The TriQL.P Browser: Filtering Information using Context-, Content- and Rating-Based Trust Policies

Christian Bizer[1], Richard Cyganiak[1], Tobias Gauss[1], and Oliver Maresch[2]

[1] Freie Universität Berlin, Germany
chris@bizer.de
richard@cyganiak.de
tobias.gauss@web.de
[2] Technische Universität Berlin, Germany
Oliver-Maresch@gmx.de

Abstract. The TriQL.P browser is a general purpose RDF browser that supports users in exploring RDF datasets containing information from multiple information sources. Information can be filtered using a wide range of user-definable trust policies. Policies can be based on information context, information content, information or information source ratings, and on the presence or absence of digital signatures. In order to help users understand the filtering decisions, the browser can explain why a piece of information fulfils the selected trust policy.

1 Trust Policies for the Semantic Web

The Semantic Web is an open, dynamic network of independent information providers all having different views of the world, different levels of knowledge, and different intentions. Thus, information found on the Semantic Web has to be seen as claims rather than as facts. Before using these claims, the information consumer has to evaluate their trustworthiness and determine the subset which he wants to trust for his specific task.

In everyday life, we use a wide range of trust assessment policies for evaluating the trustworthiness of information: We might trust Andy on restaurants but not on computers, trust professors on their research field, believe foreign news only when it is reported by several independent sources and buy only from sellers on eBay who have more than 100 positive ratings.

Which policy is chosen depends on the specific task, our subjective preferences, our past experiences and the trust relevant information available. For tasks which are economically relevant to the information consumer he might require a very strict trust policy, involving for example recommendations by people he knows. For other tasks, a looser policy like 'Accept all information that has been asserted by at least two independent information providers, no matter who they are.' might be acceptable.

The future Semantic Web is supposed to be a dense mesh of interrelated information, similar to the information perception situation we face in the offline

world. Thus, we argue, a trust policy framework for the Semantic Web can and should support a similarly wide range of trust policies as used offline [4].

Every trust policy employs one or more trust assessment methods. These methods can be classified into three categories:

Rating-Based Methods include rating systems like the one used by eBay and Web-Of-Trust mechanisms. Most trust architectures proposed for the Semantic Web so far fall into this category [1][11]. The general problem with these approaches is that they require explicit and topic-specific trust ratings. For many application domains, providing such ratings and keeping them up-to-date puts an unrealistically heavy burden on information consumers.

Context-Based Methods use meta-data about the circumstances in which information has been claimed, e.g. who said what, when and why. They include role-based trust methods, using the author's role or his membership in a specific group, for trust decisions. Example policies from this category are: 'Prefer product descriptions published by the manufacturer over descriptions published by a vendor' or 'Distrust everything a vendor says about its competitor.' Context-based trust mechanisms do not require explicit ratings, but rely on the availability of background information. Within many Semantic Web application areas, such background information might be available.

Content-Based Methods do not use meta-data about information, but rules and axioms together with the information content itself and related information about the same topic published by other authors. Example policies following this approach are: 'Believe information which has been stated by at least 5 independent sources.' or 'Distrust product prices that are more than 50% below the average price.'

2 The TriQL.P Browser

The TriQL.P browser is a general purpose RDF browser which shows how Semantic Web content can be filtered using a wide range of trust policies, combining methods from all three categories described above.

The TriQL.P browser is based on the Piggy Bank extension for the Firefox browser [10]. Piggy Bank extracts Semantic Web content from Web pages as users browse the Web. On websites where Semantic Web content is not available, Piggy Bank can invoke screen-scrapers to re-structure content into Semantic Web format. The extracted information can be browsed, sorted and searched using a comfortable user-interface, and saved into a local repository for future reference and aggregation.

In addition to the functionality provided by Piggy Bank, the TriQL.P browser gives users the ability to:

- collect provenance meta-data together with information from the Web;
- import information aggregated from multiple sources by a third party into the local repository using the RDF/XML, TriX [7] and TriG [3] syntaxes;
- load trust policy suites containing sets of policies;

– filter information in the local repository using these policies;
– explain on demand why displayed information fulfils a selected policy.

Figure 1 shows the user interface of the TriQL.P browser. Information items from the local repository are displayed on the left-hand side. The policy selection box on the right side allows users to select a policy from the policy suite currently loaded. After selecting a policy, the left-hand view updates to show only information matching this policy. There is a 'Oh, yeah?'-button [2] next to each piece of information. Pressing this buttons opens a new window with an explanation why the piece of information fulfils the selected trust policy.

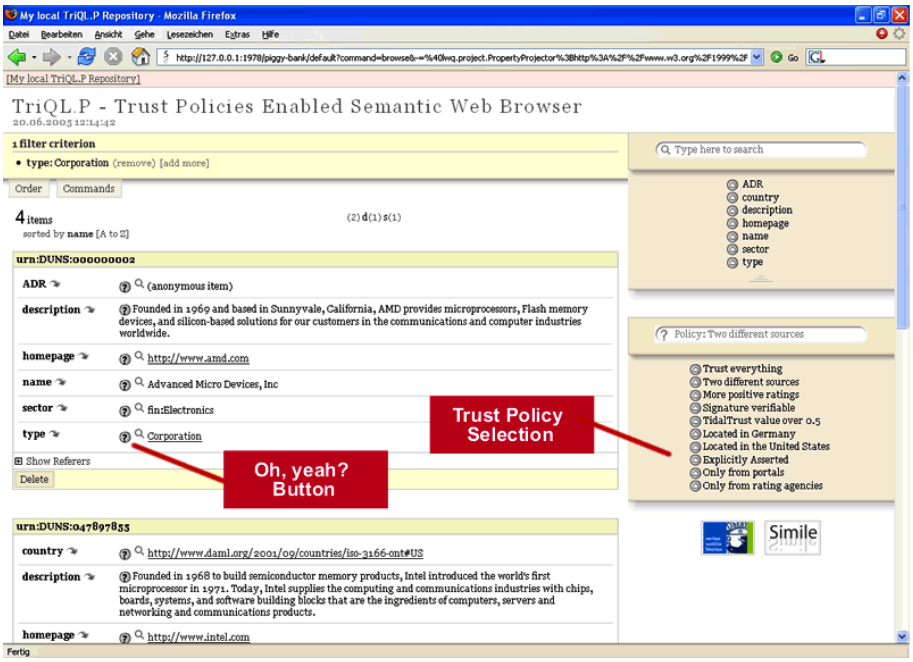

Fig. 1. The TriQL.P user interface. The user selects a trust policy from the right-hand box. The left-hand view updates to show only matching information. The 'Oh, yeah?' buttons open new windows with explanations why a piece of information fulfils the selected policy.

Figure 2 shows an explanation why information information about Peter Smith's email address fulfils the policy 'Trust only information that has been asserted by at least two different sources.'

Figure 3 shows an explanation why a news article fulfils the policy 'Trust only information from information providers who have a Tidal Trust score above 0.5'. The Tidal Trust metric calculates trust scores by determining shortest paths between individuals in a social network of weighted trust statements and calcu-

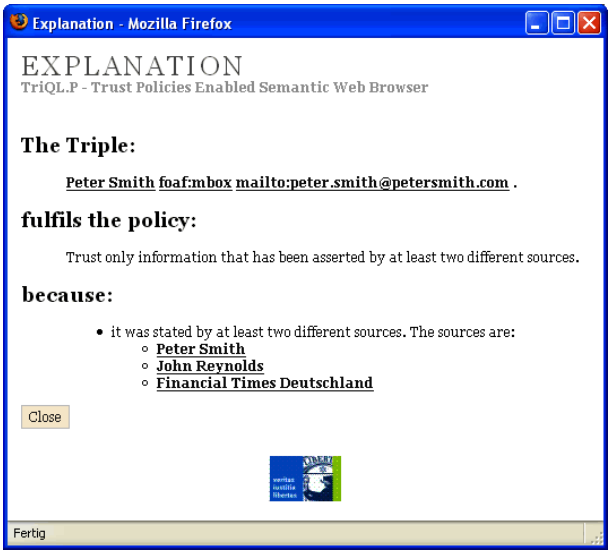

Fig. 2. The explanation window. This explanation establishes why information about Peter Smith's email address fulfils the policy 'Asserted by at least two different sources'.

lating its weighted average [11]. The explanation generated for this trust metric shows the calculation result and contains details about the calculation steps and the information sources used, allowing an information consumer to comprehend the calculations at different levels of detail.

The TriQL.P browser is pretty flexible in rendering explanations for different policies. An explanation for the policy 'Trust only information providers which are working for at least two projects about a specific topic' would contain the list of projects for each information provider. An explanation for the policy 'Trust only information that has been signed by the information providers' would contain details about the signature verification process[3].

The following sections explain how information collected from different sources is represented within the TriQL.P browser, how trust policies are expressed and applied, and how explanations are generated.

3 Representing Information

The TriQL.P browser uses Named Graphs [5] as internal data model. Named Graphs are a slight extension of the RDF abstract syntax and provide well-defined semantics for the attachment of provenance information and other meta-data to RDF graphs.

[3] Various policies and corresponding explanations are found at http://www.wiwiss.fu-berlin.de/suhl/bizer/TriQLP/browser/

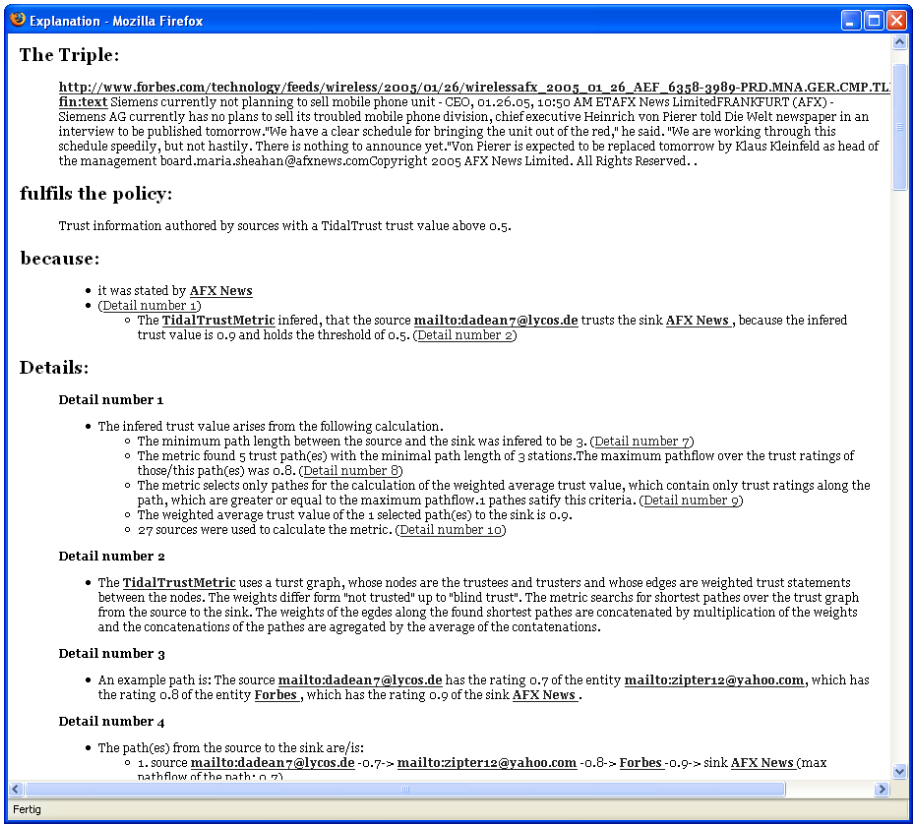

Fig. 3. Explanation for the policy 'Trust only information from information providers who have a Tidal Trust score above 0.5'

Such provenance information can be expressed with the Semantic Web Publishing Vocabulary (SWP) [6]. SWP also provides terms to indicate whether a graph is asserted or quoted and to attach digital signatures to it.

Whenever the browser saves information from a webpage into the local repository, it creates a new named graph for this visit of the page and stores the current timestamp, the URL of the page and the authority (website URL) together with the actual information. The following example shows the browser's internal representation of information about Peter Smith collected from the URL `http://www.bizer.de/myFriends.htm`, together with the recorded provenance information. The example uses the TriG syntax [3].

```
<urn:uuid:8c845860-dce7-11d9-b9c0-00112ff60c7f> {
    ex:PeterSmith a foaf:Person ;
        foaf:name "Peter Smith" ;
        foaf:mbox <mailto:peter.smith@petersmith.com> .
```

```
<urn:uuid:8c845860-dce7-11d9-b9c0-00112ff60c7f>
    swp:assertedBy
       <urn:uuid:8c845860-dce7-11d9-b9c0-00112ff60c7f> ;
    swp:authority <http://www.bizer.de> ;
    dc:date "2005-06-14T17:18:10+02:00" ;
    swp:savedFrom <http://www.bizer.de/myFriends.htm> . }
```

4 Expressing and Applying Policies

The TriQL.P browser displays information from a virtual trusted graph which contains a subset of the triples from all named graphs stored in the local repository. The browser's trust evaluation layer uses trust policies to determine which triples from the untrusted named graphs are promoted into the virtual trusted graph. The decision is made on a triple-by-triple basis, although many policies are written to accept or reject entire graphs.

The heart of every trust policy is a TriQL.P query. TriQL.P is a query language similar to but predating SPARQL [14]. In addition to basic graph pattern matching, TriQL.P offers two language constructs which are especially useful for expressing trust policies: COUNT() for formulating quantity conditions and METRIC() as an open interface to different rating metrics.

An example TriQL.P query is shown below. When executed against the untrusted repository, it selects all triples (bound to the variables `?SUBJ`, `?PRED` and `?OBJ`) which are asserted by at least two authorities.

```
SELECT ?SUBJ, ?PRED, ?OBJ
WHERE ?GRAPH (?SUBJ, ?PRED, ?OBJ)
              (?GRAPH swp:assertedBy ?warrant .
               ?warrant swp:authority ?authority)
AND COUNT(?authority) >= 2
```

In order to associate explanation templates with individual graph patterns and to provide additional metadata about a policy, TriQL.P queries are divided into single graph patterns and constraints and are recombined using the TPL - Trust Policy Language. The example below shows a TPL policy built from the query above:

```
:Policy6 rdf:type tpl:TrustPolicy ;
     tpl:policyName "Asserted by at least two sources" ;
     tpl:policyDescription "Trust only information that has
              been asserted by at least two different sources." ;
     tpl:textExplanation "it was stated by at least two
              different sources. The sources are:" ;
     tpl:graphPattern [
         tpl:pattern "(?GRAPH swp:assertedBy ?warrant .
                       ?warrant swp:authority ?authority)";
```

```
        tpl:textExplanation "@@?authority@@" ; ] ;
    tpl:constraint "COUNT(?authority) >= 2" .
```

The display layer retrieves information from the trust evaluation layer using *find* queries (queries for all triples matching a triple pattern where subject, predicate and object may be wildcards) against the virtual trusted graph.

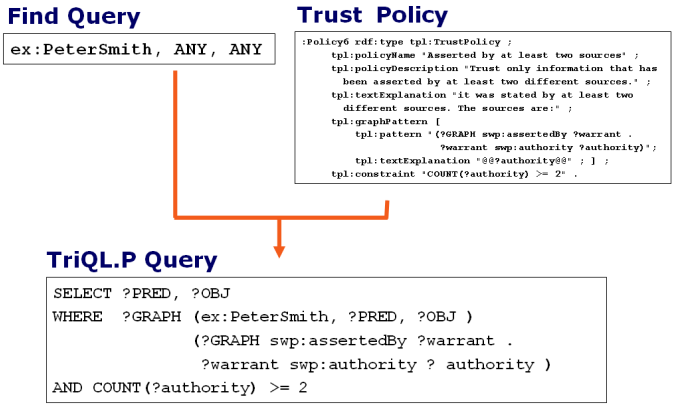

Fig. 4. A find query asking for all information about ex:PeterSmith is combined with a trust policy resulting into a complete TriQL.P query.

To display all information about the resource `ex:PeterSmith`, the following steps are performed:

1. The display layer sends a *find* query and a policy URI to the trust evaluation layer.
2. The engine combines the *find* query with the graph patterns and constraints contained within the policy into a complete TriQL.P query. The variable `?SUBJ` is pre-bound to the value `ex:PeterSmith`.
3. The TriQL.P query is executed against the untrusted repository.
4. RDF triples are created from the variables `?SUBJ`, `?PRED` and `?OBJ` of every query solution, and sent back to the display layer.

The trust evaluation layer caches the values bound to all other query variables, like `?warrant` and `?authority` in the example. They may be used later to generate explanations.

TriQL.P offers an open interface for rating metrics that cannot be expressed as graph patterns. The following example shows how the TidalTrust metric is used as a constraint within a policy:

```
tpl:constraint "METRIC(tpl:TidalTrustMetric, ?USER, ?author, 0.5)".
```

Metrics are implemented as Java plug-ins into the TriQL.P query engine. There are currently four metric plug-ins available: eBay, TidalTrust [11], Appleseed [15] and PageRank [13]. The first three metrics require explicit ratings. The PageRank metric avoids the nessesarity of explicit ratings but allows common RDF predicates like foaf:knows or rdf:seeAlso to be used as links for ranking.

5 Explaining Filtering Decisions

Each 'Oh, yeah?' button in the browser's user interface corresponds to one RDF triple of the virtual trusted graph. The following steps are executed to generate an explanation why a triple fulfils a selected policy:

1. The engine retrieves the previously cached set of variable bindings that was used to produce the triple.
2. The explanation templates associated with the trust policy are instantiated using the variable bindings.
3. The resulting text snippets are grouped into a tree which is rendered into HTML.

In addition to this basic explanation mechanism, `METRIC()` plug- ins generate their own explanations about their calculation process and information used within the process.

6 Conclusions

We have argued that trust frameworks for the Semantic Web should not rely solely on explicit ratings but also facilitate information context and information content for trust assessments. We have shown a flexible way to express trust policies and to explain filtering decisions based on these policies, and we have described how to integrate our policy framework into a general-purpose Semantic Web browser.

Our approach of expressing policies as query templates instead of expressing them as rules [8] might suit users familiar with query languages like SPARQL. The information provenance explanations generated by the browser are similar to the provenance traces used within TRELLIS [9]. Compared with TRELLIS, our explanations are more flexible as they don't assume a single provenance ontology. The explanations generated by Inference Web [12] are complementary to our work. Inference Web focuses on explaining distributed reasoning paths, while we are focusing on explaining information provenance, background knowledge used in the assessments and metric calculations.

We hope that our prototype facilitates further thinking about pragmatic ways to incorporate trust policy frameworks into Semantic Web applications, as trust is an essential topic for the Semantic Web but is often ignored by current applications.

The TriQL.P browser is available under BSD license. More information about the browser, example RDF datasets and example policy suites are found at: `http://www.wiwiss.fu-berlin.de/suhl/bizer/TriQLP/browser/`

References

1. R. Agrawal, P. Domingos, and M. Richardson. Trust Management for the Semantic Web. In *2nd International Semantic Web Conference*, 2003.
2. T. Berners-Lee. Cleaning up the user interface, section - the "oh, yeah?"-button, 1997. http://www.w3.org/DesignIssues/UI.html.
3. C. Bizer. The trig syntax. http://www.wiwiss.fu-berlin.de/suhl/bizer/TriG/.
4. C. Bizer and R. Oldakowski. Using context- and content-based trust policies on the semantic web. In *13th World Wide Web Conference (Poster)*, 2004.
5. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs website. http://www.w3.org/2004/03/trix/.
6. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *14th International World Wide Web Conference*, 2005.
7. J. Carroll and P. Stickler. TriX: RDF Triples in XML. In *Proceedings of Extreme Markup Languages*, 2004.
8. T. Gianluca. Semantic web languages for policy representation and reasoning: A comparison of kaos, rei, and ponder. In *2nd International Semantic Web Conference*, 2003.
9. Y. Gil and V. Ratnakar. Trusting information sources one citizen at a time. In *1st International Semantic Web Concerence*, 2002.
10. D. Huynh, S. Mazzocchi, and D. Karger. Piggy bank: Experience the semantic web inside your web browser. Submitted to the International Semantic Web Conference 2005.
11. Jennifer Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, 2005. http://trust.mindswap.org/papers/GolbeckDissertation.pdf.
12. D. L. McGuinness and P. P. da Silva. Infrastructure for web explanations. In *2nd International Semantic Web Conference*, 2003.
13. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.
14. E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050217/, 2005.
15. C.-N. Ziegler and G. Lausen. Spreading activation models for trust propagation. In *IEEE International Conference on e-Technology, e-Commerce, and e-Service*, 2004.