An Access Control Model for Linked Data

Serena Villata, Nicolas Delaforge, Fabien Gandon, Amelie Gyrard

An Access Control Model for Linked Data*

Serena Villata, Nicolas Delaforge, Fabien Gandon, and Amelie Gyrard

INRIA Sophia Antipolis {firstname.lastname}@inria.fr

Abstract. Linked Open Data refers to a set of best practices for the publication and interlinking of structured data on the Web in order to create a global interconnected data space called *Web of Data*. To ensure the resources featured in a dataset are richly described and, at the same time, protected against malicious users, we need to specify the conditions under which a dataset is accessible. Being able to specify access terms should also encourage data providers to publish their data. We introduce a lightweight vocabulary, called Social Semantic SPARQL Security for Access Control Ontology (S4AC), allowing the definition of fine-grained access control policies formalized in SPARQL, and enforced when querying Linked Data. In particular, we define an access control model providing the users with means to define policies for restricting the access to specific RDF data, based on social *tags*, and *contextual* information.

Keywords: LOD, Security, SPARQL 1.1, Context, Named Graphs

1 Introduction

Linked Data[1] [6] enables us to set links between items in different data sources, and to connect these sources into a single global data space. These data are provided with machine-readable annotations called *metadata*. Metadata have the aim to provide a flexible way to describe things, and how they relate to other things. However, one of the challenges of Linked Data is access control. As underlined by Bizer et al. [2, 6], the datasets are published in the Linked Open Data (LOD) cloud[2] without the addition of any kind of metadata specifying the access control conditions under which the data is accessible.

This paper addresses the research question: *How to define an access control model for Linked Data?* This is important in order to encourage as many data providers as possible to publish data in their own terms, and not only fully public data. The research question breaks down into two sub-questions: (i) *how to define fine-grained access policies?* and (ii) *how to define context-based access policies?*

The issue of defining access control policies for the Web has been addressed by the Web Access Control vocabulary (WAC)[3], which allows the user to specify

* DataLift is funded by the French National Research Agency: ANR-10-CORD-09.

[1] http://www.w3.org/DesignIssues/LinkedData.html

[2] http://richard.cyganiak.de/2007/10/lod/

[3] http://www.w3.org/wiki/WebAccessControl

access control lists (ACL). The ACL are of the form `[acl:accessTo <card.rdf>; acl:mode acl:Read, acl:Write; acl:agentClass <groups/fam#group>]`, which means that anyone in the group `<http://example.net/groups/fam#group>` may read and write `card.rdf`. The WAC vocabulary distinguishes four classes of access control privileges: `Read` (read the content), `Write` (delete or update the content), `Control` (set the ACL for the content), and `Append` (add information at the end of the content). This vocabulary grants the access to a whole RDF document, e.g., `card.rdf`. In this paper, we aim at providing fine-grained access control policies which grant the access to specific RDF data, i.e., the information providers may want to restrict the access to a few named graphs [4]. Moreover, we enable the requester to submit any SPARQL query, and resource provider to further specify the access control privilege granted to the user, and we distinguish the `Delete` and `Update` classes of privileges, included in the `Write` WAC class.

We introduce the Social Semantic SPARQL Security for Access Control vocabulary (S4AC), a lightweight ontology which allows the information providers to specify fine-grained access control policies for their RDF data (Figure 1). At the core of S4AC is the Access Condition which is a SPARQL 1.1. `ASK` clause that specifies the condition to be satisfied in order to grant the access to a named graph. Moreover, the information providers can define Access Conditions based on *tags* which restrain the conditions to named graphs tagged with such tags, e.g., named graphs tagged "friends", "amici", "ami". The conditions can be bound on specific values to provide an access evaluation context, e.g., `<''?user'', <http://myExample.net#sery>>` where the URI of the user is bound to `<http://myExample.net#sery>`. Finally, the Access Condition is associated with a temporal validity. The Access Privilege defines which kind of privilege is granted to the user satisfying the Access Conditions, e.g., `s4ac:Update` grants the user the privilege to modify the requested named graph.

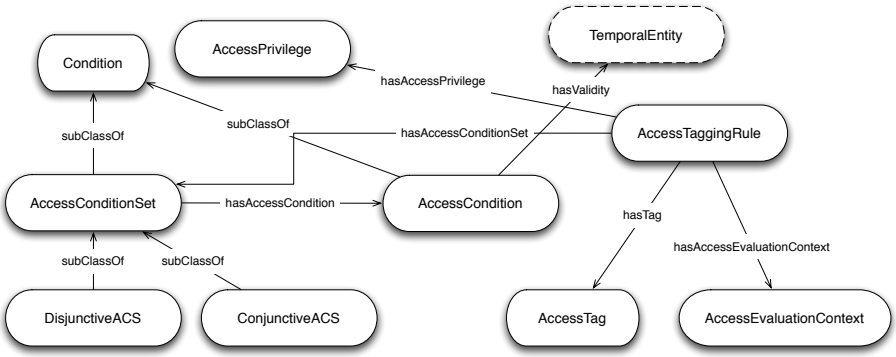

Fig. 1. An overview of the S4AC Ontology.

A key feature of our approach is to rely only on Semantic Web languages. As a consequence, our access control model is platform independent, and can be used

by any kind of system based on those languages. In particular, the semantics of our access control policies is grounded in SPARQL 1.1[4] `ASK` queries. Relying on SPARQL semantics, our model allows the user to submit arbitrary queries while enforcing fine-grained access rules on the results he will receive. If the result of the `ASK` query is *true*, then the user is provided with the information he requires. If the result is *false*, then the model returns to the user a denial coupled with one or more rule labels explaining the denial.

The reminder of the paper is organized as follows: Section 2 presents a use case of the proposed access control model. Section 3 introduces the S4AC ontology, and it details and analyses the access control polices which can be defined using our model. Related work and conclusions end the paper.

2 The DataLift use case

The DataLift project[5] aims at providing a platform to ease the publication and interlinking of datasets on the *Web of Data*. Figure 2 illustrates the Access Control Manager (ACM) which is the core module of our access control model. We now describe the features of the Access Control Manager first from the point of view of the user, and then from the point of view of the system.

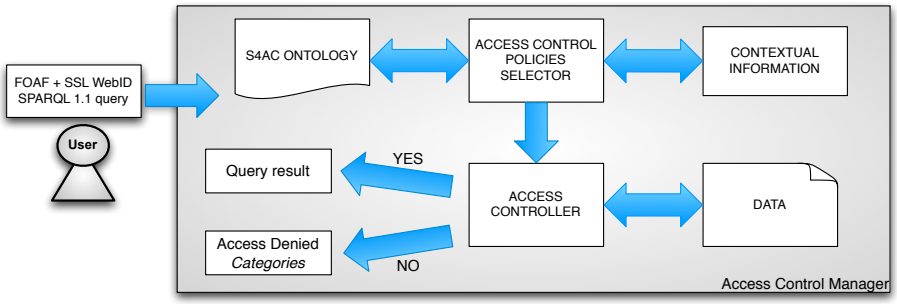

Fig. 2. The Access Control Manager.

Consider a user who wants to access some of the information published on the Web of Data by means of the DataLift platform. The user first authenticates to the ACM of the platform using the WebID protocol[6]. The user queries the datasets using a SPARQL 1.1 `SELECT`, `MODIFY`, `INSERT`, or `DELETE` query[7], depending on the kind of operation the user intends to perform on the requested

[4] http://www.w3.org/TR/sparql11-query/
[5] http://datalift.org/en/
[6] http://www.w3.org/wiki/WebID
[7] The `MODIFY`, `INSERT`, and `DELETE` queries are provided by SPARQL 1.1. See http://www.w3.org/TR/2009/WD-sparql11-update-20091022/.

data. The ACM returns the user an answer of the kind YES/NO together with the query result, or the labels of the rules that caused the failure.

The ACM receives an authentication request from the user by means of the WebID protocol. Then, after a successful authentication, it receives the query of the user. The ACM has the aim to grant or restrict the access to the RDF data published using the DataLift platform, where each SPARQL endpoint manages its requests. Once the request of the user is received, the ACM selects, by means of the module called Access Control Policies Selector (ACPS), which policy applies, depending on the requested operation. For instance, if the user submitted a `MODIFY` query, then the ACPS identifies all the policies which apply, and concern an `Update` access privilege. The ACPS handles two kinds of operations: (i) it checks the S4AC ontology in order to identify which access conditions apply, and (ii) it checks whether the contextual information, e.g., the temporal validity of the selected policies, is satisfied. Note that we check whether the contextual constraints hold before checking the reminder of the policy. If the contextual constraints are not satisfied, we already know that the access will not be granted. After the identification of the policies, and a positive checking of the contextual constraints, the Access Controller module matches the policies according to the user's profile to test what he can access. The Access Controller addresses a SPARQL `ASK` query which returns *true* if the access to the named graph is granted to the user. The Access Controller selects the set of named graphs to which the user has access, and queries this dataset adding `FROM, FROM NAMED` to the user's query. If the answer is *false*, then the Access Controller returns a failure, coupled with the categories causing the failure. These categories are provided to the Access Controller by the ACPS when it checks the ontology.

3 Access control for Linked Data

3.1 Social Semantic SPARQL Security for Access Control Ontology

The Social Semantic SPARQL Security for Access Control Ontology (S4AC), online at `http://ns.inria.fr/s4ac/v1#`, is detailed in Figure 3. One of the key features of our access control approach is to be integrated with the models adopted in the fields of the Social Web, and of the Web of Data. In particular, S4AC reuses concepts from SIOC[8], SCOT[9], NiceTag[10], WAC, TIME[11], and the access control model as a whole is grounded on further existing ontologies, as FOAF[12], Dublin Core[13], and RELATIONSHIPS[14].

The main class of the S4AC ontology is the class *AccessCondition*, which is a subclass of the class *Condition*, itself a subclass of `sioc:Item`.

[8] http://rdfs.org/sioc/spec/
[9] http://scot-project.net/
[10] http://ns.inria.fr/nicetag/2010/09/09/voc.html
[11] http://www.w3.org/TR/2006/WD-owl-time-20060927/
[12] http://xmlns.com/foaf/spec/
[13] http://dublincore.org/documents/dcmi-terms/
[14] http://vocab.org/relationship/.html

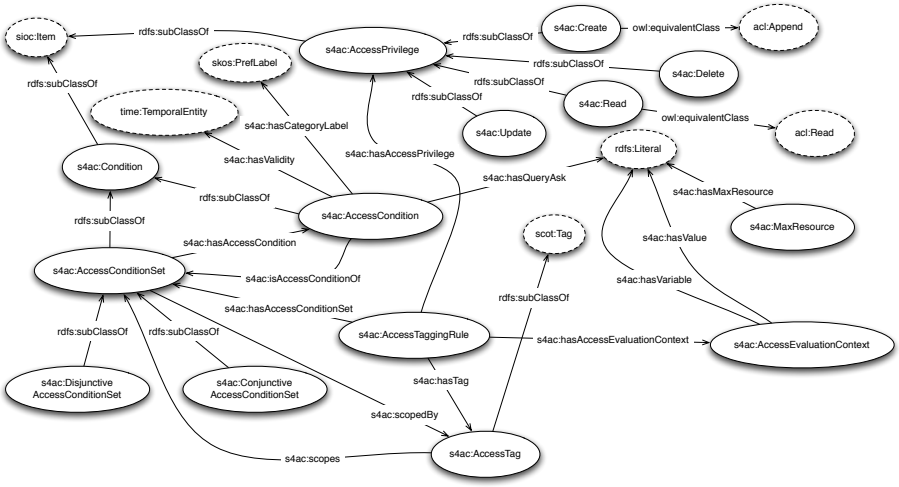

Fig. 3. The S4AC Ontology.

Definition 1. *An Access Condition (AC) is a SPARQL 1.1* `ASK` *query. If a solution exists, the* `ASK` *query returns true, and the Access Condition is said to be verified. If no solution exists the* `ASK` *query returns false, and the Access Condition is said not to be verified.*

The *Access Condition* grants or restricts the access to the data. If the `ASK` returns *true*, the access is granted to the user. In order to return the user a more informative answer if the access is denied, we introduce the property *hasCategoryLabel*. This property allows to associate to each *AC* one or more natural language labels which "identify" the access condition, and they are returned to the user to provide him the reasons of the denial. We cannot return the user all the access conditions, because this would make him aware of the policies of the provider. The *AccessCondition* defines the property of the access polices *hasValidity*. It allows to define the validity of an Access Condition. Thanks to the use of the concept `time:TemporalEntity`, the validity can be expressed in various ways: valid from/through a specific date/time, or valid in a specific interval of time. This property is used to express policies in which not only the identity of the user requesting the data is checked, but also the contextual information related to the time in which the request is performed. A further class is *MaxResource* which defines the number of times the user can access all or a specified named graph. This class has the property *maxOnResource* which is used to precise which resource is accessible by a limited number of accesses.

Definition 2. *An Access Evaluation Context (AEC) is a list L of predetermined bound variables of the form* $L = (\langle var_1, val_1\rangle, \langle var_2, val_2\rangle, \ldots, \langle var_n, val_n\rangle)$ *that is turned into a SPARQL 1.1 Binding Clause to constrain the* `ASK` *query evaluation when verifying the Access Conditions.*

The *AEC* is represented in the ontology as the class *AccessEvaluationContext* which has two properties, *hasVariable* and *hasValue*, which are respectively the variable, and the value to which the variable is bound. It is used to provide a standard evaluation context to the access conditions, e.g., requesting user, resource provider. Consider the following example: `L=(<''?resource'', ''<http://MyExample.net#doc>''>,<''?user'',''<http://MyExample.net#sery>''>)`. This list can be used to generate an additional Binding Clause for the access conditions of the form: `BINDINGS ?resource ?user {(<http://MyExample.net#doc>, <http://MyExample.net#sery>)}`.

Definition 3. *An Access Condition Set (ACS) is a set of Access Conditions.*

The *AccessConditionSet* class has a property *hasAccessCondition* which identifies which Access Conditions form the ACS. Two subclasses of *AccessConditionSet* are introduced: conjunctive, and disjunctive ACS.

Definition 4. *A Conjunctive Access Condition Set (CACS) is a logical conjunction of Access Conditions of the form* $CACS = AC_1 \wedge AC_2 \wedge \ldots \wedge AC_n$. *A CACS is verified if and only if every access conditions it contains is verified.*

Definition 5. *A Disjunctive Access Condition Set (DACS) is a logical disjunction of Access Conditions of the form* $DACS = AC_1 \vee AC_2 \vee \ldots \vee AC_n$. *A DACS is verified if and only if at least one of the access conditions it contains is verified.*

Definition 6. *An Access Tagging Rule (ATR) is a triple* $R = \langle ACS, TagSet, Bindings \rangle$ *where ACS is an Access Condition Set, TagSet is a set of tags* $\{tag_1, tag_2, \ldots, tag_m\}$, *and Bindings is an Access Evaluation Context. An ATR is verified for a named graph tagged with one or more tags from TagSet if and only if the ACS is verified for that named graph.*

An ATR declares that the access conditions in the ACS apply to any named graph tagged with one or more tags from *TagSet*. Notice that the ACS may be reduced to a single access condition. In this case, the ATR is said to be verified if and only if the single access condition is verified. Note that *TagSet* may be empty, in which case the ATR applies to any named graph. The class *AccessTaggingRule* has four properties: *hasAccessConditionSet*, associating an ACS to the ATR, *hasTag*, providing a set of tags to the ATR, *hasAccessEvaluationContext*, associating to the ATR the AEC, i.e., the bindings applied to the rule, and *hasAccessPrivilege*. The *hasAccessPrivilege* property defines the access privilege the user is granted to: *Read*, *Create*, *Update*, *Delete*. We expand the `acl:Write` class, which is used for every kind of modification on the content, and we allow fine-grained access control privileges. The class *AccessTag*, used to define the set of tags, is a sub-class of `scot:Tag`.

3.2 The access control policies

We show now which kind of access control policies are enabled by the proposed access control model. Consider the policy defined below: the data provider defines

an access policy such that only his named graphs tagged with tag "family" are constrained by the access condition which grants the access to those users which have a *hasParent* relationship with the provider, i.e., the parents of the provider. The Access Condition Set is composed only by one access condition, thus this is the only one which needs to be evaluated. The access privilege is `Update`. Thus, given a `MODIFY` query of the user, if he is granted with the access, then he is allowed to `Update` the requested named graphs. Concerning the contextual information, the Access Tagging Rule grants the access to the user if the date of the access is after December 31^{th} at 23:59. If the user is not granted with the access then the label the system returns him together with the failure message is "parents", to explain that the reasons of the failure have to be associated to the fact that the user is not a parent of the provider; the system does not return the entire policy to the user.

```
<http://MyExample.net/expolicies>
 a s4ac:AccessTaggingRule;
  s4ac:hasAccessConditionSet [
   s4ac:hasAccessCondition [
     s4ac:hasValidity [
       time:hasBeginning [
         time:inXSDDateTime 2011-12-31T23:59:00
       ];
     ];
     s4ac:hasCategoryLabel skos:PrefLabel ''parents''@en;
     s4ac:hasQueryAsk [
       ASK { ?resource dcterms:creator ?provider .
       ?provider rel:hasParent ?user }
     ];
   ];
  ];
  s4ac:hasAccessPrivilege s4ac:Update;
  s4ac:hasTag scot:Tag ''family''@en.
```

The table below presents some examples of the `ASK` queries which may be associated with the access conditions. *Cond1* grants the access to those users who have a relationship of kind "colleagues" with the provider. *Cond2* grants the access to the friends of the provider, and *Cond3* extends this access condition also to the friends of friends. *Cond4* is more complicated[15]. It grants the access to those users that are marked with a specified tag. For specifying the tag, we use the NiceTag ontology. Also negative access conditions are allowed, where we specify which specific user cannot access the data. This is expressed, as shown in *Cond5*, by means of the `FILTER` clause, and the access is granted to every user except *sery*. *Cond6* expresses an access condition where the user can access the data only if he is a minimum lucky, e.g., one chance out of two. Finally, *Cond7* grants the access to those users who are members of at least one group the provider belongs to.

An example of conjunctive ACS is as follows: $CACS_{friends-but-sery} = Cond_2 \wedge Cond_5$, where the access is granted to the users who are friends of the provider,

[15] The GRAPH keyword is used to match patterns against named graphs.

cond1	`ASK { ?resource dcterms:creator ?provider . ?provider rel:hasColleague ?user }`
cond2	`ASK { ?resource dcterms:creator ?provider . ?provider rel:hasFriend ?user }`
cond3	`ASK { ?resource dcterms:creator ?provider . ?provider rel:hasFriend{1,2} ?user }`
cond4	`ASK { ?resource dcterms:creator ?provider . ?provider dcterms:creator ?g . GRAPH ?g { ?user nicetag:hasCommunitySign ?tag }}`
cond5	`ASK { FILTER(! (?user= <http://MyExample.net#sery>))}`
cond6	`ASK { FILTER(random()>0.5) }`
cond7	`ASK { ?resource dcterms:creator ?provider . ?provider sioc:member_of ?g . ?user sioc:member_of ?g }`

but the user `<http:MyExample.net#sery>`, even if friend of the provider, cannot access the data. An example of disjunctive ACS is $DACS_{colleagues-or-friends} = Cond_1 \vee Cond_2$, where it is ensured that the users who are colleagues or friends of the provider are allowed to access the data.

The ATR detailed above can be constrained to a wider set of tags such as $ATR_{parents} = \langle Cond, \{''parent'','' parents'','' family'','' relatives''\}, \emptyset\rangle$ where no AEC is provided. Further examples of ATRs are: (i) $ATR_{friends} = \langle Cond_2, \{''friends'','' amici'','' ami''\}, \emptyset\rangle$ where the access condition constrains the access to friends, and three tags are provided without an AEC; (ii) $ATR_{group} = \langle Cond_7, \{''common'','' group'','' close''\}, \emptyset\rangle$ is the same for the belonging to the group of the provider; (iii) $ATR_{hiking} = \langle Cond_4, \emptyset, \langle''?tag'','' hiking''\rangle\rangle$ where the user can access the data if he is tagged with tag "hiking" in the graph created by the provider; (iv) $ATR_{fun} = \langle DACS_{colleagues-or-friends}, \{''fun'','' funny'','' :-)''\}, \emptyset\rangle$ where the user can access the data if the disjunctive ACS above is satisfied on the named graphs tagged with these three tags.

The prototype under development relies on the SPARQL query engine KGRAM/CORESE[16]. Briefly, the system uses the Binding SPARQL 1.1 to substitute the variable *?resource* with the URI of the named graphs to be accessed. The query is executed to obtain all the ATRs associated with the named graphs, and the data provider. CORESE returns these ATRs which contain the ACS. The `ASK` queries inside the single AC are executed on CORESE, and the returned booleans are conjunctively or disjunctively evaluated to grant or deny the access.

4 Related work

Sacco and Passant [9] present a Privacy Preference Ontology (PPO), built on top of WAC, in order to express fine-grained access control policies to an RDF file. They also specify the access queries with a SPARQL `ASK`, but their vocabulary does not consider the temporal validity of the privacy preferences, and

[16] http://www-sop.inria.fr/edelweiss/software/corese/

the number of accesses allowed for each named graph. They rely entirely on the WAC vocabulary without distinguishing different kinds of `Write` actions. Their model does not allow to specify set of tags to limit the application of the policies to the named graphs marked with those tags, and to specify conjunctive and disjunctive sets of privacy preferences. Muhleisen et al. [8] present a policy-enabled server for Linked Data called PeLDS, where the access policies are expressed using a descriptive language called PsSF, based on SWRL[17]. They distinguish only `Read` and `Update` actions, and they do not consider contextual information. Moreover, the system is based on an ontology of the actions that can be performed on the datasets, i.e., *Action*, *Rule*, *TriplePattern*, no further description is provided in [8].

Giunchiglia et al. [5] propose a Relation Based Access Control model (*RelBAC*), providing a formal model of permissions based on description logics. They require to specify who can access the data, while in our model and in [9] the provider can rely on specifying the attributes the user must satisfy. The Access Management Ontology (AMO) [3] defines a role-based access control model. Such a kind of role-based access control model applied to the world of Linked Data does not provide enough flexibility since it again needs to specify who can access the data. Abel et al. [1] present a model of context-dependent access control at triple level, where also contextual predicates are allowed, e.g., related to time, location, credentials. The policies are not expressed using Web languages, but they introduce an high level syntax then mapped to existing policy languages. They enforce access control as a layer on top of RDF stores. After the evaluation of the contextual information, the queries are expanded, and then sent to the database. Hollenbach and Presbrey [7] present a system where the users can define access controls on RDF documents, and these access controls are expressed using the WAC. Our model extends WAC for allowing the construction of more fine-grained access control policies.

5 Conclusions

In this paper, we introduce a fine-grained model of access control for Linked Data. We rely only on Semantic Web languages, namely SPARQL 1.1 queries, Update language, and Binding Clause. We present the S4AC vocabulary which allows to define various kinds of fine-grained access policies on named graphs. These policies involve both social aspects of the user who wants to access the data, e.g., social relationship with the provider, being member of a group, being tagged with a specific tag, and contextual information, e.g., the day in which the request is performed is in a particular time interval, the user is allowed to access the named graph for five times at most. Policies are evaluated together with a set of tags, which restrain the policies on data tagged in such a way, and an evaluation context which binds the variables of the query to specific values. Moreover, we introduce the four access privileges as defined by the C.R.U.D.,

[17] http://www.w3.org/Submission/SWRL/

and we map them with the SPARQL 1.1 query to identify the policies regarding this privilege which are defined on the requested named graph.

There are different research lines for future work. First, a prototype of the Access Control Manager is under definition together with a user-friendly interface allowing also non-expert users to define their own access terms. Our prototype for the DataLift platform will show a real world application of the proposed model with the aim to test its effectiveness. Second, we plan to introduce delegation in the model, in order to allow the provider to delegate some authority. An open issue remains whether this kind of delegation involves also the authority to modify the access policies defined by the provider. Third, we plan to introduce the licenses, e.g., Creative Commons[18] and Waivers[19], as a further description of the datasets. These licenses then have to be returned together with the requested data, even if the user does not ask explicitly for this information. This is needed to allow data providers to open publish their datasets together with their own terms of reuse.

References

1. Abel, F., Coi, J.L.D., Henze, N., Koesling, A.W., Krause, D., Olmedilla, D.: Enabling advanced and context-dependent access control in rdf stores. In: Proceedings of the 6th International Semantic Web Conference (ISWC-2007), LNCS 4825. pp. 1–14 (2007)
2. Bizer, C., Heath, T., Berners-lee, T.: Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems 5, 1–22
3. Buffa, M., Faron-Zucker, C., Kolomoyskaya, A.: Gestion sémantique des droits d'accès au contenu : l'ontologie AMO. In: Yahia, S.B., Petit, J.M. (eds.) EGC. Revue des Nouvelles Technologies de l'Information, vol. RNTI-E-19, pp. 471–482. Cépaduès-Éditions (2010)
4. Carroll, J.J., Bizer, C., Hayes, P.J., Stickler, P.: Named graphs. J. Web Sem. 3(4), 247–267 (2005)
5. Giunchiglia, F., Zhang, R., Crispo, B.: Ontology driven community access control. In: Proceedings of the 1st Workshop on Trust and Privacy on the Social and Semantic Web (SPOT-2009) (2009)
6. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space (1st edition), Synthesis Lectures on the Semantic Web: Theory and Technology, vol. 1:1. Morgan & Claypool (2011)
7. Hollenbach, J., Presbrey, J., Berners-Lee, T.: Using RDF Metadata To Enable Access Control on the Social Semantic Web. In: Proceedings of the Workshop on Collaborative Construction, Management and Linking of Structured Knowledge (CK-2009) (2009)
8. Muhleisen, H., Kost, M., Freytag, J.C.: SWRL-based Access Policies for Linked Data. In: Proceedings of the 2nd Workshop on Trust and Privacy on the Social and Semantic Web (SPOT-2010) (2010)
9. Sacco, O., Passant, A.: A Privacy Preference Ontology (PPO) for Linked Data. In: Proceedings of the 4th Workshop about Linked Data on the Web (LDOW-2011) (2011)

[18] http://creativecommons.org/ns

[19] http://vocab.org/waiver/terms/.html