Inf Syst Front (2007) 9:505–514
DOI 10.1007/s10796-007-9050-7

Virtual organization security policies: An ontology-based integration approach

Saravanan Muthaiyah · Larry Kerschberg

Published online: 19 October 2007

Abstract This paper addresses the specification of a security policy ontology framework to mediate security policies between virtual organizations (VO) and real organizations (RO). The goal is to develop a common domain model for security policy via semantic mapping. This mitigates interoperability problems that exist due to heterogeneity in security policy data among various (VO) and (RO) in the semantic web. We propose to carry out integration or mapping for only one aspect of security policy, which is authorization policy. Other aspects such as integrity, repudiation and confidentiality will be addressed in future work. We employ various tools such as Protégé, RacerPro and PROMPT to show proof of concept.

Keywords Ontology mapping · Semantic mapping · Security policy ontology · Security policy domain model

1 Introduction

One of the major semantic web research requirements is the need for semantic interoperability. Previous research into mapping and integration has focused on a closed and controlled environment within organizational boundaries (i.e., via EDI[1] and XML[2] technology). However those efforts did not solve semantic heterogeneity problems and only managed to mediate structural data heterogeneity requirements. With recent increased interest in ontologies and taxonomies we can see how much importance the research community has given to this area. Semantic interoperability allows systems to exchange information and services seamlessly with one another in more meaningful ways. Our research is focused on ontology mapping, which is a fairly new discipline. We believe that semantic interoperability can be achieved via ontology mapping and our work in the past has provided evidence that this is possible (Muthaiyah and Kerschberg 2006).

Understanding the causes of heterogeneity is crucial for developing accurate mapping techniques. The main reasons for heterogeneity are structural heterogeneity (difference in structures), semantic heterogeneity (interpretation of different structures) and subjective mappings. Figure 1 illustrates how the structural heterogeneity problem occurs for security policies. Assume that a VO has an authentication policy and RO has its own authentication policy, which it decided to call authorization policy.

Although the policies appear to be similar, structurally they are different. In the authentication policy structure, the VO has "UserPassword" and the RO has "Token". However the other classifications in the tree remain the same. The next example (Fig. 2) shows semantic heterogeneity that exists for the same entities, i.e., VO and RO. From the above example one would assume that the PKI

S. Muthaiyah (✉)
3500 Country Hill Drive,
Fairfax, VA 22030, USA
e-mail: smuthaiy@gmu.edu
URL: http://eceb.gmu.edu

L. Kerschberg
Department Information and Software Engineering,
George Mason University,
4400 University Drive, MS 4A4,
Fairfax, VA 22030, USA
e-mail: kersch@gmu.edu
URL: http://eceb.gmu.edu

S. Muthaiyah · L. Kerschberg
E-Center for E-Business, George Mason University,
Fairfax, USA

[1] Electronic Data Interchange
[2] Extensible Markup Language

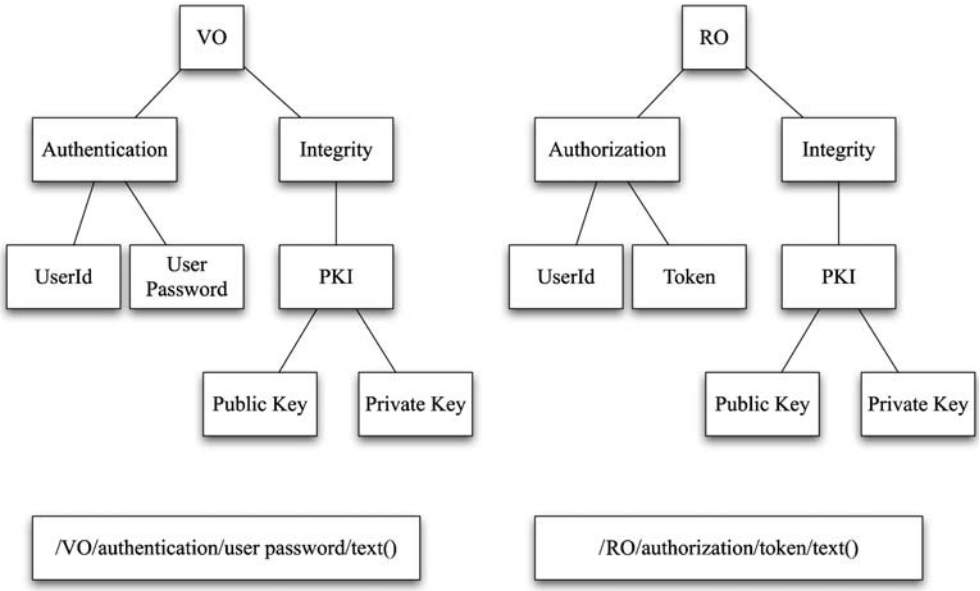


Fig. 1 Structural heterogeneity problems

data definitions of both entities are the same. However this is not always true and it is clear from the example below that the PKI data definitions are different in the final policy.

The X.509 certificate specifies the association between a public key and a set of attributes such as subject name, version, issuer name, serial number and validity interval. The OASIS[3] specification describes the use of the X.509 authentication framework in greater detail and SOAP[4] Message Security specification (WS-Security) describes procedures for message exchange. In order to achieve interoperability and to resolve the heterogeneity issues of security policies above, semantic integration is needed.

Our goal is to specify a Security Policy Domain Model (SPDM), via ontology mapping. SPDM acts as a common security policy for both entities by inheriting the existing attributes and sometimes adding or modifying attributes between them so that the policies become commonly applicable to both entities without compromising existing security standards. The contribution of our paper is twofold, one is the methodology for building the SPDM for authorization policy, and the other is the taxonomy for authentication, which we developed from scratch to test our concept.

Explicit differences in security policies can exist among organizations (Mehta et al. 2004) that transact over the Semantic Web (Lee et al. 2005). RO and VO have differences in their security policy design and we have provided examples. This is also true for any type of heterogeneous databases where having the same identical structures would be nearly impossible. SPDM fully integrates and eliminates semantic differences that exist at the object and attribute levels of the security policy data structures. In the next section we show the methodology for developing SPDM.

2 Literature review

Information systems today are evolving from static to dynamic environments. Integration of these systems using Grid technology was the very first effort to resolve data type differences. The Grid defines various abstraction layers (Foster et al. 2001), which enabled much of this work. Widely dispersed information spaces were fully interconnected and they were referred to as virtual organizations (VO) (Foster et al. 2001). The Grid infrastructure was a good start for collaborations to take place among VO participants (Foster et al. 2001). Coordination of seamless interactivity and resource sharing among VO was the main focus of that effort. However some of the heterogeneity problems still remained unresolved.

Another approach for solving heterogeneity problems has been to carry out matching. Much of this work had been done using XML technology. The match operator takes as input two graph-like structures (e.g., database schemas or ontologies) to produce mappings between its elements. If the graphs correspond semantically to each other, matching is successful. The InfoSleuth project (Fowler et al. 1999), which uses agent architecture, is a good example of this approach. Ontologies are believed to be the next trend for solving heterogeneity problems.

Ontology allows the explicit specification of a domain of discourse. By incorporating semantics, ontologies produce

[3] Organization for the Advancement of Structured Information Standards http://www.oasis-open.org/who/).

[4] Simple Object Access Protocol

Fig. 2 Semantic data heterogeneity problems

VO: Security Authentication Final Policy

PKI – X.509 Certificate

Name	Identifier	Category	Version
Final Policy	UserId	Certificate	3

Note: Object representation conflicts

RO: Security Authentication Final Policy

PKI – X.509 Certificate

Name	Identifier	Category	Version
Final Policy	UserId	Token	3

Note: Object representation conflicts

specific knowledge domains. However, ontologies per se cannot resolve any interoperability problems because it is not possible to have a single ontology that represents all kinds of domains and applications. The same analogy is also applicable to ontologies for security policies. To provide seamless and "on the fly" capabilities for semantic web services, we must implement a complete SPDM.

The semantic mapping between ontologies has not been automated and is still performed manually. A dynamic integration solution is required. Interoperability at the semantic-level is more desirable today; protocols like SOAP, UDDI and WSDL alone will not solve this problem. A shift towards semantic level integration via semantic interoperability using Reference Ontology and a Semantic Annotation Language has been proposed to resolve this problem (Missikoff et al. 2003). The Semantic Web can also be viewed in the same manner as the Grid (Howard and Kerschberg 2004) where various layers exist and heterogeneous security policies exist among them. Security policies can have a totally different meaning for different VO (Wang et al. 2004). To reconcile the differences, gap analysis is required, followed by semantic reasoning and lastly mapping of all objects and attributes. Some instances may also be merged if they have a lot in common. We can thus produce a common reference ontology, which will represent concepts, relationships, integrity constraints and business rules that both VO and RO would collaborate on. A formal definition of the ontology is necessary to do this (Höne and Eloff 2002).

3 Methodology

We first assume that only two entities exist, i.e., RO and VO, and are in our earlier examples, we specify authorization security policy ontologies, one for RO and another one for VO. In order to do this one first constructs an authorization taxonomy. To show that VO and RO have different data structures we name VO's ontology as authentication and RO's as authorization. Having identified the structural and semantic differences we proceed to carry out reasoning regarding those policies. We conducted reasoning to check for consistency in both of the ontologies that we had specified.

Inconsistencies between them had to be resolved to produce a consistent SPDM. We used a tool to check for consistency, i.e., RacerPro.(OWL reasoner by Racer Systems, http://www.sts.tu-harburg.de/~r.f.moeller/racer/) It is a very useful tool for checking the consistency between two or more ontologies. Mapping would be useless if the ontologies were inconsistent. The tool also understands the ontology to be mapped and sends a message to the reasoning server via http GET and POST messaging. This is a client-server Java-enabled engine that produces a summary report after the reasoning process is complete. This process is very time consuming and can be very cumbersome.

Integration and mapping are technically difficult; thus much of the existing work often assumes that mappings are already known. The identification of semantic relationships between different information sources is a difficult problem as well. Moreover to produce mappings and reason about them in a dynamic environment is a non-trivial task. How much processing time would be needed is still unknown. After making sure all inconsistencies have been resolved one performs gap analysis on the ontologies that were reasoned about.

The analysis provides us an overview of the similarities and differences in the structure and semantics of those security policies. Next, one maps and consolidates the policies where they were required. Ontology mapping is a fairly new area and much of it is done manually. Also, it is done in a static fashion as described in this paper. When the Semantic Web grows and becomes popular we should have automated systems to do all the mappings seamlessly in a dynamic way and on-the-fly. A method for finding semantic relations effectively is also required to make all this possible. Such systems have yet to be built and this research is a contribution in that direction.

4 Understanding security policy

The term "security policy" has many different meanings and can be interpreted in many different ways. A security policy is a statement of what is allowed, and what is not allowed (Bishop 2002). The existence of various interpretations is rooted in two observations. Firstly, security policy

is a context-dependent notion (e.g., computer security policy, information security policy, etc.) and secondly, even in the same context, specific kinds of security policies have been developed to meet specific needs (e.g., confidentiality security policies in military environments, etc.).

To effectively manage security policies we must be able to produce compatible policy representations. The existence of a large number of representation methods leads to the conclusion that security policies, even when semantically compliant, can be represented in ways that differ substantially in terms of formalism, structure, and hierarchy, thus raising obstacles to their reconciliation. Therefore, in order to effectively manage security policies one has to be able to produce compatible policy representations (i.e., SPDM).

Multiple interacting security policies require semantics to be managed and manipulated. The security policy semantics ontology is an efficient means for achieving this. Ontology is "an explicit specification of a conceptualization" (Gruber 1993). Domain-specific ontologies are used to define the terminology for a group of people that share a common view on a specific domain, effectively supporting knowledge sharing and reuse. Thus, security policies can be represented by the means of a SPDM, which elaborates on the domain of security knowledge.

SPDM can be used to describe structurally heterogeneous security policies of different levels of abstraction. Thus, by defining shared and common domain theories and vocabularies, SPDM help both people and machines to communicate in a concise manner, a manner which is based not only on the syntax of security policy statements, but on their semantics, as well.

5 SPDM lifecycle and tools

Figure 3 shows the six steps of the SPDM development lifecycle to produce the SPDM. The security policy ontology is created for RO and VO first (step 1) and then exported to OWL. We use Protégé (Open source ontology editor downloadable at: http://protege.stanford.edu/download/prerelease/full/) to build and export the OWL and perform reasoning with RacerPro (Appendix A.1 and A.2). The reasoning process in RacerPro is made up of three subprocesses, i.e., checking for consistency in the ontologies to be mapped, classifying their taxonomy and computing their inferences. Figure 3 summarizes those processes in (steps 2–3). The reasoner will eliminate inconsistency and align any new additions that are inserted into the ontology.

Upon successful elimination of inconsistency (steps 2–3) we begin mapping (Appendix A.3) using the PROMPT (Protégé plug-in that allows comparing and merging ontologies.) plug-in (step 4). Merge functions (step 5) are carried out if ontologies can be combined (Appendix A.3) and lastly a mapping log is generated (step 6). In the dynamic semantic web environment the policy mapping would be done on-the-fly seamlessly. When the VO and RO agree on a common security policy for processing a

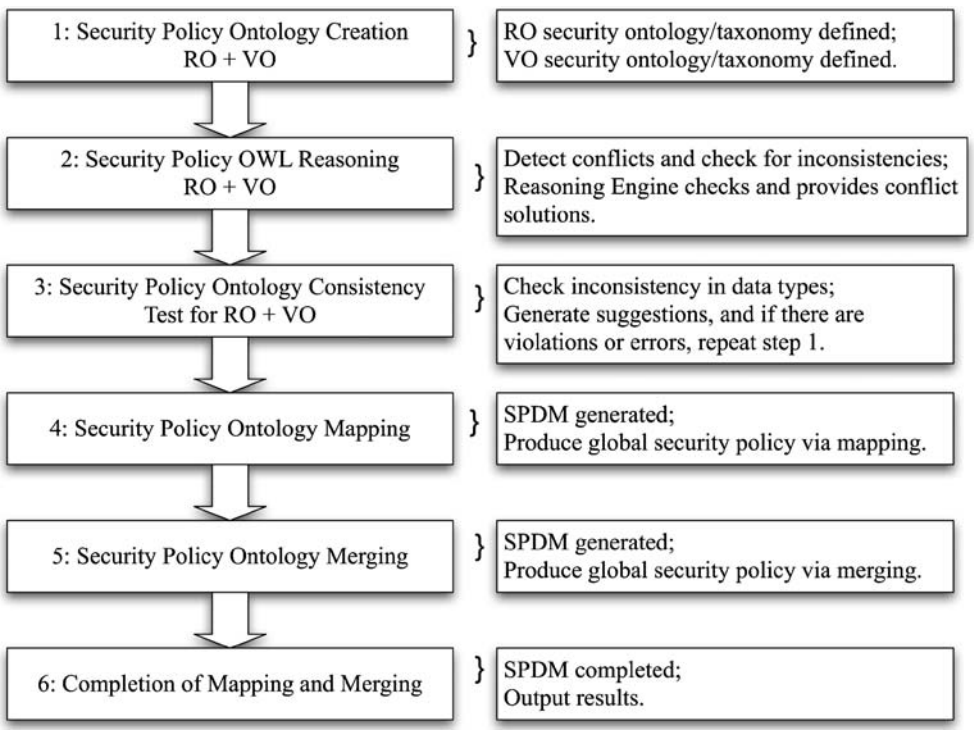


Fig. 3 SPDM development lifecycle

transaction a new contact agreement is formed. Since the Semantic Web is a dynamic environment with multiple new entrants coming into the virtual supply chain, there is a need to dynamically map local and global policies. The wrapper, as explained earlier solves this problem. Mapped policies would also represent virtual temporary contracts (VTC), which can be stored in a knowledge base for future use.

With technology similar to topic maps we can track how security polices evolve with dynamic changes in business requirements. Another benefit would be to carry out audit functions on previously contracted transactions for resolving repudiation problems and thus enhancing controls.

The methodology for semantic mapping in the semantic web would be to have a WS-Security ontology mapping system like JENA[5], which has its own rule ontology and rule parser. However, aligning and mapping the security policies of the constant influx of new entrants is not a trivial task. We demonstrate how this could be done with different plug-ins incorporated into Protégé such as Algernon[6] and PAL[7]. Also a merge[8] function was done using another plug-in called PROMPT. PROMPT has four stages (i.e., merge, extract, move frame and compare). RacerPro was used to test the overall consistency of SPDM. Table 1 illustrates the tools we have used for proof-of-concept and the steps 1 through 6 relate to the six steps in the SPDM lifecycle.

6 Mapping in Protégé and reasoning in RacerPro

There are many aspects of security policy that need to be reconciled between the global and local environments. A complete taxonomy is needed so that the reconciliation can be performed quickly. However our literature survey indicates that a comprehensive security ontology does not exist. Therefore, we have created our own ontology, one for the VO and another for the RO. As we have mentioned earlier, this is our major contribution of this paper. Security can be divided into authentication, authorization, integrity, access control, non-repudiation and confidentiality. This paper focuses on authorization security policy for SPDM.

Table 2 shows two scenarios A and B. The "final policy" data labels for authentication in scenario A are "UserId" and "UserPassword" and authorization data labels are "UserId" and "Token". SPDM shows the mapped data labels which are "UserId" and "UserPassword". The identifier remains as "UserId" but Password and TokenId becomes "Password Ticket". In scenario B, the "final policy" data labels for authentication are "UserId" and "X.509" (Certificate) before mapping. After mapping SPDM shows the new data labels as "UserId" and "X.509" (Certificate Token). Merge is complete without any discrepancies and SPDM is the common global policy for authorization, which is agreed by VO and RO. Heterogeneity in the data labels above, were reconciled to arrive at the common policy (SPDM). The scenario above is mapped into XML and in order to maintain consistency we use RacerPro to do reasoning. Typically RacerPro would produce statistics for tests carried out (step 3, Fig. 3) and produces a log report (i.e., Mlog) which will address the following:

- Total number of generated suggestions
- Number of generated suggestions that were followed by the user
- Total number of conflicts detected
- Number of conflict solutions used
- Total number of KB operations

Table 1 Tools used in the SPDM development lifecycle

Step/process	Tools used to carry out the process
1	Protégé 3.1—Ontology editor
2	RacerPro—Ontology/OWL Reasoner
3	RacerPro—Reasoner with Java API
4	Prompt—Plug-in for Protégé 3.1 ontology editor
5	Prompt—Merge function in Prompt plug-in available in Protégé 3.1 ontology editor
6	Successful completion viewable in Mlog

First separate ontologies for Scenarios A and B have to be created in Protégé. Each ontology must then be exported to OWL format prior to reasoning. After that the reasoning is done separately for A and for B. Notice that the reasoning results are error free which means mapping can be carried out from this point. If there were errors (usually in red) those would have to be corrected before mapping. We do this to ensure that the data is consistent before mapping. Errors are usually related to inconsistent classes, concepts, slots and attributes. For instance if the authorization policy for scenario A had inconsistent attributes, the output would appear in "red" and the user would be prompted to fix those errors. The errors must be amended before proceeding to the next step, which is mapping.

Figure 4 shows the successful reasoning (i.e., number of conflicts detected=0) performed for Scenario A (RO and VO). Reasoning is also done for Scenario B (RO and VO) and after successful reasoning we perform mapping. Mlog (mapping log) outputs in (Appendix A.1 and A.2) also show the number of conflicts detected which in this case is zero. SPDM (Fig. 3, step 6) would be the end result. Successful mapping with PROMPT and merge for scenario A is shown in (Appendix A.3).

[5] JENA API is a Java framework for building Semantic Web applications, http://jena.sourceforge.net/.

[6] Protégé plug-in is used for forward and backward chaining rules.

[7] Protégé plug-in is used for expressing constraints.

[8] Merging function allows projects to be merged after resolving the common concepts between them.

Table 2 Merging two SPRO policy data elements

	Security Policy – VO (Authentication) *Before Mapping*	**Security Policy – RO (Authorization)** *Before Mapping*	**Compatible Security Policy – SPDM = VO+RO (Authorization)** *After Mapping*
SCENARIO A	**A=Identifier +Password** <? xml version="1.0" ?> <Final_Policy > <Entity> <Type/> <Identifier>UserId</Identifier> <Password> UserPassword </Password>	**A=Identification with token ID** <?xml version="1.0" ?> <Final_Policy > <Entity> <Type/> <Identifier>UserId</Identifier> <TokenId>Token</TokenId>	**A=Password ticket** <? xml version="1.0" ?> <Final_Policy > <Entity> <Type/> <Identifier>UserId</Identifier> <Password Ticket> UserPassword </Password Ticket>
SCENARIO B	**B=X.509 Certificate** <?xml version="1.0" ?> <Final_Policy > <Entity> <Type/> <Identifier>UserId</Identifier> <Certificate>X.509</Certificate>	**B=X.509 token** <?xml version="1.0" ?> <Final_Policy > <Entity> <Type/> <Identifier>UserId</Identifier> <Token>X.509</Token>	**B=X.509 Certificate & token** <?xml version="1.0" ?> <Final_Policy > <Entity> <Type/> <Identifier>UserId</Identifier> <CertificateToken>X.509</CertificateToken>

7 Conclusion

Ontologies raise the level of specification of knowledge by incorporating semantics into data and promoting its exchange in an explicitly understandable form. Nevertheless one question remains unanswered that is, "are standard schemas or ontologies going to solve the semantic integration problems?" Stand-alone ontologies provide zero

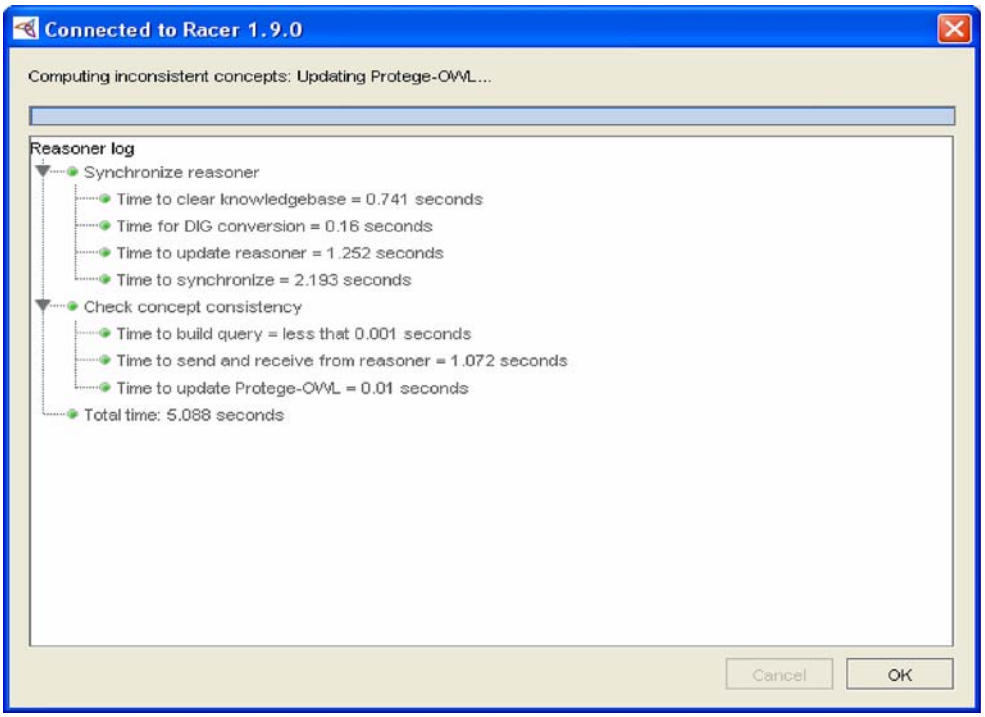


Fig. 4 Reasoning results of scenario A and B before mapping

interoperability therefore we need accurate mappings from actual systems and exploitation by real development tools like the ones mentioned above. Ontology mapping aims to overcome semantic integration problem between ontology-based systems. Ontology mapping is an important area of research due to the need for better data integration and to achieve semantic interoperability. Semantic data integration carried out in this paper gives an insight into the benefits of ontology mapping in the area of security. This is significant for building a common information model, trusted domain and federated services. More work is needed in this area, especially in conflict analysis and resolution. VO represents RO for carrying out transactions to fulfill demands from user. Policies of RO and VO of agent ontologies may differ greatly and so we need to look into implementing security taxonomy, which can be common to all. This would also make mapping and reasoning a lot faster.

Appendix A.1

OWL for RO (after RacerPro reasoning)

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Authorization"/>
  <owl:DatatypeProperty rdf:ID="password">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Authorization"/>
  </owl:DatatypeProperty>
  <owl:FunctionalProperty rdf:ID="user_id">
    <rdfs:domain rdf:resource="#Authorization"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="token_id">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Authorization"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="X.509_token">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Authorization"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >X.509 token</rdfs:label>
  </owl:FunctionalProperty>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 2.2, Build 307)  http://protege.stanford.edu -->

Mlog Output for RO
----------START----------
Total number of generated suggestions: 0
Number of generated suggestions that were followed by the user: 0
Total number of conflicts detected: 0
Number of conflict solutions used: 0
Total number of KB operations: -3
```

Note:

Appendix A.1 shows the OWL results after RacerPro had reasoned it. All the attributes and classes for RO were checked using description logic. The log shows that there were no problems with RO's ontology as such no were conflicts detected.

Appendix A.2

OWL for VO (after RacerPro reasoning)

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Authentication"/>
  <owl:DatatypeProperty rdf:ID="X.509_certificate">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >X.509 certificate</rdfs:label>
    <rdfs:domain rdf:resource="#Authentication"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="password">
    <rdfs:domain rdf:resource="#Authentication"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  </owl:DatatypeProperty>
  <owl:FunctionalProperty rdf:ID="identifier">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Authentication"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 2.2, Build 307)  http://protege.stanford.edu -->
```

Mlog Output for VO

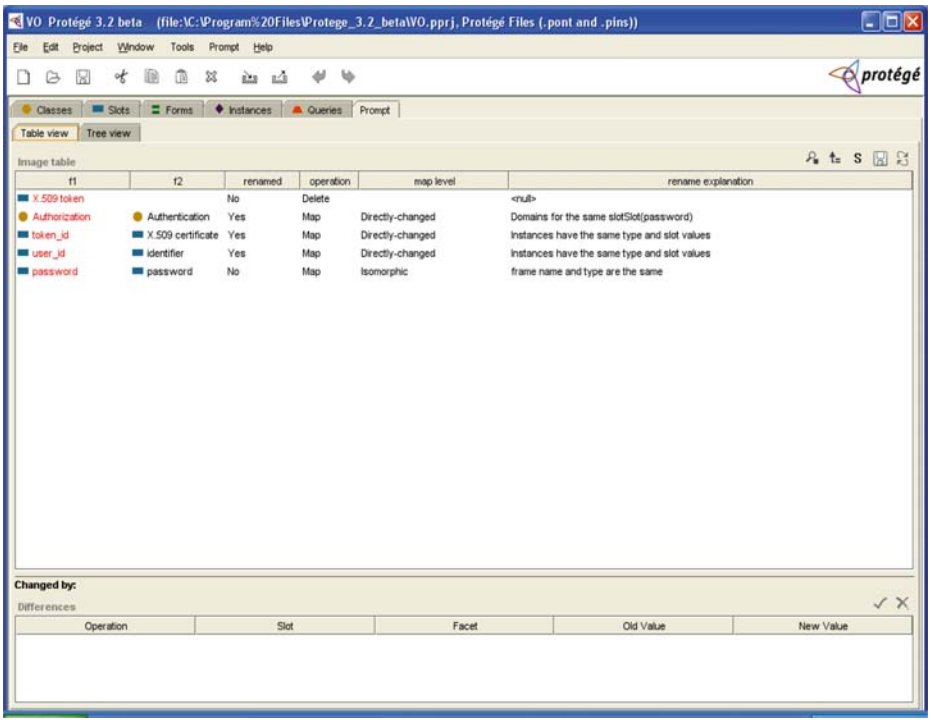
```
-------------------------START-------------------------
Total number of generated suggestions: 0
Number of generated suggestions that were followed by the user: 0
Total number of conflicts detected: 0
Number of conflict solutions used: 0
Total number of KB operations: -3
Total number of generated suggestions: 0
Number of generated suggestions that were followed by the user: 0
Total number of conflicts detected: 0
Number of conflict solutions used: 0
Total number of KB operations: -3
```

Note:

Appendix A.2 shows the OWL results after RacerPro had reasoned it. All the attributes and classes for VO were checked using description logic. The log shows that there were no problems with RO's ontology as such no were conflicts detected.

Appendix A.3

Mapping is complete for scenario A



Note:

Appendix A.3 shows the mapping results for Scenario A. We did this after the reasoning process was complete. Renamed attributes and operations are mentioned in column 3 and 4. The delete operation shows that a slot is being eliminated in the process. Map-level shows mapped operation and password in this case was not renamed as it was isomorphic.

References

Bishop, M. (2002). *Computer security: Art and science.* New York: Addison-Wesley.

Fowler, J., Brad, P., Marian, N., & Bruce, B. (1999). Agent-based semantic interoperability in InfoSleuth. *SIGMOD, 28*(1), 60–67.

Höne, K., & Eloff, J. H. P. (2002). Information security policy: What do international information security standards say? *Computers and Security, 21*(5), 402–409.

Howard, R., & Kerschberg, L. (2004). *Using facets of security within a knowledge-based framework to broker and manage semantic web services*. Paper presented at the Workshop on Secure Knowledge Management, Amberst, New York.

Foster, I., Kesselman, C., & Tuecke, S. (2001). The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications, 15*(3), 200–222.

Gruber, T. R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition, 5*(2), 199–220.

Lee, K. J., Upadhyaya, S. J., Rao, H. R., & Sharman, R. (2005). Secure knowledge management and the semantic web. *Communications of the ACM, 48*(12), 48–54.

Mehta, B., Niederée, C., Stewart, A., Muscogiuri, C., & Neuhold, E. J. (2004, June). *An architecture for recommendation based service mediation.* Paper presented at the Proceedings of International Conference on Semantics of a Networked World (ICSNW), Paris, France.

Missikoff, M., Schiappelli, F., & Taglino, F. (2003). *A controlled language for semantic annotation and interoperability in e-business applications*. Paper presented at the Proceedings of the Second International Semantic Web Conference (ISWC-03), Sanibel Island, Florida.

Muthaiyah, S., & Kerschberg, L. (2006). *Dynamic integration and semantic security policy ontology mapping for semantic web*

services (SWS). IEEE Engineering Management Society, ISSN 1-4244-0682-X, pp. 116–120.

Wang, H, Jah, S., Livny, M., & McDaniel, P. D. (2004). *Security policy reconciliation in distributed computing environments*. Paper presented at the Proceedings of the 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'04), New York.

Saravanan Muthaiyah is a senior lecturer at Multimedia University, Cyberjaya, Malaysia and currently a doctoral degree candidate in Information Technology at George Mason University, Fairfax, VA. He holds a Masters degree in Information Technology and other degrees in the area of accounting and finance for his bachelors. He is also a Fulbright scholar under the auspicious graduate research exchange program sponsored by the US Department of State. His research interests include semantic web, ontology mapping, systems integration, systems engineering, topic maps, knowledge management and enterprise architectures. His recent papers have focused on ontology mapping and mainly solving heterogeneity issues for Semantic Web.

Larry Kerschberg is Professor of Information and Software Engineering, at George Mason University, Fairfax, VA 22030 USA. He is director of E-Center for E-Business and directs the MS in E-Commerce Program. He is past Chairman of the Information and Software Engineering Department at Mason. During 1998 he was a Fellow of the Japan Society for the Advancement of Science at Kyoto University. He holds a B.S. in Engineering from Case Institute of Technology, and MS in Electrical Engineering from the University of Wisconsin-Madison, and a Ph.D. in Engineering from Case Western Reserve University. He is Editor-in-Chief of the *Journal of Intelligent Information Systems*, published by Springer. He recently served as an editor and contributor of the book "The Functional Approach to Data Management: Modeling, Analyzing and Integrating Heterogeneous Data," published at Heidelberg, Germany, Springer, 2004. His areas of expertise include expert database systems, intelligent integration of information, knowledge management, and agent-based semantic search. His recent papers have focused on ontology-driven semantic search in Knowledge Sifter, knowledge representation using Topic Maps, and methodologies for the creation and management of Semantic Web Services. These papers can be found at http://eceb.gmu.edu/publications.html.