SEMANTIC ANNOTATIONS FOR SECURITY POLICY MATCHING IN WS-POLICY

Giuseppe Di Modica and Orazio Tomarchio
Dipartimento di Ingegneria Elettrica, Elettronica e Informatica, Università di Catania
Viale A. Doria 6, 95125 Catania, Italy

Keywords: WS-policy, Policy management frameworks, Semantic annotation.

Abstract: Service computing technology enables B2B scenarios where the provision of a service may require a collaboration among several service providers across multiple independent and heterogeneous administrative domains. In these environments, several new security and privacy challenges arise, mainly related to resource sharing and interoperability among different providers. Policy management frameworks are a powerful mechanism to deal with this heterogeneity, although many issues still have to be faced with. In particular, policy matching is today carried out following a syntactical approach, which may impair the selection of suitable services on the one hand, and the flexibility of the matching process on the other one. In this work we propose a semantic approach that, by allowing WS-Policy assertions to reference semantic concepts, provides for a better matching of security requirements and capabilities. The proposed approach has been validated through a case study that shows how a pure syntactic-based mechanism of WS-Policy would have failed in matching two actually compatible policies.

1 INTRODUCTION

Policy-based management has been widely employed in enterprise information systems (Phan et al., 2008) to automate network administration tasks. Multiple approaches for policy specification have been proposed, ranging from formal policy languages to rule-based policy notation. Within the last few years policy management has extended its original scope, going beyond traditional domains. Security policies are currently used in heterogeneous service oriented architectures (SOA) to address the problem of interoperability among the different security systems and mechanisms used by each service provider.

WS-Security (OASIS, 2006) defines an abstract model for security in Web Service environments. It addresses messages confidentiality and integrity, and provides models for exchanging various security token and defining access control mechanisms (Lakshminarayanan, 2010). But, if with respect to the message security there are several well-established techniques and mechanisms, the discovery and compatibility of security requirements among "interoperable services" still lacks of an established methodology. Ensuring security in the scenarios that may arise calls for a dynamic verification of compatibility between the requestor's security requirements and the service provider's security capabilities.

In this scenario, security policies allow to abstract from the low-level security mechanisms of the specific service provider. The employment of languages for the specification of security policies will also enable, on the one hand, the providers to expose the security capabilities implemented in their administrative domain and, on the other one, the users to express their security requirements. WS-Policy (W3C, 2007) is the specification language to express Web services' policies. It is used to define requirements, preferences and capabilities of services, allowing for both simple declarative assertions as well as more sophisticated conditional assertions. Yet the comparison of policies within the WS-Policy framework is done at a syntactic level, thus limiting the selection of suitable Web services.

In this work we propose to semantically characterize the policy assertions, by defining a common security ontology that enables mechanisms for a semantic matching of security requirement and capabilities. Our approach does not propose a new semantic policy language, like instead other authors do (Uszok et al., 2003; Kagal et al., 2003). We leverages on the WS-Policy specification, allowing for a lightweight approach where existing syntactic policies could be processed by our semantic-aware tools and, viceversa, semantic policies could coexist within standard WS-Policy frameworks (though not semantically processed).

The rest of the paper is structured as follows. Section 2 presents background and motivation of our work. In Section 3 we sketch the overall architecture of the framework, and provide details on the security ontology and the matching algorithm. In Section 4 an example is reported to validate our approach. Section 5 discussed some related work. Finally we conclude the work in Section 6.

2 BACKGROUND AND MOTIVATION

The adoption of a policy based-approach for managing a system requires an appropriate language for policy representation and modeling and the design and development of a policy management framework for policy matching and enforcement. Policies will be increasingly important to effectively realize the "Internet of Service" vision, where services are mainly built through composition of services running in different providers and/or administrative domains (Phan et al., 2008).

WS-Policy (W3C, 2007) is the specification used in service oriented architectures for expressing policies. Within WS-Policy, a policy is defined as a collection of *alternatives* and each alternative is a collection of *assertions*. Assertions are XML elements identified by qualified names (*qnames*). They specify characteristics that may be used for service selection such as requirements, capabilities or behaviors of a Web service. Policy assertions can represent both requirements on service requestors and the capabilities of the service itself. Requirements represent a demand on service requestors to follow a particular behavior; capabilities are the service providers promises of behavior. Among the others non functional properties of a service, WS-Policy may be used to express security requirements and security capabilities. For example, the use of a specific protocol to protect message integrity is a requirement that a service can impose on requestors. On the other hand, the adoption of a particular privacy policy when manipulating data from a requestor is a service capability.

Policy matching in WS-Policy works on a syntactic level: it offers a domain independent mechanism to find alternatives that are compatible to two policies. Two alternatives are compatible if, for each assertion in one alternative, there is a compatible assertion in the other one. Compatibility of assertions is defined exclusively according to the equality of their *qname*, without any further mechanism involving their structure and content. Our work, as it will be described in next sections, carefully extends WS-Policy by referencing concepts from a security ontology directly in the policy assertions, so maintaining backward compatibility with existing policy management tools.

Ontologies define a vocabulary referring to concepts of a domain, their attributes and relations amongst them. Specifying this knowledge in a machine-readable language with formally defined semantics, allows computers to interpret this knowledge and possibly infer further implicit knowledge. The Web Ontology Language (OWL) (W3C, 2009) is a language, defined by W3C, for specifying ontologies with formal semantics based on description logics. There exist several implementations of OWL inference engines, so-called OWL reasoners. OWL ontologies can describe individuals, relations between individuals, classes which refer to groups of individuals that have something in common, and relations between classes.

As an example, in OWL we can define that class C is a *subclass* of class D, denoted as $C \sqsubseteq D$, which means that every individual belonging to C also belongs to D. If two classes C and D mutually are subclasses of each other, they are called *equivalent*, denoted as $C \equiv D$. Other types of complex class definitions that are worth citing here in order to better understand the case study presented in our paper, include the *intersection*, denoted as $C \sqcap D$, referring to all individuals belonging to C and D, and *existential restriction*, denoted as $\exists R.C$, referring to all individuals that have an R relationship to an individual belonging to class C. Based on such class descriptions, a reasoner can infer new relations between classes.
Let us consider the following class description as example:

$3DES \sqsubseteq SymmetricAlgorithm$ and $B \sqsubseteq \exists uses.3DES$.

From this knowledge an OWL reasoner can infer the axiom :

$B \sqsubseteq \exists uses.SymmetricAlgorithm$,

because it knows that each individual belonging to B has a *uses* relationship to an individual belonging to $3DES$, and thus also belongs to $SymmetricAlgorithm$.

From this simple example, it is possible to understand how the adoption of semantic annotations in policies provides for improved flexibility and expressiveness, and allows finer policy compatibility checks.

3 SYSTEM ARCHITECTURE

The logical view of the architecture, depicted in Figure 1, recalls that of classic SOA: the requestor, the provider and the registry keep playing their original role in the publish-find-bind cycle. The novelty is represented by the *Matchmaker* and the *Reasoner* entities, along with the flows of policy-related information. When advertising a given service, the *Service Description* information can be integrated with a *Service Policy* (step 1 in the picture). According to the WS-PolicyAttachment specification, such a policy can be either attached to service's WSDL description or added as a new document to the UDDI registry. When a requestor queries the registry for a service, the latter replies with a list of the service's feature and attached policies (steps 2-3). At this stage the requestor may want to filter out the returned list in order to select the service whose policy best matches its own *Requestor Policy*. We point out that *Service Policy* and *Requestor Policy* specify requirements and capabilities of, respectively, the service requestor and the service provider. The specification of both policies is done by extending WS-Policy. Since the specification does not impose limits on the kind of allowed assertions, as already said, for the definition of the policies we have decided to adopt semantically enriched terms. This, of course, will enable semantic-based procedures of policy comparisons.

The policies are then delivered to the Matchmaker (step 4) which is in charge for the matching process. The comparison is carried out by overlapping the requirements defined in the *Service Policy* onto the capabilities expressed in the *Requestor Policy*, and viceversa. Every requirement-capability pair is assigned a match level and, in the end, the overall match level between the requestor and the provider is evaluate. As requirements and capabilities are expressed in a semantic form, the Matchmaker will invoke the Reasoner component to perform the semantic match (step 5). The outcome of the semantic match is returned back to the Matchmaker (step 6) that will provide a final response to the requestor (step 7), who will then be able to select the appropriate service and bind to it (step 8).

3.1 Security and Policy Ontology

To support the semantic description of security requirements and capabilities two ontologies have been defined: a *security ontology*, that describes security related concepts like protocol, algorithm, credential, and a *policy ontology* that defines the concept of the policy and its characterization in term of requirements and capabilities. In the following some details about the proposed ontologies are given. We point out that it is not among the objectives of this work to provide an exhaustive view of all concepts that populate the domain of security.

In the literature many have tried to define ontologies for security. The one proposed in (Garcia and Felgar de Toledo, 2008), that makes explicit references to the WS-Security's nomenclature, addresses the problem of security when messages are exchanged among web services. In (Kim et al., 2005) the proposed ontology covers most of the concepts of the security domain: despite it was defined to address security aspects at a very high levels, there are some constructs expressly designed to semantically annotate web services.

The security ontology we propose, depicted in Figure 2, draws inspiration from the solutions proposed in literature. The main concepts at the base of the security domain are Protocol, Algorithm, Credential and Objective.

A *Protocol* is a set of rules agreed by parties that need to communicate to each other. In the context of security a protocol makes use of tools, like algorithms and credentials, in order to accomplish an objective. An *Algorithm* is a procedure for solving problems that employs a finite number of steps. In literature several security algorithms, divided in as many categories, have been proposed. We can cite, for instance, the category of encryption and that of authentication algorithms. *Credentials* play an important role in information systems that require authentication. Again, there are several categories of credentials, among which we can cite the biometric (fingerprint, voice), electronic (login, password, encrypted keys, certificates), physical (smartcard, passport, identity card). Finally, by *Objective* we mean a particular security service offered by the system: authentication, authorization, confidentiality, integrity and non-repudiation. These concepts are related to each other within the ontology, and some properties (not shown in figure for the sake of simplicity) have been also defined:

- a security protocol makes use of one or more security algorithms (*hasAlgorithm* property);
- a protocol requires one or more credentials (*reqCredential* property);
- a protocol supports one or more security objective (*hasObjective* property).

The policy ontology just defines the concept of policy in the context of our framework. A policy is nothing but a list of requirements and capabilities. In the OWL formalization, the Policy Class and the related properties *hasRequirement* and *hasCapabilities*

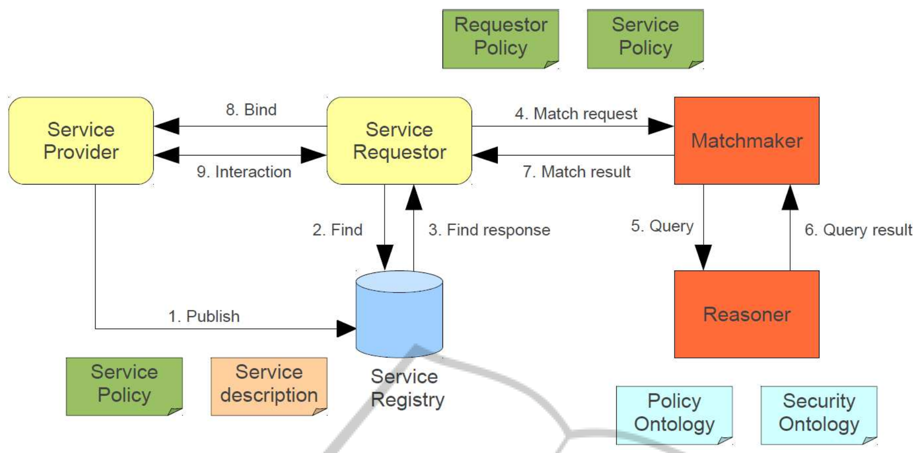

Figure 1: Overall architecture.

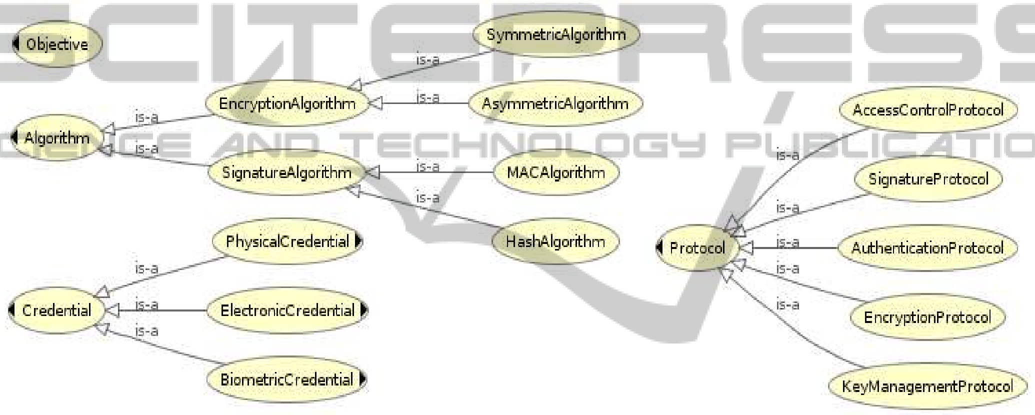

Figure 2: Security ontology.

have been created.

3.2 Matching Algorithm

The process of making the match between two policies consists in searching a correspondence between requirements and capabilities. Specifically, a) the requirements of a service are compared to the capabilities of the requestor, and b) the capabilities of the service are compared to the requirements of the requestor. In order for the comparison process to have a positive outcome, the following conditions must hold:

- the capabilities expressed in the service policy must meet the requirements expressed in the requestor policy;
- the requirements expressed in the service policy must be met by the capabilities expressed in the requestor policy.

The matchmaking process breaks down in two steps: 1) assigning the match level to each requirement-capability pair and 2) evaluating the global match level between to overall set of requirements and the overall set of capabilities. As for the first step, each requirement-capability pair can be assigned one out of a scale of four different match levels: Perfect Match, Close Match, Possible Match and No Match. For each requirement, the objective is to find the capability that matches at best. In the second step the overall match between the two policies will be evaluated. The overall match is defined to be the minimum among the individual match levels assigned in the first step for each requirement-capability pair.

In the following we detail the four levels of matching:

- **Perfect Match.** A perfect match occurs when the requirement and the capability both refer to the same concept, or to two equivalent concepts, and if properties are also specified, their respective values are identical or equivalent.
- **Close Match.** A close match occurs when the requirement is more general then the capability.

Three cases are possible: a) the requirement specifies a more general semantic concept than the capability's, i.e., a concept that is higher in the ontological hierarchy; b) the requirement and the capability refer to the same semantic concept, but more details are specified for the capability (using the "property" construct); c) the requirement is expressed in terms of security objective while the capability is expressed in terms of security concept that supports the specified objective.

- **Possible Match.** A possible match occurs when the requirement is more specific than the capability. Three cases are possible: a) the requirement specifies a more specific semantic concept than the capability's, i.e., lower in the ontological hierarchy; b) the requirement and the capability refer to the same semantic concept, but the requirement is specified more in details (using the property construct); c) the requirement is expressed as security concept while the capability is expressed in term of security objective supported by that concept.
- **No Match.** A no match occurs when the requirement and the capability have no chance to match. Two cases are possible: a) the requirement and the capability refer to semantic concepts that have no semantic relationship; b) the requirement and the capability refer to the same semantic concept but have a different specification for their properties.

Perfect, Close and No Match clearly define how requirements and capabilities match. Should the final outcome be a Possible match, nothing can be actually said about the match level of the two policies, and a further private negotiation step among the requestor and the service provider is needed to find out if there is a real chance of compatibility.

4 CASE STUDY

In this section, an example case study is presented. The description will focus on the security policies and their meaning, and on the evaluation of the matching algorithm, also explaining why a standard WS-Policy would have failed in this case.

4.1 Security Policies Definition

The definition of security policies both for the requestor requirements and the server capabilities essentially requires the use of two tools: WS-Policy, and the security ontology. The first serves as a container for the requirements and capabilities expressed through the concepts defined within the ontology.

Requestor Policy. The requestor wants to define the following policy in terms of security requirements and capabilities:

- *Requestor security requirements*
  - Message integrity
- *Requestor security capabilities*
  - SAML protocol with X.509 certificate as authentication credential
  - XML-Enc for message encryption

This policy is expressed in our system in the following way:

```
<?xml version="1.0"?>
<wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/
                        09/policy"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:security="http://www.semanticweb.org/ontologies/
                        security.owl#">
  <wsp:ExactlyOne>
        <security:DataIntegrity rdf:ID="capability0"/>
        <security:X.509Certificate rdf:ID="X.509"/>
        <security:SAML rdf:ID="capability0">
              <security:reqCredential rdf:resource=
                                  "#X.509"/>
        </security:SAML>
        <security:XML-Enc rdf:ID="capability1"/>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Service Policy. The service provider wants to define the following policy (expressed in terms of security requirements and capabilities too):

- *Service security requirements*
  - requestor authentication
  - XML-Enc protocol with AES algorithm for message encryption
- *Service security capabilities*
  - HMAC algorithm for message integrity

This policy is expressed in our system in the following way:

```
<?xml version="1.0"?>
<wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/
                        09/policy"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:security="http://www.semanticweb.org/ontologies/
                        security.owl#">
  <wsp:ExactlyOne>
        <security:Authentication rdf:ID="requirement0"/>
        <security:XML-Enc rdf:ID="requirement1"/>
           <security:hasEncryptionAlgorithm
               rdf:resource="http://www.semanticweb.org/
```

Table 1: Requestor requirements and Service capabilities matching.

Requestor requirements	**Service capabilities**	**Match level**
Message integrity	HMAC algorithm for message integrity	Close match

```
                         ontologies/security.owl#AES"/>
      </security:XMLEnc>
      <security:DataIntegrity rdf:ID="capability0">
        <security:hasMACAlgorithm
           rdf:resource="http://www.semanticweb.org/
                         ontologies/security.owl#HMAC"/>
      </security:DataIntegrity>
  </wsp:ExactlyOne>
</wsp:Policy>
```

4.2 Matching Algorithm Results

The matching algorithm tries to combine the requestor requirements with the server capabilities (and viceversa) by following the approach we are going to describe:

- The requestor's requirement of message integrity is combined with the service's capability to provide data integrity through the HMAC algorithm; the resulting level of match is *Close Match* because the objective of message integrity is satisfied by the HMAC algorithm.
- The service requirement related to requestor authentication is satisfied by the requestor capability to use SAML with X.509 certificate for authentication purposes. The resulting match level is *Close Match*, because the objective of authentication is satisfied by the SAML protocol.
- The service requirement of using XML-Enc for encryption is satisfied by the requestor using the same protocol for this purpose. The resulting match level is *Possible Match*, because both the requirement and the capability express the same concept, but the requirement is defined with greater detail (the service explicitly requires the AES algorithm).

The results are summarized in tables 1 and 2; it may be noticed that the lowest level in all pairs formed by the three capability-requirement combinations is *Possible Match*. The Matchmaker informs the requestor that the service satisfies its security requirements. As already explained in Section 3.2, in this case the requestor should further inquiry the service provider before invoking the service, in order to ascertain that the policy compatibility is granted.

Table 2: Service requirements and Requestor capabilities matching.

Service requirements	**Requestor capabilities**	**Match level**
Requestor authentication	SAML with X.509 certificate	Close match
XML-Enc with AES algorithm	XML-Enc	Possible match

We would like to outline that, if using the standard WS-Policy model, the above policies would not have matched since there is no explicit knowledge 1) that Data integrity could be supported by the HMAC algorithm and 2) that requestor authentication could be performed by SAML mechanisms.

5 RELATED WORK

Several policy languages and framework have been developed by adopting different approaches and used in different application domains (Tonti et al., 2003). Among the most notably efforts in this domain it is worth citing Ponder (Damianou et al., 2001), a declarative object-oriented language that supports the specification of several types of management policies for distributed object systems, Kaos (Uszok et al., 2003), a policy management framework where concepts and policies themselves are represented using OWL, and Rei (Kagal et al., 2003), a policy framework where OWL is extended with the expression of relations like role-value maps, making the language more expressive than the original OWL. Kaos and Rei are ontology based policy languages that, although not specifically focused on the security domain, are able to express complex security policies: policy matching is also supported by the advanced reasoning capabilities these languages offer. However, all these languages are not compatible with existing standards for policy specification in service oriented architectures. For this reason, similarly to our approach, several works in the literature (Sriharee et al., 2004; Verma et al., 2005; Speiser, 2010; Zheng-qiu et al., 2009) have been trying to enhance WS-Policy with semantic annotations. In (Sriharee et al., 2004), WS-Policy assertions refer to policies expressed in OWL: however that work is not focused on policy matching, but on modeling policies as a set of rules, which have to be evaluated using an external rule-based system, requiring reasoning beyond OWL. In (Verma et al., 2005) policies represented in WS-Policy are enhanced with semantics by using a combination of OWL and SWRL based rules to capture domain concepts and rules. In (Speiser, 2010) a lightweight approach to specify semantic annotations in WS-Policy is presented: it combines the syntactic matching with the semantic matching capability provided by OWL.

6 CONCLUSIONS AND FUTURE WORK

Defining, modeling and matching security policies is a crucial problem that needs to be faced when dealing with services spanning multiple administrative domains. The existing models inspired to syntactic approaches are not very well suited for these heterogeneous and dynamic scenarios. In this paper we proposed to leverage on the existing WS-Policy specification, proposing a semantic extension that enable semantic mechanisms to matchmaking security capabilities and requirements. The presented approach allows to go beyond the strict syntactic intersection of policy assertions. A security ontology is proposed to catch the relationships among the concepts of security Objective, Protocols, Algorithms and Credentials. A simple example has also shown the viability of the semantic approach and the actual limits of the pure syntactic one. Future works will be aimed to improve the capability to express more complex policies and to enhance the ability of inference.

ACKNOWLEDGEMENTS

The work described in this paper has been partially supported by the MIUR-PRIN 2008 project *“Cloud@Home: a New Enhanced Computing Paradigm”*.

REFERENCES

Damianou, N., Dulay, N., Lupu, E., and Sloman, M. (2001). The ponder policy specification language. In *Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, POLICY ’01, pages 18–38, London, UK. Springer-Verlag.

Garcia, D. Z. G. a. and Felgar de Toledo, M. B. (2008). Ontology-Based Security Policies for Supporting the Management of Web Service Business Processes. In *2008 IEEE International Conference on Semantic Computing*, pages 331–338. Ieee.

Kagal, L., Finin, T., and Joshi, A. (2003). A policy language for a pervasive computing environment. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, POLICY ’03, pages 63–, Washington, DC, USA. IEEE Computer Society.

Kim, A., Luo, J., and Kang, M. (2005). Security ontology for annotating resources. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, pages 1483–1499. Springer.

Lakshminarayanan, S. (2010). Interoperable security standards for web services. *IT Professional*, 12(5):42 –47.

OASIS (2006). Web Services Security (WS-Security). OASIS Standard.

Phan, T., Han, J., Schneider, J., Ebringer, T., and Rogers, T. (2008). A survey of policy-based management approaches for Service Oriented Systems. In *Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on*, pages 392–401. IEEE.

Speiser, S. (2010). Semantic Annotations for WS-Policy. In *IEEE International Conference on Web Services (ICWS 2010)*, pages 449–456. IEEE.

Sriharee, N., Senivongse, T., Verma, K., and Sheth, A. (2004). On using ws-policy, ontology, and rule reasoning to discover web services. In *Intelligence in Communication Systems*, number May 2004, pages 246–255. Springer.

Tonti, G., Bradshaw, J., Jeffers, R., Montanari, R., Suri, N., and Uszok, A. (2003). Semantic Web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder. In *International Semantic Web Conference (ISWC2003)*, pages 419–437, Florida (USA). Springer.

Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S., and Lott, J. (2003). Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, POLICY ’03, pages 93–, Washington, DC, USA. IEEE Computer Society.

Verma, K., Akkiraju, R., and Goodwin, R. (2005). Semantic matching of Web service policies. In *Semantic Web Policy Workshop (SDWP 2005)*.

W3C (2007). Web services policy 1.5 - framework. W3C Recommendation.

W3C (2009). OWL 2 Web Ontology Language. W3C Recommendation.

Zheng-qiu, H., Li-fa, W., Zheng, H., and Hai-guang, L. (2009). Semantic Security Policy for Web Service. In *2009 IEEE International Symposium on Parallel and Distributed Processing with Applications*, pages 258–262. Ieee.