) &&
       Entities.in(Subject) &&
       Policies.in(PolId) &&
       rm[s][PolId].in(GrantRight_r))
   {
      rm[Subject][PolId] += r;
      return true;
   }
   else
      return false;
```

```
}


/* withdraw some rights
*/
bool Metapolicy::DenyRight(ENT s,
                           ENT Subject,
                           POL PolId,
                           RIGHTSET r)
{  if (Entities.in(s) &&
       Entities.in(Subject) &&
       Policies.in(PolId) &&
       rm[s][PolId].in(DenyRight_r))
   {
      rm[Subject][PolId] -= r;
      return true;
   }
   else
      return false;
}


/* alter the completeness policy
*/
bool Metapolicy::SetV(ENT s,
                      POL V_Pol)
{  if (s==MetapolAdmin)
   {  V = V_Pol;
      return true;
   }
   else
      return false;
}


/* alter the conflict policy
*/
bool Metapolicy::SetK(ENT s,
                      POL K_Pol)
{  if (s==MetapolAdmin)
   { K = K_Pol;
      return true;
   }
```

```
   else
      return false;
}


/* compute all Pi's;
** called whenever any policy domain changes or when policy enters
** or leaves metapolicy
** As this implementation is running in O(|Entities|*|Policies|),
** we might consider to change the implementation for very large
** metapolicy domains.
*/
void Metapolicy::ComputePi()
{  ENT e;
   POL p;
   set<POL> Pi_e;

   Pi.clear();
   for (e=Entities.set_iterate(NoEnt);
        e=Entities.set_iterate(e); e!=NoEnt)
      for (p=Policies.set_iterate(NoPol);
           p=Policies.set_iterate(p); p!=NoPol)
         if (PolicyDomains.map(p).in(e))
         {  Pi_e = Pi.map(e);
            Pi_e.set_include(p);
            Pi.define(e,Pi_e);
         }
}


/* compute metapolicy domain;
** called whenever any policy domain changes or when policy enters
** or leaves metapolicy
*/
void Metapolicy::ComputeEntities()
{  POL p;
   Entities.set_clear();
   for (p=Policies.set_iterate(NoPol);
        p=Policies.set_iterate(p); p!=NoPol)
      Entities.set_union(PolicyDomains.map(p));
}
```

```
/* select the policy to be used for a specific entity interaction
*/
Metapolicy::POL Metapolicy::classify(ENT s, ENT o)
{  set<POL> Pi_s, Pi_o;

   Pi_s = Pi.map(s);
   Pi_o = Pi.map(o);

   if ((Pi_s.set_card()==1) &&
       (Pi_o.set_card()==1))
      return Pi_s.set_iterate(NoPol);
   else if (Pi_s.set_intersect(Pi_o).set_card()==0)
      return V;
   else
      return K;
}


/* class instance constructor
*/
Metapolicy::Metapolicy(ENT Admin)
{  int i,j;

   MetapolAdmin = Admin;

   for (i=0; i++; i<MaxEntities)
      for (j=0; j++; j<MaxPolicies)
         rm[i][j].set_clear();

   Policies.set_clear();
   Entities.set_clear();
   PolicyDomains.clear();
   Pi.clear();
}
```

```
/* outline how to create a metapolicy
*/


/* functions to
** - get completeness and conflict policies from a policy library
** - transform user name to unique id
** - get the domain of a policy
** (we assume the implementations to eb elsewhere)
*/
Metapolicy::POL GetComPolicy();
Metapolicy::POL GetConPolicy();
Metapolicy::ENT getuid(char *UserName);
Metapolicy::DOM GetDomain(Metapolicy::POL pol);


/* choose
** - a metapolicy administrator: user "root" becomes administrator
** - completeness and conflict policies
*/
#define Administrator      getuid("root")
#define CompletenessPolicy GetComPolicy()
#define ConflictPolicy     GetConPolicy()


/* two policies are to join metapolicy
*/
#define ChineseWall 1
#define CWOwner getuid("HansMuster")

#define BLP 2
#define BLPOwner getuid("ElkeMuster")
```

```
main()
{  Metapolicy ChineseBell(Administrator);
   Metapolicy::DOM CWDom, BLPDom;

   /* set completeness and conflict policies
   */
   ChineseBell.SetV(Administrator,CompletenessPolicy);
   ChineseBell.SetK(Administrator,ConflictPolicy);

   /* join chinese wall policy
   */
   ChineseBell.JoinPolicySet(Administrator, ChineseWall, CWDom, CWOwner);
   ChineseBell.SetPolicyDomain(CWOwner, ChineseWall, GetDomain(ChineseWall));

   /* join blp policy
   */
   ChineseBell.JoinPolicySet(Administrator, BLP, BLPDom, BLPOwner);
   ChineseBell.SetPolicyDomain(BLPOwner, BLP, GetDomain(BLP));
}
```

```
/* interface definitions of template classes set and mapping
*/


template <class t> class set
{
public:
   bool in(t);
   set set_union(set);
   set set_intersect(set);
   set set_clear();
   void set_include(t);
   void set_exclude(t);
   int set_card();
   bool set_mt();
   t set_iterate(t);

   operator = (set);
   operator -= (set);
   operator += (set);

   set;
};


template <class t, class u> class mapping
{
public:
   void define(t,u);
   void undefine(t);
   void clear();
   u map(t);

   mapping;
};
```

Literaturverzeichnis

[AS91] Paul E. Ammann and Ravi S. Sandhu. Safety Analysis for the Extended Schematic Protection Model. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 87–97, Oakland, California, May 1991. IEEE Press.

[BAN89] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. In *Operating Systems Principles*, pages 1–13. ACM, December 1989.

[Bib77] K.J. Biba. Integrity Considerations for Secure Computer Systems. Technical Report ESD-TR-76-372, MITRE, Bedford, Massachusetts, April 1977.

[BKAL97] Ciarán Bryce, Winfried E. Kühnhauser, Remy Amouroux, and Mauricio Lopéz. CWASAR: A European Infrastructure for Secure Electronic Commerce. *Journal of Computer Security, IOS Press*, 5(3):225–235, 1997.

[BL73a] D. Elliott Bell and Leonard J. LaPadula. Secure Computer Systems: Mathematical Foundations (Vol.I). Technical Report AD 770 768, MITRE, Bedford, Massachusetts, November 1973.

[BL73b] D. Elliott Bell and Leonard J. LaPadula. Secure Computer Systems: Mathematical Foundations (Vol.II). Technical Report AD 771 543, MITRE, Bedford, Massachusetts, November 1973.

[BL74] D. Elliott Bell and Leonard J. LaPadula. Secure Computer Systems: A Refinement of the Mathematical Model. Technical Report AD 780 528, MITRE, Bedford, Massachusetts, April 1974.

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

[BN89] David F.C. Brewer and Michael J. Nash. The Chinese Wall Security Policy. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 206–214. IEEE Computer Society Press, May 1989.

[Bry96] Ciarán Bryce. Syntax and Semantics of Skippy Type Declarations. Technical report, GMD - Forschungszentrum Informationstechnik, 1996.

[Bry97] Ciarán Bryce. Security Engineering of Lattice-Based Policies. In *Proceedings of the Computer Security Foundations Workshop*, pages 195–207, Rockport, Massachusetts, 1997. IEEE Press.

[Bun97] Bundesamt für Informationssicherheit. *IT-Grundschutzhandbuch 1997*. Bundesanzeiger-Verlag, 1997.

[CCI88] CCITT. *X.200 Reference Model of Open Systems Interconnection for CCITT Applications*, November 1988. Blue Book, FASCICLE VIII.4.

[CDK94] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems*. Addison-Wesley, 1994.

[Com96] NIST (USA), CSE (Canada), BSI (Germany), NNCSA (The Netherlands), NSA (USA), UKITSCS (UK), SCSSI (France). *Common Criteria for Information Technology Security Evaluation, Version 1.0*, January 1996.

[Com98] NIST (USA), CSE (Canada), BSI (Germany), NNCSA (The Netherlands), NSA (USA), CESG (UK), SCSSI (France). *Common Criteria for Information Technology Security Evaluation, Version 2.0*, May 1998.

[CR73] Richard Courant and Herbert Robbins. *Was ist Mathematik?* Springer-Verlag, 1973.

[CW85] Luca Cardelli and Peter Wegner. On Understanding Types, Data Abstraction, and Polymorphism. *Computing Surveys*, 17(4), Dec 1985.

[CW87] David D. Clark and David R. Wilson. A Comparsion of Commercial and Military Computer Security Policies. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 184–194. IEEE Computer Society, April 1987.

[Den76] Dorothy E. Denning. A Lattice Model of Secure Information Flow. *Communications of the ACM*, 19(5):236–242, May 1976.

[DH77] Whitfield Diffie and Martin E. Hellman. Exhaustive cryptanalysis of the nbs data encryption standard. *Computer*, 10(6):74–84, June 1977.

[Dil90] A. Diller. *Z: An Introduction to Formal Methods*. John Wiley and Sons, 1990.

[DIS96] Defense Information Systems Agency. *Technical Architecture Framework for Information Management, Volume 6: Department of Defense Goal Security Architecture*, 30. April 1996. Version 3.0.

[DoD83] Department of Defense. *Trusted Computer System Evaluation Criteria*, August 1983.

[FM98] Edward A. Feustel and Terry Mayfield. The DGSA: Unmet Information Security Challenges for Operating System Designers. *Operating Systems Review*, 32(1):3–22, January 1998.

[Fol89] Simon N. Foley. A Model for Secure Information Flow. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 248–258. IEEE Computer Society, May 1989.

[GHF89] Guy-L. Grenier, Richard C. Holt, and Mark Funkenhauser. Policy vs. Mechanism in the Secure Tunis Operating System. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 84–93. IEEE Computer Society, May 1989.

[GM82] J.A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society, April 1982.

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

[HKK93] Hermann Härtig, Winfried E. Kühnhauser, and Oliver C. Kowalski. The BirliX Security Architecture. *Journal of Computer Security, IOS Press*, 2(1):5–21, 1993.

[Hor85] P. Horster. *Kryptologie*, volume BI47 of *Reihe Informatik*. BI Wissenschaftsverlag, 1985. ISBN 3-411-03106-9.

[Hos92a] Hilary H. Hosmer. Metapolicies II. In *Proceedings of the 15th National Computer Security Conference*, pages 369–378. NIST-NCSC, United States Government Printing Office: 1992-625-512:60546, 1992.

[Hos92b] Hilary H. Hosmer. The Multipolicy Paradigm. In *Proceedings of the 15th National Computer Security Conference*, pages 409–422. NIST-NCSC, United States Government Printing Office: 1992-625-512:60546, 1992.

[HR78] Michael A. Harrison and Walter L. Ruzzo. Monotonic Protection Systems. In R. DeMillo, D. Dobkin, A. Jones, and R. Lipton, editors, *Foundations of Secure Computation*, pages 337–365. Academic Press, 1978.

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

[Int84] International Standarisation Organisation. *ISO 7498: Information Processing Systems – Open Systems Interconnection – Basic Reference Model*, 1984.

[ITS89] ZSI — Zentralstelle für Sicherheit in der Informationstechnik, Bonn, Germany. *IT-Sicherheitskriterien*, Januar 1989.

[ITS90] Bundesminister des Inneren, Bonn, Germany. *Information Technology Security Evaluation Criteria (ITSEC), Version 1*, Mai 1990.

[KM93] V. Kessler and S. Mund. Sicherheitsmodelle. Studie des Verbundprojektes ,,Referenzmodell für sichere IT-Systeme", Siemens AG, Juni 1993.

[KP99] Winfried E. Kühnhauser and Martin Protzen. An Automatic Code Generator for Skippy Types. GMD Technical Report (to appear), German National Research Center for Information Technology, St. Augustin, Germany, 1999.

[Küh95a] Winfried E. Kühnhauser. A Paradigm for User-Defined Security Policies. In *Proceedings of the 14th IEEE Symposium on Reliable Distributed Systems*, pages 135–144, Bad Neuenahr, Germany, September 1995. IEEE Computer Society Press.

[Küh95b] Winfried E. Kühnhauser. On Paradigms for Security Policies in Multipolicy Environments. In *Proceedings of the 11th International Information Security Conference (IFIP/SEC '95)*, Cape Town, South Africa, May 1995. Chapman & Hall.

[Küh98] Winfried E. Kühnhauser. A Classification of Interdomain Actions. *Operating Systems Review*, 32(4):47–61, October 1998.

[KvKO95] Winfried E. Kühnhauser and Michael von Kopp Ostrowski. A Framework to Support Multiple Security Policies. In *Proceedings of the 7th Canadian Computer Security Symposium*, Ottawa, Canada, May 1995. Communication Security Establishment Press.

[KW97] Darell M. Kienzle and William A. Wulf. A Practical Approach to Security Assessment. In *Proceedings of the New Security Paradigms Workshop*, pages 5–16. ACM SIG on Security, Audit, and Control, ACM Press, 1997.

[LCC+75] R. Levin, E. Cohen, W. Corwin, F. Pollack, and W. Wulf. Policy/Mechanism Separation in Hydra. *Operating Systems Review*, 9(5):132–140, November 1975.

[LLF+96] Jacques-Louis Lions, Lennart Lebeck, Jean-Luc Fauquembergue, Gilles Kahn, Wolfgang Kubbat, Stefan Levedag, Leonardo Mazzini, Didier Merle, and Colin O'Halloran. Flight 501 Failure. Technical report,

Flight 501 Failure Inquiry Board, Paris, July 1996. Available from http://www.esa.int/tidc/Press/Press96/ariane5rep.html.

[LS78] R. Lipton and L. Snyder. On Synchronization and Security. In R. DeMillo, D. Dobkin, A. Jones, and R. Lipton, editors, *Foundations of Secure Computation*, pages 367–385. Academic Press, 1978.

[Lux95] Wolfgang Lux. Integrating Custodians into OSF-DCE. In *Proceedings of the Workshop on Anwendungsunterstützung für heterogene Rechnernetze*, Freiberg, Germany, March 1995. Technical University of Freiberg Press.

[Man74] Zohar Manna. *Mathematical Theory of Computation.* McGraw-Hill Book Company, 1974. ISBN 0-07-039910-7.

[McL94] John McLean. Security Models. In *Encyclopedia of Software Engineering*, pages 1136–1145. John Wiley & Sons, 1994.

[Min95] Spencer E. Minear. Providing Policy Control Over Object Operations in a Mach Based System. In *Proceedings of the fifth USENIX UNIX Security Symposium*, pages 141–156. USENIX Association, 1995.

[MNSS88] S.P. Miller, B.C. Neumann, J.I. Schiller, and J.H. Saltzer. Kerberos Authentication and Authorization System. Project Athena Technical Plan Section E.2.1, Massachussetts Institute of Technology, October 1988.

[Nat77] National Bureau of Standards, National Technical Information Service, US Department of Commerce. Springfield, Virginia 22161. *Data Encryption Standard (FIPS Publication 46)*, January 1977.

[NCS87] National Computer Security Center. *Trusted Network Interpretation*, July 1987. NCSC-TG-005 Version-1.

[Neu95] Peter G. Neumann. *Computer-Related Risks.* Addison-Wesley Publishing Company, 1995.

[NS78] Roger M. Needham and Michael D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[NS87] Roger M. Needham and Michael D. Schroeder. Authentication Revisited. *Operating Systems Review*, 21(1):7, January 1987.

[OFSS96] Duane Olawski, Todd Fine, Edward Schneider, and Ray Spencer. Developing and Using a "Policy Neutral" Access Control Policy. In *Proceedings of the New Security Paradigms Workshop*, pages 60–67. ACM SIG on Security, Audit, and Control, ACM Press, 1996.

[Opp97] Rolf Opplinger. *IT-Sicherheit.* Vieweg, 1997.

[OR87] Dave Otway and Owen Rees. Efficient and Timely Mutual Authentication. *Operating Systems Review*, 21(1):8–10, January 1987.

[OSF92] Open Software Foundation. *OSF DCE 1.0 Application Development Guide*, 1992.

[San92a] Ravi S. Sandhu. Lattice-Based Enforcement of Chinese Walls. *Computers & Security*, 11:753–763, 1992.

[San92b] Ravi S. Sandhu. The Typed Access Matrix Model. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 122–136. IEEE Computer Society, May 1992.

[SB92] Ingrid Schaumüller-Bichl. *Sicherheitsmanagement.* BI Wissenschaftsverlag, 1992. Patrik Horster, editor.

[Sch93] Wolfgang Schneider. SecuDE Overview. GMD Technical Report 775, German National Research Center for Information Technology, GMD Darmstadt, Germany, 1993.

[SEN84] *ACM SIGSOFT Software Engineering Notes, RISKS Section*, January 1984.

[SEN89] *ACM SIGSOFT Software Engineering Notes, RISKS Section*, April 1989.

[SHK+94] Susann Sonntag, Hermann Härtig, Oliver Kowalski, Winfried E. Kühnhauser, and Wolfgang Lux. Adaptability Using Reflection. In *Proceedings of the Hawaii International Conference on System Sciences*, 1994.

[Spi87] J.M. Spivey. The Z Notation: A Reference Manual. Technical report, Oxford University, 1987.

[Tan95] Andrew S. Tanenbaum. *Distributed Operating Systems.* Prentice Hall, 1995.

[TW89] Ph. Terry and S. Wiseman. A New Security Model. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 215–228. IEEE Computer Society Press, May 1989.

[Wag96] Holger Wagner. Ein Konzept zur Bindung von Custodians an verteilte Anwendungen. Technical report, GMD-Studien Nr. 309, GMD Forschungszentrum Informationstechnik, December 1996. ISBN 3-88457-309-8.

[WWK96] William A. Wulf, Chenxi Wang, and Darrell M. Kienzle. A New Model of Security for Distributed Systems. In *Proceedings of the New Security Paradigms Workshop*, pages 34–43. ACM SIG on Security, Audit, and Control, ACM Press, 1996.

[Zim95] Philip R. Zimmermann. *The Official PGP User's Guide.* MIT Press, 1995.