Representing Security Policies in Web Information Systems

Félix J. García Clemente
Departamento de Ingeniería de la Información y las Comunicaciones
Campus de Espinardo, s/n
30.071 Murcia, Spain
+34 968 367645
fgarcia@dif.um.es

Gregorio Martínez Pérez
Departamento de Ingeniería de la Información y las Comunicaciones
Campus de Espinardo, s/n
30.071 Murcia, Spain
+34 968 367646
gregorio@dif.um.es

Juan A. Botía Blaya
Departamento de Ingeniería de la Información y las Comunicaciones
Campus de Espinardo, s/n
30.071 Murcia, Spain
+34 968 367317
juanbot@um.es

Antonio F. Gómez Skarmeta
Departamento de Ingeniería de la Información y las Comunicaciones
Campus de Espinardo, s/n
30.071 Murcia, Spain
+34 968 364607
skarmeta@dif.um.es

ABSTRACT

Policies, which usually govern the behaviour of networking services (e.g., security, QoS, mobility, etc.), are becoming an increasingly popular approach for the dynamic regulation of web information systems. The adoption of a policy-based approach for controlling a system requires an appropriate policy representation regarding both syntax and semantics, and the design and development of a policy management framework. In the context of the Web, the use of languages enriched with semantics (i.e. semantic languages) has been limited primarily to represent Web content and services. However the capabilities of these languages, coupled with the availability of tools to manipulate them, make them well suited for many other kinds of application, as policy representation and management. This paper provides the current trends of policy-based management enriched by semantics applied to the protection of web information systems. It also presents an approach for using DMTF Common Information Model (CIM) ontology with semantic languages.

Categories and Subject Descriptors
K.6.5 [**Management of Computing and Information System**]: Security and Protection.

General Terms
Management, Security, Languages

Keywords
Semantic Languages, Security Policy, CIM Ontology

1. INTRODUCTION

One of the main goals of policy-based management is to enable network, service and application control and management at a high abstraction layer. Using a policy language, the administrator specifies rules that describe domain-wide policies which are independent of the implementation of the particular network node, service and/or application. It is, then, the policy management architecture that provides support to transform and distribute the policies to each node and thus enforce a consistent configuration in all the elements involved. This is a prerequisite for achieving a mean to dynamically constrain and regulate the behaviour of a system without the human cooperation.

In the web information systems security field, a policy (i.e., security policy) can be defined as a set of rules and practices describing how an organization manages, protects and distributes sensitive information at several levels. Security policies can be defined to perform a wide variety of actions, from IPsec/IKE management (example of network security policy) to access control over a web server (example of application-level policy).

Researchers have proposed multiple approaches for policy specification. They range from formal policy languages that a computer can directly process, to rule-based policy notation using an *if-then-else* format, or to the representation of policies based on Deontic logic for obligation and permissibility rules.

To cover this wide range of security policies languages, this paper aims to examine the current state of policy engines and policy languages, focusing on the approaches enriched with semantics (i.e. semantic languages) using RDF [11] and OWL [2] as standards for policy specification. We intend to show the strengths and limitations of such languages by comparing three approaches: KAoS, Rei and SWRL.

The major benefit of specifying security policy rules in this way is that an organization can utilize a common ontology that can be shared amongst services and service clients. In this sense, DMTF presents the Common Information Model (CIM) standard [4] to provide a common definition of management-related information. This paper also presents an approach for using CIM ontology with semantic languages. It permits an administrator to formally describe the security policies of an administrative domain using the DMTF methodology.

This document is structured as follows. Section 2 presents the requirements of policy frameworks, focusing on policy languages and policy architectures. Then, section 3 presents a comparative analysis between “traditional” non-semantic and semantic policy frameworks to emphasize the advantages of semantic approaches. Section 4 describes and compares the three semantic approaches aforementioned. Then, section 5 presents the extension of the semantic policy language SWRL with the CIM ontology and shows an example for an authorization policy. Finally, we conclude the paper with our remarks and some future directions derived from this work.

2. REQUIREMENTS FOR A POLICY FRAMEWORK

The policy administrator needs to use a policy language that assures that the representation of policies guarantee the following requirements:

Well-defined. A policy language can be considered as well-defined if the syntax and structure is clear and no-ambiguous, and the meaning of a policy written in this language is independent of its particular implementation.

Flexibility and extensibility. A policy language has to be flexible enough to allow new policy information to be expressed, and extensible enough to allow new types of policy to be added in future versions of this language.

Interoperability with other languages. There are usually several languages that can be used in different domains to express similar policies, and interoperability is a must to allow different services or applications from these different domains to communicate with each other according to the behaviour stated in these policies.

Once the policy has been defined for a given administrative domain, a management architecture is required to transfer, store and enforce this policy in that domain. The main requirements for such policy management architecture are:

Well-defined interface. Policy architectures need to have a well-defined interface independent of the particular implementation in use. In it, the interfaces between the components need to be clear and no-ambiguous.

Flexibility and definition of abstractions to manage a wide variety of device types. The system architecture should be flexible enough to allow addition of new types of devices with minimal updates and recoding of existing management components.

Interoperability with other architectures (inter-domain). The system should be able to interoperate with other architectures that may exist in other administrative domains.

Conflict Detection. It has to be able to check that a given policy does not conflict with any other existing policy.

Scalability. It should maintain quality performance under an increased system load.

The policy framework has to support all these requirements to guarantee the correct system operation.

3. ADVANTAGES OF SEMANTIC SECURITY POLICY FRAMEWORKS

There are some non-semantic security policy frameworks such as Ponder [3] and XACML [7] that we describe briefly as follows:

Ponder, is a declarative, object-oriented language developed for specifying management and security policies. Ponder permits to express authorizations, obligations, information filtering, refrain policies, and delegation policies. Ponder can describe any rule to constrain the behaviour of components, in a simple and declarative way.

The eXtensible Access Control Markup Language (XACML) describes both an access control policy language and a request/response language. The policy language provides a common means to express subject-target-action-condition access control policies and the request/response language expresses queries about whether a particular access should be allowed and describes answers to those queries.

However, they do not take care of the description of the content of the policy (e.g., description of the specified components, the system, etc). The adoption of a semantic web language can overcome this limitation since it uses an ontology to describe the content of the policies.

In general, table 1 shows a comparative between semantic and non-semantic policy languages based on [9] and complemented with our own analysis [6].

Table 1. Comparative analysis between semantic and non-semantic policy languages

	Semantic Languages	**Non-Semantic Languages**
Abstraction	Multiple levels	Medium and low level
Extensibility	Easy and at runtime	Complex and at compile-time
Representability	Complex environments	Specific environments
Readability	Specialized tools	Direct
Interoperation	By common ontology	By interfaces
Enforcement	Complex	Easy

Semantic approaches using RDF/OWL (see Section 4) as standards for policy representation enable runtime extensibility and adaptability of the system, as well as the ability to analyse policies relating to entities described at different levels of abstraction. The representation facilitates careful reasoning about policy disclosure, conflict detection, and harmonization about domain structure and concepts. However, it is required complex policy automation mechanisms for enforcement.

4. SEMANTIC SECURITY POLICY LANGUAGES

As stated before, security policies can be specified at different levels of abstraction. The process starts with the definition of a business security policy. This can be the case of the next authorization security policy, which is defined in natural language: *"Permit the access to the e-payment service, if the user is in the group of customers registered for this service"*.

Next, the security policy is usually expressed by a policy administrator as a set of IF-THEN policy rules, for example: *IF ((<Requester> is member of Payment Customers) AND (<Server> is member of Payment Servers)) THEN (<Requester> granted access to <Server>)*

The policy languages we will be analyzing in this section are able to specify several types of security policies and will be used to provide policy examples related to this case study.

Although many semantic policy specifications exist, we have selected three of them as they are considered nowadays as promising options: KAoS, Rei and SWRL.

4.1 KAoS

KAoS [10] is a collection of services and tools that allow for the specification, management, conflict resolution, and enforcement of deontic-logic-based policies within domains describing organizations of human, agent, and other computational actors.

KAoS uses ontology concepts encoded in OWL to build policies. The KAoS Policy Service distinguishes between authorization policies and obligation policies. The applicability of the policy is defined by a set of conditions or situations whose definition can contain components specifying required history, state and currently undertaken action. In the case of the obligation policy the obligated action can be annotated with different constraints restricting possibilities of its fulfilment.

The current version of the KAoS Policy Ontologies (KPO) defines basic ontologies for actions, conditions, actors, various entities related to actions, and policies. It is expected that for a given application, the ontologies will be further extended with additional classes, individuals, and rules.

Figure 1 shows an example of the type of policy that administrators can specify using KAoS. It is related with the case study described earlier.

```
<owl:Class rdf:ID="PaymentAuthAction">
<owl:intersectionOf rdf:parseType="owl:collection">
  <owl:Class rdf:about="&action;AccessAction"/>
  <owl:Restriction>
  <owl:onProperty rdf:resource="&action;#performedBy"/>
  <owl:toClass
       rdf:resource="&domains;MembersOfPayCustomer"/>
  </owl:Restriction>
  <owl:Restriction>
  <owl:onProperty rdf:resource="&action;#performedOn"/>
  <owl:toClass
      rdf:resource="&domains;MembersOfPayServer"/>
  </owl:Restriction>
</owl:intersectionOf>
</owl:Class>
<policy:PosAuthorizationPolicy rdf:ID=”PaymentAuthPolicy1”>
  <policy:controls rdf:ID=”PaymentAuthAction”/>
  <policy:hasSiteOfEnforcement rdf:resource=”#TargetSite”/>
  <policy:hasPriority>1</policy:hasPriority>
</policy:PosAuthorizationPolicy>
```

Figure 1. Example of policy representation in KAoS

KAoS defines a Policy Framework that includes the following functionality:

Creating/editing of policies using KAoS Policy Administration Tool (KPAT). KPAT implements a graphical user interface to policy and domain management functionality.

Storing, de-conflicting and querying policies using KaoS Directory Service.

Distribution of policies to Guard, which acts as a policy decision point.

Policy enforcement/disclosure mechanism, i.e. finding out which policies apply to a given situation.

Every agent in the system is associated with a Guard. When an action is requested, the Guard is automatically queried to check whether the action is authorized based on the current policies and, if not, the action is prevented by various enforcement mechanisms. Policy enforcement requires the ability to monitor and intercept actions, and allow or disallow them based on a given set of policies. While the rest of the KAoS architecture is generic across different platforms, enforcement mechanisms are necessarily specific to the way the platform works.

4.2 Rei

Rei [5] is a policy framework that integrates support for policy specification, analysis and reasoning. Its deontic-logic-based policy language allows users to express and represent the concepts of rights, prohibitions, obligations, and dispensations. In addition, Rei permits users to specify policies that are defined as rules associating an entity of a managed domain with its set of rights, prohibitions, obligations, and dispensations.

Rei provides a policy specification language in OWL-Lite that allows users to develop declarative policies over domain specific ontologies in RDF, DAML+OIL and OWL.

A policy primarily includes a list of granting and a context used to define the policy domain. A granting associates a set of constraints with a deontic object to form a policy rule. This allows reuse of deontic objects in different policies with different constraints and actors. A deontic object represents permissions, prohibitions, obligations and dispensations over entities in the policy domain. It includes constructs for describing what action (or set of actions) the deontic is described over, who the potential actor (or set of actors) of the action is and under what conditions is the deontic object applicable.

An action is one of the most important in the Rei specifications as policies are described over possible actions in the domain. The domain actions describe application or domain specific actions, whereas the speech acts are primarily used for dynamic and remote policy management.

There are six subclasses of SpeechAct: Delegate, Revoke, Request, Cancel, Command, and Promise. A valid delegation leads to a new permission. Similarly, a revocation speech act nullifies an existing permission (whether policy based or delegation based) by causing a prohibition. An entity can request another entity for a permission, which if accepted causes a delegation, or to perform an action on its behalf, which if accepted causes an obligation. An entity can also cancel any previously made request, which leads to a revocation and/or a dispensation. A command causes an obligation on the recipient and the promise causes an obligation on the sender.

To enable dynamic conflict resolution, Rei also includes meta-policy specifications, namely setting the modality preference (negative over positive or vice versa) or stating the priority between rules within a policy or between policies themselves.

Figure 2 shows an example to illustrate the policy representation in Rei. It is related with the case study described earlier.

```
<constraint:SimpleConstraint rdf:ID=”IsPayCustomer”
   constraint:subject=”#RequesterVar”
   constraint:predicate=”&example;memberOf”
   constraint:object=”&example;payCustomer”/>
<constraint:SimpleConstraint rdf:ID=”IsPayServer”
   constraint:subject=”#PayServerVar”
   constraint:predicate=”&example;memberOf”
   constraint:object=”&example;payServer”/>
```

```
<constraint:And rdf:ID="ArePayCustomerAndPayServer"
   constraint:first="#IsPayCustomer"
   constraint:second="#IsPayServer"/>
<deontic:Permission rdf:ID="PayServerPermission">
   <deontic:actor rdf:resource="#RequesterVar"/>
   <deontic:action rdf:resource="&example;access"/>
   <deontic:constraint
         rdf:resource="#ArePayCustomerAndPayServer"/>
</deontic:Permission>
<policy:Policy rdf:ID="PaymentAuthPolicy1">
   <policy:grants rdf:resource="#PayServerPermission"/>
 </policy:Policy>
```

Figure 2. Example of policy representation in Rei

The Rei framework provides a policy engine that reasons about the policy specifications. The engine accepts policy specification in both the Rei language and in RDF-S [1], consistent with the Rei ontology. Specifically, the engine automatically translates the RDF specification into triplets of the form (subject, predicate, object). The engine also accepts additional domain-dependent information in any semantic language that can then be converted into this recognizable form of triplet. The engine allows queries according to the Prolog language about any policies, meta-policies, and domain dependent knowledge that have been loaded in its knowledge base.

The Rei framework does not provide an enforcement model. In fact, the policy engine has not been designed to enforce the policies but only to reason about them and reply to queries.

4.3 SWRL

Semantic Web Rule Language (SWRL) [8] is based on a combination of the OWL DL and OWL Lite sublanguages of the OWL with the Unary/Binary Datalog RuleML sublanguages. SWRL extends the OWL abstract syntax to include a high-level abstract syntax for Horn-like rules. A model-theoretic semantics is given to provide the formal meaning for OWL ontologies including rules written in this abstract syntax.

We distinguish between the following facts/rules for policy representation:

Structural/organizational facts and rules. These rules are used to encode domain specific ontologies.

Service definition facts and rules, provided with links to the structural rules and facts.

Task-specific rules and facts, provided by the service clients.

SWRL is defined by an XML syntax based on RuleML and the OWL XML Presentation Syntax. The rule syntax is illustrated with the following example related with the case study described earlier.

```
<ruleml:imp>
  <ruleml:_head>
    <swrlx:individualPropertyAtom
                      swrlx:property="GrantedAccess">
          <ruleml:var>requester</ruleml:var>
          <ruleml:var>server</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
  <ruleml:_body>
    <swrlx:classAtom>
          <owlx:Class owlx:name="User" />
          <ruleml:var>requester</ruleml:var>
    </swrlx:classAtom>
    <swrlx:classAtom>
          <owlx:Class owlx:name="Server" />
          <ruleml:var>server</ruleml:var>
    </swrlx:classAtom>
    <swrlx:individualPropertyAtom swrlx:property="Member">
          <ruleml:var>requester</ruleml:var>
          <owlx:Individual owlx:name="#PayCustomer" />
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom swrlx:property="Member">
          <ruleml:var>server</ruleml:var>
          <owlx:Individual owlx:name="#PayServer" />
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
</ruleml:imp>
```

Figure 3. Example of policy representation in SWRL

A useful restriction in the form of the rules is to limit antecedent and consequent classAtoms to be named classes, where the classes are defined purely in OWL. Adhering to this format makes it easier to translate rules to or from existing or future rule systems, including Prolog.

4.4 Comparative Analysis

Table 2 shows a comparison of the aforementioned security policy languages. Many aspects can be identified as part of this comparison, although the most relevant are:

Approach. Two types of approaches have been identified: rule-based and deontic logic-based.

Specification language. It can be XML, RDF-S or OWL.

Tools for policy specification.

Reasoning engine for policy analysis and verification.

Enforcement support to the policy deployment.

Table 2. Comparative analysis between KAoS, SWRL and Rei

	KAoS	Rei	SWRL
Approach	Deontic Logic	Deontic Logic + Rules	Rules
Specification language	DAML/OWL	Prolog-like syntax + RDF-S	Prolog-like syntax + OWL
Tools for specification	KPAT	No	No
Reasoning	KAoS engine	Prolog engine	Prolog engine
Enforcement	Supported	External Functionality	External Functionality

OWL has a limited way of defining restrictions using the tag owl:Restriction. This limitation also appears in KAoS, but SWRL overcomes it by the extending the set of OWL axioms including horn-like rules. On the other hand, SWRL is not limited to deontic policies as it happens in Rei and KAoS.

5. USING CIM ONTOLOGY WITH SEMANTIC LANGUAGES

The Common Information Model (CIM) is an approach from the DMTF that applies the basic structuring and conceptualization techniques of the object-oriented paradigm to provide a common definition of management-related information for systems, networks, users, and services.

The CIM model is independent of any implementation or specification. However, for an information model to be useful, it must be mapped into some implementation. As Figure 4 showed, CIM can be mapped to several structured specifications.

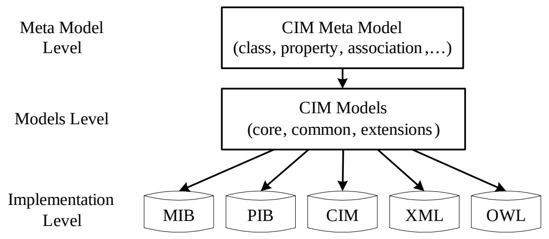
Meta Model Level
CIM Meta Model (class, property, association,…)
Models Level
CIM Models (core, common, extensions)
Implementation Level
MIB
PIB
CIM
XML
OWL

Figure 4. CIM modelling levels

An advantage of CIM is that the model can be mapped to structured specifications such as OWL, which can then be used to define management resources for Web Information System (WIS). Also note that the mapping of CIM to a valid representation for WIS is beneficial, since it permits to model WIS components using the DMTF methodology and hence obtain a standard and interoperable representation of it.

According to our approach, regarding the mapping of CIM into OWL, the main principles identified as part of this process are:

Every CIM class generates a new OWL class using the tag <owl:Class>.

Every CIM generation (inheritance) is expressed using the tag <rdfs:subClassOf>.

Every CIM class attribute is specified using the tag <owl:DatatypeProperty> for literal values or <owl:ObjectProperty> as references to class instances.

Every CIM association is expressed as an OWL class with two <owl:ObjectProperty> where their identifiers (i.e., <rdf:ID>) are the names of the properties of the CIM association; this is the most suitable general-purpose mechanism currently available.

An example of these transformations for the CIM classes related to the user authorization is now presented and explained. CIM defines the classes depicted in Figure 5 to represent the management concepts that are related to an authorization privilege. Privilege is the base class for all types of activities, which are granted or denied to a subject by a target. Authorized-Privilege is the specific subclass for the authorization activity.

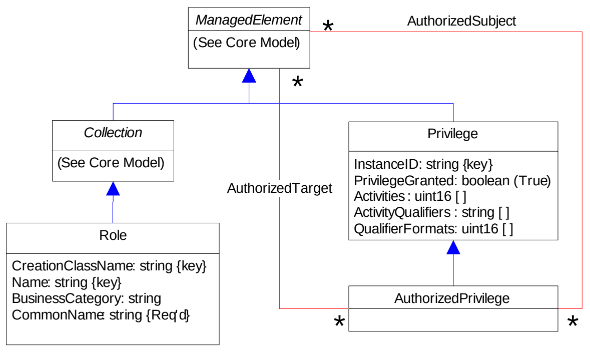

Figure 5. UML diagram of User-Authentication classes

Whether an individual Privilege is granted or denied is defined using the PrivilegeGranted boolean. The association of subjects to AuhorizedPrivileges is accomplished explicitly via the association AuthorizedSubject. The entities that are protected (targets) can be similarly defined via the association AuthorizedTarget. Note that AuthorizedPrivilege and its AuthorizedSubject/Target associations provide a static mechanism to represent authorization policies.

An example of the mapping of these CIM classes to OWL is illustrated in the Figure 6. This example shows a fragment of the mapping of CIM class Privilege and CIM association AuthorizedSubject.

```
<owl:Class rdf:ID=”CIM_Privilege”>
        <rdfs:subClassOf
                rdf:resource=”CIM_ManagedElement”/>
</owl:Class>
<owl:Class rdf:ID=”CIM_AuthorizedSubject”>
        <rdfs:subClassOf rdf:resource=”LogicalEntity”/>
</owl:Class>
<rdf:DatatypeProperty rdf:ID=”InstanceID”>
        <rdfs:domain rdf:resource=”CIM_Privilege”/>
        <rdfs:range rdf:resource=”String”/>
</rdf:DatatypeProperty>
<rdf:DatatypeProperty rdf:ID=”PrivilegeGranted”>
        <rdfs:domain rdf:resource=”CIM_Privilege”/>
        <rdfs:range rdf:resource=”Boolean”/>
</rdf:DatatypeProperty>
<rdf:DatatypeProperty rdf:ID=”Activities”>
        <rdfs:domain rdf:resource=”CIM_Privilege”/>
        <rdfs:range rdf:resource=”Uint16”/>
</rdf:DatatypeProperty>
<rdf:DatatypeProperty rdf:ID=”ActivityQualifers”>
        <rdfs:domain rdf:resource=”CIM_Privilege”/>
        <rdfs:range rdf:resource=”String”/>
</rdf:DatatypeProperty>
<rdf:DatatypeProperty rdf:ID=”QualiferFormats”>
        <rdfs:domain rdf:resource=”CIM_Privilege”/>
        <rdfs:range rdf:resource=”Uint16”/>
</rdf:DatatypeProperty>
<rdf:ObjectProperty rdf:ID=”Privilege”>
        <rdfs:domain rdf:resource=”CIM_AuthorizedSubject”/>
        <rdfs:range rdf:resource=”CIM_ManagedElement”/>
</rdf:ObjectProperty>
<rdf:ObjectProperty rdf:ID=”PrivilegedElement”>
        <rdfs:domain rdf:resource=”CIM_AuthorizedSubject”/>
        <rdfs:range rdf:resource=”CIM_ManagedElement”/>
</rdf:ObjectProperty>
```

Figure 6. A fragment of the mapping of Privilege and AuthorizedSubject into OWL

Note that the ontological representation of CIM (i.e., OWL representation) permits to represent a CIM ontology that can be used in semantic policy languages (e.g., SWRL).

SWRL uses ontology concepts encoded in OWL to build rules. It can be extended with the OWL CIM ontology. For example, rule syntax is illustrated in the Figure 7 related with the case study described earlier.

```
<ruleml:imp>
 <ruleml:_body>
  <swrlx:classAtom>
        <owlx:Class owlx:name="CIM_Role"/>
        <ruleml:var>server</ruleml:var>
  </swrlx:classAtom>
  <swrlx:classAtom>
          <owlx:Class owlx:name="CIM_Role" />
          <ruleml:var>requester</ruleml:var>
  </swrlx:classAtom>
  <swrlx:classAtom>
          <owlx:Class owlx:name="CIM_AuthorizedPrivilege" />
          <ruleml:var>privilege</ruleml:var>
  </swrlx:classAtom>
  <swrlx:individualPropertyAtom swrlx:property="Name">
          <ruleml:var>server</ruleml:var>
          <owlx:Individual owlx:name="#PayServer" />
  </swrlx:individualPropertyAtom>
  <swrlx:individualPropertyAtom swrlx:property="Name">
          <ruleml:var>requester</ruleml:var>
          <owlx:Individual owlx:name="#PayCustomer" />
  </swrlx:individualPropertyAtom>
  <swrlx:individualPropertyAtom swrlx:property="Name">
          <ruleml:var>privilege</ruleml:var>
          <owlx:Individual owlx:name="#GrantedAccess" />
  </swrlx:individualPropertyAtom>
 </ruleml:_body>
 <ruleml:_head>
  <swrlx:classAtom>
          <owlx:Class owlx:name="CIM_AuthorizedTarget" />
          <ruleml:var>authtarget</ruleml:var>
  </swrlx:classAtom>
  <swrlx:classAtom>
          <owlx:Class owlx:name="CIM_AuthorizedSubject" />
          <ruleml:var>authsubject</ruleml:var>
  </swrlx:classAtom>
  <swrlx:individualPropertyAtom swrlx:property="Privilege">
          <ruleml:var>authtarget</ruleml:var>
          <ruleml:var>privilege</ruleml:var>
  </swrlx:individualPropertyAtom>
  <swrlx:individualPropertyAtom
          swrlx:property="TargetElement">
          <ruleml:var>authtarget</ruleml:var>
          <ruleml:var>server</ruleml:var>
  </swrlx:individualPropertyAtom>
  <swrlx:individualPropertyAtom swrlx:property="Privilege">
          <ruleml:var>authsubject</ruleml:var>
          <ruleml:var>privilege</ruleml:var>
  </swrlx:individualPropertyAtom>
  <swrlx:individualPropertyAtom
          swrlx:property="PrivilegedElement">
          <ruleml:var>authsubject</ruleml:var>
          <ruleml:var>requester</ruleml:var>
  </swrlx:individualPropertyAtom>
 </ruleml:_head>
</ruleml:imp>
```

Figure 7. Example of policy representation in SWRL using the CIM ontology

6. CONCLUSIONS

This paper has provided some discussions of the most relevant security-aware semantic specification languages and information models. Our perspective on the main issues and problems of each of them has also been presented, based on different criteria such as their approach or the specification technique they use. It has also presented an approach for using CIM ontology with the semantic languages.

Our future work is being planned to investigate how the CIM information model can be used as ontology for other semantic security policy languages. In this sense the current research work undertaken in the POSITIF EU IST project [12] is gathering requirements of security management in web and information systems and defining, based on the work presented in this paper, a semantic security policy language able to formally define the desired security policy.

7. ACKNOWLEDGMENTS

This work has been partially funded by the EU POSITIF (Policy-based Security Tools and Framework) IST project (IST-2002-002314).

8. REFERENCES

[1] Brickley, D., and Guha, R. V. (2004, January). Rdf vocabulary description language 1.0: Rdf schema. Technical report, W3C Working Draft.

[2] Connolly, D., Dean, M., Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Scneider, P. F., and Stein, L. A. (2003, February). Web ontology language (owl) reference version 1.0. Technical report, W3C Working Draft.

[3] Damianou, N., Dulay, N., et al. (2001). The Ponder Policy Specification Language. Policy 2001: Workshop on Policies for Distributed Systems and Networks. Springer-Verlag.

[4] Distributed Management Task Force, inc. (2005). Common Information Model (CIM) Standards, version 2.9.0.

[5] Kagal, L., Finin, T., and Johshi, A. (2003). A Policy Language for Pervasive Computing Environment. Policy 2003: Workshop on Policies for Distributed Systems and Networks. Springer-Verlag.

[6] Martinez Perez, G., Garcia Clemente, F.J., Gomez Skarmeta, A.F. (2005), Policy-Based Management of Web and Information Systems Security: an Emerging Technology, Idea Group Inc., in press.

[7] OASIS (2004, December). Extensible Access Control Markup Language (XACML), version 2.0, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[8] The Rule Markup Initiative (2004, May). SWRL: A Semantic Web Rule Language Combining OWL and RuleML, version 0.6.

[9] Tonti, G., Bradshaw, J. M., Jeffers, R., Montanari, R., Suri, N., and Uszok, A. (2003). Semantic Web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder. The Semantic Web—ISWC 2003. Proceedings of the Second International Semantic Web Conference. Springer-Verlag.

[10] Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., et al. (2003). KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement. Policy 2003: Workshop on Policies for Distributed Systems and Networks. Springer-Verlag.

[11] W3C. (1999, February). Resource description framework (rdf), data model and syntax. W3C Recommendation.

[12] EU IST POSITIF (Policy-based Security Tools and Framework) Project, http://www.positif.org/