Using RDF for policy specification and enforcement

Barbara Carminati¹, Elena Ferrari¹, Bhavani Thuraisingham²

¹Università dell’Insubria, Como, Italy
{barbara.carminati, elena.ferrari}@uninsubria.it
²National Science Foundation, USA
bthurais@nsf.gov

Abstract

Security issues for the Resource Description Framework (RDF) [RDF] are today crucial due to the key role played by RDF for the semantic web [LEE01]. However, despite its importance, RDF security is still a novel area and very little work has been done so far. In this paper, we make a first step in this direction by proposing an approach for using RDF for policy specification and enforcement.

1. Introduction

Today we are operating in an environment that is very insecure with Trojan horses, viruses and worms. Daily we hear about security breaches on the Internet. As the web evolves into the semantic web [LEE01], there are more and more possibilities for security breaches as we introduce new technologies. Therefore, it is critical that the security be considered right from the beginning when we develop the semantic web. To secure the semantic web, we need to secure all its components such as XML [XML], RDF [RDF], Agents, Databases, Web Services, and Ontologies and ensure the secure interoperation of all these technologies. Security research for the semantic web is just at the beginning. While we have some work carried out in securing XML documents [G00], secure agents, secure databases and secure web services, there is virtually no work on RDF security as well as secure information interoperability. This paper describes our preliminary investigation on RDF security. In particular, we can distinguish between two main issues: that of securing RDF documents and that of using the semantic richness of RDF for expressing security information and for making policy specification and enforcement easier. In this paper, we focus on this latter issue, by showing how RDF can be used for policy specification.

The organization of this paper is as follows. Section 2 provides some background information on RDF. Section 3 shows how RDF can be used for access control policy specification and enforcement. Finally, Section 4 concludes the paper.

2. A Brief Introduction to RDF

RDF (Resource Description Framework) [RDF] is a key component of the semantic web, since it provides a standardized way to describe resource and their relationships. RDF is essentially the foundation for processing metadata. While XML is limited in providing machine understandable documents, RDF handles this limitation. That is, while XML provides syntax and notations, RDF supplements this by providing semantic information in a standardized way. As a result, RDF provides better support for interoperability as well as searching and cataloging.

The basic RDF data model is built on the concepts of resources, properties and statements, which can be represented by means of an intuitive and simple graph model. The term resource denotes anything that can be identified by an URI reference [URI]. For instance, a resource could be a web page or a collection of pages, as well as a person or an organization. In the graph model, the resource is designed by a node containing the corresponding URI reference. Information of the resource can be expressed by way of the property. As an example, a property could be the date of creation of a web page, as well as the author of the page. The properties are associated with resources by means of the property-types, expressed by a URI reference, and graphically represented as an arc connecting the resource with the corresponding URI reference. Information of the resource can be expressed by way of the property. More precisely, the property-types state the relationships between an atomic value and the associated resources (i.e., the atomic property), as well as relationships with other resources, which in turn may have their own properties. The RDF statements can thus be expressed as the resources together with their properties plus the values of the properties. Moreover, a set of RDF statements describing a common scenario is referred to as an RDF description (see Figure 1).

The RDF working groups have defined a formal syntax for representing the graph model into machine-readable files. More precisely, it provides XML syntax for
3. Using RDF for access control policy specification and enforcement

Policy specification is the first step in defining a system able to protect any web resource. Access control policies are basically high level rules regulating how access control should take place. An example of access control policy can be the rule stating that: “Only the creator of a data object can modify it”. The definition of the policies usually results by the analysis, carried out by the Security Administrator (SA) or the owner of the data, of the security requirements of the scenario being protected. Traditionally, such policies are expressed in natural language, without any kind of formalization. Thus, in order to make the access control mechanism able to enforce the specified policies, each policy is implemented as a set of authorizations, that is, a set of tuples \((s, o, a, m)\), stating that subject \(s\) can access object \(o\) under access mode \(m\). When a subject submits an access request to the system, the access control mechanism verifies whether the access can be granted or not by searching among the specified authorizations.

The main drawback of this approach is the lack of tools that help the SA in policy specification and in generating the corresponding authorizations. For instance, it would be nice to have a tool for specifying policies for a general scenario (e.g., the management of research projects) and then to generate, starting from this high level policies, the set of corresponding authorizations for a particular instance of this scenario (e.g., the authorizations regulating the management of research projects for a specific organization). As an example, consider again the policy above introduced, stating that: “Only the creator of a data object can modify it”. This general policy can be instantiated in several contexts. For instance, if we apply it to a medical scenario, data objects can be patient records or medical prescriptions. Thus, examples of entailed authorizations can be (Dr Smith, patient_record123, modify). By contrast, if we consider the same policy applied to technical reports issued by a university, then examples of entailed authorizations can be: (Professor E. Ferrari, TR#78, modify) or (Dr. B.Carminati, TR#76, modify). These simple examples show the need of policy specification tools that consider only the high level semantics of the involved resources and their relationships, leaving out the details of the particular scenario.

In this paper, we make a step in this direction, by showing how RDF can help the policy specification task and the deployment of the specified policies in different scenarios. More precisely, we show how RDF can be exploited in two different phases of the authorization generation process: the first is the specification of the access control policies for a general scenario, hereafter called high level policies, whereas the second is the generation of the authorizations for a specific scenario, starting from the high level policies and an RDF description of the considered domain.

3.1 Overall strategy

The idea underlying our approach is that of using RDF to provide an high level description of different scenarios with similar characteristics, that is, scenarios with similar resource types and semantic relationships among them, and with similar security requirements. Moreover, we use RDF also to specify policies on such general scenario. To better clarify our approach, we use as running example a general scenario for the management of research projects. Figure 1 depicts part of the RDF description for such a scenario. The main resources in the considered scenario are: Project Manager, Administrative Staff, Technical Staff, Funds, and Technical Reports. Technical Staff and Technical Reports. The high level scenario depicted in Figure 1 is general enough to denote a set of more specific scenarios, all conforming to such a general template, which we call scenario instances.
The overall architecture is depicted in Figure 2. The first component is the High level policy generation that, given the RDF description of the general scenario, specifies the corresponding high level policies. In particular, as it will be explained in the next section, this module helps the SA in defining the access control constraints directly on the RDF description, generating the so-called "Security-enhanced RDF description". Starting from this document, the module automatically generates the corresponding policies. Then, the Authorizations entailment module takes as input the RDF description of a scenario instance and the high level policies generated in the previous phase, and generates all the corresponding authorizations. The last component is the Reference monitor that receives as input an access request and checks whether it can be granted or not. In the following sections, we explain the strategies underlying the two main phases, that is, the high level policy generation and the authorization entailment.

3.2 High level policy generation
Starting from the RDF description of a general domain, the SA (or the data owner) has to specify, on the RDF description itself, the access control policies for that domain. In particular, in our approach high level policies can be formalized as tuples of the form (class of subjects, class of objects, set of access modes specification). In particular, class of subjects/objects are the resources appearing in the RDF description of the general scenario, and the last component is a set of tuples of the form (cond_subj, cond_obj, set of access modes) specifying that subjects belonging to the class of subjects and respecting the condition cond_subj can access under the set of specified access modes the objects belonging to the class of objects in the policy and respecting the condition cond_obj. For instance, with the above format it is possible to express policy "All the funds with amount greater than 1000k have to be managed only by administrative staff belonging to a level higher than 5".

Thus, the SA has to specify the class of subjects, the class of objects, and the corresponding access specifications for each policy it would like to enforce. To do that, we suppose that all resources of the RDF model, that is, objects uniquely identified by an URI, can play both the role of the subject and object of a policy. Thus, given an RDF description of a general scenario, the SA can assign to all the resources a role either of subject or object. For instance, considering the example in Figure 1, the resources ‘Technical Staff’ and ‘Technical report’ play the role of subject and object, respectively. To keep track of the role that each resource plays, the RDF description of the general domain is extended, by associating with each resource an atomic property-type “role”, with value “subject” or “object” (see Figure 3).

Once the class of subjects and objects of an high level policy have been specified, the SA has to state the access mode under which that class of subjects can access that class of objects, and, optionally, conditions on the subject and object properties. To model such information, the security-enhanced RDF description is complemented with a new atomic property-type “access rights” that can be associated with all the

![Figure 2](image)

**Figure 2** The overall architecture

On the RDF description of Figure 1, we can define general policies applying to all the scenario instances. An example can be the policy stating that the fund resources must be managed only by the administrative staff.

The basic idea of our approach is that from the RDF description of a general domain, the SA is able to define high level policies for that domain, which are then applied to all the domain instances to derive the set of entailed authorizations.

Once the high level policies for a general domain are specified on its RDF description, and given the RDF description of a specific scenario conforming to that domain, the proposed framework is able to automatically entail all the authorizations implied by the application of the high level policies on the specific scenario.

The overall architecture of our framework is depicted in Figure 2. The first component is the High level policy generation that, given the RDF description of the general scenario, specifies the corresponding high level policies. In particular, as it will be explained in the next section, this module helps the SA in defining the access control constraints directly on the RDF description, generating the so-called "Security-enhanced RDF description". Starting from this document, the module automatically generates the corresponding policies. Then, the Authorizations entailment module takes as input the RDF description of a scenario instance and the high level policies generated in the previous phase, and generates all the corresponding authorizations. The last component is the Reference monitor that receives as input an access request and checks whether it can be granted or not. In the following sections, we explain the strategies underlying the two main phases, that is, the high level policy generation and the authorization entailment.
resources playing the role of object. The value of such
property models the \( (\text{cond}_{\text{subj}}, \text{cond}_{\text{obj}}, p) \)
component of a policy.

Upon the end of the policy specification phase, the
High level policy generator parses the security-
enhanced RDF description and automatically derives
the corresponding high level policies, which are then
stored in the “High level policy base”.

3.3 Authorization Entailment

The Authorization Entailment module receives as
input a scenario instance, and starting from it and
from the high-level policies, it generates all the
authorizations implied by the policies for the specific
scenario. The goal of the scenario instance is to
provide a detailed description of a specific scenario,
conforming to the RDF description of the general
domain of which it is an instance. This detailed
description consists of one or more copies of the
resources and properties in the RDF description of
the general domain, plus some additional information
referring to the instances of each resource. With the
term instance of a resource, we denote a subject/object of a specific domain belonging to the class of subjects/objects denoted by the resource in the general domain description. For instance, with reference to Figure 1, instances of the resource administrative staff are the specific persons that in the considered domain belong to the administrative staff. A domain instance of the RDF description in Figure 1 is given in Figure 4.

More formally, we assume that an RDF description
\( \text{R}_{\text{inst}} \) is a scenario instance of an RDF description
\( \text{R}_{\text{dom}} \), if the following conditions hold: 1) \( \text{R}_{\text{inst}} \)
contains one or more copies of each resource and
property described in \( \text{R}_{\text{dom}} \); 2) \( \text{R}_{\text{inst}} \) preserves the
relationships among resources and properties defined
in \( \text{R}_{\text{dom}} \); 3) for each resource \( R \) in \( \text{R}_{\text{inst}} \)
corresponding to a resource in \( \text{R}_{\text{dom}} \), let \( \text{inst}(R) \) be
the set of instances of \( R \) in the specific domain: then,
for each \( \text{re} \) \( \text{inst}(R), \text{R}_{\text{inst}} \) contains a corresponding
resource connected to \( R \) by the property-type
“http://www.RDFsec.org/RDFsec#belongs”.

We are now ready to describe the authorization
entailment process. The process iteratively considers
the high-level policies defined for the general domain
and computes the set of entailed authorizations for the
scenario instance being considered. An algorithm
performing such task is reported in Figure 5. For the
sake of simplicity, we consider only policies with a
single access mode specification. However, the
algorithm we present can be easily extended for
considering more than one access mode specification.
The algorithm takes as input a scenario instance
\( \text{R}_{\text{inst}} \), an high level policy \( \text{hlp} \), and the authorization
base \( \text{AB} \), and it returns the updated authorization base,

Algorithm 3.1: The authorization Entailment

INPUT: \( \text{R}_{\text{inst}} \): the domain instance; \( \text{hlp} \): an high level policy applied to
\( \text{R}_{\text{inst}} \); \( \text{AB} \): the authorization base

OUTPUT: updated Authorization Base

1. let \( \text{copies}_{\text{subj}} \) be the set of resources in \( \text{R}_{\text{inst}} \) with URI equal to
\( \text{hlp.class.ofsubj} \) and satisfying the condition in
\( \text{hlp.access.spec.cond}_{\text{subj}} \).
2. let \( \text{copies}_{\text{obj}} \) be the set of resources in \( \text{R}_{\text{inst}} \) with URI equal to
\( \text{hlp.class.ofobj} \) and satisfying the condition in
\( \text{hlp.access.spec.cond}_{\text{obj}} \).
3. For each \( cs \in \text{copies}_{\text{subj}} \):
   a. For each \( co \in \text{copies}_{\text{obj}} \) connected with \( cs \):
      i. For each \( s \in \text{inst}(cs) \)
         1. For each \( o \in \text{inst}(co) \)
            a. For each \( am \in \text{hlp.set.of.access.modes} \)
               i. add \((s,o,am)\) to \( \text{AB} \)
               End For
      End For
4. Return \( \text{AB} \)

Figure 4 An RDF description of a specific domain

Figure 5 The authorization entailment algorithm
containing the authorizations entailed by the input policy. The algorithm derives the implied authorizations by iteratively considering each copy of a resource playing the role of subject, with the corresponding copy of the resource playing the role of object. More precisely, the algorithm inserts into the authorization base a new authorization \((s, o, am)\), where \(s\) is an instance of the copy of the resource playing the role of subject and satisfying the condition stated in the input policy, \(o\) is an instance of the copy of the resource playing the role of object and satisfying the condition stated in the input policy, and \(am\) are the access modes in \(hlp\). In the algorithm, we use the dot notation to denote the various components of a policy. As an example of algorithm execution, consider the high level policy
\[
\text{hlp}=(\text{http://www.pm.org/resources#Administrative_staff, http://www.pm.org/resources#Funds}, \{(\text{level}>5, \text{amount}=>1000k, \{\text{modify, read}\})\}), \text{and the scenario instance in Figure 4.}
\]

The first step of the algorithm determines the URI(s) of the resources playing the role of subjects and objects. By using these URI(s) the algorithm determines the copies of the corresponding resources in the scenario instance, which satisfy the conditions in the access mode specification. Referring to the domain instance in Figure 4, the only copy of the resource identified by the URI http://www.pm.org/resources#Administrative_staff and satisfying condition “level>5” is the administrative staff with level 6. Whereas, the copies of the resource identified by the URI http://www.pm.org/resources#Funds and satisfying condition “amount=>1000k” are the two funds with amount 1500k. For each copy of the subject resource, the algorithm determines the corresponding copies of objects. Referring to Figure 4, the algorithm considers the resource associated to the administrative staff with level 6, and the fund with amount greater than 1500k connected to it. In step 3.a.i, the algorithm considers each instance \(cs\) of the subject resource determined in step 3. Then, for each instance \(co\) of the object resource (determined in 3.b.i) and for each access mode \(am\) specified in the access mode specification, the algorithm adds the authorization \((cs, co, am)\) to the authorization base. In our running example, the inserted authorizations are: (Patty, account09327, modify), (Patty, account09327, read), (Bob, account09327, modify), (Bob, account09327, read).

4. Summary and Directions

RDF security is a novel research area and thus very little work has been done so far. To the best of our knowledge, the only work dealing with RDF security is the one by [GOW03]. In [GOW03] the authors propose a framework for enforcing access control on XML documents, which allows a security classification of XML nodes, and associations among nodes. In particular, they exploit RDF to express security information (i.e., security policy and association among XML node). By contrast, in this paper we exploit RDF model to allow the specification of high level policy on a general scenario, and to automatically entail all the implied authorizations for a specific scenario. Another work that has some relationships with our approach is [QIN], where a concept-level access control model has been proposed. The main difference between the two strategies is that in [QIN] the access authorizations are stated on the concepts specified by ontologies, whereas in our approach we specify policies in the underlying level (i.e., RDF).

This paper has explored the relationships between RDF and security. Specifically, we focused on the use of RDF for specifying security policies and related authorizations.

While the ideas discussed in this paper are preliminary, to our knowledge we are the first to discuss such ideas. We need to develop the ideas further. For example, in Section 4 we described security architecture. We need to carry out a detailed design of the various components of the architecture. We need to essentially work with the RDF standards organizations (e.g., W3C) and incorporate policies specification into the standards.

5. References