

A Model-driven Approach to enable Access Control for Ontologies

Willy Chen and Heiner Stuckenschmidt 1)

Abstract

Semantic Technologies, in particular ontologies are becoming more and more relevant for practical applications. Typical use cases for ontologies in industrial practice are the integration of heterogeneous information sources and the management of information resources and services. Currently, a wider uptake of semantic technologies is still hindered by the failure of these technologies to meet some of the basic requirements of practical applications. One of these requirements is the support for sophisticated security policies for protecting the knowledge contained in ontologies and their instances from unauthorized use. In this paper, we propose a model-driven approach for access control for a lightweight ontology model that uses a role-based security model. The approach utilizes standard technologies such as (a subset of) OWL and the SPARQL query language as well as the Extensible Access Control Markup Language (XACML). We present the security model, show how it can be linked to the knowledge model using a meta-modeling approach and discuss different options for enforcing policies on SPARQL queries.

1. Motivation

1.1. Ontologies and Industrial Applications

Semantic technologies originally developed for knowledge representation and reasoning on the web are gaining attention from industry as an adequate means for solving complex information management tasks. A constantly growing number of companies that offer professional services or tools in this area reflect this trend [4]. Typical applications of ontologies include knowledge and skill management [18] as well as web service and business process management [8]. Studies [10], prototypes [14], and first successful projects [19] emphasize the relevance of these technologies in the area of product lifecycle management (PLM), in particular in the automotive sector. Their application in large-scaled industrial use cases (e.g., [2]) quickly shows that theoretical issues like expressiveness and decidability are less a limiting factor for their uptake in industry than non-functional aspects. In commercial scenarios we are particularly faced with ever changing domains that need to be reflected within a knowledge base, restricted resources for its development and maintenance, and the uncertain perspicuity and acceptance of such an information source by the actual users [10].

1.2. The Need for Access Control

Another important requirement of many real world applications in industrial practice is the need to protect knowledge against unauthorized access. This need is evident in many potential application areas of ontologies. In particular in the area of data and application integration, which is one of the primary applications of ontologies, there is a strong need for security mechanisms. The integrated data sources often hold confidential information that should not be accessible for everybody. In

¹ Mannheimer Zentrum für Wirtschaftsinformatik, A5, 6 68159 Mannheim, mail@chenwilly.info / Heiner.Stuckenschmidt@uni-mannheim.de

some cases, such as the deregulation of the energy markets, there are even legal constraints on the accessibility of strategic information between different parts of the same enterprise.

Tailoring the integration of data sources for different target groups is too much effort in most cases. Therefore, the only sensible solution is a complete integration of information along with the implementation of a fine-grained access control mechanism that grants access to parts of the integrated model based on a suitable set of security policies. While this problem has been investigated in details for standard technologies such as relational databases, so far there are no convincing solutions for providing fine-grained access control for ontology-based knowledge. This hinders the uptake of semantic technologies in industrial applications dealing with sensible information. Providing a solution to this problem therefore increases the usefulness and the potential impact of semantic technologies in practice.

1.3. Outline and Contributions

Within this paper we present an approach based on the Model-driven Architecture (MDA) to enable access control for ontologies. We thereby utilize the Ontology Definition Metamodel (ODM, [16]), a MDA standard for developing ontologies provided by the Object Management Group (OMG), to create a metamodel for lightweight ontologies, which is tailored to the use in commercial settings. We extend this metamodel with elements to define access control policies, which can directly refer to ontology elements. This integrated approach to access control for ontologies enables building a loosely coupled security architecture for enabling authorization within existing semantic web infrastructures. In fact, reasoners do not need to be modified, whereas we can utilize existing access control frameworks to enforce the defined security policies. Based on the architecture, we discuss different options for enforcing security policies in the presence of logical inference.

The paper is organized as follows. In section 2 we briefly recall previous work on a model-driven approach for defining lightweight ontologies for industrial applications and provide an example of its use. In section 3, we extend the approach with a role-based access control model that uses the XACML standard for defining policies and show how it applies to our example. In section 4, we discuss different approaches for enforcing security policies by rewriting SPARQL queries posed to the knowledge model. We conclude with a brief review of related work in section 5 and a discussion of the strength and weaknesses of our approach.

2. Industrial-Strength Ontologies

In previous work [1] we have proposed a lightweight knowledge model and an associated metamodel that meets the representation requirements of typical industrial applications while still being easy to understand and implemented using existing technology. In the following, we briefly present parts of this model for which we investigate the development of a security framework.

2.1 A Metamodel for Lightweight Ontologies

Gasevic et al. [9] have shown the relevance of the Model-driven Architecture (MDA) for ontology development. The Object Management Group (OMG), which provides all standards related to MDA in software engineering, has recently finalized the standardization process of the Ontology Definition Metamodel (ODM, [16]) - a metamodel for OWL Full ontologies based on the MetaObjects Facitiliy (MOF), the standard for describing such models. The participation of big players of the branch such as IBM and AT&T and a first implementation by the Eclipse foundation, the EODM project¹, shows the high relevance of the ODM particularly to the industry.

The ODM covers OWL Full, which results in a fairly complex metamodel. In fact, for OWL DL and Lite the ODM could be significantly simplified, since the use of RDF constructs are strongly restricted in these subsets. Because OWL Full is not relevant to commercial applications due to its undecidability, we focus on more practical subsets of OWL. To enable high performance reasoning even with large A-Boxes we propose the use of a subset of OWL Lite, which falls into plain Datalog. Such a subset, namely OWL Lite^{-p}, has been identified by de Bruijn et al. [5]. The relevance of such a subset has been discussed by Volz [20] and Hitzler et al. [12]. In order to enable the direct representation of data from relational database, we added the use of primitive datatypes. The resulting subset, OWL Lite^{-p}, has been presented in more details in [1]. Table 1 gives an overview of the included language elements and also presents some mappings to other knowledge representation languages.

OWL Abstract Syntax	DL Syntax	F-Logic Syntax	Datalog Syntax
restriction(R allValuesFrom(C))*	$\forall R.C$		
Class (A partial $C_1 \dots C_n$)	$A \sqsubseteq C_i$	$\bigwedge^1 A :: C_i$ $\bigwedge^2 x_1 : A \wedge x_1[R_i \rightarrow x_2] \rightarrow x_2 : C_i$	$\bigwedge^1 A(x) \rightarrow C_i(x)$ $\bigwedge^2 A(x_1) \wedge R(x_1, x_2) \rightarrow C_i(x_2)$
Class (A complete $C_1 \dots C_n$)	$A \equiv C_1 \sqcap \dots \sqcap C_n$	$\bigwedge \left\{ \begin{array}{l} A :: C_i \\ C_i :: A \end{array} \right.$	$\bigwedge \left\{ \begin{array}{l} A(x) \rightarrow C_i(x) \\ C_i(x) \rightarrow A(x) \end{array} \right.$
EquivalentClasses ($C_1 \dots C_n$)	$C_1 \equiv \dots \equiv C_n$	$\bigwedge_{i \neq j} C_i :: C_j$	$\bigwedge_{i \neq j} C_i(x) \rightarrow C_j(x)$
ObjectProperty (R)	$R \sqsubseteq R_i$	$\bigwedge x[R \rightarrow y] \rightarrow x[R_i \rightarrow y]$	$\bigwedge R(x, y) \rightarrow R_i(x, y)$
super (R_1) ... super (R_n)	$\top \sqsubseteq \forall R^-.C_i$	$\bigwedge x[R \rightarrow y] \rightarrow x : C_i$	$\bigwedge R(x, y) \rightarrow C_i(x)$
domain (C_1) ... domain (C_n)	$\top \sqsubseteq \forall R.C_i$	$\bigwedge x[R \rightarrow y] \rightarrow y : C_i$	$\bigwedge R(x, y) \rightarrow C_i(y)$
range (C_1) ... range (C_n)	$R \equiv R_0^-$	$\bigwedge \left\{ \begin{array}{l} x[R \rightarrow y] \rightarrow y[R_0 \rightarrow x] \\ x[R_0 \rightarrow y] \rightarrow y[R \rightarrow x] \end{array} \right.$	$\bigwedge \left\{ \begin{array}{l} R(x, y) \rightarrow R_0(x, y) \\ R_0(x, y) \rightarrow R(x, y) \end{array} \right.$
[inverseOf (R_0)]	$R \equiv R^-$	$x[R \rightarrow y] \rightarrow y[R \rightarrow x]$	$R(x, y) \rightarrow R(y, x)$
[Symmetric]	Trans (R)	$x[R \rightarrow y] \wedge y[R \rightarrow z] \rightarrow x[R \rightarrow z]$	$R(x, y) \wedge R(y, z) \rightarrow R(x, z)$
[Transitive]	$R_1 \sqsubseteq R_2$	$x[R_1 \rightarrow y] \rightarrow x[R_2 \rightarrow y]$	$R_1(x, y) \rightarrow R_2(x, y)$
SubPropertyOf (R_1 R_2)	$R_1 \equiv \dots \equiv R_n$	$\bigwedge_{i \neq j} x[R_i \rightarrow y] \rightarrow x[R_j \rightarrow y]$	$\bigwedge_{i \neq j} R_i(x, y) \rightarrow R_j(x, y)$
EquivalentProperties ($R_1 \dots R_n$)			
DatatypeProperty (U)	$\top \sqsubseteq \forall U^-.C_i$	$\bigwedge x[U \rightarrow y] \rightarrow x : C_i$	(not supported)
domain (C_1) ... domain (C_n)	$\top \sqsubseteq \forall U.C_i$	$C_i[U \rightarrow D]$	(not supported)
range (D)**			
Individual (o type (C_1) ... type (C_n))	$o \in C_i$	$\bigwedge o : C_i$	$\bigwedge C_i(o)$
value (R_1 o_1) ... value (R_n o_n)	$\langle o, o_i \rangle \in R_i$	$\bigwedge o[R_i \rightarrow o_i]$	$\bigwedge R_i(o, o_i)$
value (U_1 v_1) ... value (U_n v_n)	$\langle o, v_i \rangle \in U_i$	$\bigwedge o[U_i \rightarrow v_i]$	(not supported)

* only allowed in partial class definitions

** only primitive datatypes allowed

¹: for named classes

²: for universal class restriction

Figure 1 Languages Features of OWL Lite^{-p}

For the development of the metamodel for OWL Lite^{-p} we have cut down the not needed elements of the ODM. The resulting MOF-based metamodel is thus highly simplified [1]. To actually take advantage of the metamodel and utilize it for ontology development, we have decided to build on the Eclipse Modeling Framework, a project aiming at providing Eclipse-based tools for model-driven development. It provides Java implementations of MDA-standards and concrete reusable tool support. It uses Ecore as its meta-metamodel language. In fact, it is a subset of MOF. The EMF provides means to generate a Java object representation of a metamodel defined in Ecore. Figure x shows the Ecore equivalent of the metamodel for OWL OWL Lite^{-p}.

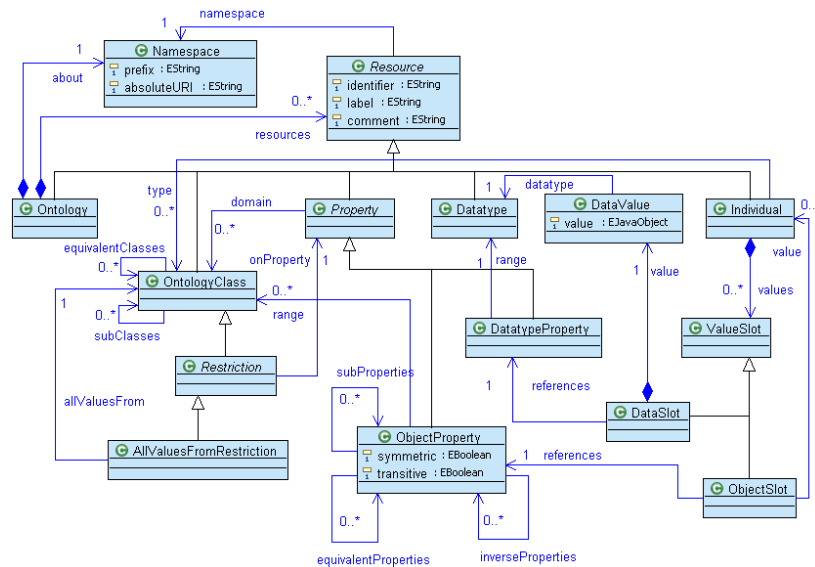


Figure 2 Lightweight Ontology Metamodel

2.2 An Example

Consider a product ontology, which integrates different information about products and their parts that may exist in different departments of an car manufacturer. Clearly, some information from e.g. sales such as the purchase price of a part should not be visible for developers. Also, external staff that works on specific parts should only see data, which they require for their work. For example, somebody who is working on the braking system should only be able to see brake parts and specific electronic control units such as the Anti-Lock Braking System.

The ontology is defined using the abstract OWL syntax: Class(Manufacturer partial), Class(Supplier partial Manufacturer restriction(produces allValuesFrom Part)), Class(Product partial), Class(Car partial Product), Class(Part partial Product), Class(BodyPart partial Part), Class(PrototypeBodyPart partial BodyPart), Class(ElectronicControlUnit partial Part), Class(AntiLockBrakingSystem partial ElectronicControlUnit), Class(TransmissionControlUnit partial ElectronicControlUnit), Class(TCU partial ElectronicControlUnit), Class(Brake partial Part), EquivalentClasses(TCU, TransmissionControlUnit), ObjectProperty(produces domain(Manufacturer) range(Product)), ObjectProperty(contains domain(Car) range(Part)), DatatypeProperty(hasSalesPrice domain(Product) range(float)), DatatypeProperty(hasPurchasePrice domain(Product) range(float)).

3. The Security Framework

3.1 Access Control and Policy Specification

Access control systems enable the regulation of access to protected resources (i.e. objects) in distributed systems by subjects such as users or system processes (cf. [6]). They can be categorized in discretionary access control (DAC), mandatory access control (MAC), and role-based access control (RBAC) models. In DAC-based systems, the permissions to access an object are defined by its owner. In MAC models, the system determines the access to objects either by utilizing access

rules or lattices for assigning permissions to subjects. It thus removes the ability of the users to control access to their resources. RBAC systems finally remove the explicit use of subjects within access rules or lattices and replace them with roles, which form a logical group of a number of subjects. In fact, permissions are assigned to roles and the subjects are assigned members of a number of roles. Thus changes of single subjects do not necessarily have consequences in the actual access control policies.

The Extensible Access Control Markup Language (XACML) is an OASIS standard that describes both a policy language and an access control decision request/response language (both written in XML) [15]. The policy language consists of following basic constructs:

- PolicySet: A PolicySet is a container for a number of Policies or other PolicySets.
- Policy: A policy represents a single access control policy, expressed by a set of Rules. It applies for a defined Target.
- Rule: A Rule consists of a Target element and has an attribute Effect, which can be either 'permit' or 'deny'.
- Target: The Target element declares a set of Subjects, Resources, and Actions for which a Rule or Policy applies.

Note that the actual XACML specification defines a number of additional constructs to further refine the access control policies.

3.2 Applying Access Control to Ontologies

Since the T-Box of an ontology reflects a common vocabulary of a specific domain it is not reasonable to restrict access at this level. In fact, we aim at controlling the access to instances of specific classes and properties – i.e. the A-Box, while the T-Box remains unrestricted. We thereby assume that the access to individuals of all classes and all property values are denied by default for avoiding unintended security leaks. Because of this, access to specific classes and thus corresponding individuals as well as property values have to be explicitly granted for subjects. To refine this basic security policy, we consider following basic access rule types: (1) permit access to all individuals and values respectively of a class, a datatype property, or an object property; (2) deny access to a specific individual. To refine these rules, we provide means to (3) deny the access to individuals and values of subclasses and subproperties of prior permitted resources. As common in modern access control systems, we envision a RBAC model for simplifying the maintenance of the required access control policies.

3.3 Extending the Metamodel

In order to enable a uniform and interdependent maintenance of the protected ontologies together with the security policies, we extend the earlier presented metamodel of lightweight ontologies by means to model access control policies. Thus, we do not need to commit to a specific representation formalism for either ontologies or security policies. We initially cover a subset of the XACML specification, which provides basic features for realizing the access control for ontologies as presented above (cf. Figure 3). We thereby leave out the PolicySet element and only consider simple restrictions on ontology resources. The use of the Target construct allows to group the Policies by Subjects, Resources or any combination of both. If no Action, Subject, or Resource is defined for a Target, we consider that the Policy or Rule applies for all Actions, Subjects, and Resources respectively. Within our work, we assume that Policies are grouped by Subjects, whereas the Rules specify both Actions and Resources that should be restricted.

Kommentar [willy1]: Policy/Rule Combining Algorithmen sind in unserem Rahmen wohl nicht interessant (All specified PolicySets are jointly evaluated upon request by the access control system using specific Policy-combining and Rule-combining algorithms, which are specified in [6] such as Deny-overrides or Permit-overrides.)

Kommentar [willy2]: Weitere mögliche Fälle:
(Deny access to all individuals having a specific datatype property value.)
(Deny access to all individuals having a specific object property value.)

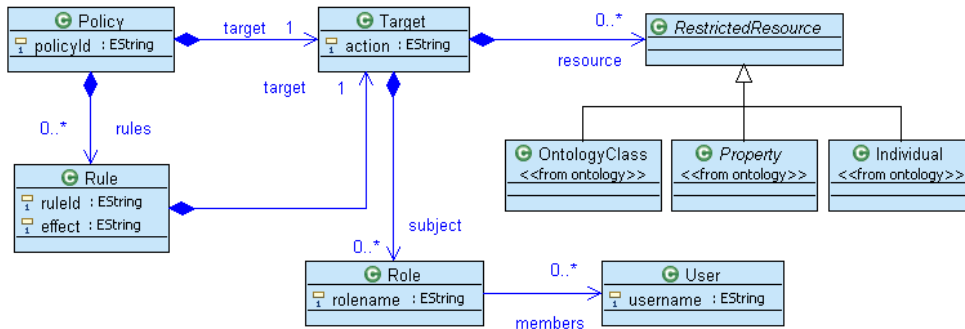


Figure 3 Access Control Policies Metamodel

3.4 Security Architecture

We can utilize existing reasoners for OWL Lite to infer implicit knowledge and query the ontology. This is typically done by sending a SPARQL query to the appropriate endpoint of the used reasoner. In order to ensure confidentiality in such an infrastructure, we enforce the use of a security proxy, which intercepts all communication to the reasoner. Similar to web proxy systems, we require a basic user authentication prior to its first use. The proxy thereby acts as the Policy Enforcement Point (PEP) within the XACML architecture and constructs an authorization decision query to the Policy Decision Point (PDP). The decision of the PDP is based on the access control policies provided by the Policy Retrieval Point (PRP) and the information provided by Policy Information Point (PIP) about for instance the role of a user. The PEP finally enforces the decision of the PDP either by completely denying the query, by rewriting the query so it meets the security policies, or by directly redirecting the query to the reasoner. Details of this part of the process are discussed in section 4 below. Within this work we assume that the security proxy unifies the different components (PxP) defined by the XACML architecture so there is no need to exchange messages between them. In future work, we consider splitting the single parts into separated components. Finally, we need to setup a trust network so that the reasoner only accepts queries from the security proxy. The resulting security architecture ensures that enforcement of access control policies in a loosely coupled way by utilizing existing frameworks and tools without the need to modify existing reasoners, so they can directly support such a task.

3.5 Example

An external developer works on the braking system of a car and thus should be able to access the required product information – i.e. information about Brake and AntiLockBrakingSystem. Nevertheless, a prototype ABS-system should not be visible for him. We define following policy for the role 'ExternalBrakeDeveloper':

Policy $P_1 = \{\text{subject}='ExternalBrakeDeveloper', \text{rules}=\{R_1, R_2, R_3\}\}$
 Rule $R_1 = \{\text{'read'}, \text{'permit'}, \{\text{Class}(\text{Brake})\}\}$
 Rule $R_2 = \{\text{'read'}, \text{'permit'}, \{\text{Class}(\text{AntiLockBrakingSystem})\}\}$
 Rule $R_3 = \{\text{'read'}, \text{'deny'}, \{\text{Individual}(\text{ABS1})\}\}$

For employees working in the sales department, we define similar rules, but explicitly permit the access to the datatype property 'hasPurchasePrice':

Policy $P_2 = \{\text{subject}='Sales', \text{rules}=\{R_3, R_4, R_5\}\}$

Rule $R_3 = \{ \text{'read'}, \text{'permit'}, \{ \text{Class(Product)} \} \}$
 Rule $R_4 = \{ \text{'read'}, \text{'deny'}, \{ \text{Individual(ABS2)} \} \}$
 Rule $R_5 = \{ \text{'read'}, \text{'permit'}, \{ \text{DatatypeProperty(hasPurchasePrice)} \} \}$

4. Enforcing Access Policies

As outlined above, security policies are enforced at a security proxy that receives queries to the ontology and rewrites them in such a way that no policies are violated. In this section, we provide details about this rewriting step. The goal of the rewriting step is to create a query that returns the same results as the original query except for those answers that contain restricted information. For this purpose, the proxy first determines the access restrictions that apply for the specific query by determining the role of the issuing user and looking up the restrictions that apply to this role in the policy rules. As defined in the security framework, such rules can apply to classes, properties and instances. In this paper, we further focus on read restrictions on these elements.

4.1 Filtering Query Results

A read restriction on individuals i_1, \dots, i_n is interpreted in the way that the user is not allowed to see answers that contain this specific individual. This condition can easily be fulfilled by extending the query with the following filter condition on each of the return variables $?v_k$

FILTER ($?v_k \text{ != ex:i1 \&\& .. \&\& ?v_k != ex:i_n}$)

This condition removes all results where restricted individuals are bound to any of the return variables.

We interpret read restriction on a class C in such a way that the user is not allowed to see answers that contain instances of the restricted class. Therefore not allowing the access to a class is equivalent to restricting the access to any individual of the class. Matters are complicated by the fact that class membership can be inferred using the ontology. Therefore, it is not possible to simply add filter restrictions for all instances that are defined to be member of restricted classes. We can however, delegate this problem to the underlying inference engine by using a filter expression on the schema level:

OPTIONAL { $?v_k \text{ rdf:type ?v2 . FILTER (?v2 = ex:C) } \text{ FILTER (! bound (?v2)) }$ }

This expression checks for each return variable, whether the instance bound to the variable is of type C . If this is true, the corresponding answer is filtered out based on the criterion that variable $?v2$ is bound to the restricted class.

Read permission on classes can be enforced in a similar manner by adding a filter expression, which ensures that the queried instances are at least member of one of the permitted classes C_i :

$?v_k \text{ rdf:type ?v2. FILTER (?v2 = ex:C1 || ... || ?v2 = ex:Cn)}$

We further interpret restrictions on a relation P in the way that the user is not allowed to see answers in which two of the return variables are bound to instances that are in relation R to each other, because queries might be formulated in such a way, that they model the restricted relation without actually mentioning it. We again use a filter expression to filter out the corresponding answers from the result set.

OPTIONAL { ?v_k ?r ?v_j . FILTER (?r = ex :P) } FILTER (! bound (?r))

Here ?v_k ≠ ?v_j are return variables of the query. The expression checks whether the existence of relation P between ?v_k and ?v_j can be derived from the ontology. If this is the case, the corresponding answer is filtered out based on the criterion that the variable ?r is bound to P. This solution again delegates the problem of dealing with derivable information to the underlying inference engine which is used to check whether (?v_k ?r ?v_j) can be derived from other information such as inverse relations, etc.

In the case a user queries the Cartesian product of two variables ?v_k and ?v_j the filter above would cut out sensitive pairs of values, which enables attackers to possibly derive confidential information from. We therefore need to ensure that such queries are detected by the security proxy and rejected by default.

Similar to explicitly granting read access to individuals of a specific class, we can add a filter for permitting access to property values P_i:

?v_k ?r ?v_j . FILTER (?r = ex:P₁ || ... || ?r = ex:P_n)

4.2 Controlling the re-writring Process

In section 4.1 we have presented the concrete filter expressions that can be used to ensure that no restricted information is returned by a query. This section now deals with linking the manipulation of the queries with the policy specifications in order to control the rewriting process. Following the assumption from 3.2, we can formulate a simple rewriting strategy that first extends the query with filters for those individuals explicitly restricted for the role under consideration and then also adds filters for concepts and relations access to which is not explicitly permitted. The corresponding simple algorithm is given below:

ReWrite(Q:Query, R:Role, P:Policy)

FORALL {i, read, R, deny} ∈ P DO	AddDenyFilter(Q,i)
FORALL {C, read, R, permit} ∈ P DO	AddPermitFilter(Q,C)
FORALL {C, read, R, deny} ∈ P DO	AddDenyFilter(Q,C)
FORLAL {P, read, R, permit} ∈ P DO	AddPermitFilter(Q,P)
FORALL {P, read, R, deny} ∈ P DO	AddDenyFilter(Q,P)
IF {C, read, R, permit} ∉ P DO	AddDefaultClassDenyFilter(Q)
IF {P, read, R, permit} ∉ P DO	AddDefaultPropertyDenyFilter(Q)

The conditions for adding the filter expressions are defined in such a way that in the case of a direct conflict – the policy set contains rules both for grating and denying permission on a certain element – the filter is added to protect the data.

4.3. Verifying Anomalies in Access Policies

| During the creation and maintenance anomalies such as [contradictory](#) rules could occur in access policies. Since we use filters to enforce the policies, a deny rule always overrides a permit rule on the same resource. Thus, there is not possible information leak. In order to detect such anomalies,

Feldfunktion geändert

Formatiert: Englisch
(Großbritannien)

we provide following verification rules that can be applied at the metamodel level on groups of rules that apply for the same subject:

- IF EXISTS $R1 = \{\text{'read'}, \text{'permit'}, \text{RestrictedResource}(R)\}$, $R2 = \{\text{'read'}, \text{'deny'}, \text{RestrictedResource}(R)\}$ THEN Anomaly($R1, R2$)
- IF EXISTS $R1 = \{\text{'read'}, \text{'permit'}, \text{OntologyClass}(C1)\}$, $R2 = \{\text{'read'}, \text{'deny'}, \text{OntologyClass}(C2)\}$ AND EquivalentClasses($C1, C2$) THEN Anomaly($R1, R2$)
- IF EXISTS $R1 = \{\text{'read'}, \text{'permit'}, \text{ObjectProperty}(P1)\}$, $R2 = \{\text{'read'}, \text{'deny'}, \text{ObjectProperty}(P2)\}$ AND (EquivalentProperties($P1, P2$) OR ObjectProperty($P1$ inverseOf($P2$))) THEN Anomaly($R1, R2$)
- IF EXISTS $R1 = \{\text{'read'}, \text{'permit'}, \text{ObjectProperty}(P1)\}$, $R2 = \{\text{'read'}, \text{'deny'}, \text{OntologyClass}(C1)\}$ AND (ObjectProperty($P1$ domain($C1$)) OR ObjectProperty($P1$ range($C1$))) THEN Anomaly($R1, R2$)
- IF EXISTS $R1 = \{\text{'read'}, \text{'permit'}, \text{DatatypeProperty}(P1)\}$, $R2 = \{\text{'read'}, \text{'deny'}, \text{OntologyClass}(C1)\}$ AND DatatypeProperty ($P1$ domain($C1$)) THEN Anomaly($R1, R2$)

5. Related Work

Recently, there has been quite a bit of work on combining semantic web technologies with security issues. Most of this work, however, is about applying semantic web technologies to the problem of specifying and checking security policies and related aspects (e.g. [13]). The aspect we are interested in, the protection of knowledge encoded in an ontology has so far received less attention. Qin and Atluri [17] have presented an approach to enable access control at concept level for ontologies. Their approach is based on a propagation mechanism that derives access rights for concepts based on the semantic of the ontology and partial access rights explicitly stated. A similar approach is envisioned by Knechtel [?] who proposes to derive the access rights of derived facts from the access rights based on access restrictions imposed on stated knowledge. In contrast to this our approach relies on the re-writing on queries based on security policies. This allows us to leave the task of optimizing the execution of the query to the query engine.

The inference problem, which is central to our work has been investigated by researchers in the field of database research [7]. The problem has been investigated in the context of statistical databases, where statistical inference can be used to derive classified information from unclassified ones [?]. Special attention has been paid to the inference problem in the context of multilevel secure databases. Here it also has to be ensured that access policies are not violated between the different security layers. In our work, we have so far ignored such problems arising in multilevel security policies. Most of the work on secure databases has addressed the relational model and problems that arise from the use of database constraints [3]. The problem of enforcing security constraints in deductive databases, a problem that is closely related to our work has so far only been addressed on a theoretical level with little concerns on implementability [?].

6. Conclusions

In this paper we have presented an approach to restrict the access to confidential knowledge that is represented using ontologies. We consider this feature essential for a greater uptake of ontologies in real world applications. We thereby fully build on existing technologies and infrastructures. The access control system relies on a security proxy, which intercepts all communication to the SPARQL endpoint of the used reasoner for OWL Lite. When enforcing fine-grained role-based access policies we need to take possible inferences from the ontology into account. Instead of

realizing this complex issue within the security proxy, we delegate this task to the reasoner by intelligently adding SPARQL filter statements to incoming queries from users. This allows us to apply our solution with all reasoners supporting OWL Lite and SPARQL.

Since the access policies and rules are highly dependent to the ontologies they have been defined for, we utilize a model-driven approach to enable their interdependent development and maintenance. Moreover, we do not need to decide for a specific representation language for either a priori, but can transform the created models to various formalisms we required. For ontologies, we can directly support OWL, F-Logic, and Datalog, while we may rely on the XACML specification for access policies. Within this work we are not even required to explicitly export the access policies to a formal representation but can rely on their representation as Java objects, which are then directly used by the security proxy.

Bibliography

- [1] CHEN, W. and STUCKENSCHMIDT, H., Towards industrial strength knowledge bases for product lifecycle management, in: 16th European Conference on Information Systems, Galway, Ireland, 2008.
- [2] CHEN, W., SYLDATKE, T. and HESS, C., Business-oriented CAx-Integration with Semantic Technologies, in: 3rd International Applications of Semantic Technologies Workshop, Munich, Germany, 2008.
- [3] BONATTI, P., KRAUS, S. and SUBRAHMANYAN, V.S., Foundations of Secure Deductive Databases IEEE Transactions on Knowledge and Data Engineering, 1995.
- [4] DAVIS, M., Semantic Wave 2008 Report - Industry Roadmap to Web 3.0 & Multibillion Dollar Market Opportunities 2008
- [5] DE BRUIJN, J., POLLERES, A. and FENSEL, D., OWL Lite², WSML Deliverable D20 v 0.1, 2004.
- [6] ECKERT, C., IT-Sicherheit, Oldenburg, 2003.
- [7] FARKAS, C. and JAJODIA, S., The inference problem: a survey, SIGKDD Explor. Newsl., ACM, 2002.
- [8] FENSEL, D., Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce, Springer Verlag, 2001
- [9] GASEDVIC, D., DJURIC, D., and DEVEDIC, V., Model Driven Architecture and Ontology Development. Springer, 2006
- [10] HAHN, A., Integration verteilter Produktmodelle durch Semantic-Web-Technologien, Wirtschaftsinformatik, 47, 278-284, 2005.
- [11] HEPP, M., Possible Ontologies: How Reality Constrains the Development of Relevant Ontologies, IEEE Internet Computing, 2007.
- [12] HITZLER, P., HAASE, P., KRÖTZSCH, M., SURE, Y., and STUDER, R., DLP is not so bad after all, in: OWL: Experiences and Directions, 2005.
- [13] KOLOVSKI, V., HENDLER, J. and PARSIA, B., Analyzing Web Access Control Policies, in: Proceedings of the Sixteenth International World Wide Web Conference, 2007.
- [14] KÜSTERS, K., Teilautomatisierte Überführung der Produktstruktur in eine lastfallabhängige Modellstruktur mittels Ontologien, Diploma Thesis Technische Hochschule Aachen, 2006.
- [15] OASIS, A Brief Introduction to XACML, <http://www.oasisopen.org/committees/download.php/2713/BriefIntroductionto-XACML.html>, March 2003.

- [16] OMG Adopted Specification, Ontology Definition Metamodel, <http://www.omg.org/docs/ptc/07-09-09.pdf>, November 2007.
- [17] QIN, L. and ATLURI, V., Concept-level access control for the semantic web, in: Proceedings of the 2003 ACM workshop on XML security, 2003.
- [18] STAAB, S., Wissensmanagement mit Ontologien und Metadaten, Habilitation Thesis, University of Karlsruhe, 2002.
- [19] SYLTAKE, T., CHEN, W., ANGELE, J., NIERLICH, A. and ULLRICH, M., How Ontologies and Rules Help to Advance Automobile Development Advances in Rule Interchange and Applications, in: Proceedings International Symposium RuleML 2007, Orlando, Florida, 2007.
- [20] VOLZ, R., Web Ontology Reasoning with Logic Databases, PhD thesis, Universität Fridericiana zu Karlsruhe, 2004.