A Demonstration of Formal Policy Reasoning Using an Extended Version of BaseVISor

Christopher J. Matheus
VIStology, Inc.
cmatheus@vistology.com

Mieczyslaw M. Kokar
Northeastern University
mkokar@ece.neu.edu

Robert Dionne
VIStology, Inc.
rdionne@vistology.com

1. Introduction

This demonstration will show the practical application of a formal reasoning engine for both policy invocation and policy reconciliation. The primary features of this demonstration include the use of formal ontologies, context based policy reasoning, and support for both policy invocation and reconciliation. BaseVISor [1], a forward chaining inference engine optimized for reasoning about RDF triples, is the formal reasoning engine that we have extended to support automated reasoning about policies through the addition of context-based semantics introduced by cwm/REIN [2] and a formal ontology that we have extended/developed based on concepts from REIN and Ismeme [3]. We will demonstrate the system's application to some existing scenarios taken from the body of work described in [2] and [3].

2. Policy Reasoning

In this work we are concerned with the ability to use policies to ensure compliance during runtime as well as with the ability to do policy reconciliation. Policy compliance involves the run-time process of ensuring that all of the conditions defined by a policy hold true; a common example is the checking of credentials required before granting access to a document. In policy reconciliation, the goal is to take multiple polices and generate a policy instance that simultaneously satisfies all of them; a typical example here is determining specific conditions under which a communication session is to be established between nodes in a VPN where the ends of the connection are governed by different policies.

A primary objective in our work is to develop the means by which these operations governing policies can be handled automatically by computer. For this reason we believe it is important to be able to describe policies in a formal, declarative way that will permit them to be automatically processed by formal reasoning engines.

3. Formal Reasoning

A formal reasoner or inference engine is a system capable of applying the formal axioms of a language to a body of data/facts/knowledge resulting in the derivation of additional inferable facts. A rule-based system, for example, may be used as a formal reasoner if it is provided with a set of axioms for the language in which the data/knowledge is represented. Such axiom sets are available for a number of ontology languages as discussed in the next section.

An important principle employed by many systems including policy based reasoners is the use of the *closed world assumption* which permits systems to assume that everything that is known to be true of the "world" is available in the facts that have been provided about it; if a fact is not explicitly stated it is assumed to be false. The closed world defined by a set of facts can be thought of as a "context" in which reasoning is to occur. The ability to define multiple contexts and to reason across them becomes important in policy reconciliation and is a capability provided by the *log:semantics* construct described in Section 5.

4. Ontologies

By ontology we mean a formal representation of the classes and properties relevant to a particular domain of interest. The Web Ontology Language OWL [4] and the Resource Description Framework [5,6] are formal ontology languages commonly used within the Semantic Web community and elsewhere. Formal semantics exist for these languages making them suitable for automated processing by inference engines. Both of these languages provide a restricted subset of first order logic and as such cannot represent a range of concepts that are important to policy reasoning; in particular they are not able to capture composite properties (e.g., "uncle" as the "brother of parent") or express joins on property values across instances. To overcome this limitation many reasoning systems that employ OWL/RDF also provide a means for representing knowledge in the form of rules; see for example cwm [7] and BaseVISor [1].

For representing policies we have adopted the policy ontology proposed by REIN and depicted in Figure 1. To this we added concepts and properties necessary to support policy reconciliation for the scenarios used in the testing of Ismeme. In this way our approach extends that of Ismeme by affording it the advantages of formal ontological reasoning (e.g., concise representation, interoperability, class subsumption, instance classification, etc.)

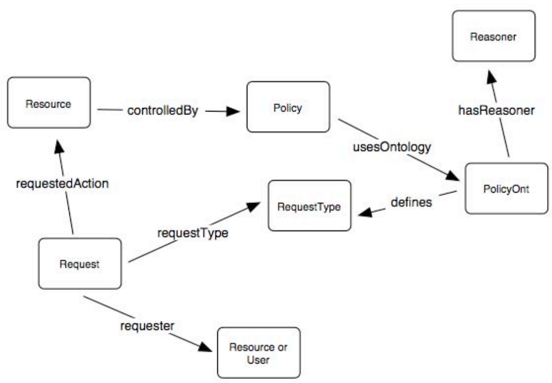

Figure 1. REIN Policy Ontology (from [2])

5. BaseVISor

BaseVISor is a forward-chaining, rule-based inference engine based on an implementation of the Rete algorithm [8] optimized for the processing of RDF triples. BaseVISor is written as an embeddable Java component and is extensible through the definition of user defined procedural attachments. It also comes with the set of axioms required to support R-Entailment [9], a language consisting of RDF, RDFS and a subset of OWL that has attractive computational characteristics.

To use BaseVISor for policy reasoning we extended it with three procedural attachments that implement the *log:semantics*, *log:includes* and *log:notIncludes* elements of N3 rules [10]. *log:semantics* takes the URI for a document and loads it into a separate instance of a Rete network (preloaded with the axioms for R-Entailment) which is then run to derive all triples (facts) entailed by the document. The derived fact base (an RDF graph) is referenced by a variable which can then be used as arguments to *log:includes* and *log:notIncludes*. *log:includes* takes two RDF graphs and returns true if the first graph is contained within the second graph; *log:notIncludes* works the same but returns true if the first RDF graph is not contained in the second. Through these constructs BaseVISor is able to reason over multiple policies, each represented within its own context (i.e., a fact base) that can be queried against.

6. Demonstration

In the demonstration we will show how the extended version of BaseVISor can be used to perform the same kind of simple policy invocation for controlling document access that has been demonstrated in REIN. For this purpose the REIN policies [11] have been translated from N3 rules into BaseVISor rules via a Python script. These are then run within the system to process the policies and determine whether access should be granted.

We will also demonstrate how BaseVISor can be used to do basic policy reconciliation using the scenarios demonstrated with Ismeme. The manner in which the policies are processed by the system for reconciling a VPN session between two parties is depicted in Figure 2. The Ismeme policies employ a rule-like structure but their semantics differ from the Horn clause nature of BaseVISor rules and thus their translation into BaseVISor rules was done manually. In our implementation of these scenarios we leveraged the power afforded by the use of formal ontologies, an ability Ismeme does not currently support.

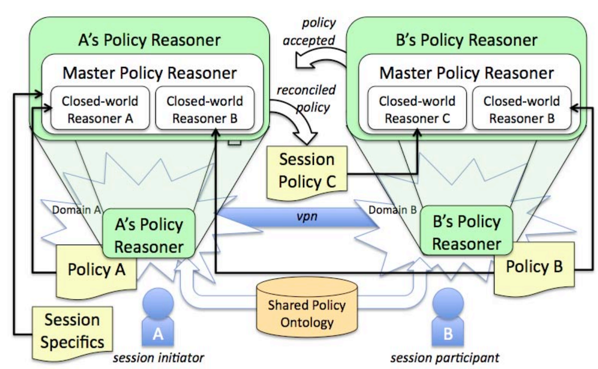

Figure 2. Policy Reconciliation Scenario

7. Acknowlegments

This work was partially funded by the United States Missile Defense Agency under contract number W9113M-08-C-0053.

8. References

[1] C. Matheus, K. Baclawski and M. Kokar. BaseVISor: A Triples-Based Inference Engine Outfitted to Process RuleML and R-Entailment Rules. In Proceedings of the 2nd International Conference on Rules and Rule Languages for the Semantic Web, Athens, GA, Nov. 2006.

[2] L. Kagal and T. Berners-Lee. Rein : Where policies meet rules in the semantic
web. Technical report, MIT, 2005.

[3] P. McDaniel and A. Prakash. Methods and Limitations of Security Policy Reconciliation. ACM Transactions on Information and System Security (TISSEC), Association for Computing Machinery, 9(3):259-291, August 2006.

[4] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L. A. Stein. OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004. http://www.w3.org/TR/owl-ref/ .

[5] W3C: RDF/XML Syntax Specification (Revised). http://www.w3.org/TR/rdf-syntax-grammar/ (2004).

[6] W3C: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, http://www.w3.org/TR/rdf-schema/ (2002).

[7] CWM: Closed World Machine. http://infomesh.net/2001/cwm/ 2001.

[8] C. L. Forgy, "Rete: a fast algorithm for the many pattern/many object pattern match problem," Artificial Intelligence, 1982, pp.17-37.

[9] H. ter Horst. Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity. In Proc. of the Fourth Int'l Semantic Web Conference. Y. Gil et al. (Eds.): ISWC 2005, LNCS 3729, pp. 668–684, 2005.

[10] T. Berners-Lee, Notation 3. http://www.w3.org/DesignIssues/Notation3.html (1998).

[11] http://dig.csail.mit.edu/Rein