TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

Master's Thesis

Access Control Policy Editor and Analyzer for Policies on a Business Level

carried out at the
Institute of Computer Aided Automation
Automation Systems Group
Vienna University of Technology

under the guidance of
Dipl.-Ing. Dr.techn. Christopher Krügel
and
Dipl.-Ing. Dr.-Ing. Marcel Graf
as advisor at the IBM Zurich Research Laboratory

by

Franz-Stefan Preiss
Kesselbodengasse 47
3910 Zwettl, Austria

Date _____ Signature _____

Abstract

Access control is extensively used as security technology to prevent unauthorized access to protected information and system resources in accordance with a policy. However, the formulation of such policies is a complicated task and requires a lot of technical knowledge. This task can therefore only be performed by security experts but not by the persons that are familiar with the business domain for which the access shall be controlled.

This thesis introduces therefore a policy language that expresses the access control behavior on a higher abstraction level — on the business level. The level elevation is achieved by formulating the policies around the data itself and its meaning to the business. The business meaning is introduced by formalizing business concepts and classifying the data according to these concepts.

On top of this new policy language, this thesis proposes a number of analysis algorithms that may be performed by the policy authors in order to answer common questions that arise during the authoring process and to simulate an evaluation of the policies. Moreover, since the authoring of access control policies is not a task that is performed by a single person, a policy delegation mechanism is proposed that allows multiple authors to formulate their policies collaboratively. In order to enable an enforcement of the policies in an existing IT infrastructure without making any changes to the infrastructure, it is shown how the policies on the business level are translated into the standardized policy language XACML.

Finally a prototype of a user friendly policy editor and analyzer is created that puts all the pieces together in one tool. This tool enables non-technical policy authors to formulate policies on the business level collaboratively and to perform the introduced analysis algorithms on the policies. To facilitate the formulation of the policies for the non-technical authors, significant emphasis was put on the usability aspect of the prototype.

Kurzfassung

Zugriffskontrolle auf Basis von bestimmten Regelwerken, sogenannten Policies, ist eine weit verbreitete Technik um unautorisierten Zugriff auf geschützte Informationen und Systemressourcen zu verhindern. Die Formulierung von solchen Policies ist eine schwierige Aufgabe, welche ein hohes Maß an technischem Wissen erfordert. Diese Aufgabe kann demnach nur von Sicherheitsexperten durchgeführt werden, aber nicht von den eigentlichen Verantwortlichen, die mit dem jeweiligen Geschäftsbereich vertraut sind, in dem der Zugriff kontrolliert wird.

Diese Arbeit stellt daher eine neue Sprache zur Formulierung von Policies vor, welche das Zugriffsverhalten auf höherer Abstraktionsebene, auf der Geschäftsebene, ausdrückt. Die Abstrahierung wird dadurch erreicht, dass die Policies nun mit Hilfe von Begriffen formuliert werden, die in dem jeweiligen Geschäftsbereich geläufig sind.

Auf Basis dieser neuen Sprache wird eine Reihe von Analysealgorithmen eingeführt, die den Autoren von Policies dabei helfen, Fragen zu beantworten, die beim Verfassen der Policies häufig auftreten, und die ihn den Entscheidungsfindungsprozesses für einen Zugriff auf Informationen simulieren lassen. Da das Verfassen von Policies eine Aufgabe ist, die meist von mehreren Personen in Zusammenarbeit durchgeführt wird, beschreiben wir einen Mechanismus zur Delegierung der Zugriffskontrolle. Damit die Policies auf der Geschäftsebene auch in bereits bestehenden IT Infrastrukturen eingesetzt werden können ohne Änderungen an dieser Infrastruktur vornehmen zu müssen, wird gezeigt, wie eine Übersetzung der Policies in die standardisierte Sprache namens XACML ausgeführt werden kann.

Schließlich wird ein Prototyp eines Policyeditors gezeigt, welcher alle zuvor erwähnten Konzepte und Funktionen in sich vereint. Dieses Werkzeug ermöglicht es auch Personen, die nicht über technisches Wissen verfügen, Policies auf der Geschäftsebene zu Verfassen und die erwähnten Analysefunktionen zur Beantwortung auftretender Fragen durchzuführen. Um für die technisch nicht versierten Autoren den Erstellungsprozess der Policies so einfach wie möglich zu gestalten, wurde bei dem erstellten Editor besonders Wert auf die Benutzerfreundlichkeit gelegt.

Acknowledgements

I want to thank Christopher Krügel, my professor at the Vienna University of Technology, who immediately agreed in supervising me without even knowing me personally and although I stayed in Switzerland during the time I was working on the thesis. I am very grateful for the support of my study abroad and for the quick and uncomplicated handling of all the administrative issues I had to deal with to finish my study.

Many thanks to Marcel Graf, my advisor at the IBM Research Laboratory in Zurich, for his guidance and support in introducing me to scientific research. Thank you for all the helpful discussions we had and for the excellent feedback you gave me during my time in the lab.

The friends that I found during my stay in Switzerland deserve my gratitude as well. I sincerely have to thank all of you for the great time I was allowed to experience. Especially I have to thank my friends at IBM for contributing to such a pleasant and comfortable work environment and for all the fun we had at our non work related activities. Alexandru, Caroline, Dieter, Lydia, Martin, Mathias, Rafik and Urko, thank you!

Furthermore, I have to thank my colleague and very good friend Christian Distelberger for going through all the pain of the computer science study together with me and for the constant push he gave me that enabled us both to finish the study in that short time.

Finally I want to thank my extraordinary parents. I am very grateful for giving me the opportunity to receive so much education, for supporting me in everything I did during the last twenty four years and for always being there when I need something. Mum and Dad, thank you very much, without you I would not be where I am.

Contents

1 Introduction **9**
- 1.1 Motivation . . . 9
- 1.2 Contributions of this Thesis . . . 10
- 1.3 Thesis Outline . . . 11

2 Related Work **12**
- 2.1 Data Centric Security . . . 12
- 2.2 eXtensible Access Control Markup Language . . . 14
  - 2.2.1 XACML Architecture . . . 14
  - 2.2.2 XACML Policy Language . . . 15
- 2.3 Role Based Access Control . . . 17
  - 2.3.1 Core RBAC . . . 17
  - 2.3.2 Hierarchical RBAC . . . 18
- 2.4 Resource Description Framework . . . 19
- 2.5 Related Tools . . . 20
  - 2.5.1 IBM P3P Policy Editor . . . 20
  - 2.5.2 UMU XACML Editor . . . 20
- 2.6 Access Control Lists vs. Data Classification . . . 21
- 2.7 MAP vs. Data Centric Security . . . 22
- 2.8 User Interface Issues . . . 23
- 2.9 Intentional Access Management . . . 25
- 2.10 Policy Evaluation Issues . . . 25
- 2.11 Summary . . . 26

3 Access Control for Data Centric Security **27**
- 3.1 Introduction . . . 27
- 3.2 Business Level Policies . . . 28
  - 3.2.1 Reduction of Policy Quantity . . . 29
- 3.3 Basic Concepts . . . 29
  - 3.3.1 Resource . . . 29
    - 3.3.1.1 Resource Attributes . . . 30
  - 3.3.2 Classification Scheme . . . 30
  - 3.3.3 Subject . . . 31
  - 3.3.4 Role . . . 31
    - 3.3.4.1 Role Assignment. . . . 31
  - 3.3.5 Action . . . 32

3.4 Data Classification 32
3.4.1 Label Propagation 32
3.5 Policy Author Collaboration 33
3.5.1 Policy Delegation 33
3.6 Business Level Policy Language 34
3.6.1 Business Level Rule 34
3.6.1.1 Business Level Condition 35
3.6.2 Business Level Target 35
3.6.3 Business Level Policy 36
3.6.4 Business Level Policy Set 36
3.7 Matching Semantics 37
3.7.1 Subject Match 37
3.7.2 Action Match 37
3.7.3 Resource Match 37
3.8 Hierarchy Semantics 38
3.8.1 Role Hierarchy Semantics 38
3.8.2 Resource Hierarchy Semantics 38
3.8.3 Label Hierarchy Semantics 39
3.9 Policy Application 39
3.9.1 Business Level Rule Application 39
3.9.2 Business Level Policy Application 39
3.10 Scenarios 40
3.10.1 Common Roles, Subjects and Labels 40
3.10.2 Physical Access Control 40
3.10.3 Travel Expense Data 43
3.10.4 Backup Data 45
3.10.5 Discovering Issues in the Policies 45
3.10.6 Policy Delegation 46

4 Analysis Algorithms 47
4.1 Introduction 47
4.2 Business Level Decision Request 47
4.2.1 Decision Rendering on XACML Level 49
4.2.2 Use of Roles instead of Subjects 49
4.2.3 Use of Labels instead of Resources 50
4.2.4 Use of Roles and Labels 51
4.2.5 Policy Application 52
4.3 Contribution Analysis 52
4.4 Policy Override Detection 54
4.5 Authorization Analysis 55
4.5.1 Subject Sets 56
4.5.2 Role Sets 56
4.5.3 Action Sets 56
4.5.4 Resource Sets 57
4.5.5 Label Sets 57
4.6 Coverage Analysis 58

5 Implementation of Analysis Algorithms 60
5.1 Policy Translation . . . 60
5.1.1 Label Concept . . . 61
5.1.2 Policy Delegation Concept . . . 62
5.1.3 Role Concept . . . 63
5.1.4 Conditions . . . 64
5.1.5 XACML Policy File Size . . . 64
5.1.6 Example Rule . . . 64
5.2 Formulation of Business Level Decision Request . . . 66
5.2.1 Example Request . . . 67
5.3 Contribution Analysis . . . 68
5.4 Policy Override Detection . . . 69
5.5 Authorization Analysis . . . 70

6 Policy Editor and Analyzer 71
6.1 Design . . . 71
6.1.1 Eclipse Rich Client Platform . . . 71
6.1.2 RDF Store . . . 73
6.1.3 Jena - Semantic Web Framework for Java . . . 73
6.1.4 IBM XACML Light Library . . . 76
6.1.5 Sun XACML Implementation . . . 76
6.1.6 Assumptions . . . 76
6.2 Perspectives . . . 77
6.3 Policy Authoring and Review . . . 77
6.3.1 Author Authentication . . . 79
6.3.2 Policy Authoring . . . 79
6.3.3 Rule Authoring . . . 81
6.3.4 Policy Review . . . 82
6.3.4.1 Policy Override Detection . . . 85
6.4 Domain Browsing . . . 87
6.5 Data Classification . . . 88
6.6 Policy Simulation . . . 91
6.6.1 Contribution Analysis . . . 92
6.7 Authorization Analysis . . . 93

7 Conclusions and Future Work 96
7.1 Conclusions . . . 96
7.2 Future Work . . . 97

A Example XACML Policy 99

Bibliography 107

List of Figures

2.1 The Data Centric Security Model [13] . . . 13
2.2 XACML Architecture . . . 14
2.3 XACML Policy Language Model [10] . . . 16
2.4 Hierarchical RBAC [1] . . . 18
2.5 RDF Graph of a Single RDF Statement [20] . . . 19
2.6 IBM P3P Policy Editor . . . 21
2.7 UMU XACML Editor . . . 22

3.1 Abstract Level Elevation . . . 28
3.2 Concrete Level Elevation . . . 29
3.3 Business Level Policy Language Model . . . 35
3.4 Organization Hierarchy . . . 46

4.1 Decision Request Evaluation . . . 48

6.1 Policy Editor Components . . . 72
6.2 Author Authentication . . . 79
6.3 Policy Authoring and Review Perspective . . . 80
6.4 Business Level Policy Editor . . . 81
6.5 Business Level Rule Editor . . . 82
6.6 Subject Dialog . . . 83
6.7 Condition Selection . . . 83
6.8 Grouping in Policy Explorer View . . . 84
6.9 Rule Outline . . . 85
6.10 Policy Override Detection . . . 86
6.11 Domain Browsing Perspective . . . 88
6.12 Data Classification Perspective . . . 89
6.13 Data Classification Outline Views . . . 90
6.14 Policy Simulation Perspective . . . 92
6.15 Contribution Analysis View . . . 93
6.16 Authorization Analysis Perspective . . . 94
6.17 Authorization Analysis Result View . . . 95

List of Tables

3.1 Common Role Scheme: JobType 40
3.2 Common Role Scheme: JobDuration 41
3.3 Common Role Scheme: WorkComunity 41
3.4 Common Role Scheme: SpecialPermissions 41
3.5 Common Subjects 41
3.6 Common Classification Scheme: RoomType 42
3.7 Common Classification Scheme: DataOrigin 42
3.8 Physical Access Control: Resources 43
3.9 Physical Access Control: Business Level Rules 44
3.10 Travel Expenses: Resources 44
3.11 Travel Expenses: Business Level Rules 44
3.12 Backup Data: Resources 45
3.13 Backup Data: Business Level Rules 45
3.14 Final Policy by Administration Office of Canton Zurich 46
3.15 Recommended Policy by IBM Switzerland Headquarters 46

6.1 DCSM Ontology: RDFS Classes 74
6.2 DCSM Ontology: RDF Properties 75
6.3 Perspectives and the corresponding Workbench Parts. An (i) entry denotes that the workbench part is available *initially* in the corresponding perspective whereby a (d) entry denotes that the part is shown on *demand*. 78

Chapter 1

Introduction

This thesis describes the specification and implementation of an access control policy editor and analyzer for policies on a business level. The goal of this policy editor is to enable non-technical decision makers to author and analyze access control policies, until now a domain reserved to security experts. Therefore the policies are not any more formulated on a — low — system level, but on a — higher — business level. The idea to build such a policy editor came up within the *Data Centric Security* project at the IBM Zurich Research Laboratory.

1.1 Motivation

Access control is extensively used as security technology to prevent unauthorized access to information and system resources in accordance with a policy. Different models have been developed and studied throughout the last years to construct and manage access control systems. The vocabulary that is used by all these existing access control models to formulate the corresponding policies is hard to understand for non-technical policy authors. In February 2005, the Organization for the Advancement of Structured Information Standards (OASIS) approved the eXtensible Access Control Markup Language (XACML) 2.0 [10]. XACML is amongst others a standard language to express access control policies. Nowadays more and more companies and institutions follow this standard to formulate their access control policies, but still security experts are needed to formulate these standardized policies. This is because the XACML policies are rich XML-based documents with complex syntax and it is very difficult and time consuming to write error free XML documents by hand, especially for people who do not know the syntax well. To support the security experts in authoring their policies, different policy editors may be used to speed up and facilitate the authoring process and avoid errors in the policy syntax. However, the policy author must be very familiar with the concepts of the underlying policy language in order to use the policy editor.

Therefore, on the one hand this thesis provides a policy language that may not only be used by security experts but also by authors that are familiar with the business concepts of the domain for that the access shall be controlled. In

this new language the policies are formulated around the data — that is contained in the protected resources — itself and its meaning to the business. The business meaning is introduced by formalizing business concepts and classifying the resources according to these concepts. The new policy language elevates the policies from the technical level to the business level and makes it therefore understandable for non-technical policy authors. The level elevation is achieved by allowing subjects to be addresses in terms of roles, and resources in terms of labels that capture their business meaning. On top of the policy language, the thesis defines different analysis algorithms that may be performed by the policy authors in order to answer a number of questions that arise during the authoring process and to simulate the evaluation of the policies in an XACML environment. Because policy authoring is not a task that is performed by a single person or a single institution within an organization, furthermore a policy delegation mechanism is proposed that allows to delegate or override specific access control behavior within this organization. In order to deploy the new policy language in existing IT infrastructure without making any changes to the infrastructure, a translation from policies on the business level to XACML is defined later on. This allows the formulation of policies on a business level while the enforcement can take place in any domain that supports the XACML standard.

On the other hand, a user friendly prototype of an policy editor and analyzer is presented in this thesis, which puts all the theoretical work of the thesis into practice. This tool enables non-technical policy authors to formulate policies in the new policy language and to analyze them according to the specified analysis algorithms. Since the policy authoring tool is primarily built to be used by non-technical authors instead of security experts that have high technical knowledge, significant emphasis was put on the usability aspect of the policy editor. It is intended to give business users, who have no technical skills but high-level knowledge of the business domain instead, an appealing tool to express access control policies while using concepts they are familiar with. The analysis algorithms that have been integrated in the tool support the user in better understanding the current state of the policy repository as well as the effects that result from the delegation of access control behavior. Moreover the implementation of the policy editor and analyzer is intended to show that the business level policies are based on sound and easily understandable concepts.

1.2 Contributions of this Thesis

The major contributions of this thesis are the following:

- Using a labeling concept — whereby business meaning is attached to resources that shall be protected — together with a role concept to express access control behavior on a business level.

- Creating a policy language model to formulate policies on the business level.

- Introducing a policy delegation mechanism that allows policy authors to formulate their policies collaboratively.
- Specifying and implementing analysis algorithms for this new policy language model to answer questions that arise during an authoring process and to simulate an evaluation process of the policies in an XACML environment.
- Defining and implementing a translation between policies on a business level and standardized XACML policies.
- Building a user friendly prototype that supports the authoring and analysis of access control policies on the business level.

1.3 Thesis Outline

This thesis is organized as follows:

Chapter 2 reviews the background of the policy editor and analyzer by introducing IBM's Data Centric Security project, the eXtensible Access control Markup Language as well as Role Based Access Control and the Resource Description Framework. Moreover, it summarizes the research findings that that have been made in the domains that are related to the development of the policy editor and analyzer. It reviews existing ideas and concepts related to data classification and discusses design principles and guidelines for the building of user interfaces for systems in the security domain.

Chapter 3 specifies the underlying concepts of policies on a business level. It proposes a language to formulate the policies on the business level and defines appropriate semantics to interpret them.

Chapter 4 introduces and formalizes a number of analysis algorithms that can be performed on a repository of policies on a business level in order to answer questions that arise during the authoring process, to see the effects of policy delegation and to simulate the evaluation of the policies in an XACML environment.

Chapter 5 describes how the analysis algorithms introduced in chapter 5 can be implemented. Moreover, it discusses how a translation from policies on a business level to a standardized XACML policy can be done.

Chapter 6 presents the implementation of the actual policy editor and analyzer. It describes the solution components that have been used to implement the policy editor and it provides detailed insight into the developed user interfaces.

Chapter 7 gives the conclusions and discusses the future work.

Chapter 2

Related Work

This chapter provides general background information about underlying concepts for the policy editor and analyzer that is proposed in this thesis, it introduces Data Centric Security which is the base of the main ideas that are proposed in this thesis, and it contains a summary of tools and research findings that are related to the policy editor.

At first Data Centric Security, XACML, RBAC and RDF are described. Then two policy editors and their capabilities as well as a a policy analysis and verification tool are discussed. The next section discusses why it is better to use the concept of data classification to control access than to use access control lists. Afterwards a project called MAP that uses a labeling concept is compared to Data Centric Security. Later on a number of articles are discussed whose authors propose concepts of access control where namespaces are no longer a restrictive factor. At last some ideas related to user interfaces for systems in the security domain are discussed, which are relevant for the implementation of the policy editor.

2.1 Data Centric Security

Today adapting the IT infrastructure to become compliant with changing regulations and guidelines is slow and costly. This is because it is hard for enterprises to correctly configure security mechanisms such as access control that governs who can access what data. Traditionally, writing security policies requires a lot of knowledge of technical details, therefore only security experts are able to write policies. However, as each security policy incurs the business certain costs, decision about security and the right level of protection of data assets are increasingly being made at the executive level.

With *Data Centric Security* (DCS) a new approach of formulating security policies was developed by IBM. This approach introduces high-level business concepts into the policies which enables non-technical decision makers to understand and formulate such policies on a business level. Figure 2.1 shows these concepts as well as a general overview of DCS. The main issue that is new in DCS and that makes policy authoring easier is *data classification*. Through data classification business meaning is attached to data assets in the form of meta-

data, thus the data is elevated to the level of business processes. For example, metadata that states that the assets contain *travel expense data* or *backup data* may be attached. Policy authors can then refer in their policies to the travel expense or backup label and therefore govern the access for assets that are classified by these labels. Together with an appropriate role assignment, it is possible to formulate business level policies on a base of corporate security guidelines and law regulations. For the formulation of the policies, the vocabulary that was introduced for data classification and role assignment is used to describe the intended regulations at a business level. These business level policies can not only state access control but also data retention behavior as well as a specific data lifecycle etc. In order to *enforce* the policies within the IT infrastructure an automatic process derives low-level security policies and deploys them in the infrastructure. Becoming compliant to changing regulations and guidelines is now only a matter of changing the business level policies.

This thesis pays special attention to policies that state access control behavior in Data Centric Security. Amongst others it points out how to formulate policies on a business level and how to translate them into a representation that can be used in existing IT infrastructure. Note that the question how to enforce these policies is not an issue addressed by this thesis.

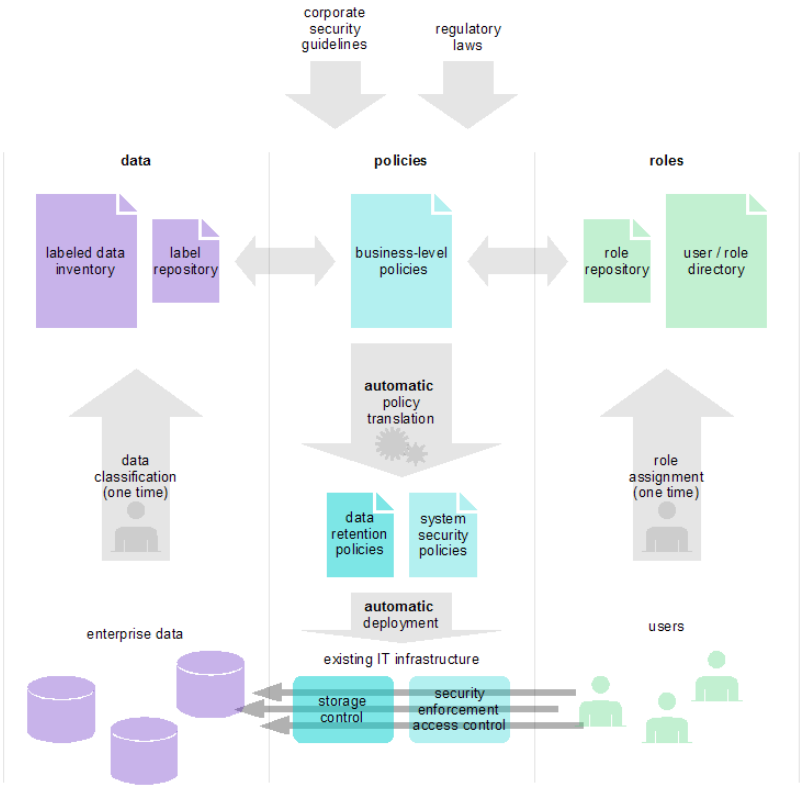


Figure 2.1: The Data Centric Security Model [13]

2.2 eXtensible Access Control Markup Language

In 2005 OASIS[1] published the eXtensible Access Control Markup Language (XACML) [10] that defines a standard language to express access control policies as well as a standard language for expressing queries over these policies. Both languages are formulated in XML[2] and are based on XML schema. The policy language is used to describe general access control requirements for protected application resources and has standard extension points for defining new functions, data types, etc. The query language lets one ask whether a given action should be allowed for a resource or not, and therefore renders authorization decisions. In case a policy is found for the queried resource, given attributes in the request are compared against the attributes contained in the policy. Finally a response is issued containing an answer about the given request where four possible return values are possible: Permit, Deny, Indeterminate (an error occurred or some required value was missing, so a decision cannot be made) or NotApplicable (there is no policy that states access control behavior for this request).

2.2.1 XACML Architecture

A typical XACML architecture is shown in Figure 2.2.

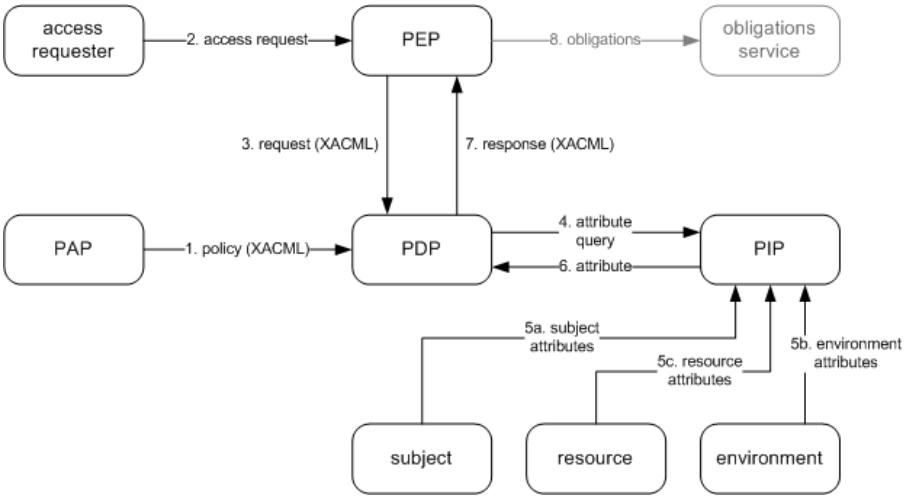


Figure 2.2: XACML Architecture

(1) The Policy Administration Point (PAP) writes a policy in XACML format and makes it available to the Policy Decision Point (PDP).

(2) The access requester sends a request for access in its native request format to the Policy Enforcement Point (PEP).

[1] Organization for the Advancement of Structured Information Standards
[2] eXtensible Markup Language

(3) The PEP creates a request for access in XACML format, optionally including attributes of the subject, action, resource and environment.

(4) (5) (6) The PDP queries the Policy Information Point (PIP) service to retrieve attributes of the subject, action, resource and environment that were *not* included in the request.

(7) The PDP compares the available attributes against attributes in the policy and determines an answer about whether access should be permitted or not. That answer is returned as response in XACML format to the PEP.

(8) Optionally the PEP fulfills obligations that are bound to the responded decision.

2.2.2 XACML Policy Language

As mentioned previously and shown in Figure 2.2, XACML defines how a request, a response and the policies for a PDP are formulated. The components of the XACML policy language are shown in Figure 2.3. The main components are *Rule*, *Policy* and *PolicySet*.

A *PolicySet* comprises the following components:

a) A set of Policies and/or PolicySets or references to them.

b) A Target to which this PolicySet is intended to apply. The Target defines which characteristics a subject, action, resource or environment must have that the PolicySet applies. For example, it can be stated that the PolicySet applies to subjects that have an email address within the zurich.ibm.com domain.

c) A Policy Combining Algorithm, which determines how the individual decisions of the contained Policies and/or PolicySets are reconciled into a single decision. Examples of Policy Combining Algorithms include *policy deny overrides*, *policy permit overrides*, *policy first applicable* and *policy only-one applicable*.

d) A set of optional obligations that must be fulfilled by the PEP in conjunction with the enforcement of the authorization decision.

A *Policy* comprises the following components:

a) A set of Rules.

b) A Target to which this Policy is intended to apply.

c) A Rule Combining Algorithm, which determines how the individual decisions of the contained Rules are reconciled into a single decision. Examples of Rule Combining Algorithms include *rule deny overrides*, *rule permit overrides* and *rule first applicable*.

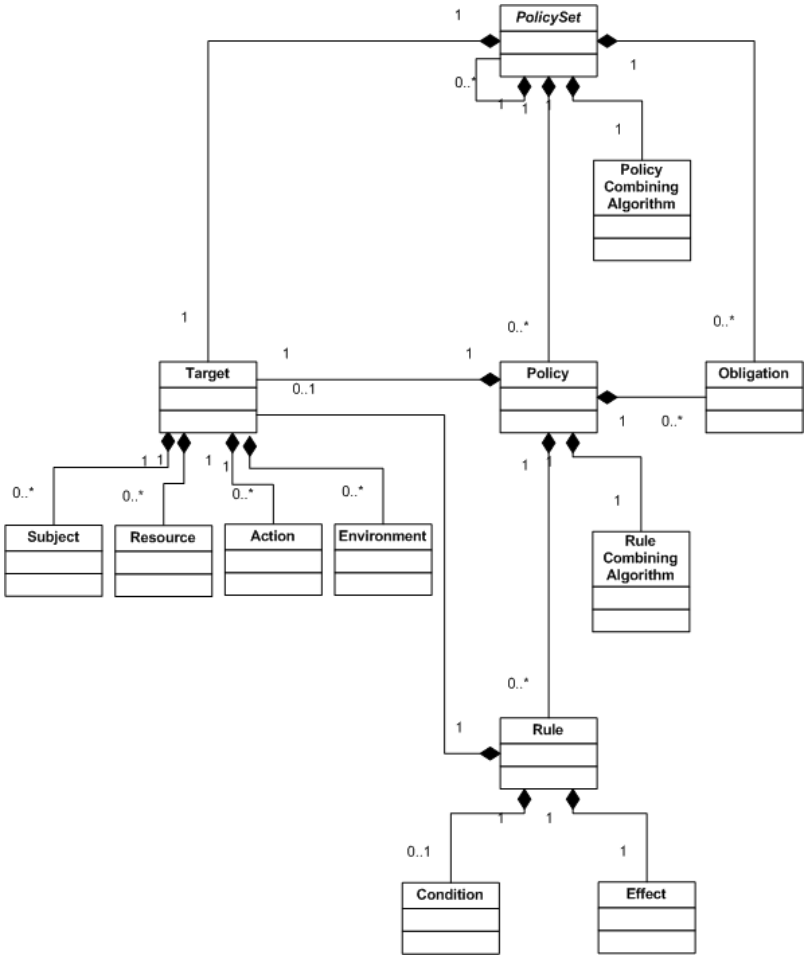


Figure 2.3: XACML Policy Language Model [10]

d) A set of optional Obligations that must be fulfilled by the PEP in conjunction with the enforcement of the authorization decision.

A *Rule* comprises the following components:

a) A Target to which this Rule is intended to apply.

b) An optional Condition that is a Boolean expression used to evaluate attributes of subjects, actions, resources and environment.

c) An Effect which defines the access decision of this rule. The possible values are "Permit" or "Deny".

To get an impression how an XACML policy looks like, see Section 5.1 on page 60 to find some example policies.

2.3 Role Based Access Control

Role Based Access Control (RBAC) is an approach to restrict system access to authorized users. It is a newer alternative approach to mandatory access control (MAC) and discretionary access control (DAC). MAC refers to means of restricting access to objects based on the sensitivity of the information contained in the objects and the formal authorization of subjects to access information of such sensitivity. DAC refers to means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission on to any other subject.

In Data Centric Security (see Section 2.1), RBAC serves as the basis to formulate policies on a business level whereby this basis is extended by the data classification concept to achieve the intended abstraction from the system level to the business level.

The RBAC model was first proposed in 1992 [6]. Since then it has been widely discussed and further developed [23]. In 2001, NIST[3] proposed a consensus model for RBAC which after further refinement has been adopted by the American National Institute(ANSI), International Committee for Information Technology Standards(ANSI/INCITS) as ANSI INCITS 359-2004 [1].

In the ANSI INCITS 359-2004 standard, the RBAC model is defined in terms of four model components: Core RBAC, Hierarchical RBAC, Static Separation of Duty Relations and Dynamic Separation of Duty Relations. The important part for my thesis are Core RBAC as well as Hierarchical RBAC which are described in the following.

2.3.1 Core RBAC

Core RBAC embodies the essential aspects of RBAC. The basic concept of RBAC is that users are assigned to roles, permissions are assigned to roles, and users acquire permissions by being members of roles. The same user can be

[3]National Institute of Standards and Technology

assigned to many roles and a single role can have many users. Similarly, for permissions, a single permission can be assigned to many roles and a single role can be assigned to many permissions. Core RBAC also includes the concept of user sessions, which allows selective activation and deactivation of roles.

A *user* is defined as a human being (or autonomous agent). A *role* is a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role. *Permission* is an approval to perform an operation on one or more RBAC protected objects. An *operation* is an execution action performed by a user. When a user logs on to a system, she establishes a *session* during which the user activates a subset of roles that are assigned to him [1].

2.3.2 Hierarchical RBAC

Hierarchical RBAC (see Figure 2.4) adds requirements for supporting role hierarchies. A hierarchy is mathematically a partial order defining a seniority relation between roles, whereby senior roles acquire the permissions of their juniors, and junior roles acquire the user membership of their seniors. The standard recognizes two types of role hierarchies [1]:

- *General Hierarchical RBAC.* General role hierarchies support the concept of multiple inheritance, which provides the ability to inherit permission from two or more role sources and to inherit user membership from two or more role sources.
- *Limited Hierarchical RBAC.* Roles in a limited role hierarchy are restricted to a single immediate descendant whereby the role hierarchy results in a simple "tree" structure.

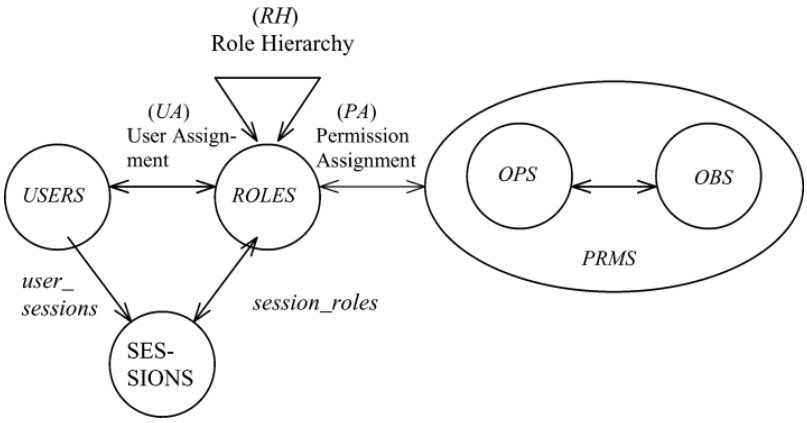


Figure 2.4: Hierarchical RBAC [1]

2.4 Resource Description Framework

The Resource Description Framework (RDF) [15] is a language for representing information about resources in the World Wide Web. It is particularly intended for representing metadata about Web resources, such as the title, author, and modification date of a Web page, copyright and licensing information about a Web document, etc. However, by generalizing the concept of a "Web resource", RDF can also be used to represent information about objects that can be identified on the Web, even when they cannot be directly retrieved. Examples include information about items available from online shopping facilities (e.g., specifications, prices, and availability), or information about a movie in the local cinema (e.g., date, time, location) [17]. Before resources can be described with semantic metadata, an *ontology* has to be defined for the domain of discourse. An ontology formally describes the vocabulary that is used in the domain that it describes. For example, in the domain of an online record shop concepts such as "composer", "album" and "track" and properties such as "composed by", "has track" and "has title" have to be defined.

To formalize the metadata about resources, the RDF data model is based on *subject - predicate - object* triples, so called *RDF statements*. The subject is the entity the statement is about, the predicate (also called property) defines the kind of information that is expressed about the subject, and the object defines the value of the predicate. For example, the fact:

> "The CD with the item number "1234" has the title "Alanis Unplugged" [20]

can be expressed as RDF statement. This is shown in Figure 2.5 where the statement is represented as RDF Graph. RDF also provides an XML-based syntax called RDF/XML [2] for recording and exchanging these graphs.

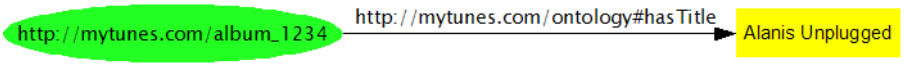


Figure 2.5: RDF Graph of a Single RDF Statement [20]

The big advantage of RDF is that it is domain independent in that no assumptions are made about the domain of discourse. The users are responsible for defining their own ontology in some ontology definition language (such as RDF Schema (RDFS) [3] or OWL[4] [18]). The vocabulary of an ontology is then used to describe resources. A resource is a "thing" we want to make a statement about. It can really be everything, for example, a movie, a person, a tree, an access control policy(!) and so on. Every resource is identified by a Uniform Resource Identifier (URI). This URI does not necessarily enable an access via the Web to this resource, it just has to unambiguously identify the resource.

For the policy editor and analyzer tool proposed in this thesis, a collection of RDF statements is used as data store to query and persist the data the policy

[4]Web Ontology Language

editor is operating on.

2.5 Related Tools

In the field of policy editors only few tools are existing. We describe a policy editor for P3P policies on the one hand and an XACML policy editor on the other hand.

A third related tool is a policy verification and change analysis tool called Margrave [7]. It represents an API that can be used in analyzing access control policies written in a subset of XACML. Margrave provides functions to answer queries about one policy or about the relationship between two policies.

2.5.1 IBM P3P Policy Editor

The Platform for Privacy Preferences Project (P3P) is a specification from the World Wide Web Consortium (W3C) that enables web sites to post their privacy practices in a machine-readable format. P3P user agents can inform users about a site's data collection practices and allow users to either accept or reject data transfer based on their own preferences.

IBM's P3P Policy Editor [14] is a tool for creating or updating such web site policies that are formulated in XML. For the user no knowledge of XML is required. The policies are created by a number of drag and drop operations and some additional changes in property windows. The policy editor generates an XML policy as well as a human readable version of the policy for review. Furthermore it includes a set of template policies to help site owners create a policy quickly.

The user interface of the editor is kept very simple. It mainly consists of three parts (see Figure 2.6):

- The available data elements to protect, structured in a tree,
- a list of the elements that were already chosen by the user, and
- a review area where the user can see his policy in XML and in human readable format.

The policy editor for P3P policies is in a sense similar to the policy editor that is created in this thesis in that both editors create an XML policy in the background through a users interaction on user interfaces. The idea behind the editors is the same, however, as P3P policies are very simple compared to XACML policies the requirements for a P3P editor are much simpler than on an XACML editor.

2.5.2 UMU XACML Editor

Another editor that is currently available is a policy editor that was developed at the University of Murcia (UMU). This tool is related to the editor proposed in this thesis since it allows users to create XACML policies, i.e., the final output

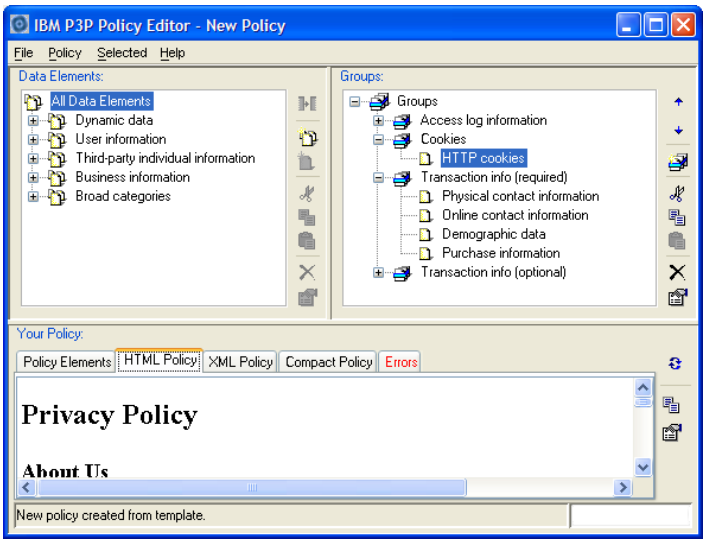


Figure 2.6: IBM P3P Policy Editor

format of both editors is the same. However, UMU's policy editor has neither support for policies on business level nor any analysis functionality. It supports a user in creating and reviewing plain XACML policies with the help of a user interface (see Figure 2.7). The creation of a policy with this editor helps a user to avoid spelling mistakes since the most identifiers for common attributes are predefined. An author who wants to formulate XACML policies with this editor needs very detailed knowledge of XACML.

2.6 Access Control Lists vs. Data Classification

An access control list (ACL) allows policy authors to create access control policies for individual resources by specifying lists of subjects who can access the resource (as well as the kinds of access that are permitted). However, ACLs fail to provide the mechanisms needed to categorize and group information objects throughout the namespace or to apply ranked levels of trust to subjects. These criteria are often the ones used to express security policies. In addition, ACLs do not have any generally accepted well-defined semantics. Every system is different, and the inconsistency makes using ACLs difficult when translating between different systems or working in a heterogeneous environment [25].

The concepts of data classification (also named data labeling) together with business level rules used in DCS meets these shortcomings of ACLs. Through the classification the resources are categorized according to user defined schemes that are later used in business level policies to specify access control behavior. Access is no longer controlled by the location of a resource but by its category. It does no longer matter where a resource is stored within the enterprise as long it is classified by the right label(s) [25].

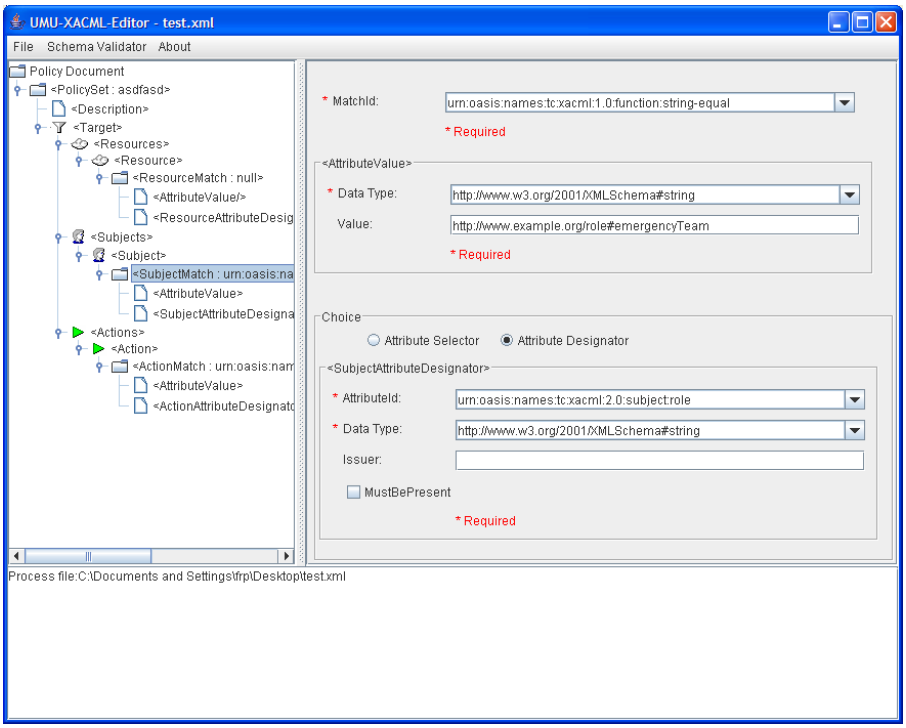


Figure 2.7: UMU XACML Editor

2.7 MAP vs. Data Centric Security

In [25] a project called MAP[5] is described in which a prototype was built that extended the existing authorization engine of the DCE[6] Web Server [21] with a label and rule mechanism.

In DCE users are arranged in groups. These user groups are then used in ACLs to make authorization decisions. However, no similar concept of arranging objects in groups is defined in DCE. MAP now used the DCE with the user group concept and extended it with a concept of labels which can be applied to objects and the ability to define rules that grant or deny access based on the relationship between the subjects's group and the object's label.

So far the user groups and the object labels look very similar to the concepts of Data Centric Security (while DCS uses user roles instead of user groups). But the big difference now is that MAP uses the existing user groups to label the objects whereas in DCS independent label schemes are used to label objects.

Allowing objects in MAP to be labeled and collected into groups fixes a basic asymmetry between how subjects and objects were treated in DCE. Labeling and grouping objects according to sensitivity or function is a natural thing to do, and relying on namespace partitions to accomplish this — by placing different kinds of objects in different places — is unnecessarily restrictive and difficult to

[5]Management of Authorization for site Policy
[6]Distributed Computing Environment

manage. Explicitly adding object groups and labels allows objects with similar labels to be protected in the same way, no matter where they are located.

Since data labeling is seen in [25] as a very powerful und intuitive method to control access on resources, it is also mentioned that the way the data labeling is applied in MAP has several shortcomings. A major problem of MAP was that rules were attached to namespaces. Thus it was not possible to specify exceptions to rules within a namespace branch, i.e., the granularity of protection was not fine enough. The authors of [25] mention that if the connection between rules and namespaces were broken, then finer grained control could be obtained more easily, for example, by having the scope of rules be governed by the labels on objects no matter where they were, rather than by their position in the namespace. Exactly these shortcomings are met in DCS. A fine grained granularity is possible because rules connect no longer roles and namespaces but roles and object labels. And the resources that are classified by these object labels may be located anywhere in the namespace.

Although the labeling concepts are different in DCS and in MAP, the method *how* the labels are applied to the objects was considered for DCS. In MAP, labels may be applied to single objects or to all objects in one or more branches of the object namespace. The branches covered by different labels may overlap so objects may have more than one label applied to them. The effective group membership of an object at MAP is the union of all the groups in all the labels applied to the object (with redundancies removed). This concept is similar to the one that was defined as *label propagation* for the DCS (see Section 3.4.1).

In [24] the authors tried to find ways of allowing administrators to specify authorization policy about targets without being tied to their namespace, but still allowing them to take advantage of the natural grouping in their namespace, by using features such as wildcarding. They said that users should think in terms of labels and collections and it should be possible to combine the advantages of using labels together with the advantages of namespaces.

2.8 User Interface Issues

In [22] a number of design principles are mentioned that shall be considered when creating secure systems. Among them are the following principles that have been considered while the policy editor and analyzer was built.

- *Economy of mechanism.* Keep the design as simple and small as possible.
- *Fail-safe defaults.* Base access decisions on permission rather than exclusion. The default situation is lack of access, and the protection scheme identifies conditions under which access is permitted. The alternative, in which mechanisms attempt to identify conditions under which access should be refused, presents the wrong psychological base for secure system design.
- *Psychological acceptability.* It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly.

– *Clarity.* The effect of any security-relevant action must be clearly apparent to the user before the action is taken.

In addition the authors of [22] mention the design philosophy of "affordances"[7], whereby the affordance of an object helps a potential user to determine how that object can be used, and they point out that mechanisms and models that are confusing to the user will be misused. Further it is underlined that clarity of effect is crucial to the implementation of a coherent security policy. This can only be done through a carefully thought-out GUI that facilitates rule, label and group (in DCS this means roles) definition and perception of the overall protection structure that is in force. The minimal requirements on this GUI according to the authors would be:

– Integrated management of rules, labels and groups.

– Convenient data entry of complex structures like rules.

– Consistency checking between current rules and one being entered.

– High-level overview of rules, labels and groups and their effects.

– Convenient querying of current policy constraints (how is a given object or group of objects protected?).

They further mention that user-centered security is as much about user interfaces as about security mechanisms. A coherent, consistent GUI is itself a security tool, not just window dressing.

A major usability problem in current systems is that applications unnecessarily export the underlying data structure as the user model. The user is given a rudimentary formatted display of the information in the data structure (or perhaps just a literal display of its values) and must learn the algorithm that the computer software will use to evaluate the data structure in order to understand what access control policy is actually instantiated [22].

In [4] is mentioned that "security experts are mainly blind to normal users (lack of) ability to understand security concepts and terminology" and that great care and early user testing is required in order to bridge the knowledge gap between the security experts and the normal users.

The authors of [25] mention that they did not explicitly include the notion of roles in their user interface, in order to keep the range of concepts to a minimum. An additional reason for that was that they already had a large number of defined terms for the user to understand.

They further mention that in MAP it was hard to know in which groups a given object was and what rules would be applied in any given decision. For DCS that means that it must be possible through a user interface to determine the labels a resource has associated and which rules contribute to a specific authorization decision.

In the system the authors of [24] propose, they allow the user to try various policy alternatives in a "debugging" mode before actually deploying a policy

[7] In German: "Aufforderungscharakter"

or policy change. Users can build and test latent policies off-line, and then activate them when appropriate. A query capability allows authorization rules to be tested.

In a usability test of a policy editor system in [24] the users indicated that they need a feature that would compare two entities (like rules or actors) and show the differences. This usability test also showed that users wondered how rules would combine and that they are concerned that an administrator could make a rule that would override an existing rule without knowing it.

A user study in [5] showed that confusion between stated and effective privileges is one of the main sources of error for end-users in manipulating ACLs. The Policy Editor for the DCSM shall mitigate such human errors by presenting the needed information including the effective permissions on the GUI to the user.

2.9 Intentional Access Management

In a paper about intentional access management [5] it is stated that before even considering user interface issues one should determine whether too much may be asked of the user in terms of sheer involvement with the process. The end user is primarily interested in the output of the ACL system and not in the means by which this output is achieved. They state that the end user is only interested in one of the following two goals that are called *primary user intentions*:

- G1: Principal X must have privilege Y on object Z.
- G2: Principal X must not have privilege Y on object Z.

Based on these two possible intentions they present an algorithmic approach on how to achieve the desired goals. This intentional approach is related to the *authorization analysis* algorithm that is later proposed in this thesis. Thereby a user can ask the system which values his business level rules must comprise in order to achieve an intended access control behavior.

2.10 Policy Evaluation Issues

The system that is specified in [4] denies everything except that what is specifically granted in the current authorization policy. The system that is specified in [24] denies access if any rule fails or if no rules are found that pertain to the decision. The WebDAV system that is described in [5] denies access if during the ACL evaluation a deny entry is encountered for a privilege which has not yet been granted.

Conform to the design principle of “fail safe defaults” and to the evaluation algorithms used in the three systems mentioned above, in the policy editor the XACML combining algorithm “Deny-Overrides” is used as default value wherever possible. In [24] is further mentioned that it is better to start with a simple too restrictive policy because a loose policy would not get tightened up. But the too restrictive will be loosened over time naturally.

2.11 Summary

Based on several papers data classification — or labeling — is in contrast to ACLs a very promising and intuitive concept for usable access control. Data classification makes fine grained granularity in access control possible because the connection between subjects and namespaces is broken.

At the design of the user interface for the policy editor, the consideration of the mentioned design principles was a good starting point. We where thereby aware of the (lack of) ability to understand security concepts and terminology and tried to keep the range of concepts that are presented to the user to a minimum.

The following list captures requirements that were stated from the referenced papers and finally implemented in the policy editor and analyzer that focuses on usability.

- Keep the number of concepts at a minimum.
- Convenient querying of current policy constraints.
- Check which labels are assigned to a resource.
- Check which roles are assigned to a subject.
- Check which rules contribute to a specific authorization decision.
- Present effective permissions to the user, not only the stated permission.

Chapter 3

Access Control for Data Centric Security

3.1 Introduction

Access is the ability to perform an action on a resource (e.g., read, delete). Access control is the means by which the ability is explicitly enabled or restricted in some way. Data Centric Security (see 2.1 on page 12) seeks to control access by allowing non-technical decision makers to understand and formulate access control policies on a business level. This chapter describes what the underlying concepts of these access control policies are, how the access control policies can be formulated and how they have to be interpreted.

In the following a new policy language model is proposed that was mainly inspired by the policy language model of XACML. The modifications that have been made to deduce the new model from XACML are the following:

- *Complexity removal.* The XACML model is very expressive and can become very complex. A non-technical policy author can not be expected to handle this complexity and expressivity. Therefore we simplified the model for the user by removing the nesting of of PolicySets and by reducing the number of allowed combining algorithms and conditions.

- *Extension with business level.* To be able to express our policies at business level we extended the model in a way that we can address subjects also in terms of (hierarchic) roles and resources in terms of (hierarchic) labels.

The goal was to define a proper policy language model on a business level that can still be translated into the policy language model of XACML. A set of access control policies formulated on business level can then be translated into access control policies on XACML level. Therefore, in every domain where an XACML PDP is used to render an authorization decision, policies can be formulated on business level and then — after a translation process — enforced on XACML level.

3.2 Business Level Policies

When we are talking about business level policies, the term *business* itself does not tell anything about the appearance or formal construction of the policies. It just states that the policies are formulated on a level that is understandable by people that are familiar with the business domain of the policies, thus the policies are formulated on a high level of abstraction. In an ideal environment the policies would be formulated as sentences of natural language, which are then parsed and translated into policies on a lower, more technical, level.

Access control policies, regardless of which abstraction level, express the information *"who is permitted to access what data in what way"*. In access control policies that are formulated at a system level, terms like *subject*, *action* and *resource* are used to define access control, i.e., it is stated that subject s is permitted to perform action a on resource r. For example, it may be stated that Sue is permitted to perform a read action on the salary database. In order to express the policies at business level to support DCS, two concepts are used to increase the level of abstraction.

With DCS, for the data classification — also called data labeling — a classification scheme is used to classify the enterprise data, and within the role assignment an assignment scheme is used to define the responsibilities of the users. The terms contained in these two schemes are now used to achieve the intended abstraction. So, instead of addressing subjects directly, one may use *roles*, and instead of addressing resources directly, one may use *labels*. This is illustrated in Figure 3.1.

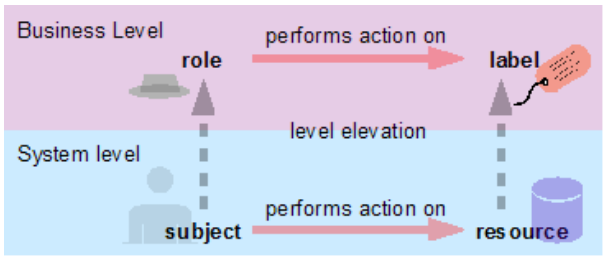


Figure 3.1: Abstract Level Elevation

With the use of the roles and labels, policy authors now get the opportunity to express the policies at a business level. For example, it can be expressed that an employee from the human resources (HR) department may access data that is HR related, i.e., *"HR role is permitted to perform action A on HR data"*. Therefore in a previous data classification the HR data has to be labeled accordingly as well as an HR role has to be assigned to the HR employees (see Figure 3.2). The flexibility of security policies on a business level lies in the possibility to associate any subject with the intended roles and to label any resource with the intended labels.

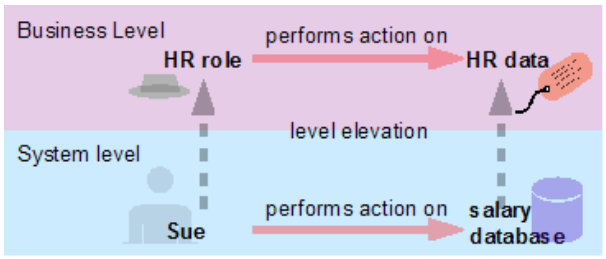


Figure 3.2: Concrete Level Elevation

3.2.1 Reduction of Policy Quantity

Using a formulation of policies at a business level, the easier understanding due to the higher level of abstraction is not the only benefit one can achieve. Another important outcome of the level elevation is that this enables a significant reduction of the number of policies. This reduction can be obtained through a structuring of the used roles and labels in a hierarchical way:

- An inheritance of role privileges along the role hierarchy as defined in RBAC establishes the opportunity to cover lower role levels with a policy targeting a higher role level.
- A hierarchical structure of labels allows the coverage of labels on lower levels with a policy targeting a label on a higher (hierarchical) level.

The possibility to retain the labels of resources throughout a copy process of this resource is another reason for the quantity reduction of the policies. Imagine a resource that is classified by a "Backup" label during a backup process. For redundancy reasons this copied version of the resource is again copied to different machines. If during this copy process the labels are retained, then a policy that addresses the "Backup" label addresses all copies of the resource and not just the original one.

3.3 Basic Concepts

Before describing a language for an access control policy on business level, the concepts that are used in this language must be defined precisely. This section provides definitions for resources, classification schemes, subjects, roles and actions.

3.3.1 Resource

An object for which access shall be controlled is referred to as a *resource* (Note: Do not confuse with resources used in RDF). Resources may be organized in a hierarchy. A set of resources organized in a hierarchy is a tree, i.e., a hierarchy with a single root that may not have cycles. Each resource in the hierarchy has zero or more child resources. A resource that has a child is called the child's

parent resource. A resource has at most one parent. All direct or indirect children of a resource are called the *descendants* of the resource. All direct or indirect parents of a resource are called the *ancestors* of the resource.

An XML document for example is structured as a tree. Other types of hierarchical resources may be structured in a way where a resource may have multiple parents. We do not consider such structures here, i.e., only tree structures are used to describe resources. If access shall be controlled for a hierarchy with multiple parents, this structure has to be transformed to a less expressive tree structure first.

A consistent representation for the identity of the resources is used. This is done by including the resource's position in the hierarchy as part of the resource's identity. Thus, the identity of a resource in a hierarchy depends on the position of the resource in the hierarchy.

3.3.1.1 Resource Attributes

Authorization decisions may be based on characteristic of the resource other than its identity. This approach is supported by means of *resource attributes*. Although a resource may have an arbitrary number of attributes, for DCS especially an *owner* attribute is important. Thereby the value of the owner attribute refers to the subject that owns the resource. Every resource is allowed to have at most one owner attribute (whereby this constraint is just true for the owner attribute but not for other resource attributes).

3.3.2 Classification Scheme

A *classification scheme* is an arrangement or division of objects into groups based on characteristics that the objects have in common. Classification schemes are used as a way to classify administered items, such as resources, in a metadata registry. The names of the groups that characterize the objects are referred to as *labels*. Data can be classified according to any criteria, not only access control or security related ones.

The labels in a classification scheme may be organized in a hierarchy. A classification scheme organized in a hierarchy is a tree, i.e., a hierarchy with a single root that may not have cycles. Each label in the hierarchy has zero or more child labels. A label that has a child is called the child's parent label. A label has at most one parent. All direct or indirect children of a label are called the *descendants* of the label. All direct or indirect parents of a label are called the *ancestors* of the label. A classification scheme has an associated *scheme name* and may contain labels — which may be organized in a hierarchy — as well as other classification schemes, i.e., a classification scheme may be part of another classification scheme.

Every label has a *label name*. The label name is an arbitrary string associated with the label that is intended to indicate the semantics of the label. For example, imagine a label with the name "Zurich". Since labels may be structured in a hierarchy, the label name alone is not expressive enough to identify a label in the label repository unambiguously. Therefore a label has a *label*

identity that is a concatenation of all the label names of the ancestors together with label's name. For example, the identity of a "Zurich" label could be "Europe>Switzerland>Zurich". Within the same hierarchy level of a classification scheme the label names have to be unique, i.e., it is not allowed that two European countries with the name Switzerland exist, but it might be the case the in another European country a city with the name Zurich exists. In business level policies always the label identity is used to reference a label.

3.3.3 Subject

A *subject* represents an actor who may request access to a resource. A subject may be assigned to several roles, depending on the job function of that subject.

3.3.4 Role

A *role* represents a job function that has — according to the access control policy — associated a collection of access privileges. At the same time a role represents — according to a role assignment — a collection of subjects. The role serves as an intermediary to bring these two collections together, i.e., which subject has which access privileges.

Roles may be organized in a hierarchy. The role hierarchy is defined according to *Limited Hierarchical RBAC* (see 2.3.2 on page 18). Thereby the hierarchy defines a seniority relation between roles, whereby senior roles acquire the access privileges of their juniors, and junior roles acquire the subject membership of their seniors. A set of roles organized in a hierarchy is a "tree", i.e., a hierarchy with a single root that may not have cycles. Each role in the hierarchy has zero or more senior roles. A role has at most one junior role. All direct or indirect seniors of a role are called the *descendants* of the role. All direct or indirect juniors of a role are called the *ancestors* of the role.

Roles are divided into *role schemes*. A role scheme has an associated *schema name* and may contain roles — which may be organized in a hierarchy — as well as other role schemes, i.e., a role scheme may be part of another role scheme.

In XACML, roles are implemented by means of individual characteristics of subjects, which are called subject attributes. A subject may have an arbitrary number of attributes at the same time, and therefore an arbitrary number of roles.

A role that was assigned to a subject by a role assignment (see 3.3.4.1) is referred to as *explicit role* of the subject. The roles that a subject has due to the fact that junior rules acquire the subject membership of their seniors are referred to as *implicit roles*. For example, think of a role "EmergencyTeam" that has a senior role, i.e., a child role, "FireDepartement". If the role "FireDepartement" is now — explicitly — assigned to the subject John, then John has an implicit role assignment of the role "EmergencyTeam".

3.3.4.1 Role Assignment.

The process by which a role is assigned to a subject is referred to as *role assignment*, i.e., the process by which a subject gets the role assigned. At the same

time a *role assignment* may refer to a repository that contains the information which roles are assigned to which subjects.

3.3.5 Action

An operation performed on a resource is referred to as an *action*. The actions that may be performed are not predefined, i.e., appropriate actions for the particular target domain may be chosen freely. Each of the chosen actions refers to one specific operation that may be performed on a resource.

3.4 Data Classification

The process by which a label defined in a classification scheme is assigned to a resource is referred to as *data classification*, i.e., the process by which a resource is classified by a label according to a classification scheme. At the same time a *data classification* may refer to a repository that contains the information which labels are assigned to which resources. A label that is associated to a resource acts as a tag, which assigns particular meaning to the resource. Each node of a classification scheme, no matter if organized in a hierarchy or not, may be used to classify a resource. Multiple classification schemes may be used to classify a resource, i.e., a resource may have labels out of different classification schemes associated at the same time. The labels that classify a resource are referred to as *explicit labels*.

A label hierarchy defines a specialization relation between the labels, i.e., a child label is more specific than its parent label, and a parent label is more general than its child labels. Hence, for a resource that is explicitly classified by a label exist implicit classifications by the ancestors of that label. The labels that classify a resource due to an implicit classification are referred to as *implicit labels*. For example, think of a label “Europe” that has a child label “Zurich”. If a resource is now — explicitly — classified by the label “Zurich”, then this resource has an implicit classification by the label “Europe”.

3.4.1 Label Propagation

If a resource that is classified by a label is a node in a hierarchy, the label may also be assigned to the descendants of that resource. This mechanism is referred to as *label propagation*. Label propagation is mainly motivated by the containment relation that is true for most hierarchically structured resources. For example, at databases where a database scheme contains tables and a table contains columns, this containment relation is obvious. In the file system domain different points of view exist though where the the containment relation is not always adopted.

A resource’s label that is inherited through label propagation is referred to as *inherited label*. For example, think of a resource “EmployeeTable” that has a child resource “SurnameColumn”. If the “EmployeeTable” is now — explicitly — classified by the label “PersonalData” using label propagation, then the resource “SurnameColumn” inherits the label “PersonalData”.

As for an explicit label, also for an inherited label exists an implicit classification by the ancestors of that label, i.e., it does not matter if a resource's label was inherited by label propagation or not. Thus, a resource may have explicit, implicit and inherited labels associated at the same time.

Label propagation allows adapting the granularity of access control because at the beginning of the data classification a coarse grained classification may be done using label propagation. This coarse grained classification is then refined in further classification steps. This is done by removing the explicit classifications on the coarse grained level and classify the intended resources on a finer level.

3.5 Policy Author Collaboration

Policy authoring is not a task that is done by one specific facility. Policies may be issued by multiple policy authors, where each author is likely to formulate policies for a specific domain she is designated for. To meet the requirements of each policy author, all formulated policies have then to be considered at the same time. However, policy authors may reside in different levels of an organization that is structured in a hierarchy. Authors on higher levels in the hierarchy have more power of decision than authors on lower levels in the hierarchy. The authors on the higher levels want to be able to formulate policies that are either recommended or final. Authors formulate *final policies* if they know exactly which access control behavior is desired. Authors formulate *recommended policies* if they do not know exactly which access control behavior is desired on the one hand, but want to specify a "default" behavior for lower levels on the other hand. Therefore, recommended policies may be overridden by authors on lower levels, but final policies may not be overridden by authors on lower levels. The sequence in which the policies are considered to decide about the access control behavior is determined by the policy author's level in this hierarchy and by the type of the policy which is either final or recommended.

For example, an author that resides on a higher hierarchy level is concerned about the safety of financial data and formulates therefore a recommended policy that denies every employee the access to this kind of data. At the same time she intends institutions on lower hierarchy levels to allow later on the the access for authorized personnel.

3.5.1 Policy Delegation

An access control policy on a business level has a hierarchy level which determines the sequence in which the policies are considered. Different policies may have the same hierarchy level. The higher the hierarchy level of a policy, the earlier the policy is considered to decide about the access control behavior. If a policy does not specify a particular access control behavior for a given decision request, then the next lower hierarchy level is considered. If a policy specifies access control behavior that was already specified on a higher level, the behavior on the lower level overrides the behavior on the higher level, i.e., a policy author on a higher level may specify a recommended access control behavior that can be overridden by policy authors on lower levels.

A policy on a business level may be a final policy or a recommended policy. A final policy overrides all policies — no matter if final or not — that reside on a lower hierarchical level than the policy itself, i.e., a final policy can not be overridden. At each hierarchy level there may be zero or more final and zero or more recommended policies.

Since every policy has an associated hierarchy level, contradicting access control behavior between policies on different levels is always resolved by the delegation mechanism (whereby the higher level determines the behavior). Contradicting behavior between policies on the same hierarchy level is resolved by *combining algorithms* that are introduced in the next section.

3.6 Business Level Policy Language

The policy language model on business level is shown in Figure 3.3. This model was inspired by the XACML policy language model (see Figure 2.3 on page 16). The XACML model was simplified on the one hand in order to create a policy language that is easier to understand for non-technical policy authors, on the other hand it was extended in order to enable the policy delegation mechanism. Due to the simplification it is not as expressive as the XACML model. Our model is less expressive because policy sets can not be nested and only two possible kinds of conditions are allowed. The main components of the model are:

- Business Level Rule,
- Business Level Target,
- Business Level Policy, and
- Business Level Policy Set.

These components are described in detail in the following subsections.

3.6.1 Business Level Rule

An access control rule on business level, called *business level rule*, is the most elementary unit of policy. It defines authorization behavior for a set of subjects to perform an action on a set of resources. The set of entities to which the rule is intended to apply is called the business level target of the rule. The authorization behavior of a business level rule is specified through the effect of the rule, which may be:

- “Permit”, or
- “Deny”.

The effect indicates the intended consequence of a rule. If the consequence shall be to grant access then the effect “Permit” is used. Otherwise the effect “Deny” is used to indicate that access is not granted.

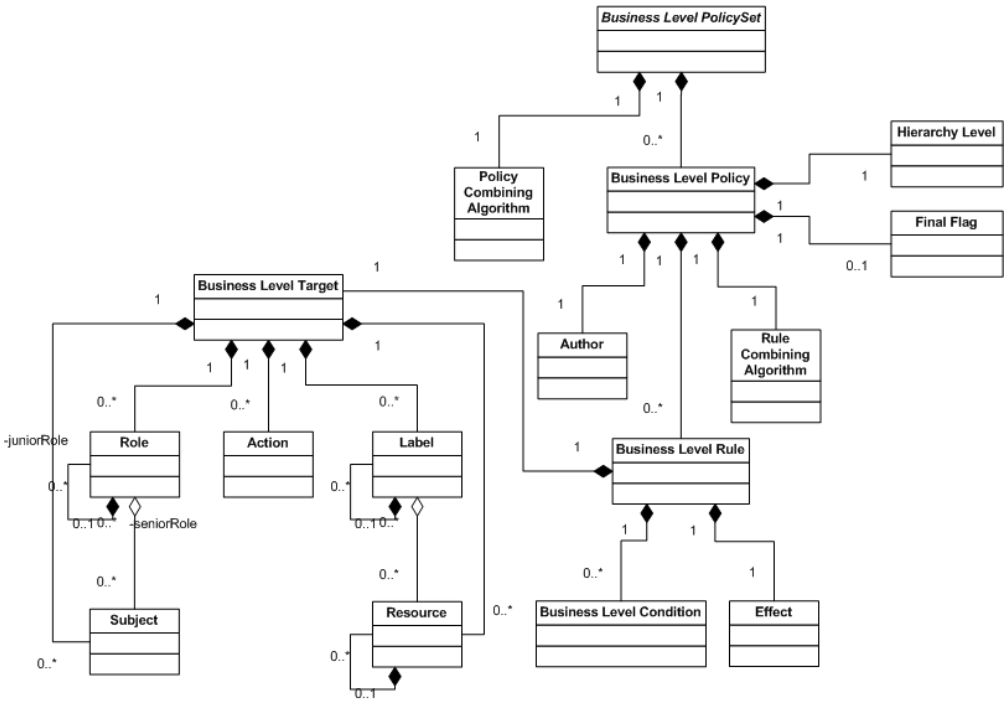


Figure 3.3: Business Level Policy Language Model

3.6.1.1 Business Level Condition

A set of business level conditions may refine the applicability established by a business level target. A *business level condition* represents a function that can evaluate to "True" or "False". Only if all associated conditions evaluate to "True" in conjunction, the particular business level rule is applicable. Two possible types of conditions may be used to specify if a business level rule is applicable or not:

– "Owner Condition", and

– "Time Range Condition".

The "Owner Condition" evaluates to "True" if the resource that shall be accessed has exactly one owner attribute and its value is equal to the subject that requests the access to the resource. The "Time Range Condition" evaluates to "True" if the decision request was issued between two specified points of time.

3.6.2 Business Level Target

A *business level target* specifies the set of subjects, actions and resources that are governed by the associated rule. The subjects can either be specified directly or in terms of roles, i.e., subjects and roles in a target are mutually exclusive. Similarly the resources can either be specified directly or in terms of labels, i.e., resources and labels in a target are mutually exclusive.

The mutual exclusivity of subjects and roles is defined because combinations of subjects have different matching semantics than combinations of roles (see Section 3.7). A non-technical policy author could get easily confused by the different semantics if they would be used within the same rule. The same reasoning is true for the mutual exclusivity of resources and labels.

3.6.3 Business Level Policy

Business level rules are combined in a *business level policy*. Such a policy comprises the following components:

- an author,
- a hierarchy level,
- an optional final flag,
- a set of business level rules, and
- a rule combining algorithm.

The hierarchy level and the final flag refer to the corresponding concepts of policy author collaboration (see Section 3.5 on page 33). The *rule combining algorithm* specifies the procedure by which the individual results of evaluating the contained rules are combined when evaluating the business level policy. The rule combining algorithms that may be used are:

- "Deny-overrides", and
- "Permit-overrides".

In case of the "Deny-overrides" algorithm, if a single rule is encountered that evaluates to "Deny", then, regardless of the evaluation result of the other applicable rules in the policy, the combined result is "Deny". Likewise, in case of the "Permit-overrides" algorithm, if a single "Permit" result is encountered, then the combined result is "Permit". The notion and semantics of the used combining algorithms are taken directly from [10].

3.6.4 Business Level Policy Set

A *business level policy set* is a container for business level policies. Throughout this thesis, only one business level policy set is used which comprises all business level policies that are created from any author. The individual policies are combined with a *policy combining algorithm*. The policy combining algorithms that may be used for business level policy sets are the same as for business level policies. The notion and semantics of the used combining algorithms are taken directly from [10].

3.7 Matching Semantics

As described in Section 3.6.2, a business level target specifies the subjects, actions and resources that are intended to be matched by the target whereby the subjects and resources can either be addressed directly or in terms of roles and labels respectively. In the cases where the subjects or resources are not addressed directly, a proper semantics has to be provided that states which subjects are matched by which roles and which resources are matched by which labels. This semantics is introduced in the following subsections. Such a matching semantics is necessary to make a translation from access control policies on business level to access control policies on XACML level possible.

3.7.1 Subject Match

Subjects may either be matched directly by a business level target or in terms of roles, i.e., a business level target may not use both terms of roles and subjects at the same time to address the subjects that are intended to be matched.

A business level target that uses roles to match subjects, is intended to match subjects that have *all the target's roles assigned in conjunction*, i.e., the subjects that have an appropriate role assignment for every role the target contains. For matching a subject, it does not matter whether the subject's roles are explicit or implicit.

A business level target that matches subjects directly, is intended to match subjects that are contained in the target's subjects. If a target contains more than one subject, the target matches *any of the target's subjects*.

If a business level target contains no role and no subject, i.e., if the roles and the subjects are omitted, the target matches *any subject*.

3.7.2 Action Match

A business level target is intended to match actions that are contained in the target's actions. If a target contains more than one action, the target matches *any action out of the target's actions*. If a target contains no action, i.e., if the actions are omitted, the target matches *any action*.

3.7.3 Resource Match

Resources may either be matched directly by a business level target or in terms of labels, i.e., a business level target may not use both terms of labels and resources at the same time to address the resources that are intended to be matched.

A business level target that uses labels to match resources, is intended to match resources that are classified by *all the target's labels in conjunction*, i.e., the resources that have an appropriate classification by every label the target contains. For matching a resource, it does not matter whether the labels of the resource are explicit, implicit or inherited.

A business level target that matches resources directly, is intended to match resources that are contained in the target's resources. If a target contains more than one resource, the target matches *any of the target's resources*.

If a business level target contains no label and no resource, i.e., if the labels and the resources are omitted, the target matches *any resource*.

3.8 Hierarchy Semantics

According to the basic concepts that have been defined previously, the roles, labels and resources of a business level target may be nodes in a hierarchically organized structure (see also Figure 3.3). This leads to different possible interpretations on how a policy for a particular node governs access for nodes that are ancestors or descendants of that node. In the following subsections, the semantics of the hierarchic elements are defined. Thereby the described semantics for the roles and labels derive from their definitions provided in the Sections 3.3.4 and 3.3.2, i.e., the corresponding subsections just make these concepts more clear.

3.8.1 Role Hierarchy Semantics

Senior roles acquire the access privileges of their juniors, and junior roles acquire the subject membership of their seniors (see Section 3.3.4 on page 31). A single policy for a particular role governs therefore also the access for all descendants of that role. For example, if a role "Admin" has a senior role "BackupAdmin" and a policy governs access for the "Admin" role (which is the junior role), then the policy also governs access for the "BackupAdmin" role since the "BackupAdmin" role is a descendant of the "Admin" role. However, a policy for the "BackupAdmin" role does not govern the access for the "Admin" role.

Being able to express a single policy constraint that applies to an entire path in a role hierarchy up to a particular role, rather than having to specify a separate constraint for each role, increases ease of use and reduces the amount of policies that have to be formulated (see 3.2.1 on page 29).

3.8.2 Resource Hierarchy Semantics

The nodes in a hierarchical resource are treated as individual resources, i.e., an authorization decision that permits access to an interior node in a resource hierarchy does not imply that access to its descendant nodes is permitted. At the same time an authorization decision that denies access to an interior node does not imply that access to its descendant nodes is denied.

Note that OASIS has published an XACML Profile for hierarchical resources [11] where is defined how to represent the identity of a node and how to request access to a node. However, this Profile is not considered here.

3.8.3 Label Hierarchy Semantics

An implicit classification by the ancestors of the label exists for resources that are classified by a label (see Section 3.4 on page 32). A single policy for a particular label governs therefore also the access for all descendants of that label. For example, if a label "Financial" has a child label "Salaries" and a policy governs access for the "Financial" label (which is the parent label), then the policy also governs access for the "Salaries" label since the "Salaries" label is a descendant of the "Financial" label. However, a policy for the "Salaries" label does not govern the access for the "Financial" label.

Being able to express a single policy constraint that applies to a label as well as the entire sub tree of the label in the label hierarchy, rather than having to specify a separate constraint for each label, increases ease of use and reduces the amount of policies that have to be formulated (see 3.2.1 on page 29).

In the same way as for hierarchical resources, the nodes in a hierarchical classification scheme are treated as individual labels. That means an authorization decision for a decision request on business level (see Section 4.2 on page 47) that governs access to resources labeled by an interior label does not imply that access to resources labeled with its descendant labels is governed.

3.9 Policy Application

To let an existing XACML policy decision point (PDP) perform an *authorization decision*, which states if a subject is permitted to perform an action on a resource or not, a *decision request* is issued that comprises three components:

- a *subject* that wants to perform an action on a resource,
- an *action* that is intended to be performed, and
- a *resource* on that the action is intended to be performed.

The set of business level policies and business level rules that govern access for such a decision request is referred to as the *applicable policy* of the decision request. The complete policy applicable to a decision request may be composed of a number of individual rules or policies. In the following is defined which policies and rules apply to a decision request.

3.9.1 Business Level Rule Application

A business level rule is applicable to a decision request if the rule's target matches the subject *and* the action *and* the resource of the decision request *and* the rule's business level condition is fulfilled.

3.9.2 Business Level Policy Application

A business level policy is applicable to a decision request if *at least one* business level rule that is contained in the policy is applicable.

Table 3.1: Common Role Scheme: JobType

- ▷ Employee
  - ▷ ComputerScience
    - ▷ Security&Assurance
    - ▷ Security&Cryptography
  - ▷ Science&Technology
    - ▷ PhysicsOfNanoscaleSystems
  - ▷ SiteOperations
    - ▷ InformationServices
    - ▷ Finance&Administration
      - ▷ FinanceAnalyst
      - ▷ FinancePayroll
      - ▷ Contolling
  - ▷ HumanResources

3.10 Scenarios

In the previous sections was specified on a theoretical level what the underlying concepts of access control policies on a business level are, how they can be formulated and how they have to be interpreted. This section is intended to show some examples inspired by real requirements in the IBM Zurich Research Laboratory (ZRL) to illustrate the practical aspect of business level policies. In Chapter 6, where the actual policy editor is presented, these scenarios are used again.

3.10.1 Common Roles, Subjects and Labels

This section shows the roles, the subjects together with associated role assignments as well as the labels — arranged in classification schemes — the scenarios have in common. The Tables 3.1, 3.2, 3.3 and 3.4 show the common roles arranged in role schemes.

Table 3.5 shows the common subjects as well as associated role assignments. The role assignments just state the *explicit* role assignments. The implicit role assignments are not shown here. In Chapter 6, the policy editor is used to show some of the implicit role assignments.

The Tables 3.6 and 3.7 show the common classification schemes that are used in the scenarios.

3.10.2 Physical Access Control

This scenario is inspired by the physical access control at the IBM Zurich Research Laboratory. To control the physical access in this scenario, the action

Table 3.2: Common Role Scheme: JobDuration

▷ Regular
▷ ResearchStaffMember
▷ Supplemental
▷ Contractor

Table 3.3: Common Role Scheme: WorkComunity

▷ OfficeCommunityC231
▷ OfficeCommunityIS

Table 3.4: Common Role Scheme: SpecialPermissions

▷ LaboratoryAccess
▷ EmergencyTeam

Table 3.5: Common Subjects

Subject	**Explicit roles**
Alice	Security&Assurance, Supplemental, OfficeCommunityC231
Bob	Security&Assurance, ResearchStaffMember, LaboratoryAccess
Carol	Security&Assurance, Contractor, OfficeCommunityC231
Dave	Security&Cryptography, ResearchStaffMember, LaboratoryAccess
Emily	PhysicsOfNanoscaleSystems, ResearchStaffMember
Francis	InformationServices, Regular, OfficeCommunityIS
George	FinancePayroll, Regular
Helen	Controlling, Regular
Isaac	HumanResources, Regular
John	EmergencyTeam

Table 3.6: Common Classification Scheme: RoomType

- ▷ Area
  - ▷ Office
  - ▷ ConferenceRoom
  - ▷ PrinterRoom
  - ▷ ConfidentialPrinterRoom
  - ▷ Lounge
  - ▷ RoomWithServer
  - ▷ Laboratory

Table 3.7: Common Classification Scheme: DataOrigin

- ▷ ComputerScience
  - ▷ Security&Assurance
  - ▷ Security&Cryptography
- ▷ Science&Technology
  - ▷ PhysicsOfNanoscaleSystems
- ▷ SiteOperations
  - ▷ InformationServices
    - ▷ Backup
  - ▷ Finance&Administration
    - ▷ TravelExpenses
- ▷ HumanResources

Table 3.8: Physical Access Control: Resources

Resource	**Owner**	**Explicit classifications**
▷ ZRL		
▷ BuildingC		Security&Ass., Security&Cryptography
▷ C201		ConferenceRoom
▷ C202		ConferenceRoom
▷ C230		PrinterRoom
▷ C231		Office, RoomWithServer
▷ C273		Office
▷ C247		Laboratory
▷ C350	Dave	Office
▷ C375	Bob	Office
▷ BlueLagoon		Lounge
▷ BuildingL		Science&Technology
▷ BuildingM		Science&Technology
▷ Lobby		Lounge
▷ Cafeteria		Lounge

"enter" is used. Table 3.8 shows the used resources and Table 3.9 shows the most interesting part of the scenario: the access control rules on business level.

3.10.3 Travel Expense Data

This scenario is inspired by the access control to travel expense data at the IBM ZRL. The following requirements have to be met:

a) Every financial employee may create travel expense entries in the database.

b) A financial analyst has to check the travel expenses in random intervals.

c) A financial payroll employee marks a travel expense as disbursed after disbursement.

d) A controller has to check disbursed travel expenses.

Therefore the actions that are used in this scenario are *"read"*, *"write"* and *"create"*. The *"TravelExpenses"* label is used to classify data. Table 3.10 shows the used resources and Table 3.11 shows the the access control rules on business level.

Table 3.9: Physical Access Control: Business Level Rules

[1]	Permit	Employee	enter	Area	ifOwner
[2]	Deny	Contractor	enter	Area	between 8pm and 6am
[3]	Permit	OfficeCommunityC231	enter	C231	
[4]	Permit	OfficeCommunityIS	enter	C273	
[5]	Permit	Employee	enter	ConferenceRoom	
[6]	Permit	Employee	enter	PrinterRoom	
[7]	Permit	Employee	enter	Lounge	
[8]	Permit	InformationServices	enter	RoomWithServer	
[9]	Permit	Security&Assurance, LaboratoryAccess	enter	Security&Assurance, Laboratory	
[10]	Permit	Security&Assurance, ResearchStaffMember	enter	Security&Assurance, ConfidentialPrinter-Room	
[11]	Permit	Science&Technology	enter	Science&Technology	

Table 3.10: Travel Expenses: Resources

Resource	**Owner**	**Explicit classifications**
▷ HostComputer1234		
▷ DB2DataBaseServer		
▷ DataBaseSchemaXY		
▷ TravelExpensesTable		TravelExpenses

Table 3.11: Travel Expenses: Business Level Rules

[1]	Permit	Finance&Administration	create	TravelExpenses
[2]	Permit	FinanceAnalyst	read	TravelExpenses
[3]	Permit	FinancePayroll	read, write	TravelExpenses
[4]	Permit	Controlling	read	TravelExpenses

Table 3.12: Backup Data: Resources

Resource	**Owner**	**Explicit classifications**
▷ HostComputer1234		
▷ DB2DataBaseServer		
▷ DataBaseSchemaXY		
▷ TravelExpensesTable		TravelExpenses, Backup

Table 3.13: Backup Data: Business Level Rules

[1]	Permit	InformationServices	read, write, create, delete	Backup

3.10.4 Backup Data

This scenario is inspired by the access control to backup data at the IBM ZRL. It aims to meet the requirement that "Administrators may manipulate all backup data". Therefore the actions that are used in this scenario are *"read"*, *"write"*, *"create"* and *"delete"*. A *"Backup"* label is used to classify data. Table 3.12 shows the used resources and Table 3.13 shows the the access control rules on business level.

3.10.5 Discovering Issues in the Policies

After data backup, data in the resource "TravelExpensesTable" is labeled with the labels "Backup" *and* "TravelExpenses". This means that a subject with the "InformationServices" role has access to all travel expense data. This behavior is not intended. But the good news is, the policy editor can help in discovering such issues. With the help of an analysis functions of the policy editor that shows which subjects are allowed to access travel expense data, this error in access control can be discovered. The person responsible for formulating the access control policies for travel expense data will then contact the person responsible for the access control policies for backup data and they negotiate a different access control behavior. For example, they might negotiate to create a new "BackupAdmin" role — which is assigned to explicitly authorized employees — and to restrict the access to backup data to employees who hold this new role.

In a first attempt of creating the common classification scheme RoomType, the label "ConfidentialPrinterRoom" (Table 3.6) was a child of the label "PrinterRoom". This lead to a problem in the label hierarchy since a policy that permits the access to resources labeled with the "PrinterRoom" label, would also permit access to resources labeled with the "ConfidentialPrinterRoom" label. This issue

is fixed in the current scenario.

3.10.6 Policy Delegation

This scenario aims to show how a possible policy delegation scenario could look like and gives examples of final and recommended policies. In Section 3.5.1 it was mentioned that each business level policy resides on a specific hierarchy level. For this scenario we assume an organization hierarchy as shown in Figure 3.4 where each organization has an associated hierarchy level. A detailed description on how the delegation mechanism is implemented can be found in Section 5.1.2 on page 62.

The administration office of Canton Zurich wants to make sure that emergency teams have the possibility to safe peoples' lives. Therefore it creates a final policy that ensures this behavior (see Table 3.14). The IBM Switzerland Headquarters are concerned about the safety of financial data. To ensure that access to this kind of data is controlled properly, they create a recommended policy that denies every employee the access to financial data (see Table 3.15). The headquarters intend that institutions on lower hierarchy level allow later on the access for authorized personnel. The access control requirements and policies for the IBM ZRL have already been described in Section 3.10.3. Therefore, assume the policies of Table 3.11 on the lowest hierarchy level in the organization hierarchy. As the access control policy of the IBM Switzerland Headquarters is only recommended, the policy of the IBM ZRL overrides, and the roles "Finance&Administration", "FinanceAnalyst", "FinancePayroll" and "Controlling" are allowed to access data that is classified as "TravelExpenses".

Figure 3.4: Organization Hierarchy

Level 10: Administration Office of Canton Zurich
|
Level 20: IBM Switzerland Headquarters
|
Level 30: IBM Zurich Research Laboratory

Table 3.14: Final Policy by Administration Office of Canton Zurich

[1]	Permit	EmergencyTeam	enter	Area

Table 3.15: Recommended Policy by IBM Switzerland Headquarters

[1]	Deny	Employee	read	Finance&Administration

Chapter 4

Analysis Algorithms

4.1 Introduction

Multiple policy authors may formulate access control policies independently. The set of all individual policies jointly forms the *policy repository*, which is the base for a policy decision point (PDP) to render an authorization decision. This chapter specifies different analysis algorithms that may be performed on a policy repository. Most of the introduced algorithms are later implemented within the policy editor. The types of analysis that are proposed in this chapter are the following:

- A *decision request on business level* is the base for most of the other analysis algorithms. By formulating the request on the business level an author can render decisions for roles instead of subjects and/or for labels instead of resources. Consequently the author gets the possibility to simulate an evaluation process that is performed by a PDP to determine a decision for an arbitrary request.
- *Contribution Analysis* tells an author *which* rules and policies in particular are the reason that a specific decision is rendered.
- *Policy Override Detection* lets authors find out which of their policies are overridden by other authors or which policies they override themselves.
- *Authorization Analysis* is used by an author to determine how she has to formulate a decision request in order to get a specific authorization decision by a PDP.
- *Coverage Analysis* lets an author identify the subjects that may never access any resource as well as the resources that may never be accessed by anyone.

4.2 Business Level Decision Request

As mentioned before, an XACML PDP renders an authorization decision for a specific decision request that comprises a subject, an action and a resource.

From now on we refer to such a decision request as *decision request on XACML level*. As we are formulating access control policies on a business level we also want to be able to issue decision requests that are on a business level. Such a *decision request on business level* comprises roles instead of subjects and/or labels instead of resources. It lets an author therefore issue requests that do not contain specific subjects or specific resources any more. Using the policy editor prototype that is presented later, a policy author can issue decision requests on both levels, i.e., she can therefore simulate an evaluation process and render an authorization decision.

Our goal is to use an existing XACML PDP to render an authorization decision for a decision request. Such a decision request is now either on the business level or on XACML level. However, a PDP can only answer requests on XACML level, thus we give a translation from the business level to XACML whereby the XACML request addresses the intentions of the request on business level.

Every time an author uses the policy editor to simulate a decision request, requests that are formulated on the business level are translated to XACML first before an XACML PDP is used to render a decision. In case a user provides roles instead of a subject, the translation process comes up with a temporary subject that has *exactly* the given roles. In case a user provides labels instead of resources, the translation process comes up with a temporary resource that has *exactly* the given labels. A modified decision request, that contains the temporary subject and the temporary resource, is handed over to an XACML PDP in order to reach a decision. In case a user provides already a subject and a resource a lookup process determines the corresponding roles and labels and lets the PDP render a decision. This is illustrated in Figure 4.1.

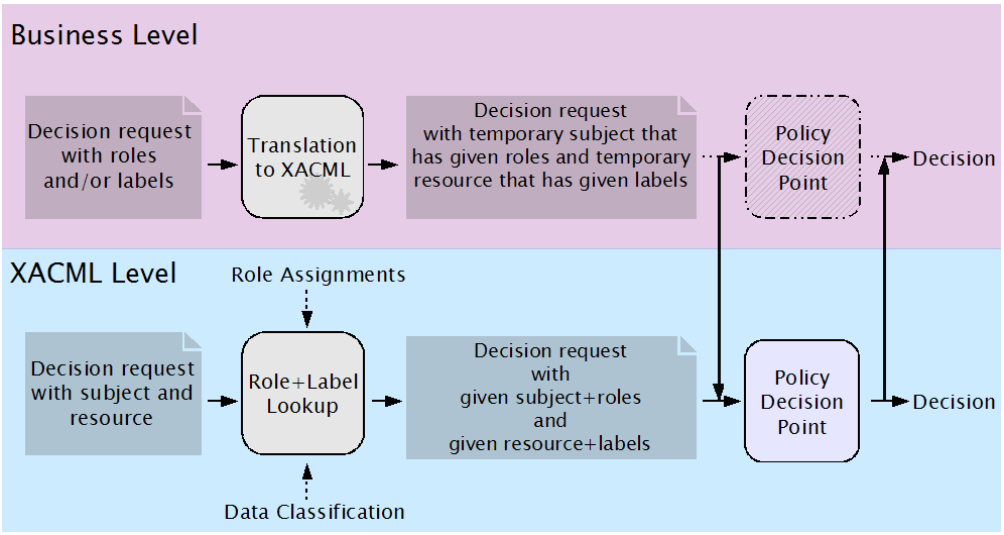


Figure 4.1: Decision Request Evaluation

Decision requests on business level are necessary for the simulation in the policy editor and they serve as basis for more analysis algorithms. However,

in an operational system only requests on the XACML level are issued. To formalize decision requests on business level we use the following definitions:

- S, O, A, R, and L, which refers to subjects, roles, actions, resources and labels respectively,
- E, which refers to environments,
- $LA \subseteq L \times R$, a many-to-many label to resource assignment relation, and
- $OA \subseteq O \times S$, a many-to-many role to subject assignment relation.

The environment components E address attributes that are relevant to an authorization decision but are independent of a particular subject, action, and resource. For example, an attribute that specifies the date and time of the decision request would be provided as environment attribute. This environment attribute is required if a business level rule has a time range condition, which needs a reference point to evaluate its value.

4.2.1 Decision Rendering on XACML Level

An XACML PDP is able to render an authorization decision for a subject s, an action a, a resource r and an environment e, i.e., the PDP is able to evaluate a "permission" function

$$p(s, a, r, e) \rightarrow \{true, false\} \tag{4.1}$$

and thus to answer the question "Is subject s permitted to perform action a on resource r within the environment e?". If the function $p(\cdot)$ evaluates to true, then the subject is permitted to perform the action. Otherwise the subject is not permitted to perform the action. With the help of the function $p(\cdot)$, which is the only function that can be evaluated by an XACML PDP, decision requests on a business level can be evaluated by using values for s and r which satisfy special properties. These properties are defined in the following.

4.2.2 Use of Roles instead of Subjects

When formulating policies on a business level, we are able to target roles instead of subjects, so we also want to be able to issue a decision request on a business level that comprises roles instead of a subject. Therefore we want to be able to evaluate a "role permission" function

$$po(\{o_1, o_2, \ldots, o_n\}, a, r, e) \rightarrow \{true, true\} \tag{4.2}$$

and thus to answer the question "Is a subject with exactly the roles o_1, o_2,..., o_n permitted to perform action a on resource r within environment e?". For example, recall the rule "Permit Employee enter ConferenceRoom" from the physical access control scenario (see Table 3.9). This rule is intended to permit all subjects that have the role employee the physical access to resources labeled as conference room. A decision request on business level could therefore be

$$po(\{Employee\}, enter, C201, 9pm)$$

which answers the question "Is a subject with the role "Employee" permitted to "enter" the resource "C201" at 9p.m.?". In contrast to a decision request on XACML level where we would ask for the permission of a specific subject, we are asking if *any* subject that has the role "Employee" — no matter if such a subject exists or not — has permission or not.

Note that the answer the function $po(\cdot)$ returns is only valid for *exactly* the given role set, i.e., for subjects that would have the roles o_1, o_2,..., o_n and *only these* roles. This is because for a subject that has a role in addition to the given roles, a business level rule could state an access control behavior that is different to the one that is stated for the given roles. For example, imagine a subject "Kelly" that has just the role "Employee". The example decision request from above would give the same answer as a request for "Kelly" herself. So far so good. But now imagine that "Kelly" has also the "Contractor" role. Then is not guaranteed any more that the example decision request from above returns the same answer as a request for "Kelly" herself. Especially if there exists a business level rule like the one in Table 3.9 that states "Deny Contractor enter Area between 8pm and 6am". Furthermore the given answer is only valid for subjects (1) for that no business level rules exist that target the subject directly and (2) that are not owner of the requested resource. This is because rules that target the subject directly or have owner conditions could state different access control behavior.

The function $po(\cdot)$ is defined as

$$\begin{aligned} &po(\{o_1, o_2, \ldots, o_n\}, a, r, e) \\ &\quad = p(s_{temp}, a, r, e) \end{aligned} \quad \left| \begin{array}{l} \\ (o_1, s_{temp}) \in OA \wedge (o_2, s_{temp}) \in OA \wedge \ldots \wedge \\ (o_n, s_{temp}) \in OA \wedge |\{(x, s_{temp}) \in OA\}| = n \end{array} \right. \tag{4.3}$$

where s_{temp} is a subject that has (1) the roles o_1, o_2,..., o_n and no other roles and is (2) not existing in the subject repository and (3) just existing temporarily while the decision request is evaluated by the PDP. This temporary subject is necessary because only for such a *new* subject, which has not existed before, is guaranteed that no business level rules exists that govern access for this subject directly. For example, for this subject s_{temp} is guaranteed that there exists no rule like "Deny s_{temp} enter Area".

4.2.3 Use of Labels instead of Resources

Since we are now able to address roles instead of subjects, we also want to be able to issue a decision request that comprises labels instead of subjects. Note that the methods to use labels instead of resources that are described in the section are identical to the ones to use roles instead of subjects.

To address labels instead of resources we want also be able to evaluate a "label permission" function

$$pl(s, a, \{l_1, l_2, \ldots, l_n\}, e) \rightarrow \{true, true\} \tag{4.4}$$

and thus to answer the question "Is subject s permitted to perform action a on a resource labeled with exactly the labels l_1, l_2,..., l_n within environment e?".

For example, recall again the rule "Permit Employee enter ConferenceRoom'. A decision request on business level could therefore be

$$pl(Alice, enter, \{ConferenceRoom\}, 11am)$$

which answers the question "Is "Alice" at 11a.m. permitted to "enter" resources labeled with the label "ConferenceRoom?". In contrast to a decision request on XACML level where we would ask for the permission to enter a specific resource, we are asking if there is permission to enter "any" resource that has the label "ConferenceRoom", no matter if such a resource exists or not.

Note that the answer the function $pl(\cdot)$ returns is only valid for exactly the given label set, i.e., for resources that would have the labels l_1, l_2,..., l_n and *only these* labels. The reasoning is similar to the previous one for exact role sets. Furthermore the given answer is only valid for resources, for which no business level rules exist that target the resource directly.

The function $pl(\cdot)$ is defined as

$$\begin{array}{l|l} pl(s, a, \{l_1, l_2, \ldots, l_n\}, e) & \\ \quad = p(s, a, r_{temp}, e) & (l_1, r_{temp}) \in LA \wedge (l_2, r_{temp}) \in LA \wedge \ldots \wedge \\ & (l_n, r_{temp}) \in LA \wedge |\{(x, r_{temp}) \in LA\}| = n \end{array} \tag{4.5}$$

where r_{temp} is a resource that has (1) the labels l_1, l_2,..., l_n and no other labels and is (2) not existing in the resource repository and (3) just existing temporarily while the decision request is evaluated by the PDP.

4.2.4 Use of Roles and Labels

As combination of the two previous requirements, we want to be able to issue a decision request that comprises roles instead of subjects *and* labels instead of resources. Therefore we want to be able to evaluate a "role and label permission" function

$$pol(\{o_1, o_2, \ldots, o_m\}, a, \{l_1, l_2, \ldots, l_n\}, e) \rightarrow \{true, true\} \tag{4.6}$$

and thus to answer the question "Is a subject with exactly the roles o_1, o_2,..., o_m permitted to perform action a on a resource labeled with exactly the labels l_1, l_2,..., l_n within environment e?". A decision request on business level could therefore be

$$pol(\{Employee\}, enter, \{ConferenceRoom\}, 9pm)$$

The function $pol(\cdot)$ is defined as

$$
\begin{aligned}
pol(\{o_1, o_2, \ldots, o_m\}, a, \{l_1, l_2, \ldots, l_n\}, e) \\
= p(s_{temp}, a, r_{temp}, e)
\end{aligned}
\Bigg|
\begin{array}{l}
(o_1, s_{temp}) \in OA \wedge (o_2, s_{temp}) \\
\in OA \wedge \ldots \wedge (o_m, s_{temp}) \in OA \\
\wedge\, |\{(x, s_{temp}) \in OA\}| = m \wedge \\
(l_1, r_{temp}) \in LA \wedge (l_2, r_{temp}) \\
\in LA \wedge \ldots \wedge (l_n, r_{temp}) \in LA \wedge \\
|\{(x, r_{temp}) \in LA\}| = n
\end{array}
\tag{4.7}
$$

where the same requirements are put on s_{temp} and r_{temp} as above.

4.2.5 Policy Application

In Section 3.9 it has been defined under which conditions a business level rule is *applicable* to a decision request on XACML level. Because it is also important for the following analysis algorithms under which conditions a business level rule is applicable to a decision request on the business level, this is defined in the following.

The rule application semantics for a decision request on a business level is the same as for a decision request on XACML level (see Section 3.9.1) with two extensions:

1. In case the decision request addresses roles instead of subjects, a subject match is only accomplished if (1) the rule's target also addresses *roles instead of subjects* and (2) the rule's target addresses *the same roles* as the decision request.

2. In case the decision request addresses labels instead of resources, a resource match is only accomplished if (1) the rule's target also addresses *labels instead of resources* and (2) the rule's target addresses *the same labels* as the decision request.

4.3 Contribution Analysis

Using a decision request, a policy author is able to determine if a certain access is permitted or not. Such a decision request can either be on the XACML level, which means evaluating $p(\cdot)$, or on the business level, which implies evaluating $po(\cdot)$, $pl(\cdot)$ or $pol(\cdot)$. In both cases the answer the policy author gets is either true or false, i.e., a certain access is, or is not, permitted. For a PEP that has to enforce the decision, such an answer is sufficient. But for a policy author it is interesting *why* a true decision, or *why* a false decision was rendered. Therefore she is interested in the particular business level rules that are responsible for the rendered decision, i.e., she is interested in the rules that *contribute* to the decision. Furthermore she can be interested in the rules that are not contributing,

or in the rules that may never contribute to a particular decision. The corresponding analysis algorithm is referred to as *contribution analysis*. To formalize this kind of analysis, the following definitions are used:

- PR, which refers to policy repositories (although we are always referring to the same policy repository — the one the policy editor is operating on — we talk for formalization reasons about repositor*ies*),
- DR, which refers to decision requests, either on XACML or on a business level,
- BLR, which refers to business level rules, and
- a "contribution" function $c(blr, dr, pr) \rightarrow \{true, true\}$, which evaluates to true if the business level rule blr *contributes* to an authorization decision for the decision request dr according to the policy repository pr, and to false otherwise. A business level rule contributes to an authorization decision for a decision request if it is *applicable* to the decision request (see Section 3.9 on page 39 for policy application on XACML level as well as Section 4.2.5 for policy application on a business level) and the business level condition is fulfilled.

The set of business level rules that contribute to an authorization decision for a decision request dr according to a policy repository pr is defined by

$$\{blr \mid c(blr, dr, pr)\} \tag{4.8}$$

This set answers the question "Which business level rules contribute to an authorization decision for the decision request dr according to the policy repository pr?".

The set of business level rules that do not contribute to an authorization decision for a decision request dr according to a policy repository pr is defined by

$$\{blr \mid \neg c(blr, dr, pr)\} \tag{4.9}$$

This set answers the question "Which business level rules blr do not contribute to an authorization decision for the decision request dr according to the policy repository pr?".

The set of business level rules that may never contribute to a decision request dr according to a policy repository pr is defined by

$$\{blr \mid \forall dr \ \neg c(blr, dr, pr)\} \tag{4.10}$$

This set answers the question "Which business level rules blr may never contribute to an authorization decision for any decision request according to a policy repository pr?".

4.4 Policy Override Detection

Policy authors formulate business level policies that reside on a specific hierarchy level. For a policy author it is interesting to know if other policy authors override his policy or which policies she overrides herself. To determine if a policy overrides another policy, the contained rules are investigated first for overrides. Thereby a rule overrides another rule if it addresses the same subjects, actions and resources as the other rule and resides at the same time on a higher level in the policy delegation hierarchy. If then a rule override is found for at least one pair of rules out of two policies, one policy overrides the other one. To formalize *policy override detection* we use the following definitions:

- BLP, which refers to business level policies,
- BLR, which refers to business level rules,
- $RA \subseteq BLR \times BLP$, a many-to-one rule to policy assignment relation,
- a "delegation level" function $dl(blp) \rightarrow \mathbb{N}^+$, which returns the level of the business level policy blp in the policy delegation hierarchy as integer value where a lower value means a higher priority,
- a "target match" function $tm(blr_1, blr_2) \rightarrow \{true, true\}$, which evaluates to true if the business level targets of the business level rules blr_1 and blr_2 have intersections in the subjects *and* actions *and* resources that are matched (for definitions on matching see Section 3.7 on page 37 as well as Section 3.8 on page 38), and to false otherwise.

To be able to decide if a business level rule blr_1 overrides another business level rule blr_2, we have to evaluate a "rule override" function

$$or(blr_1, blr_2) \rightarrow \{true, true\} \tag{4.11}$$

and thus to answer the question "Does the business level rule blr_1 override the business level rule blr_2?". The function $or(\cdot)$ is defined as

$$\begin{aligned} or(blr_1, blr_2) = tm(blr_1, blr_2) \wedge \\ dl(blp_1 \mid (blr_1, blp_1) \in RA) > dl(blp_2 \mid (blr_2, blp_2) \in RA) \end{aligned} \tag{4.12}$$

In the current definition of $or(\cdot)$, an *override* is defined by a target match of two rules where one rule resides on a higher hierarchy level. Since a target match requires that actual subjects and actions and resources exist in the subject, action and resource repositories, a policy override is only detected if this requirement is fulfilled. For example, remember the rules "Deny Employee read Finance&Administration" and "Permit FinanceAnalyst read TravelExpenses" from the policy delegation scenario, which reside on different hierarchy levels. With the current definition of $or(\cdot)$, an override of these rules is only detected if there exists at least one subject that has the "FinanceAnalyst" role, and at least one resource that has the "TravelExpenses" label. Thus, the definition of $or(\cdot)$ detects overrides that are *currently present* whereby the presence is dependent

on the policy repository, the role assignment and the data classification. Another approach for defining $or(\cdot)$ would be, to consider also overrides that *might* occur later on. Note also that $or(\cdot)$ only defines *potential* overrides since it just takes the matching into account and disregards the business level conditions of the rules.

After knowing that one business level rule overrides another business level rule, it is interesting to know which business level policy as a whole overrides another business level policy. To be able to decide if a business level policy blp_1 overrides another business level policy blp_2, we have to evaluate a "policy override" function

$$op(blp_1, blp_2) \rightarrow \{true, true\} \tag{4.13}$$

and thus to answer the question "Does the business level policy blp_1 override the business level policy blp_2?". The function $op(\cdot)$ is defined as

$$\begin{aligned} op(blp_1, blp_2) = \exists blr_1 \exists blr_2 \; or(blr_1, blr_2 \mid (blr_1, blp_1) \in RA \\ \wedge (blr_2, blp_2) \in RA) \end{aligned} \tag{4.14}$$

4.5 Authorization Analysis

A single decision request on business level makes a statement about authorization of one particular combination of request parameters. However, often it is required not to determine the authorization decision itself but the set of request parameters that evaluate to a particular authorization decision. This process is referred to as *authorization analysis*. With authorization analysis a policy author can find answers to common questions that may arise during the policy authoring process. A policy author can for example answer questions such as:

- Which subjects are permitted to "enter" the room "C230"?
- Which subjects are permitted to "enter" resources that have the label "ConferenceRoom"?
- Which roles must a subject have to be permitted to "enter" the room "C230"?
- Which roles must a subject have to be permitted to "enter" resources that have the label "ConferenceRoom"?
- etc.

The pattern of the questions above can be continued, i.e., a user can ask for subjects, roles, actions, resources or labels. Depending on the value she is interested in, she has to provide the remaining parameters to form a complete question.

4.5.1 Subject Sets

The set of subjects that are permitted to perform action a on resource r within environment e is defined by

$$\{s \mid p(s, a, r, e)\} \tag{4.15}$$

This set answers the question "Which subjects s are permitted to perform action a on resource r within environment e?".

The set of subjects that are permitted to perform action a on a resource that has the labels l_1, l_2,..., l_n and only these labels assigned within environment e is defined by

$$\{s \mid pl(s, a, \{l_1, l_2, \ldots, l_n\}, e)\} \tag{4.16}$$

This set answers the question "Which subjects s are permitted to perform action a on a resource labeled with exactly the labels l_1, l_2,..., l_n within environment e?".

4.5.2 Role Sets

The set of role sets, which has a subject exactly to hold to be permitted to perform action a on resource r within environment e is defined by

$$\{\{o_1, o_2, \ldots, o_n\} \mid po(\{o_1, o_2, \ldots, o_n\}, a, r, e)\} \tag{4.17}$$

This set answers the question "Which roles o_1, o_2,..., o_n has a subject exactly to hold to be permitted to perform action a on resource r within environment e?".

The set of role sets, which has a subject exactly to hold to be permitted to perform action a on a resource r labeled with the labels l_1, l_2,..., l_n and only these labels within environment e is defined by

$$\{\{o_1, o_2, \ldots, o_m\} \mid pol(\{o_1, o_2, \ldots, o_m\}, a, \{l_1, l_2, \ldots, l_n\}, e)\} \tag{4.18}$$

This set answers the question "Which roles o_1, o_2,..., o_m has a subject exactly to hold to be permitted to perform action a on a resource labeled with exactly the labels l_1, l_2,..., l_n within environment e?".

4.5.3 Action Sets

The set of actions, which may be performed by subject s on resource r within environment e is defined by

$$\{a \mid p(s, a, r, e)\} \tag{4.19}$$

This set answers the question "Which actions a may be performed by subject s on resource r within environment e?".

The set of actions, which may be performed on resource r by a subject that holds the roles o_1, o_2,..., o_n and only these roles within environment e is defined by

$$\{a \mid po(\{o_1, o_2, \ldots, o_n\}, a, r, e)\} \tag{4.20}$$

This set answers the question "Which actions a may be performed on resource r by a subject that holds exactly the roles o_1, o_2,..., o_n within environment e?".

The set of actions, which may be performed by subject s on a resource labeled with the labels l_1, l_2,..., l_n and only these labels within environment e is defined by

$$\{a \mid pl(s, a, \{l_1, l_2, \ldots, l_n\}, e)\} \tag{4.21}$$

This set answers the question "Which actions a may be performed by subject s on a resource labeled exactly with the labels l_1, l_2,..., l_n within environment e?".

The set of actions, which may be performed by a subject that holds the roles o_1, o_2,..., o_m and only these roles on a resource labeled with the labels l_1, l_2,..., l_n and only these labels within environment e is defined by

$$\{a \mid pol(\{o_1, o_2, \ldots, o_m\}, a, \{l_1, l_2, \ldots, l_n\}, e)\} \tag{4.22}$$

This set answers the question "Which actions a may be performed by a subject that holds exactly the roles o_1, o_2,..., o_m on a resource labeled exactly with the labels l_1, l_2,..., l_n within environment e?".

4.5.4 Resource Sets

The set of resources on which action a may be performed by subject s within environment e is defined by

$$\{r \mid p(s, a, r, e)\} \tag{4.23}$$

This set answers the question "On which resources r may action a be performed by subject s within environment e?".

The set of resources on which action a may be performed by a subject that holds the roles o_1, o_2,..., o_n and only these roles within environment e is defined by

$$\{r \mid po(\{o_1, o_2, \ldots, o_n\}, a, r, e)\} \tag{4.24}$$

This set answers the question "On which resources r may action a be performed by a subject that holds exactly the roles o_1, o_2,..., o_n within environment e?".

4.5.5 Label Sets

The set of label sets that must be assigned to a resource exactly so that subject s is permitted to perform action a on the resource within environment e is defined by

$$\{\{l_1, l_2, \ldots, l_n\} \mid pl(s, a, \{l_1, l_2, \ldots, l_n\}, e)\} \tag{4.25}$$

This set answers the question "Which labels l_1, l_2,..., l_n must be assigned to a resource exactly so that subject s is permitted to perform action a on the resource within environment e?".

The set of label sets that must be assigned to a resource exactly so that a subject that holds the roles o_1, o_2,..., o_m and only these roles is permitted to perform action a on the resource within environment e is defined by

$$\{\{l_1, l_2, \ldots, l_n\} \mid pol(\{o_1, o_2, \ldots, o_m\}, a, \{l_1, l_2, \ldots, l_n\}, e)\} \tag{4.26}$$

This set answers the question "Which labels l_1, l_2,..., l_n must be assigned to a resource exactly so that a subject that holds exactly the roles o_1, o_2,..., o_m is permitted to perform action a on this the resource within environment e?".

4.6 Coverage Analysis

A policy decision point answers to a decision request with an authorization decision according to a policy repository. The policy repository contains rules which define the access control behavior for the decision request. With *coverage analysis* it can be evaluated if there are gaps in the rule set, i.e., if there are subjects or roles that may never access any resource or if there are resources or labels that may never be accessed by anyone.

To be able to analyze the policy coverage, we have to redefine the function $p(\cdot)$ that was introduced earlier. Since an XACML PDP has actually not only the outcomes "true" and "false", but also "NotApplicable" and "Indeterminate" we define

$$p_a(s, a, r, e) \rightarrow \{true, false, NotApplicable, Indeterminate\} \tag{4.27}$$

as the function that is evaluated by a real world PDP. This function evaluates to *NotApplicable* if no statement can be made by the PDP because the underlying policy repository defines no behavior for the given input, and it evaluates to *Indeterminate* if an error occurs during the evaluation process. To keep the formalisms defined earlier consistent we redefine $p(\cdot)$ so that

$$p(\cdot) = \begin{cases} p_a(\cdot) & \text{if } p_a(\cdot) \in \{true, false\} \\ true & \text{otherwise} \end{cases} \tag{4.28}$$

With this redefined $p(\cdot)$ we are now capable to define coverage analysis.

The set of subjects which may not perform any action on any resource within environment e is defined by

$$\{s \mid \forall a \forall r \; p_a(s, a, r, e) = (false \vee NotApplicable)\} \tag{4.29}$$

This set answers the question "Which subjects s are not permitted to perform any action on any resource within environment e".

The set of role sets that has a subject exactly to hold so that no action may be performed on any resource within environment e is defined by

$$\begin{aligned} \{\{o_1, o_2, \ldots, o_n\} \mid \forall a \forall r \; po(\{o_1, o_2, \ldots, o_n\}, a, r, e) \\ = (false \vee NotApplicable)\} \end{aligned} \tag{4.30}$$

This set answers the question "Which roles o_1, o_2,..., o_n has a subject exactly to hold so that no action may be performed on any resource within environment e?".

The set of resources, on which no action may be performed by any subject within environment e is defined by

$$\{r \mid \forall s \forall a \; p(s, a, r, e) = (false \vee NotApplicable)\} \tag{4.31}$$

This set answers the question "On which resources r may no action be performed by any access requestor within environment e?".

The set of label sets that has a resource exactly to have so that no access requestor is permitted to perform any action on the resource within environment e is defined by

$$\{\{l_1, l_2, \ldots, l_n\} \mid \forall s \forall a \; pl(s, a, \{l_1, l_2, \ldots, l_n\}, e) = (false \vee NotApplicable)\} \tag{4.32}$$

This set answers the question "Which labels l_1, l_2,..., l_n has a resource exactly to have so that no access requestor is permitted to perform any action on the resource within environment e?".

Chapter 5

Implementation of Analysis Algorithms

This chapter gives an insight on how policies on a business level are translated to XACML and describes how the concepts and functions introduced and specified in the previous chapter have been implemented in the policy editor.

First, the translation of all the underlaying concepts of business level policies into XACML is described in detail. Then we explain how a decision request on business level is formulated in XACML and give an example for such a formulation. Finally, it is shown how we implemented the contribution analysis, the override detection and the authorization analysis.

5.1 Policy Translation

Since XACML emerged as standard for expressing access control policies, more and more companies follow this standard to implement access control in their IT infrastructure [8]. For policies on a business level, as they are defined in this thesis, it shall be possible to use them in the existing IT infrastructures of these companies without making any changes to the infrastructure. Therefore, the policies on business level have to be transformed into XACML policies. Although some concepts can be translated directly — remember that the policy language for business level policies was inspired by the XACML model — the new concepts that are only used on the business level are not supported by XACML. Thus, these concepts have to be translated accordingly. The concepts that need special attention during the translation process from the business level into XACML are the following:

- Label Concept
- Policy Delegation Concept
- Role Concept

The translation of these concepts is described in the following subsections. Furthermore the implementation of the translation from business level conditions to XACML conditions as well as the size of XACML policy files are discussed.

Note that the described translation is a one way translation from business level to XACML. A translation from XACML back to the business level is possible under certain conditions and with some restrictions, but not discussed here. Since the role repository as well as the label repository are input data for the translation algorithm, every change in one of the repositories requires that a new translation of the policies has to be performed.

5.1.1 Label Concept

A business level rule's target may address resources in terms of labels instead of addressing the resources directly. In XACML this approach can be implemented with the help of *resource attributes*. In an XACML rule, resource attributes are used to specify the characteristics of the resources that shall be matched by the rule. In a typical XACML policy, a resource is addressed by an *identity attribute*, but what we want is to address the resource in terms of label attributes. Each attribute has an *attribute identifier*, which states the kind of characteristic and an *attribute value*. Due to the fact that XACML is extensible[1], a *label attribute* with the identifier

"urn:zurich:ibm:names:xacml:2.0:resource:label"

to expresses such a label characteristic of a resource is introduced here. Using this new label attribute, we can address resources in XACML rules by their labels instead of their identity.

For every business level rule that addresses labels instead of resources, in the corresponding XACML rule an XACML `<Resource>` element is created that comprises the business level rule's target labels expressed by our label attribute. The XACML rule is then always considered if the access requestor tries to access a resource that has at least the labels contained in the `<Resource>` element.

Since the labels in a classification scheme may be organized in a hierarchy whereby the label hierarchy defines a specialization relation between the labels, this specialization relation is considered during the policy translation. Whenever a policy on business level states access control behavior in terms of labels, then the intention is that access control behavior is also stated for all the descendants of that label. For example, remember the rule "Permit Employee enter Area if Owner" from the physical access control scenario (see Table 3.9). This rule is intended to address all resources that have the "Area" label as well as resources that have some descendant label like "Office", "ConferenceRoom", "PrinterRoom" etc. Therefore, at a policy translation of a rule from business level to XACML, an XACML <Resource> element is created not only for the business level rule's target label itself, but also for every descendant of that label.

A business level rule's target may contain more than one label at the same time, i.e., a whole label set. As defined in Section 3.7.3, in this case, the rule targets resources that have all these labels together. For the policy translation of such a rule, a `<Resource>` element is created for every *consistent* combination of the set's labels together with their descendants whereby a label set is

[1] *"eXtensible"* Access Control Markup Language

considered as consistent if no label of the set is ancestor of another label of this set. For example, consider a rule "Permit Alice enter Area,ComputerScience". This rule intends to let "Alice" enter every resource that has the labels "Area" *and* "ComputerScience", i.e., "Alice" is allowed to enter every room that belongs to the computer science department. The policy translation algorithm creates therefore also `<Resource>` elements for every consistent combination of the "Area" and the "ComputerScience" label together with their descendants, such as

- "Area", "Security&Assurance',
- "Area", "Security&Cryptography',
- "Office", "Security&Assurance",
- "Office", "Security&Cryptography",
- ...

5.1.2 Policy Delegation Concept

A delegation concept for access control policies like the one proposed in this thesis (see Section 3.5.1) is not supported explicitly in XACML. Nevertheless, we can translate the delegation mechanism of the policies on business level to XACML.

The proposed concept of policy delegation in this thesis is in the end a problem of finding the first policy that was not delegated to somebody else. To find this policy, we arrange all the policies in a specific sequence according to their hierarchy level and final status, and then we use the first policy that is applicable in this sequence. Since XACML has the built in policy combining algorithm "First-applicable" [10] whereby the first policy that states access control behavior for the given decision request determines the overall outcome (if no access control behavior is stated in the first policy, then the second policy in the list is considered, etc.), the remaining part is to find and specify the right sequence of the policies. Recall that the requirements for the proposed delegation mechanism are:

- The higher the hierarchy level, the earlier a policy must be considered to decide about the access control behavior.
- Final policies can not be overridden by policies on lower hierarchy levels.
- Recommended policies may be overridden by policies on lower hierarchy levels.

According to these requirements, the policies have to be split in two groups. The first group contains the final policies, and the second group contains the recommended policies. The policies in the first group are then sorted *ascending* by the hierarchy level, and the policies in the second group are sorted *descending* by the hierarchy level. Then, an XACML policy set with the policy combining

algorithm "First-applicable" is created, whereby at first the sorted policies of the first group and then the sorted polices of the second group are attached to this policy set.

The following provides an example sequence of some business level policies as they would appear and be considered in an XACML policy set with the combining algorithm "First-applicable" after a policy translation. Note that a higher hierarchy level is associated with a smaller integer value. Also, observe that the final policies are sorted ascending, and the recommend policies are sorted descending:

1. Policy with Hierarchy Level 10 [final]
2. Policy with Hierarchy Level 20 [final]
3. Policy with Hierarchy Level 30 [final]
4. Policy with Hierarchy Level 50 [recommended]
5. Policy with Hierarchy Level 30 [recommended]
6. Policy with Hierarchy Level 20 [recommended]
7. Policy with Hierarchy Level 10 [recommended]

At each hierarchy level there may be more than one final or more than one recommended policy. In this case it is necessary to pool all policies at the same hierarchy level together in single policy. In XACML this can be done by attaching the relevant XACML policies to a XACML `<PolicySet>` and combining the policies with the combining algorithm that is defined for the business level policy set.

5.1.3 Role Concept

Although OASIS published a Profile that defines how to implement access control policies in conjunction with RBAC [9], this Profile cannot be applied here. The reason for that is the way in which the policy delegation mechanism has to be translated from business level to XACML. Thereby the policies have to be sorted — according to the described requirements (see 5.1.2) — and combined *as a whole* with the "First-Applicable" combining algorithm. However, the RBAC Profile resolves the role hierarchy by (1) splitting the individual policies up into parts where every part contains just the access control behavior for a specific role and (2) connecting the parts with references. This splitting mechanism is not compatible with our method to translate the policy delegation mechanism.

The approach which is used instead is the same as the one for translating the label concept. The attribute identifier that is used to specify the *role attribute* of a subject is

"urn:oasis:names:tc:xacml:2.0:subject:role"

which is defined by OASIS in the RBAC Profile.

For every business level rule that addresses roles instead of subjects, in the corresponding XACML rule an XACML `<Subject>` element is created that comprises the business level rule's target roles expressed by OASIS's role attribute. The XACML rule is then always considered if the access requestor has at least the roles of the `<Subject>` element associated.

The translation of the role hierarchy as well as the translation of role sets is implemented in the same way as for the label concept (see Section 5.1.1).

5.1.4 Conditions

For the translation of an owner condition to XACML an *owner attribute* with the identifier

"urn:zurich:ibm:names:xacml:2.0:resource:owner"

has been introduced. The translation of an owner condition can be seen in Appendix A on lines 149–168.

An example translation of a time range condition can be seen in Appendix A on lines 277–285.

5.1.5 XACML Policy File Size

The reduction of the policy quantity, which was mentioned in Section 3.2.1, has some side effect on the size of the XACML policy file. The reduced amount of policies that have to be formulated derives from the possibility to cover an entire path in a role or label hierarchy up to a particular depth. While for policies on a business level the role and label hierarchies are represented in the role and label repositories, after the translation from business level to XACML, these hierarchies are partially reflected in the XACML policy itself. Remember that for a label that is targeted on business level, a `<Resource>` element is created for every descendant of that role. The same is true for roles and their `<Subject>` elements. Thus, the deeper the role and label hierarchy trees are, the bigger the XACML policies files may get. For example, consider a rule that states "Permit Alice enter Area,SiteOperations". Since the label "Area" has seven descendant labels and the label "SiteOperations" has four descendant labels, an overall amount of

$$40 = (1 + 7) \cdot (1 + 4)$$

`<Resource>` elements is created for the XACML representation of this rule.

5.1.6 Example Rule

The translation of the business level rule "Permit Finance&Administration create TravelExpenses" to XACML has the following XML representation.

```
1  <Rule Effect="Permit" RuleId="urn:zurich:ibm:names:xacml:2.0:ruleid:4">
2    <Target>
3      <Subjects>
4        <Subject>
5          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
               string-equal">
6            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                 string">http://www.example.org/role#
                 financeAndAdministration</AttributeValue>
7            <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:
                 xacml:2.0:subject:role" DataType="http://www.w3.org/2001/
                 XMLSchema#string"/>
8          </SubjectMatch>
9        </Subject>
10       <Subject>
11         <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
               string-equal">
12           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                 string">http://www.example.org/role#financeAnalyst</
                 AttributeValue>
13           <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:
                 xacml:2.0:subject:role" DataType="http://www.w3.org/2001/
                 XMLSchema#string"/>
14         </SubjectMatch>
15       </Subject>
16       <Subject>
17         <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
               string-equal">
18           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                 string">http://www.example.org/role#controlling</
                 AttributeValue>
19           <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:
                 xacml:2.0:subject:role" DataType="http://www.w3.org/2001/
                 XMLSchema#string"/>
20         </SubjectMatch>
21       </Subject>
22       <Subject>
23         <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
               string-equal">
24           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                 string">http://www.example.org/role#financePayroll</
                 AttributeValue>
25           <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:
                 xacml:2.0:subject:role" DataType="http://www.w3.org/2001/
                 XMLSchema#string"/>
26         </SubjectMatch>
27       </Subject>
28     </Subjects>
29     <Resources>
30       <Resource>
31         <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
               string-equal">
32           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                 string">http://www.example.org/label#travelExpenses</
                 AttributeValue>
33           <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:names:
                 xacml:2.0:resource:label" DataType="http://www.w3.org/2001/
                 XMLSchema#string"/>
34         </ResourceMatch>
35       </Resource>
36     </Resources>
37     <Actions>
38       <Action>
39         <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
               string-equal">
```

```
40              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                    string">http://www.example.org/action#create</
                    AttributeValue>
41              <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:
                    xacml:1.0:action:action-id" DataType="http://www.w3.org
                    /2001/XMLSchema#string"/>
42            </ActionMatch>
43          </Action>
44        </Actions>
45    </Target>
46 </Rule>
```

Line 1 introduces the XACML Rule. The effect of the rule is to *permit* the access.

Line 2 introduces the target of the rule. The target of a rule describes the decision requests to which this rule applies. If the subject, resource, action or environment in a decision request do not match the values specified in the rule's target, then the remainder of the rule does not need to be evaluated, and a value of "NotApplicable" is returned to the rule evaluation.

Line 3 introduces the subjects that are matched by this rule. This rule specifies four kinds of subjects that are matched.

The lines 4–9 specify one of the four kinds of subjects that are matched by this rule. In this case it is stated that subjects are matched that have an attribute with the identifier *urn:oasis:names:tc:xacml:2.0:subject:role* that has the value *http://www.example.org/role#financeAndAdministration*.

The subjects that are introduced in the lines 10, 16 and 22 state that also subjects are matched that have the roles *http://www.example.org/role#finance Analyst*, *http://www.example.org/role#controlling* or *http://www.example.org/ role#financePayroll*. These roles are the descendants of the "Finance& Administration" role.

Line 29 introduces the resources that are matched by this rule. The resources that are matched must have an attribute with the identifier *urn:zurich :ibm:names:xacml:2.0:resource:label* — which was introduced in this thesis — that has the value *http://www.example.org/label#travelExpenses*.

Line 37 introduces the actions that are matched. The actions that are matched must have the identifier *http://www.example.org/action#create*.

5.2 Formulation of Business Level Decision Request

To let an XACML PDP render an authorization decision — which means evaluating the function $p(\cdot)$ — an XACML `<Request>` context is created that comprises the following elements:

- A mandatory `<Subject>` element containing an attribute with the identifier *urn:oasis:names:tc:xacml:1.0:subject:subject-id*.
- A mandatory `<Action>` element containing an attribute with the identifier *urn:oasis:names:tc:xacml:1.0:action:action-id*.
- A mandatory `<Resource>` element containing an attribute with the identifier *urn:oasis:names:tc:xacml:1.0:resource:resource-id*.

– An optional `<Environment>` element containing attributes that are not related to a subject, an action or a resource.

The elements given in the `<Request>` context are then compared — using string comparison — with the `<Target>` elements of the rules in the XACML policy to determine finally an authorization decision. Since the subject-id, the action-id and the resource-id attributes are mandatory, we always have to render the decision for a specific subject, a specific action and a specific resource. To render an authorization decision for a decision request on business level, special requirements have been specified in Section 4.2 that are put on the subject and/or the resource.

In order to fulfill the first requirement that is put on the subject to evaluate the function $po(\cdot)$, a subject attribute — with the attribute identifier *urn:oasis:names:tc:xacml:2.0:subject:role* — for *every* element in the given role set and *only these* elements is included in the `<Request>` context. To fulfill the second and third requirement, the value of the subject-id attribute must be a subject name that is not existing in the subject repository and just existing temporarily. Therefore we choose an empty string as value and ensure at the same time, that no subject with an empty string as identifier is allowed in the policy editor and therefore in the business level policies.

The same methods are used to evaluate the function $pl(\cdot)$. A resource attribute with the identifier *urn:zurich:ibm:names:xacml:2.0:resource:label* is created for every element in the given label set and only these elements. Further, an empty string is chosen as value of the resource-id attribute while making sure that no resource with an empty string as identifier is allowed in the policy editor and therefore in the business level policies.

To evaluate the function $pol(\cdot)$, the methods to evaluate $po(\cdot)$ and $pl(\cdot)$ are combined. An example on how to formulate a request to evaluate the function $pol(\cdot)$ is given in the next section.

5.2.1 Example Request

The following gives an example on how to evaluate a decision request on business level that corresponds to the function call

$$pol(\{FinanceAnalyst\}, read, \{TravelExpenses\}, 9am)$$

which answers the question "Is a subject with the role "FinanceAnalyst" permitted to "read" resources labeled with the label "TravelExpenses" at 9a.m?".

```
1 <Request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="urn
      :oasis:names:tc:xacml:2.0:context:schema:os" xsi:schemaLocation="urn:
      oasis:names:tc:xacml:2.0:policy:schema:os access_control-xacml-2.0-
      context-schema-os.xsd">
2   <Subject>
3     <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-
          id" DataType="http://www.w3.org/2001/XMLSchema#string">
4       <AttributeValue></AttributeValue>
5     </Attribute>
6     <Attribute AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
          DataType="http://www.w3.org/2001/XMLSchema#string">
```

```
7         <AttributeValue>http://www.example.org/role#financeAnalyst</
              AttributeValue>
8       </Attribute>
9     </Subject>
10    <Resource>
11      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:
            resource-id" DataType="http://www.w3.org/2001/XMLSchema#string">
12        <AttributeValue></AttributeValue>
13      </Attribute>
14      <Attribute AttributeId="urn:zurich:ibm:names:xacml:2.0:resource:owner
            " DataType="http://www.w3.org/2001/XMLSchema#string">
15        <AttributeValue></AttributeValue>
16      </Attribute>
17      <Attribute AttributeId="urn:zurich:ibm:names:xacml:2.0:resource:label
            " DataType="http://www.w3.org/2001/XMLSchema#string">
18        <AttributeValue>http://www.example.org/label#travelExpenses</
              AttributeValue>
19      </Attribute>
20    </Resource>
21    <Action>
22      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id
            " DataType="http://www.w3.org/2001/XMLSchema#string">
23        <AttributeValue>http://www.example.org/action#read</AttributeValue>
24      </Attribute>
25    </Action>
26    <Environment>
27      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:environment:
            current-time" DataType="http://www.w3.org/2001/XMLSchema#time">
28        <AttributeValue>09:00:00</AttributeValue>
29      </Attribute>
30    </Environment>
31  </Request>
```

The lines 2–9 describe the subject for that the request is made. The request is issued for a subject which subject-id is an empty string and that has just the role “FinanceAnalyst”.

The lines 10–20 describe the resource for that the request is made. The request is issued for a resource which resource-id is an empty string and that has just the label “TravelExpenses”. The empty owner attribute is necessary to let the PDP evaluate owner conditions properly.

The lines 21–25 describe the action for that the request is made. We issue the request for the “read” action.

The line 26–30 describe the environment. It contains the current time which is necessary to let the PDP evaluate time range conditions properly.

5.3 Contribution Analysis

The idea of the contribution analysis is to determine the rules and policies that apply to a given decision request. Because the application functionality for individual rules already exists in the used XACML PDP implementation[2], this functionality was reused and has been extended.

This extension implements an *evaluation recording* functionality. Thereby every rule and policy that is evaluated by the PDP — only rules and policies that *apply* to the decision request are evaluated — is recorded in a central buffer, together with the evaluation result of the evaluated rule or policy. After the

[2]In the policy editor Sun’s open source XACML implementation is used

evaluation process this buffer contains a collection of all the rules and policies the PDP has evaluated to come up with a authorization decision, as well as the corresponding evaluation results of the individual rules and policies.

The individual rules in a policy as well as the individual policies in a policy set are combined with a combining algorithm. Depending on the particular combining algorithm, the PDP stops the evaluation process as soon as it has enough information to come up with a combining result. For example, when evaluating a policy that combines the individual rules with the "Deny-overrides" combining algorithm, the PDP can stop the evaluation process as soon as the first rule with an effect of "Deny" matches. Hence, after the evaluation process our buffer contains so far not all the rules and policies that apply to the decision request, since some may not have been evaluated because the PDP had already enough information to come up with the combining result. However, in contribution analysis we are interested in *all* rules and policies that apply to a given decision request.

Therefore we came up with modified versions for all the combining algorithms that may appear in a translation of a business level policy to XACML. These modified versions perform the *entire* evaluation, i.e., they keep the overall evaluation result in mind, but *do not stop* the evaluation process.

When performing contribution analysis in the policy editor, the policy on business level is translated to XACML level whereby all the combining algorithm identifiers are replaced with the corresponding identifiers of our versions that perform an entire evaluation. In this way our recording buffer contains after the evaluation process *all* rules and policies that apply to the given decision request.

5.4 Policy Override Detection

According to the specifications in Section 4.4, an implementation of a "target match" function $tm(\cdot)$ is needed to detect overrides between rules or policies.

To implement $tm(\cdot)$, at first it has to be determined if two given business level rules have intersections in the set of subjects they govern. Therefore, for both rules, the following steps are taken to determine these individual set of subjects:

1. If the rule's target contains subjects — and matches them therefore directly — we take these subjects and are done.

2. Otherwise, if the rule's target contains roles — and matches the subjects therefore indirectly — we determine the subjects that have associations for all these roles and are done.

3. Otherwise the rule's target targets all subjects, so we take all subjects from the repository and are done.

Now it can be verified if the two determined subject sets have an intersection. If this is the case, similar methods as above are used to determine the set of actions and the set of resources that are matched. If there are intersections in all three types of sets, then we found a "target match".

Given a target match between two rules, the delegation levels of the policies that contain the rules determine if one rule *overrides* the other or *is overridden* by the other. The outcome of the delegation level function $dl(\cdot)$ is defined by the evaluation sequence of the policies that is described in Section 5.1.2.

5.5 Authorization Analysis

To determine the authorized subject, action or resource sets, as defined in Section 4.5, we iterate through all subjects, actions or resources in the repository respectively, and evaluate the corresponding decision request function.

Before we can determine the authorized role or label sets as defined in the mentioned section, we have to define for which sets in particular we want to evaluate the corresponding function. While it is typically too costly to iterate through all possible combinations of roles or labels respectively, the user is often only interested in role sets with a small number of elements. Therefore, we let the user decide the size of role sets up to which she is interested and then, we determine for all the role sets up to the specified size whether they are (1) consistent and for the consistent ones if (2) the corresponding decision request function evaluates to true.

For example, we assume that a user uses the policy editor to determine authorized role sets and specifies at the same time that she is interested in result sets up to a size of three. For a role repository that contains n different roles, this means that

$$\sum_{k=1}^{3} \binom{n}{k} = \binom{n}{1} + \binom{n}{2} + \binom{n}{3}$$

different role sets have to be checked if they are authorized or not. Therefore, the more roles the repository contains, the more role sets have to be checked.

Chapter 6

Policy Editor and Analyzer

This chapter describes the implementation and user interfaces of the policy editor and analyzer tool that was created for this thesis. It shows an overview of the components that have been used to build the policy editor and describes each individual component in detail. Different *perspectives* have been introduced to perform different user tasks, whereby a perspective shows only the user interface elements that are needed to perform a given task. These particular perspectives, together with their corresponding user tasks and the user interface elements that are needed to fulfill these tasks are described in detail as well.

6.1 Design

The Eclipse RCP was chosen as platform for the user interface, which is therefore used by the policy authors to interact with the system. A collection of RDF statements is used as data store for the editor. This data store contains the data model the policy editor is operating on. The Jena Library is used to manipulate the statements in the RDF store programmatically. Sun's XACML implementation is used as decision engine for XACML requests. Since this implementation takes its XACML inputs in XML format, an existing IBM internal library is used to generate these XACML inputs programmatically.

Figure 6.1 shows the components that are used in the design of the policy editor. The components that have been provided or extended throughout this thesis are shaded in gray.

6.1.1 Eclipse Rich Client Platform

The Eclipse[1] Rich Client Platform (RCP) is the platform that was extracted from the well known Eclipse IDE. Now, one can use the same platform that was used to build the Eclipse IDE to build applications with powerful user interfaces and rich functionality. Eclipse RCP was chosen as platform since it offers several advantages over other UI platforms for building rich client applications. For example, it has a plug-in architecture that allows other developers to extend the application easily and fast SWT widgets that have a native look and feel on

[1] `http://www.eclipse.org/`

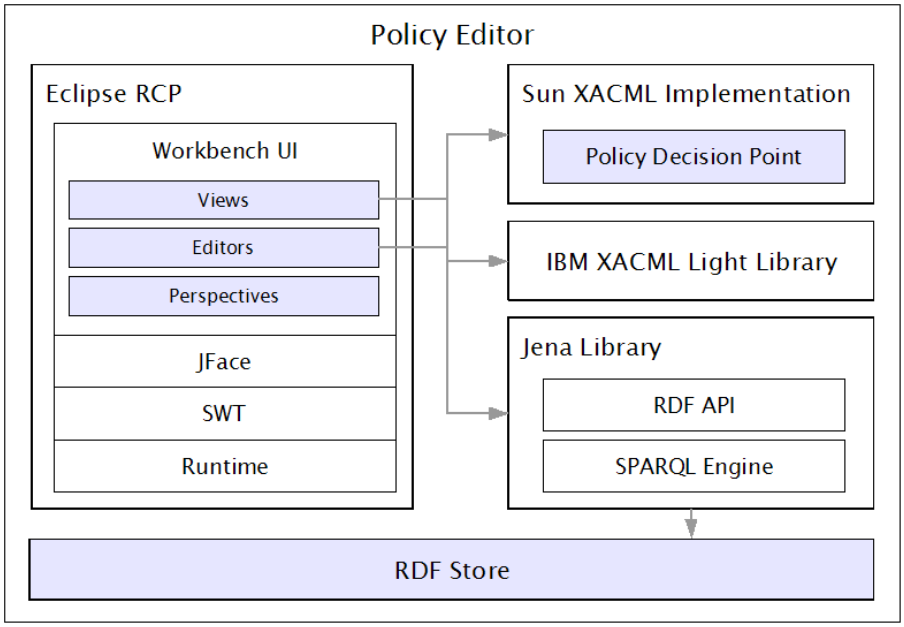


Figure 6.1: Policy Editor Components

multiple platforms. Furthermore the language to develop the application is the open source programming language Java.

However, the Eclipse RCP was not the only platform that was considered as platform for the user interface. Initially we intended to implement the interfaces as web interface. To this end we evaluated the Google Web Toolkit (GWT) [12]. Although the GWT offers interesting facilities for building web applications it turned out that the user interfaces of that toolkit are not yet powerful and user friendly enough to create a policy editor that has the focus on usability for non-technical authors.

The components of the RCP are a platform Runtime, a low level widget toolkit called Standard Widget Toolkit (SWT), a UI toolkit called JFace that provides helper classes for developing UI features with the SWT faster, and a Workbench implementation that provides the look and feel and therefore the UI personality of the Eclipse Platform. The UI paradigm of the Workbench is centered around Editors, Views and Perspectives. Editors are parts that have an "open, edit, save, close" lifecycle. Views are parts that provide information on some object and augment editors or other views. Perspectives are arrangements of views and editors whereby different perspectives are suited for different user tasks.

The plug-in architecture of the Eclipse RCP requires that all the components of the application are implemented as plug-ins. Even the Runtime, SWT, JFace and the Workbench themselves are provided as plug-ins. Because the Sun XACML implementation and the Jena library are originally not provided as plug-in — only IBM's XACML Light library is provided as plug-in — these two components had to be transformed into corresponding Eclipse plug-ins.

6.1.2 RDF Store

The advantage of using RDF as storage technology is that the RDF data model can be extended without any influence on the already present data. Whenever new information has to be stored in the RDF data store this fact is just expresses as RDF statement and there is no impact on the existing queries that read information out of the data store.

To be able to build up the RDF store with statements about the access control domain on a business level, an ontology was defined that describes the vocabulary that can be used for this purpose. The classes and the properties of this ontology are listed in the Tables 6.1 and 6.2. The data that is stored in the RDF store with this ontology comprises

- subjects, role schemes, roles and the role assignments,
- actions,
- resources with their owner and type, classification schemes, labels and the data classification,
- business level policy sets and policies together with their combining algorithms and rules together with their effect and conditions,
- organizations and their associated hierarchy levels, and
- users with their associated organization, username and password.

An example namespace is used to identify the RDF resources that are described by the ontology. Some examples of URIs as they are used in the RDF store of the policy editor are:

- http://www.example.org/subject#alice
- http://www.example.org/label#office
- http://www.example.org/businessLevelPolicy#policy002

SPARQL[2] [19] is used as RDF query language to access information contained in an RDF data store. Since only simple queries can be generated by Jena programmatically, a number of more sophisticated queries was created to get the right information out of the RDF store.

6.1.3 Jena - Semantic Web Framework for Java

Jena [16] is an open source Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS, OWL and SPARQL and includes a rule-based inference engine. We use Jena

- as RDF API,
- for reading and writing RDF in RDF/XML to create in-memory models and persistent storage, and
- as SPARQL query engine.

[2]SPARQL Protocol and RDF Query Language

Table 6.1: DCSM Ontology: RDFS Classes

- ▷ Subject
- ▷ Role
- ▷ Action
- ▷ Resource
- ▷ Label
- ▷ BusinessLevelPolicySet
- ▷ BusinessLevelPolicy
- ▷ BusinessLevelRule
- ▷ Effect
  - ▷ Permit
  - ▷ Deny
- ▷ BusinessLevelCondition
  - ▷ OwnerCondition
  - ▷ TimeRangeCondition
- ▷ CombiningAlgorithm
  - ▷ PolicyCombiningAlgorithm
    - ▷ DenyOverrides
    - ▷ PermitOverrides
  - ▷ RuleCombiningAlgorithm
    - ▷ DenyOverrides
    - ▷ PermitOverrides
- ▷ AuthorizationDecision
  - ▷ Permit
  - ▷ Deny
  - ▷ NotApplicable
  - ▷ Indeterminate
- ▷ ClassificationType
  - ▷ Explicit
  - ▷ Inherited
  - ▷ Implicit
- ▷ ResourceType
  - ▷ DefaultResource
  - ▷ HostComputer
  - ▷ RelationalDataStore
  - ▷ RelationalSchema
  - ▷ RelationalTable
  - ▷ RelationalColumm
- ▷ Scheme
  - ▷ RoleScheme
  - ▷ ClassificationScheme
- ▷ User
- ▷ Organization

Table 6.2: DCSM Ontology: RDF Properties

Property	**Domain**	**Range**
subRoleOf	Role	Role
subLabelOf	Label	Label
subResourceOf	Resource	Resource
owner	Resource	Subject
classification	Resource	Label
resourceType	Resource	ResourceType
name	rdfs:Class	xsd:string
schemaContainer	rdfs:Class	Scheme
classificationType	rdfs:Class	ClassificationType
policyCombiningAlgorithm	BusinessLevelPolicySet	PolicyCombiningAlgorithm
hasPolicy	BusinessLevelPolicySet	BusinessLevelPolicy
hierarchyLevel	BusinessLevelPolicy	xsd:int
isFinal	BusinessLevelPolicy	xsd:boolean
authorUser	BusinessLevelPolicy	User
ruleCombiningAlgorithm	BusinessLevelPolicy	RuleCombiningAlgorithm
hasRule	BusinessLevelPolicy	BusinessLevelRule
effect	BusinessLevelRule	Effect
hasTargetSubject	BusinessLevelRule	Subject
hasTargetRole	BusinessLevelRule	Role
hasTargetAction	BusinessLevelRule	Action
hasTargetResource	BusinessLevelRule	Resource
hasTargetLabel	BusinessLevelRule	Label
hasCondition	BusinessLevelRule	BusinessLevelCondition
organization	User	Organization
password	User	xsd:string
username	User	xsd:string
role	Subject	Role
email	Subject	xsd:string
phone	Subject	xsd:string
timeRangeBegin	TimeRangeCondition	xsd:time
timeRangeEnd	TimeRangeCondition	xsd:time

6.1.4 IBM XACML Light Library

While the data in the policy editor is represented as collection of RDF statements, the Sun XACML implementation demands its input in XML format. To achieve a transformation between the given RDF and the needed XML, an existing IBM internal library is used. In particular this library is used for the translation process of business level policies into XACML, as well as at the creation of a decision request. With the help of the library, the transformation from the RDF statements to XACML is achieved programmatically, but the content and the semantics of the data remain unchanged.

6.1.5 Sun XACML Implementation

Sun's XACML implementation is used in the policy editor as decision engine to evaluate decision requests. Throughout this thesis, Sun's implementation was extended with

- an EvaluationRecorder class which records all Rules and Policies that are evaluated by the PDP,
- modified versions of the combining algorithms — our versions perform an *entire* evaluation — that may appear in a translation of a business level policy to XACML, and
- the necessary modifications in the PDP to include these new combining algorithms.

6.1.6 Assumptions

Before being able to formulate and analyze policies, a proper data foundation is needed on that the policies can be formulated on. Since the main focus of the established tool is to support a policy author in formulating and analyzing his business level policies, some less interesting features of the editor were not fully implemented due to time constraints. Therefore, we created as input for the policy editor an RDF store that contains already

- subjects,
- role schemes,
- role assignments,
- actions,
- resources with an optional associated owner,
- classification schemes,
- organizations with associated hierarchy levels, and
- users with associated organization, username and password.

A data classification, which assigns the labels of the classification schemes to the resources, may also be part of the input. However, the policy editor provides functionality for classifying resources with labels.

6.2 Perspectives

Developers who build RCP applications do this by creating their own views and editors which are made visible in the workbench through perspectives. The policy editor supports a policy author during different tasks that need different user interfaces for their proper completion. Therefore different perspectives have been introduced whereby each perspective shows only the user interfaces that are needed to perform a given task. Table 6.3 lists these perspectives together with the views and editors that are available at each perspective. A summary of the tasks that can be performed in these perspectives is given in the following:

- The *Policy Authoring and Review Perspective* supports an author in creating, modifying and deleting policies and rules, in reviewing the policy repository and in detecting overrides between rules.
- The *Domain Browsing Perspective* provides an overview of the subjects, roles, actions, resources and labels that are contained in the current repository and in determining the assignments between subjects and roles and between resources and labels.
- The *Data Classification Perspective* is used by an author to classify resources by labels.
- The *Policy Simulation Perspective* lets an author simulate a decision request and perform contribution analysis accordingly.
- The *Authorization Analysis Perspective* supports an author in determining the values she has to provide in order to get a specific authorization decision from a PDP.

A policy author who wants to carry out a specific task chooses the appropriate perspective in the policy editor and is then able to use the user interface parts that are associated to this perspective.

6.3 Policy Authoring and Review

The *Policy Authoring and Review Perspective* supports an author in performing all tasks that are related to the authoring of policies and rules, as well as to the review of already existing policies. The policies that can be reviewed contain the policies the author created himself as well as the policies that have been created by other policy authors. The policy authoring related tasks that a policy author can carry out in this perspective are:

- Create a new policy.

Table 6.3: Perspectives and the corresponding Workbench Parts. An (i) entry denotes that the workbench part is available *initially* in the corresponding perspective whereby a (d) entry denotes that the part is shown on *demand*.

	Policy Authoring and Review Perspective	Domain Browsing Perspective	Data Classification Perspective	Policy Simulation Perspective	Authorization Analysis Perspective
Policy Explorer View	(i)				
Rule Outline View	(i)			(i)	
Override Detection View	(d)				
Data Classification View			(i)		
Domain Explorer View		(i)			
Classification Outline View		(i)	(i)		
Classified Resources Outline View		(i)	(i)		
Role Outline View		(i)			
Subject Outline View		(i)			
Policy Simulation View				(i)	
Policy Simulation Result View				(i)	
Decision Contribution View				(i)	
Authorization Analysis View					(i)
Authorization Analysis Result View					(i)
Business Level Policy Editor		(d)		(d)	
Business Level Rule Editor		(d)		(d)	

- Modify a policy she created.
- Delete a policy she created.
- Add a rule to a policy she created.
- Modify a rule she created.
- Remove a rule she created.

The policy reviewing related tasks that a policy author can carry out in this perspective are:

- View any policy and the associated rules in the policy repository.
- Determine the set of rules that are overridden by a specific rule.
- Determine the set of rules that override a specific rule.

6.3.1 Author Authentication

In order to be allowed to use the policy editor, an author has to identify himself to the system, i.e., every user who wants to use the application needs a valid user with username and password (see Figure 6.2). Every user has an associated organization which provides a specific hierarchy level for the policy delegation support. Thereby a user may only create business level policies for his particular hierarchy level while she is not allowed to modify the policies of other users. The policy delegation mechanism is only useable after the particular authors have authenticated themselves to the system. Otherwise anybody could override all other policies by creating a policy on the lowest level.

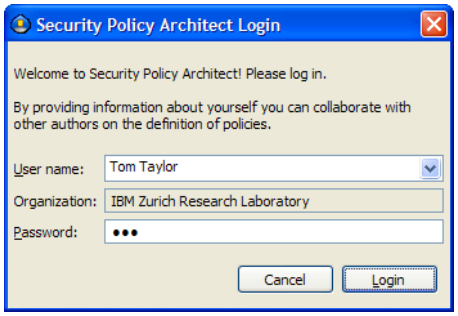


Figure 6.2: Author Authentication

6.3.2 Policy Authoring

Figure 6.3 shows the policy editor in the Policy Authoring and Review Perspective. The left part of the workbench shows the *Policy Explorer View*. The upper right part shows the *editor area* where editors of policies and rules are displayed later on. The lower right part shows the *Rule Outline View* which gives additional information to a displayed policy. Actions in the *global toolbar*,

which is available in every perspective, allow the author to *save* modifications that she made in an editor and to *edit* an element that is currently selected in the policy editor. All the actions that are available in a toolbar or a menu are context aware, i.e., they are grayed out if the corresponding action is not compatible with the current selection or state of the application.

A policy author who wants to create a new policy does this by using the Policy Explorer View. This view displays the policies and rules of the policy repository as tree structure and has a *view toolbar* that allows the author to perform actions on the contained elements. Furthermore, the elements shown in the tree have an associated *context menu* that contains some more specific actions than the toolbar.

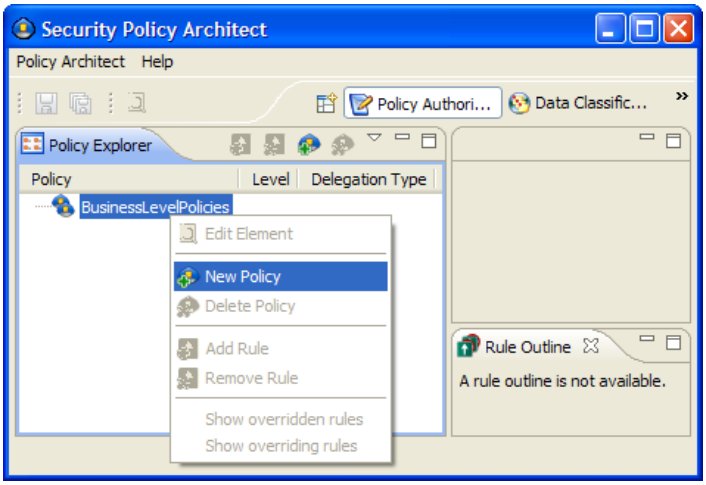


Figure 6.3: Policy Authoring and Review Perspective

After creating a new policy, the Policy Explorer View displays this policy and a *Policy Editor* for the policy is opened in the editor area of the workbench (see Figure 6.4). The editor contains a header area where a general description of a policy is provided so that even users that are not familiar with the tool can get an idea what a business level policy is. An author defines the name, the delegation type — final or recommended — and the combining algorithm of the policy by modifying the corresponding fields in the editor. Conform to the design principle of "fail-safe defaults" [22] the default value for the combining algorithm is "Deny-Overrides". The input field for the combining algorithm is augmented by a descriptive text that explains the currently selected algorithm. The editor shows additional information about the author, the associated organization and the hierarchy level of the policy.

As soon as the editable fields in an editor are modified, this is indicated by a star (*) in the caption of the editor and at the same time the "save" action in the global toolbar is activated. An author uses this action to apply the changes she made in the editor. Otherwise she closes the editor and confirms that she wants to discard the changes.

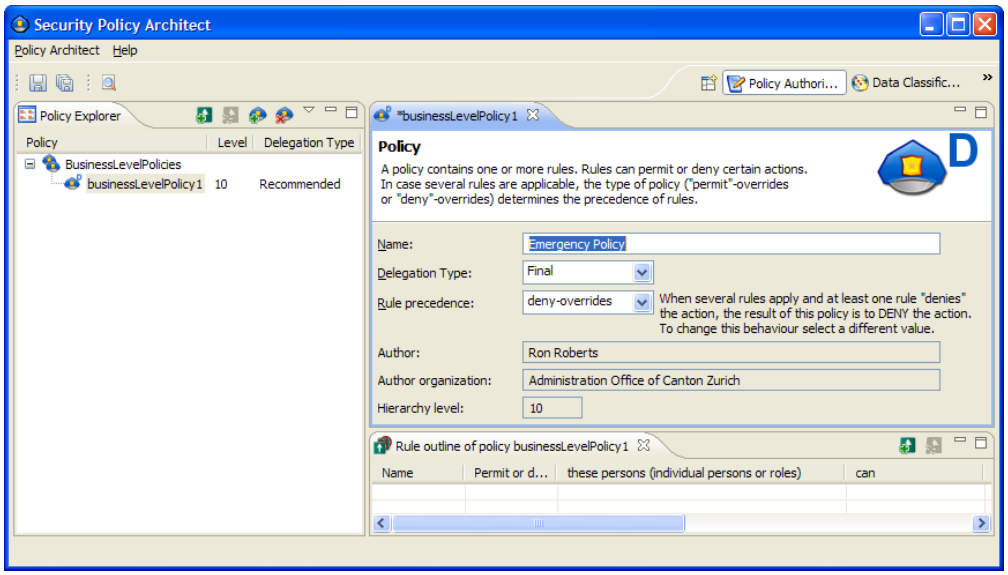


Figure 6.4: Business Level Policy Editor

6.3.3 Rule Authoring

To add a rule to a policy, the policy author uses the corresponding actions in the toolbar or context menu of the Policy Explorer View. The new rule is then displayed in a *Rule Editor* that contains a general description of a rule as well as a set of input fields that are arranged in a way that the policy author can read them as completion exercise[3] (see Figure 6.5). The author is expected to read the completion exercise from top to bottom and fill in the missing parts in order to form a meaningful sentence. By reading this sentence, policy authors can quickly grasp the intentions of the author who created this rule. The policy editor keeps the grammar of the sentence always consistent, i.e., the fillers between the missing parts in the completion exercise change according to the content of the rule. The editor also uses fillers to express the semantics of multiple entries in one of the completion exercise's parts. For example, if an author selects multiple roles for a rule, the filler "and" is used to express that the rule targets persons that have the given roles in conjunction (e.g., "Permit that the person that simultaneously have the roles Security&Assurance *and* LaboratoryAccess can..."). The filler "as well as" is used if an author selects multiple subjects (e.g., "Permit that the persons Alice *as well as* Carol can...").

The mandatory parts of the completion exercise that an author has to define are an effect, subjects or roles, actions as well as resources or labels. Optionally the author can define conditions for the rule. Conform to the design principle of "fail-safe defaults" [22] the default value for the effect is "Deny".

To select the subjects, actions or resources for the rule, the author presses the corresponding button next to the missing part in the completion exercise. Figure 6.6 shows the dialog that requests an author to select the rule's subjects or roles. The left part of the dialog contains the available subjects and roles

[3] In German: Lückentext

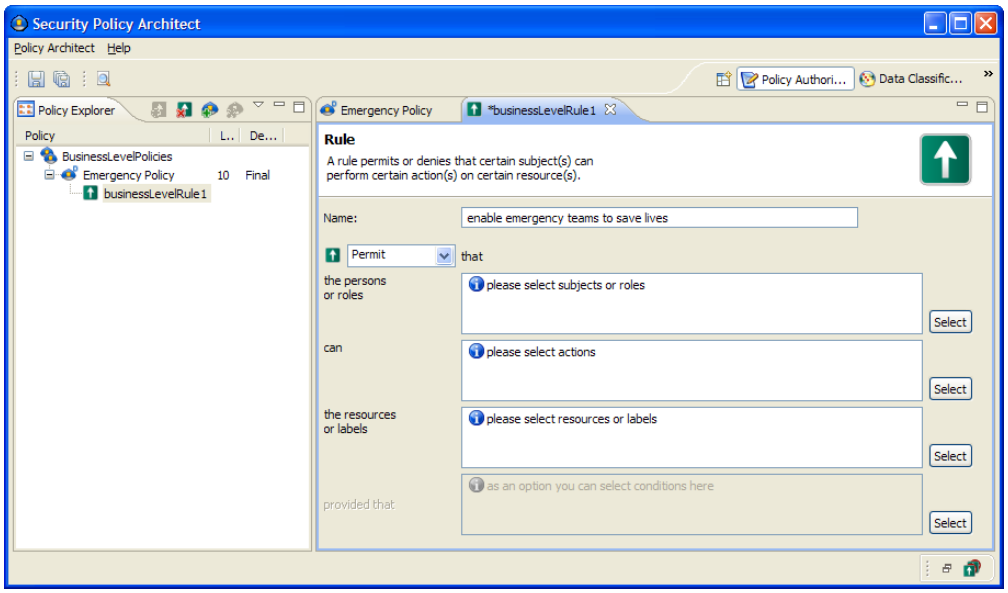


Figure 6.5: Business Level Rule Editor

as well as an entry to address any subject. A search field — that also allows wildcards — supports an author in finding the intended available items quickly since she does not have to navigate through the scheme and role hierarchies any more. The right part contains the list of items that are already selected. With the help of the buttons "add", "remove" and "remove all" that are located between the two parts, the available items can be marked or unmarked as selected. Since a rule allows to address either subjects or roles but not both at the same time, a selection of a subject disallows the selection of a role and vice versa. The items that are already selected as well as the items that are not allowed to be selected together with the already selected ones are grayed out to indicate this fact.

The dialog for selecting actions and the dialog for selecting resources are similar to the one for selecting subjects. Conditions are specified with the button next to the corresponding missing part in the completion exercise (see Figure 6.7). Thereby an owner condition and/or a time range condition can be defined for the rule. While for the owner condition no more information is needed, an author has to specify the begin and the end of the time range in a separate dialog.

As soon as an author has defined all the mandatory parts of the completion exercise — and optionally the conditions as well — she saves the rule by using the "save" action in the global toolbar.

6.3.4 Policy Review

A policy author who wants to do a review of the policies and rules available in the repository may have the intention to review the policy repository as a whole, to review a particular policy and its content or to review a particular rule.

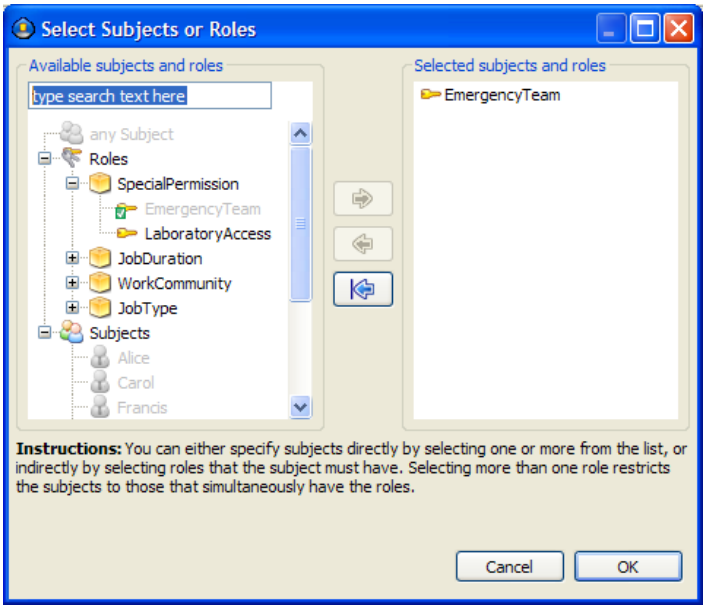


Figure 6.6: Subject Dialog

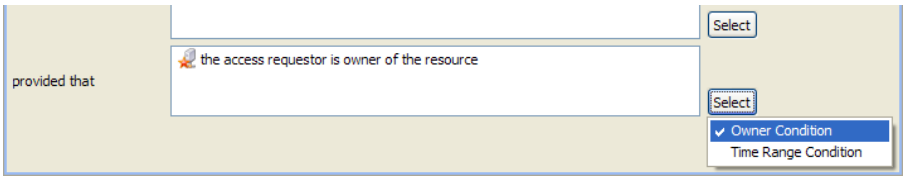
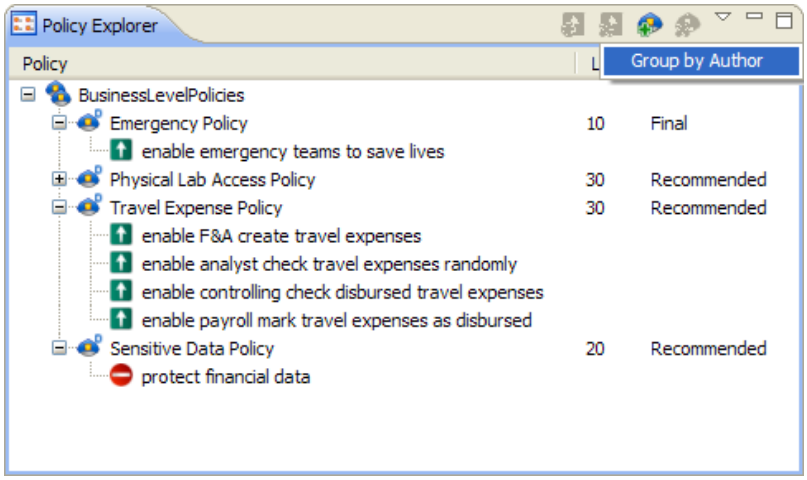
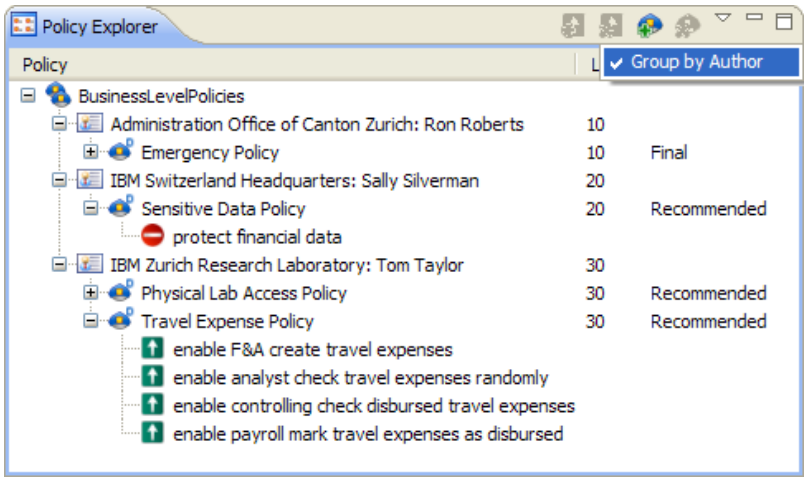


Figure 6.7: Condition Selection

The Policy Explorer View supports an author in reviewing the repository as a whole, and therefore in getting an overall idea of the content of the repository. Thereby the Explorer can display the policies in two different ways. The policies are either displayed in the sequence as they are evaluated by a PDP, i.e., sorted according to the delegation type and the hierarchy level (see Section 5.1.2), or they can be grouped by their author(s). To group the policies by author, the corresponding option has to be activated in the view menu of the Policy Explorer. The type of an element that is shown in the Policy Explorer is indicated by an icon. Different icons indicate that an element is a policy with deny-override combining, a policy with permit-override combining, a deny rule, a permit rule or an author group element. At the same time the hierarchy levels of the policies as well as their delegation types are shown. Figure 6.8 shows the grouping mechanism as well as the different icons.



(a) Evaluation sequence



(b) Group by Author

Figure 6.8: Grouping in Policy Explorer View

Although the Policy Explorer View shows which rules are contained in a pol-

icy, it does not give an overview of the actual contents of these rules. Therefore the *Rule Outline View* was created, which shows such an overview as soon as a policy is displayed in a Policy Editor View (see Figure 6.9). The Rule Outline displays the rules of a policy in a way that they can be read as sentences so that the policy author gets an idea of the policy's content.

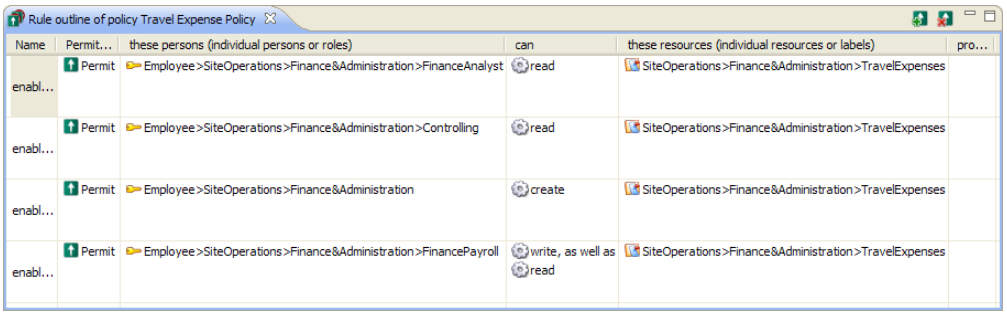


Figure 6.9: Rule Outline

To review a particular policy or a particular rule, the policy author selects the element in the Policy Explorer View and uses the "edit" action in the global toolbar or in the context menu. Then the corresponding Editor, which shows the content of the element, is displayed. If the user that reviews the element is also its author, then the content can be modified as this was shown in the previous section. Otherwise the controls are not editable and the reviewer can only read the element's content.

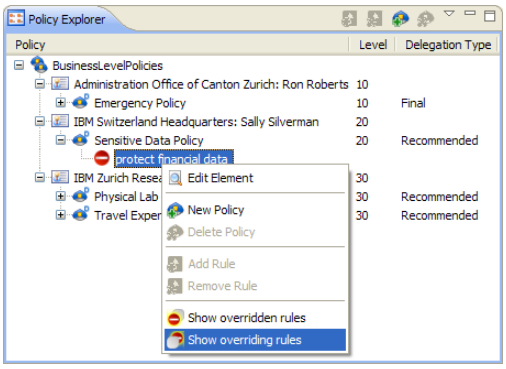
6.3.4.1 Policy Override Detection

For a policy author it might be interesting to know whether the rules in his formulated policy are overridden by rules from another policy author, or if his own rules override the rules of another author. The policy editor's override detection function helps an author to determine which rules in particular override a rule of his interest or which rules get overridden by a rule of his interest.

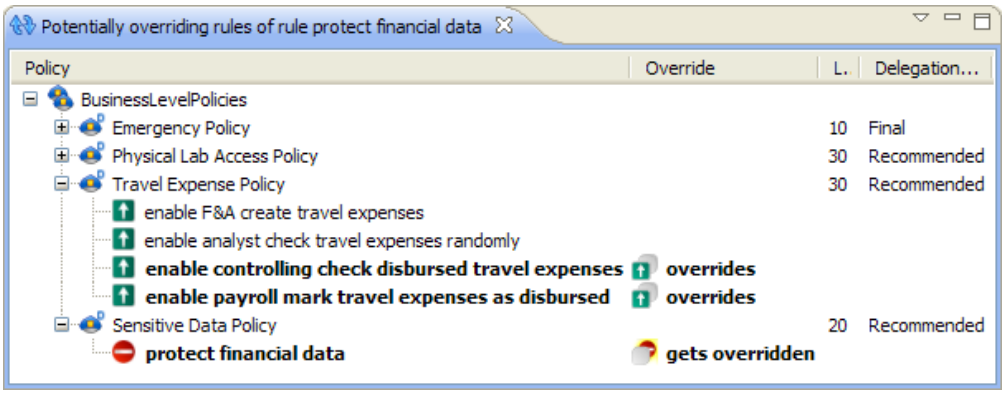
The override detection is available through the context menu of a business level rule in the Policy Explorer View. The policy author selects the desired rule in the Explorer's tree and specifies in the context menu if she is interested in the rules that are overridden by this rule or in the rules that override this rule, i.e., if the rules below or above in the evaluation sequence shall be checked for an override (see Figure 6.10a).

The results of the override detection are then shown in the *Override Detection View*. This view is similar to the Policy Explorer View but contains an additional column to mark the rules that satisfy the given override requirement. Thereby rules that override others are marked with "override", and rules that get overridden are marked with "gets overridden" (see Figure 6.10b). Additionally every element that overrides or is overridden is displayed with bold font.

Recall the policy that was formulated by the IBM Switzerland Headquarters in the policy delegation scenario (see Section 3.10.6). This recommended policy was intended to be overridden by institutions on lower hierarchy levels. The



(a) Show overriding Rules



(b) Override Detection View

Figure 6.10: Policy Override Detection

IBM Switzerland Headquarters uses the policy editor now to see if other policy authors have overridden their recommended policy. This is illustrated in Figure 6.10. There the Override Detection View shows the policies in the evaluation sequence and one can see that two rules override the "protect financial data" rule. Note that since the implementation of override detection just detects potential overrides, the rule "enable analyst check travel expenses randomly" is not identified as override because there exists no subject in the subject repository that has the role "FinancialAnalyst".

6.4 Domain Browsing

Formulating policies on a business level allows a policy author to use various terms in order to express his intention. The terms that can be used comprise all the subjects, roles, actions, labels and resources that exist in the individual repositories whereby some of the terms may also be organized in a hierarchy. Moreover, the assignments between subjects and roles as well as between labels and resources are relevant for the formulation of the policies.

The *Domain Browsing Perspective* of the policy editor supports an author in getting an overview of the whole term domain that she can use for formulating policies. The tasks that can be carried out in this perspective are:

- Browse the non-hierarchic repositories of subjects, and actions.
- Browse the hierarchic repositories of roles, labels and resources whereby the roles and labels are divided into schemes.
- Determine the assignments between subjects and roles and vice versa.
- Determine the assignments between resources and labels and vice versa.

The Domain Browsing Perspective comprises a *Domain Explorer View* and four outline views for labels, resources, roles and subjects. The Domain Explorer View shows the individual repositories as tree structures and comes with a search field on top that also supports wildcards in order to support the author in finding specific elements quickly. While an author browses the repositories in the Domain Explorer View, the outline views adapt according to his selection. She can also browse the entries in one of the outline views and the corresponding outline views adapt accordingly as well.

An author who is for example interested in the roles of the subject "Francis" selects this subject in the Domain Explorer View (see Figure 6.11). The *Role Outline View* adapts to this selection and shows that Francis has the explicit roles "OfficeCommunityIS", "Regular" and "InformationServices" and the implicit roles "Employee" and "SiteOperations" whereby the implicit ones are triggered by the "InformationServices" role. The author is now for example interested in all the subjects that have the role "Regular" and selects this role either in the Domain Explorer View or in the Role Outline View itself. Then the *Subject Outline View* adapts accordingly and shows that "Helen", "Isaac", "Francis" and "George" have this role explicitly and "Bob", "Dave" and "Emily" have it implicitly because they also have the role "ResearchStaffMember".

Similarly an author can select resources in one of the perspective's views and determine thereby the assigned labels, or she selects a label and determines the assigned resources.

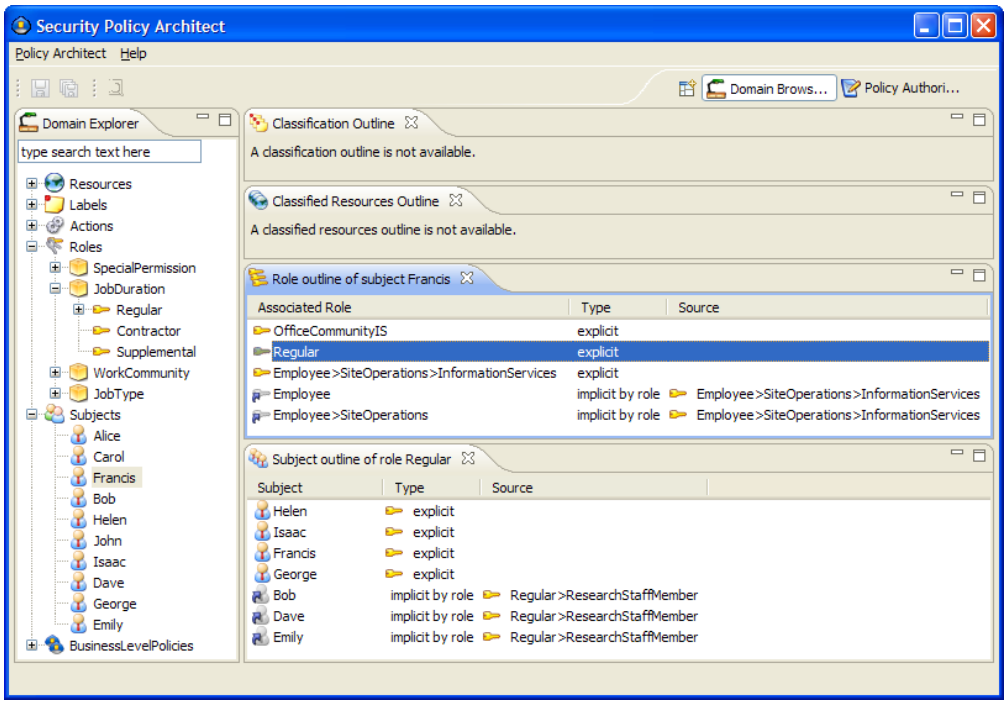


Figure 6.11: Domain Browsing Perspective

6.5 Data Classification

The policy editor can be used to classify resources by labels, i.e., to assign a label that is contained in a classification scheme to a resource. This is done in the *Data Classification Perspective* that contains a *Data Classification View* as well as outline views for labels and resources. The tasks that can be carried out in this perspective are:

- Classify a resource by a label (explicitly).
- Remove an explicit classification from a resource.

The Data Classification View is divided into three parts. The first part shows the labels of the repository as tree structure and contains also a search field. The second part contains buttons for the tasks that can be carried out in the perspective and the third part shows the resources of the repository as tree structure and contains a search field as well. A user has the possibility to display a column in the resource pane that marks all the resources that have a particular label assigned. This is done by activating a checkbox in front of this label in the label pane.

A user who wants to perform one of the possible tasks selects the desired label in the label tree and the desired resource in the resource tree. According to his selection the buttons in the task area are enabled or disabled. The "Add label" button is enabled if the resource *does not have* the label assigned explicitly or inherited, and the "Remove label" button is enabled if the resource *has* the label assigned explicitly.

Figure 6.12 shows the Data Classification Perspective after classifying the resource "TravelExpensesTable" with the "TravelExpenses" label. Because the checkboxes of the "TravelExpenses" and the "SiteOperations" label have been activated by the user, the resource tree shows two additional columns that mark the resources that have these labels assigned. The figure shows that three different icon types are used to indicate that a label is explicit, inherited or implicit.

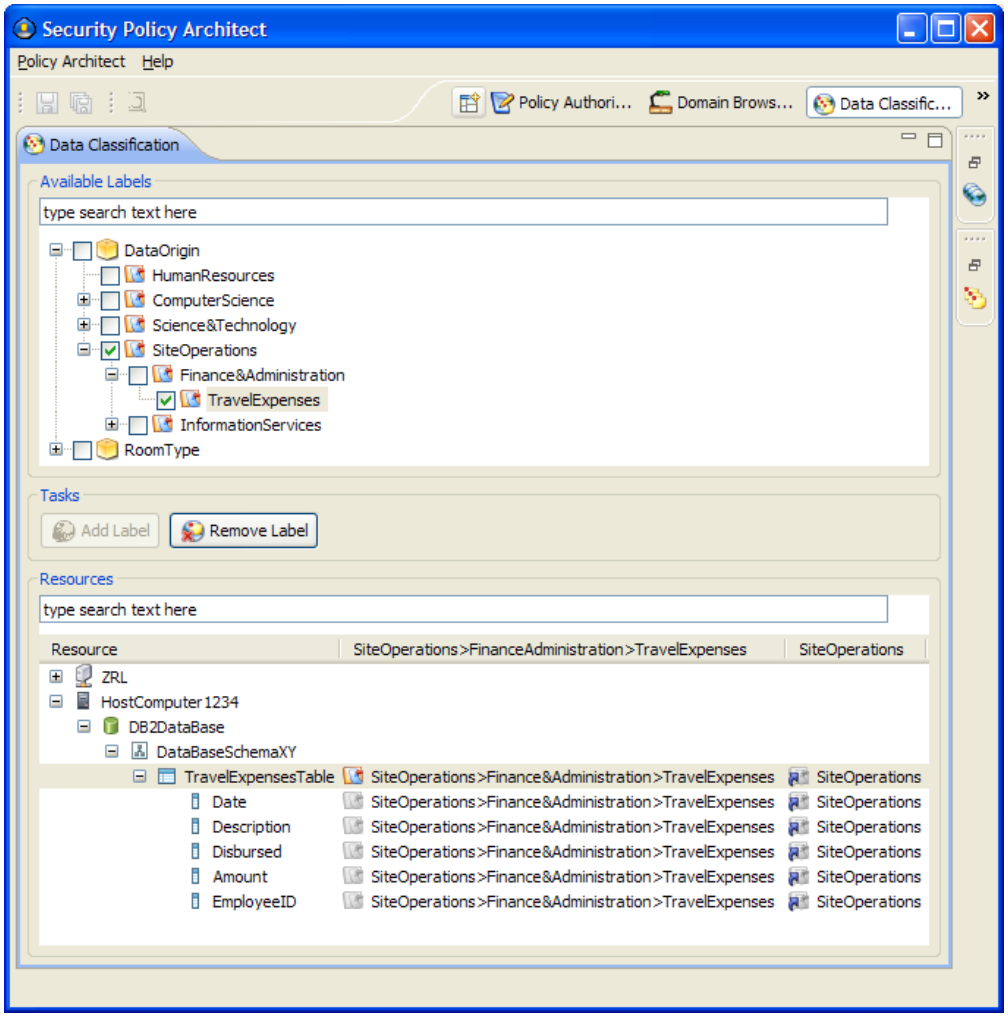
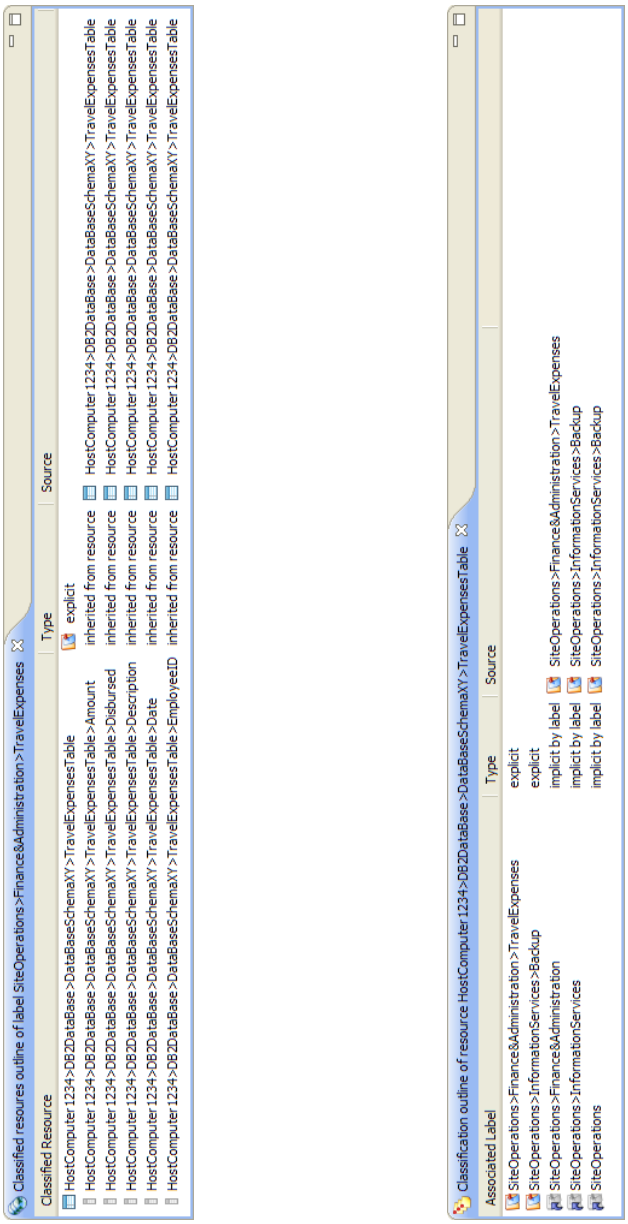


Figure 6.12: Data Classification Perspective

Since the horizontal space on a screen is limited, the Data Classification Perspective also contains the mentioned outline views for labels and resources.

In the figure these views are displayed minimized by placeholders on the right border. The views are adapted accordingly to the selections of the user. The maximized outline views, as they appear with selections of the "TravelExpenses" label and the "TravelExpensesTable" resource in the Data Classification View, are shown in Figure 6.13.



(a) Classified Resources Outline View (b) Classification Outline View

Figure 6.13: Data Classification Outline Views

6.6 Policy Simulation

A policy repository typically contains a big number of policies and rules that have been formulated by multiple authors. Since a single author easily looses track of the current state of the repository she can review it with the help of the Policy Authoring and Review Perspective. However, even if an author got an idea of the policies in the repository, it is not clear how the individual policies and rules cooperate at an evaluation process of a PDP. Therefore the *Policy Simulation Perspective* allows an author to simulate such an evaluation as it would appear in a running system to see if she captured his intentions properly and if a PDP would behave like she expect it. The tasks that can be carried out in this perspective are:

- Issue a decision request and receive an authorization decision from a PDP.
- Perform contribution analysis, i.e., see which rules in particular contribute to a decision for a given request.
- Review the policies and rules that have been discovered in a contribution analysis.

As described earlier, a decision request resides either on XACML level or on a business level. Nevertheless, for an author who wants to issue a request in the Policy Simulation Perspective this is irrelevant. This is because the user interface hides the different levels from the user, she even does not have to know about them.

For issuing a decision request, the Policy Simulation Perspective contains the *Policy Simulation View*. The user interface of this view is very similar to the already described Rule Editor. It contains input fields that are arranged in a way such that a user can read them as a completion exercise, and the user is expected to fill in the missing parts of the exercise to form a meaningful sentence.

The mandatory parts of the completion exercise are subjects or roles, actions, resources or labels, and a point in time for which the request is evaluated. To select these elements, the author presses the corresponding button next to the missing part in the exercise. The request time can either be the current point in time when the user issues the request or a specific one that the user specifies in a separate dialog. After the user presses the "Evaluate" button, the policy editor determines in the background, depending on the inputs of the author, if the given request is on XACML or on a business level and evaluates the corresponding function using an XACML PDP. For example, if a user specifies the subjects in terms of roles then the policy editor evaluates the function $po(\cdot)$. Figure 6.14 illustrates a user input that leads to an evaluation of the function

$$pol(\{FinanceAnalyst\}, read, \{TravelExpenses\}, 9am)$$

and it shows the result of the evaluation of that function. Thereby the *Policy Simulation Result View* shows the result of the evaluation. This view contains

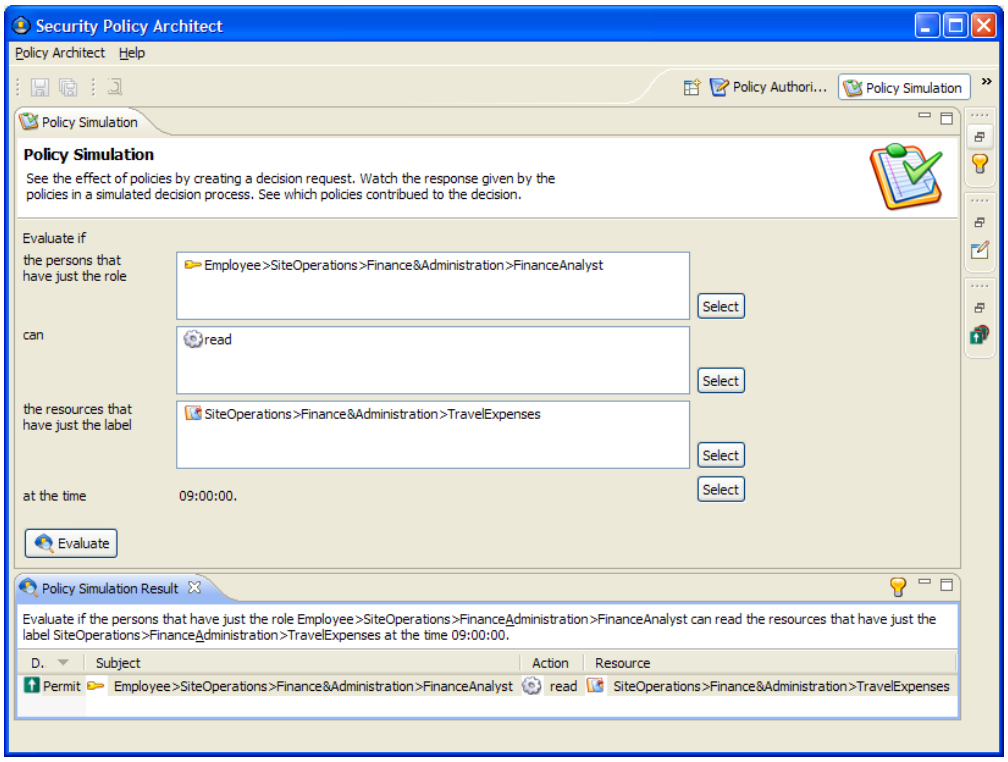


Figure 6.14: Policy Simulation Perspective

a header area, where the given request is repeated in sentence representation, and the result of the evaluation process.

The input fields in the Policy Simulation View allow a user to formulate a sentence that addresses more than one subject, more than one action and more than one resource. Since a single decision request can only comprise one subject, one action and one resource, the policy editor splits the users request in multiple requests and evaluates them independently. In that case the Policy Simulation Result View contains multiple result lines where each line represents an individual authorization decision. For example, a user that formulates the sentence "Evaluate if the persons Alice, as well as Carol can..." is split into two requests, one for Alice and one for Carol. A sentence like "Evaluate if any Subject can..." is split into individual requests for all subjects in the repository. The same methods are used to resolve multiple actions and multiple resources.

6.6.1 Contribution Analysis

To perform contribution analysis on a particular authorization decision that is shown in the Policy Simulation Result View, a user selects the corresponding line in the view and uses the "Authorization Analysis" button in the view toolbar or in the context menu of the entry. The results of the contribution analysis are then shown in the *Decision Contribution View*. This view is similar to the Policy Explorer View but contains an additional column to mark the rules that contribute to the given authorization decision. Thereby rules that evaluated to

a permit are marked with "Permit" and the ones that evaluated to a deny are marked with "Deny". The policies that contain contributing rules are marked with the evaluation result of the corresponding combining algorithm in the same way. Additionally every element that contributes to the decision is displayed with bold font.

Figure 6.15 shows the results of a contribution analysis for the authorization decision of the decision request given in the last section. Thereby the "Travel Expense Policy" and the "Sensitive Data Policy" contribute with a "Permit" and a "Deny" decision. Since the "Travel Expense Policy" is the first policy in the evaluation sequence that gives a decision, this decision is also the overall decision and the "Sensitive Data Policy" is overridden.

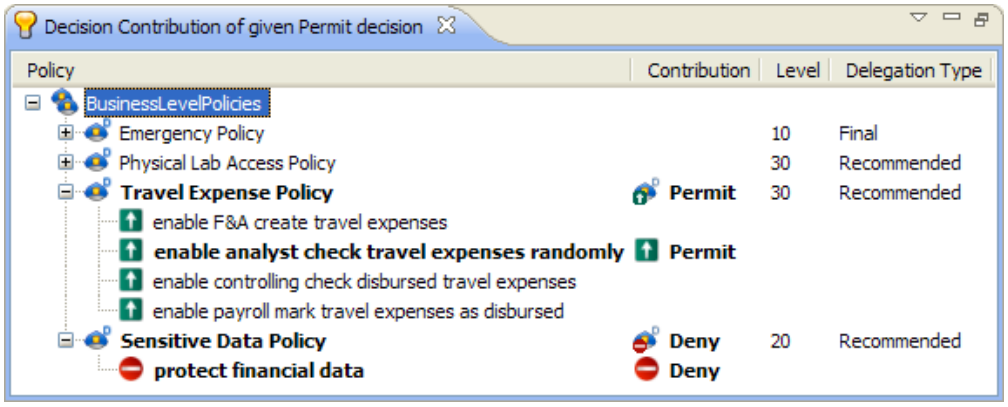


Figure 6.15: Contribution Analysis View

6.7 Authorization Analysis

With authorization analysis a user can determine what the values of certain parameters of a decision request must be, in order to receive a permit decision. An author can use the policy editor to determine the values for the role set parameter (defined in Section 4.5.2) and the values for the label set parameter (defined in Section 4.5.5) in order to receive a permit decision. Therefore a user can ask questions of the form "Which roles has a subject exactly to hold to..." and "Which labels has a resource exactly to have so that...".

A user does not have to specify or even know which of the sets that are defined in the mentioned sections she wants to determine. She just formulates his question with the user interface as meaningful sentence (see Figure 6.16), and the the policy editor determines in the background which set corresponds to the given question. As in the Policy Editor and the Policy Simulation View, the user interface is designed in a way that the user can read the input fields as completion exercise and formulate his question by specifying the variable and missing parts.

An author who wants to formulate a question uses the *Authorization Analysis View* to define the first part of his question which can either be "Which roles..." or "Which labels...". According to his decision the user interface adapts its

appearance since the word order in these two types of questions is not the same. In case the user is interested in roles, she has to define an action and resources or labels. In case she is interested in labels, she has to define subjects or roles as well as an action. In both cases it has to be defined for which point in time the request has to be evaluated. To select the missing parts of the completion exercise the author presses the corresponding button next to it. Before a user can start the analysis process, she has to define in which size of result sets she is interested in, i.e., what the highest number of role or label combinations is that the policy editor shall check for permit decisions.

Figure 6.16 shows the user input that represents the question "Which roles has a subject exactly to hold to be permitted to "enter" resources that are labeled with exactly the labels "Security&Assurance" and "ConfidentialPrinterRoom" at 9a.m.?" whereby we are interested in role sets up to the size of two.

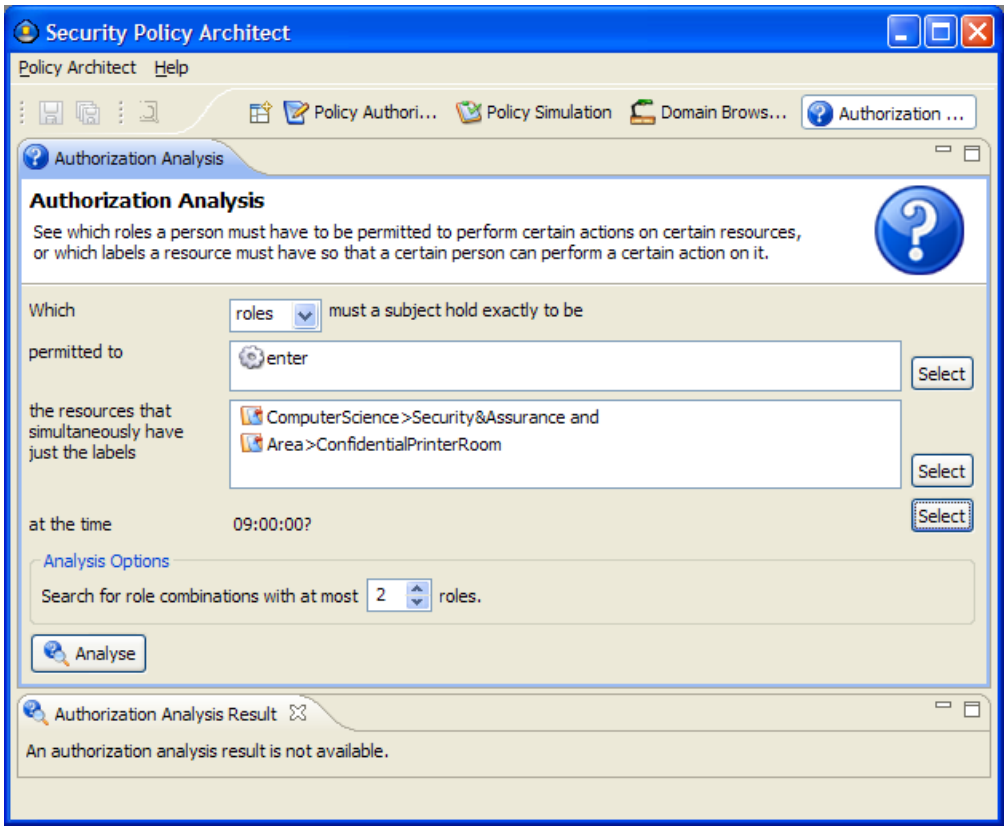


Figure 6.16: Authorization Analysis Perspective

The *Authorization Analysis Result View* shows the result of the analysis after pressing the "Analysis" button. The view contains a header area where the given question that was evaluated by the PDP is repeated. Furthermore it displays the list of role sets or label sets, that result in a permit decision from the PDP for the given question. Figure 6.17 shows the result of the question that was given in the last paragraph. All the results that are shown in the figure would result in a "Permit" decision if an author used the Policy Simulation View with the corresponding values. Since we specified we are interested in result sets

up to the size of two elements, we received result sets with one or two elements. In our example we got a number of results with two elements, and one result with one element. The first element in the list — which is highlighted in blue — is found because of a concrete rule in the policy repository that permits the access for this role set. All the other results are found because subjects with the "EmergencyTeam" role are permitted to enter all areas.

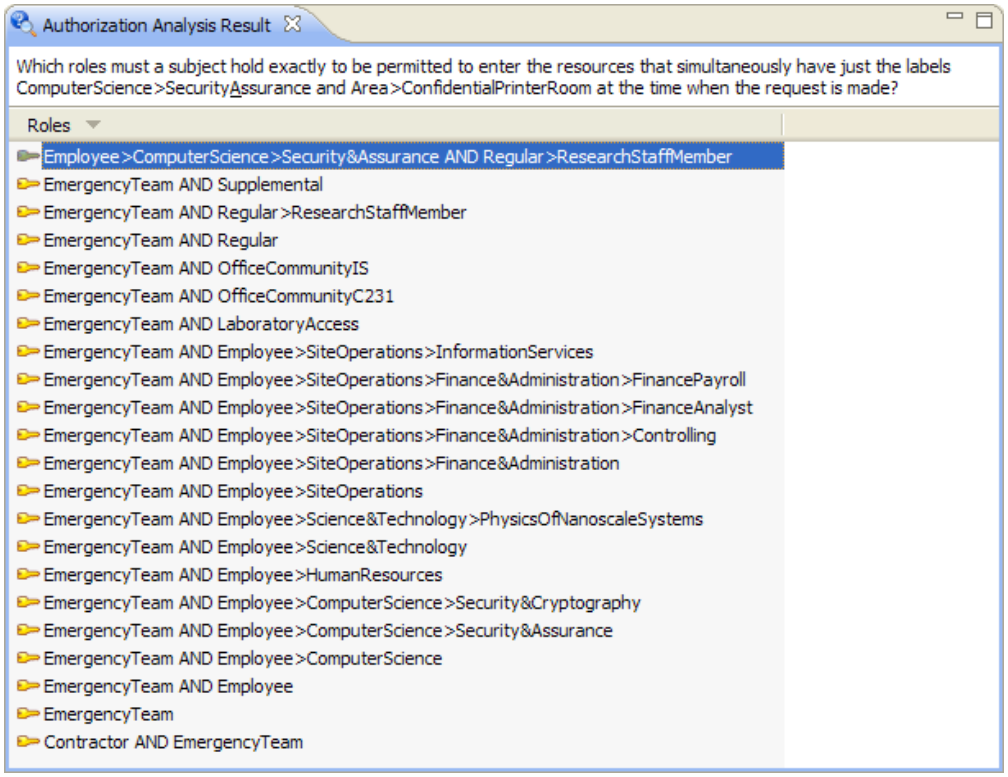


Figure 6.17: Authorization Analysis Result View

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In the security domain different models to formulate access control policies have been developed and studied throughout the last years. However, all these models use concepts that are hard to understand for non-technical policy authors and furthermore the authoring of the corresponding policies is not supported by user-friendly and easy-to-use tools. This thesis proposes a policy language on a higher abstraction level that leverages business terms to formulate access control behavior and provides analysis as well as tool support for the authoring of policies formulated in this language. The abstraction is achieved by addressing subjects in terms of roles and by using data classification to attach business meaning to resources. In order to deploy the business level policies in existing IT infrastructure, they can be translated to the standardized XACML access control policy language.

Up to now there were no approaches of formulating policies on such a business level. A survey in the IBM Zurich Research Laboratory has shown that there are no policies on this abstraction level available so far, which makes a communication between the high level executives who are responsible for the access control policies and the actual authors of the policies difficult. At the moment there are only policies on an even higher level than the one used in this thesis (like recommendations, guidelines, etc.), and policies on a lower, more technical level. Our policy language on business level closes therefore the existing gap between these levels.

We found that the used concept of data classification appears to be very intuitive and natural, compared to access control models in which policies are tied to namespaces. It does no longer matter where the resources to protect reside in the namespace since the policies address the resources now by their labels.

Due to the hierarchical structure of roles, labels and resources, various advantages in the formulation of policies can be achieved. The hierarchies of roles and labels reduce the number of policies that have to be formulated. Furthermore the resource hierarchy enables an individual degree of granularity in data

classification. While the number of policies to be formulated is reduced, the size of a policy in its XACML representation can become very big if the used role and label hierarchies are large.

The proposed policy delegation mechanism allows authors to formulate policies throughout multiple levels in an organization that is structured in a hierarchy. This offers the possibility to let every policy be formulated by the person who is responsible for it, without the need to communicate the policy to technical employees. However, policy authors who use the delegation mechanism have to agree on common role and label schemes they use in their policies.

With respect to policy analysis we found that a number of analysis algorithms with value for the policy authors can be formulated. The analysis algorithms we propose support an policy author in issuing a decision request on business level, in determining the individual policies and rules that contribute to an authorization decision, in detecting overrides between policies, in determining the values she has to provide in order to receive a specific authorization decision and in identifying gaps in the access control behavior that prevent subjects from accessing in any resource and resources from being accessed by any subject.

The results of a decision request on business level as well as of the policy override detection are subject to certain restrictions that have to be kept in mind when performing the analysis. For a decision request on business level that targets roles instead of subjects it has to be considered that the given answer is only valid for subjects that (1) hold the given role sets *exactly*, (2) for that no rules exist that target the subject *directly* and (3) that are not owner of the requested resource. Similar restrictions are put on the results of requests that target labels instead of resources. For the results of policy override detection it has to be kept in mind that just overrides are detected that are (1) *currently present* — whereby the presence is dependent on the policy repository, the role assignment and the data classification — and that are (3) *potential* because the conditions of the rules are not considered.

For the development of the tool to support policy authors in formulating their policies, we could see that the consideration of the design principles described in Section 2 is a very good starting point. While keeping them in mind we could develop an graphically appealing and user friendly policy editor that is kept small and simple but nevertheless powerful enough to let authors formulate real world policies in a very intuitive and easy way.

7.2 Future Work

With time as restricting factor, not all ideas that came up during the work on this thesis could be explored or even implemented. In the following an unordered list captures some of the unimplemented ideas we had over the last six months as well as possible improvements of analysis algorithms and the policy editor.

– A *more efficient policy translation algorithm.* At the moment only the entire business level policy can be translated. For big policies an improvement in time could be made by an incremental translation approach where

just new or modified policies are translated.

- A *new policy editor perspective that focuses on one specific subject* and where a user can obtain all kind of information that is related to that single subject. For example, if a specific subject is exposed as crook, this perspective shall enable to identify the resources as well as the labels of these resources that might be compromised. A similar perspective would also be possible for resources.

- New policy editor functions to (1) *export the current policy* in XACML format to enforce it in an IT infrastructure and to (2) allow the *import and export of repositories* for labels, roles, resources, etc.

- A *change impact analysis* algorithm that determines the changes or side effects in access behavior for certain actions. For example, which subjects would have access to a resource if this resource is classified with a particular label.

- Use *more expressive and powerful OWL ontologies* to describe the role and label repositories. This would open up new methods to formulate and analyze the policy repository.

- A *conflict analysis* algorithm that identifies potential inconsistencies of access control behavior within one specific business level policy (the conflicts throughout multiple policies are resolved by the policy delegation mechanism).

- A *time range analysis* algorithm that allows to determine the time ranges for which different access control behavior is stated. The authorization decisions that are given with the current implementation are only valid for one specific point of time. The time range analysis could help in loosening this restriction.

Appendix A

Example XACML Policy

```
1  <PolicySet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="
       urn:oasis:names:tc:xacml:2.0:policy:schema:os" xsi:schemaLocation="
       urn:oasis:names:tc:xacml:2.0:policy:schema:os access_control-xacml
       -2.0-policy-schema-os.xsd" PolicyCombiningAlgId="urn:oasis:names:tc:
       xacml:1.0:policy-combining-algorithm:first-applicable" PolicySetId="
       urn:zurich:ibm:names:xacml:2.0:policysetid:1">
2    <Description>Contained policies are ordered according to their
         hierarchy level and final type</Description>
3    <Target/>
4    <Policy PolicyId="urn:zurich:ibm:names:xacml:2.0:policyid:1"
         RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
         algorithm:deny-overrides">
5      <Description>Corresponds to business level policy the name 'Physical
           Lab Access Policy'</Description>
6      <Target/>
7      <Rule Effect="Permit" RuleId="urn:zurich:ibm:names:xacml:2.0:ruleid
           :1">
8        <Description>Corresponds to business level rule with id 'http://www
             .example.org/businessLevelRule#rule0003' and name 'enable
             owners enter their rooms'</Description>
9        <Target>
10         <Subjects>
11           <Subject>
12             <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                   string-equal">
13               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                     string">http://www.example.org/role#employee</
                     AttributeValue>
14               <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                     :xacml:2.0:subject:role" DataType="http://www.w3.org
                     /2001/XMLSchema#string"/>
15             </SubjectMatch>
16           </Subject>
17           <Subject>
18             <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                   string-equal">
19               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                     string">http://www.example.org/role#computerScience</
                     AttributeValue>
20               <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                     :xacml:2.0:subject:role" DataType="http://www.w3.org
                     /2001/XMLSchema#string"/>
21             </SubjectMatch>
22           </Subject>
23           <Subject>
24             <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                   string-equal">
```

```
25              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                    string">http://www.example.org/role#
                    scienceAndTechnology</AttributeValue>
26              <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                    :xacml:2.0:subject:role" DataType="http://www.w3.org
                    /2001/XMLSchema#string"/>
27            </SubjectMatch>
28          </Subject>
29          <Subject>
30            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                  string-equal">
31              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                    string">http://www.example.org/role#informationServices
                    </AttributeValue>
32              <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                    :xacml:2.0:subject:role" DataType="http://www.w3.org
                    /2001/XMLSchema#string"/>
33            </SubjectMatch>
34          </Subject>
35          <Subject>
36            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                  string-equal">
37              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                    string">http://www.example.org/role#
                    physicsOfNanoscaleSystems</AttributeValue>
38              <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                    :xacml:2.0:subject:role" DataType="http://www.w3.org
                    /2001/XMLSchema#string"/>
39            </SubjectMatch>
40          </Subject>
41          <Subject>
42            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                  string-equal">
43              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                    string">http://www.example.org/role#siteOperations</
                    AttributeValue>
44              <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                    :xacml:2.0:subject:role" DataType="http://www.w3.org
                    /2001/XMLSchema#string"/>
45            </SubjectMatch>
46          </Subject>
47          <Subject>
48            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                  string-equal">
49              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                    string">http://www.example.org/role#
                    securityAndAssurance</AttributeValue>
50              <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                    :xacml:2.0:subject:role" DataType="http://www.w3.org
                    /2001/XMLSchema#string"/>
51            </SubjectMatch>
52          </Subject>
53          <Subject>
54            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                  string-equal">
55              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                    string">http://www.example.org/role#financeAnalyst</
                    AttributeValue>
56              <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                    :xacml:2.0:subject:role" DataType="http://www.w3.org
                    /2001/XMLSchema#string"/>
57            </SubjectMatch>
58          </Subject>
59          <Subject>
60            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                  string-equal">
```

```
61              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                    string">http://www.example.org/role#humanResources</
                    AttributeValue>
62              <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                    :xacml:2.0:subject:role" DataType="http://www.w3.org
                    /2001/XMLSchema#string"/>
63            </SubjectMatch>
64          </Subject>
65          <Subject>
66            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                  string-equal">
67              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                    string">http://www.example.org/role#
                    securityAndCryptography</AttributeValue>
68              <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                    :xacml:2.0:subject:role" DataType="http://www.w3.org
                    /2001/XMLSchema#string"/>
69            </SubjectMatch>
70          </Subject>
71          <Subject>
72            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                  string-equal">
73              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                    string">http://www.example.org/role#controlling</
                    AttributeValue>
74              <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                    :xacml:2.0:subject:role" DataType="http://www.w3.org
                    /2001/XMLSchema#string"/>
75            </SubjectMatch>
76          </Subject>
77          <Subject>
78            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                  string-equal">
79              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                    string">http://www.example.org/role#financePayroll</
                    AttributeValue>
80              <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                    :xacml:2.0:subject:role" DataType="http://www.w3.org
                    /2001/XMLSchema#string"/>
81            </SubjectMatch>
82          </Subject>
83          <Subject>
84            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                  string-equal">
85              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                    string">http://www.example.org/role#
                    financeAndAdministration</AttributeValue>
86              <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                    :xacml:2.0:subject:role" DataType="http://www.w3.org
                    /2001/XMLSchema#string"/>
87            </SubjectMatch>
88          </Subject>
89        </Subjects>
90        <Resources>
91          <Resource>
92            <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                  :string-equal">
93              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                    string">http://www.example.org/label#area</
                    AttributeValue>
94              <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                    names:xacml:2.0:resource:label" DataType="http://www.w3
                    .org/2001/XMLSchema#string"/>
95            </ResourceMatch>
96          </Resource>
97          <Resource>
```

```
98              <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                    :string-equal">
99                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/label#lounge</
                      AttributeValue>
100               <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                      names:xacml:2.0:resource:label" DataType="http://www.w3
                      .org/2001/XMLSchema#string"/>
101             </ResourceMatch>
102           </Resource>
103           <Resource>
104             <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                    :string-equal">
105               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/label#office</
                      AttributeValue>
106               <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                      names:xacml:2.0:resource:label" DataType="http://www.w3
                      .org/2001/XMLSchema#string"/>
107             </ResourceMatch>
108           </Resource>
109           <Resource>
110             <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                    :string-equal">
111               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/label#
                      confidentialPrinterRoom</AttributeValue>
112               <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                      names:xacml:2.0:resource:label" DataType="http://www.w3
                      .org/2001/XMLSchema#string"/>
113             </ResourceMatch>
114           </Resource>
115           <Resource>
116             <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                    :string-equal">
117               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/label#laboratory</
                      AttributeValue>
118               <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                      names:xacml:2.0:resource:label" DataType="http://www.w3
                      .org/2001/XMLSchema#string"/>
119             </ResourceMatch>
120           </Resource>
121           <Resource>
122             <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                    :string-equal">
123               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/label#roomWithServer</
                      AttributeValue>
124               <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                      names:xacml:2.0:resource:label" DataType="http://www.w3
                      .org/2001/XMLSchema#string"/>
125             </ResourceMatch>
126           </Resource>
127           <Resource>
128             <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                    :string-equal">
129               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/label#printerRoom</
                      AttributeValue>
130               <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                      names:xacml:2.0:resource:label" DataType="http://www.w3
                      .org/2001/XMLSchema#string"/>
131             </ResourceMatch>
132           </Resource>
133           <Resource>
```

```
134             <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                    :string-equal">
135               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/label#conferenceRoom</
                      AttributeValue>
136               <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                      names:xacml:2.0:resource:label" DataType="http://www.w3
                      .org/2001/XMLSchema#string"/>
137             </ResourceMatch>
138           </Resource>
139         </Resources>
140         <Actions>
141           <Action>
142             <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                    string-equal">
143               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/action#enter</
                      AttributeValue>
144               <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:
                      xacml:1.0:action:action-id" DataType="http://www.w3.org
                      /2001/XMLSchema#string"/>
145             </ActionMatch>
146           </Action>
147         </Actions>
148       </Target>
149       <Condition>
150         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
151           <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string
                  -equal">
152             <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
                    string-one-and-only">
153               <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                      :xacml:1.0:subject:subject-id" DataType="http://www.w3.
                      org/2001/XMLSchema#string"/>
154             </Apply>
155             <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
                    string-one-and-only">
156               <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                      names:xacml:2.0:resource:owner" DataType="http://www.w3
                      .org/2001/XMLSchema#string"/>
157             </Apply>
158           </Apply>
159           <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:not">
160             <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
                    string-equal">
161               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string"></AttributeValue>
162               <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
                      string-one-and-only">
163                 <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                        names:xacml:2.0:resource:owner" DataType="http://www.
                        w3.org/2001/XMLSchema#string"/>
164               </Apply>
165             </Apply>
166           </Apply>
167         </Apply>
168       </Condition>
169     </Rule>
170     <Rule Effect="Permit" RuleId="urn:zurich:ibm:names:xacml:2.0:ruleid
          :2">
171       <Description>Corresponds to business level rule with id 'http://www
            .example.org/businessLevelRule#rule0011' and name 'enable
            laboratory users of S&amp;amp;A enter their labs'</Description>
172       <Target>
173         <Subjects>
174           <Subject>
```

```
175             <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                    string-equal">
176               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/role#
                      securityAndAssurance</AttributeValue>
177               <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                      :xacml:2.0:subject:role" DataType="http://www.w3.org
                      /2001/XMLSchema#string"/>
178             </SubjectMatch>
179             <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                    string-equal">
180               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/role#laboratoryAccess</
                      AttributeValue>
181               <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                      :xacml:2.0:subject:role" DataType="http://www.w3.org
                      /2001/XMLSchema#string"/>
182             </SubjectMatch>
183           </Subject>
184         </Subjects>
185         <Resources>
186           <Resource>
187             <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                    :string-equal">
188               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/label#laboratory</
                      AttributeValue>
189               <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                      names:xacml:2.0:resource:label" DataType="http://www.w3
                      .org/2001/XMLSchema#string"/>
190             </ResourceMatch>
191             <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                    :string-equal">
192               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/label#
                      securityAndAssurance</AttributeValue>
193               <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                      names:xacml:2.0:resource:label" DataType="http://www.w3
                      .org/2001/XMLSchema#string"/>
194             </ResourceMatch>
195           </Resource>
196         </Resources>
197         <Actions>
198           <Action>
199             <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                    string-equal">
200               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/action#enter</
                      AttributeValue>
201               <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:
                      xacml:1.0:action:action-id" DataType="http://www.w3.org
                      /2001/XMLSchema#string"/>
202             </ActionMatch>
203           </Action>
204         </Actions>
205       </Target>
206     </Rule>
207     <Rule Effect="Deny" RuleId="urn:zurich:ibm:names:xacml:2.0:ruleid:3">
208       <Description>Corresponds to business level rule with id 'http://www
              .example.org/businessLevelRule#rule0004' and name 'contractors
              only during the day'</Description>
209       <Target>
210         <Subjects>
211           <Subject>
212             <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                    string-equal">
```

```
213                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                        string">http://www.example.org/role#contractor</
                        AttributeValue>
214                 <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc
                        :xacml:2.0:subject:role" DataType="http://www.w3.org
                        /2001/XMLSchema#string"/>
215               </SubjectMatch>
216             </Subject>
217           </Subjects>
218           <Resources>
219             <Resource>
220               <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                      :string-equal">
221                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                        string">http://www.example.org/label#area</
                        AttributeValue>
222                 <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                        names:xacml:2.0:resource:label" DataType="http://www.w3
                        .org/2001/XMLSchema#string"/>
223               </ResourceMatch>
224             </Resource>
225             <Resource>
226               <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                      :string-equal">
227                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                        string">http://www.example.org/label#lounge</
                        AttributeValue>
228                 <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                        names:xacml:2.0:resource:label" DataType="http://www.w3
                        .org/2001/XMLSchema#string"/>
229               </ResourceMatch>
230             </Resource>
231             <Resource>
232               <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                      :string-equal">
233                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                        string">http://www.example.org/label#office</
                        AttributeValue>
234                 <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                        names:xacml:2.0:resource:label" DataType="http://www.w3
                        .org/2001/XMLSchema#string"/>
235               </ResourceMatch>
236             </Resource>
237             <Resource>
238               <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                      :string-equal">
239                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                        string">http://www.example.org/label#
                        confidentialPrinterRoom</AttributeValue>
240                 <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                        names:xacml:2.0:resource:label" DataType="http://www.w3
                        .org/2001/XMLSchema#string"/>
241               </ResourceMatch>
242             </Resource>
243             <Resource>
244               <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                      :string-equal">
245                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                        string">http://www.example.org/label#laboratory</
                        AttributeValue>
246                 <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                        names:xacml:2.0:resource:label" DataType="http://www.w3
                        .org/2001/XMLSchema#string"/>
247               </ResourceMatch>
248             </Resource>
249             <Resource>
```

```
250             <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                    :string-equal">
251               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/label#roomWithServer</
                      AttributeValue>
252               <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                      names:xacml:2.0:resource:label" DataType="http://www.w3
                      .org/2001/XMLSchema#string"/>
253             </ResourceMatch>
254           </Resource>
255           <Resource>
256             <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                    :string-equal">
257               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/label#printerRoom</
                      AttributeValue>
258               <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                      names:xacml:2.0:resource:label" DataType="http://www.w3
                      .org/2001/XMLSchema#string"/>
259             </ResourceMatch>
260           </Resource>
261           <Resource>
262             <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function
                    :string-equal">
263               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/label#conferenceRoom</
                      AttributeValue>
264               <ResourceAttributeDesignator AttributeId="urn:zurich:ibm:
                      names:xacml:2.0:resource:label" DataType="http://www.w3
                      .org/2001/XMLSchema#string"/>
265             </ResourceMatch>
266           </Resource>
267         </Resources>
268         <Actions>
269           <Action>
270             <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:
                    string-equal">
271               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
                      string">http://www.example.org/action#enter</
                      AttributeValue>
272               <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:
                      xacml:1.0:action:action-id" DataType="http://www.w3.org
                      /2001/XMLSchema#string"/>
273             </ActionMatch>
274           </Action>
275         </Actions>
276       </Target>
277       <Condition>
278         <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:time-in-
                  range">
279           <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-
                    one-and-only">
280             <EnvironmentAttributeDesignator AttributeId="urn:oasis:names:
                    tc:xacml:1.0:environment:current-time" DataType="http://
                    www.w3.org/2001/XMLSchema#time"/>
281           </Apply>
282           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time
                  ">20:00:00</AttributeValue>
283           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time
                  ">06:00:00</AttributeValue>
284         </Apply>
285       </Condition>
286     </Rule>
287   </Policy>
288 </PolicySet>
```

Bibliography

[1] ANSI INCITS 359-2004. American National Standard for Information Technology, 2004.

[2] D. Beckett. RDF/XML Syntax Specification (Revised). W3C Recommendation, 10 February 2004. `http://www.w3.org/TR/rdf-syntax-grammar/` [cited 2007 August].

[3] D. Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, 10 February 2004. `http://www.w3.org/TR/rdf-schema/` [cited 2007 August].

[4] S. Brostoff, M.A. Sasse, D. Chadwick, J. Cunningham, U. Mbanaso, and S. Otenko. "R-what?" Development of a role-based access control policy-writing tool for e-Scientists. *Software- Practice & Experience*, 35(9):835–856, 2005.

[5] X. Cao and L. Iverson. Intentional access management: making access control usable for end-users. *Proceedings of the second symposium on Usable privacy and security*, pages 20–31, 2006.

[6] D. Ferraiolo and R. Kuhn. Role-based access control. In *15th NIST-NCSC National Computer Security Conference*, 1992.

[7] K. Fisler, Krishnamurthi S., Meyerovich L.A., and Tschantz M.C. Margrave. An API for XACML Policy Verification and Change Analysis. `http://www.cs.brown.edu/research/plt/software/margrave/` [cited 2007 August].

[8] Organization for the Advancement of Structured Information Standards. XACML Products and Deployments. `http://docs.oasis-open.org/xacml/xacmlRefs.html#Products` [cited 2007 August].

[9] Organization for the Advancement of Structured Information Standards. Core and hierarchical role based access control (RBAC) profile of XACML Version 2.0, 2005. Available from: `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf` [cited 2007 August].

[10] Organization for the Advancement of Structured Information Standards. eXtensible Access Control Markup Language (XACML) Version 2.0,

2005. Available from: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf [cited 2007 August].

[11] Organization for the Advancement of Structured Information Standards. Hierarchical resource profile of XACML Version 2.0, 2005. Available from: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-hier-profile-spec-os.pdf [cited 2007 August].

[12] Google. Google Web Toolkit - Build AJAX apps in the Java language. http://code.google.com/webtoolkit/ [cited 2007 August].

[13] M. Graf and M. Swimmer. Data Centric Security. IBM internal slide set, 2007.

[14] IBM. P3P Policy Editor. http://www.alphaworks.ibm.com/tech/p3peditor/ [cited 2007 August].

[15] G. Klyne and J.J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, 10 February 2004. http://www.w3.org/TR/rdf-concepts/ [cited 2007 August].

[16] HP Labs. Jena - A Semantic Web Framework for Java. http://jena.sourceforge.net/ [cited 2007 August].

[17] F. Manola and E. Miller. RDF Primer. W3C Recommendation, 10 February 2004. http://www.w3.org/TR/rdf-primer/ [cited 2007 August].

[18] D.L. McGuinness and F. Van Harmelen. OWL Web Ontology Language. W3C Recommendation, 10 February 2004. http://www.w3.org/TR/owl-features/ [cited 2007 August].

[19] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Candidate Recommendation 14 June 2007. http://www.w3.org/TR/rdf-sparql-query/ [cited 2007 August].

[20] G. Reif. *WEESA - Web Engineering for Semantic Web Applications*. PhD thesis, TU Wien, 2005.

[21] W. Rosenberry, D. Kenney, and G. Fisher. *Understanding DCE*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1992.

[22] J.H. Saltzer and M.D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63:1278–1308, 1975.

[23] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[24] M.E. Zurko, R. Simon, and T. Sanfilippo. A User-Centered, Modular Authorization Service Built on an RBAC Foundation. *IEEE Symposium on Security and Privacy*, pages 57–71, 1999.

[25] M.E. Zurko and R.T. Simon. User-centered security. *Proceedings of the 1996 workshop on New security paradigms*, pages 27–33, 1996.