A MODEL FOR AUTOMATIC MATCHING OF SECURITY REQUIREMENTS DURING SEMANTIC WEB SERVICE DISCOVERY

Andreas Friesen, Danna Feng
SAP Research CEC Karlsruhe,SAP AG,Vincenz-Priessnitz-Str.1 ,D-76131 Karlsruhe,Germany

Keywords: Semantic Web Services, Security, Security Ontology, Security Policy, Security Requirements, Semantic Discovery, Semantic Selection, QoS.

Abstract: This paper describes a semantic approach for modelling security requirements of requesters and providers of Semantic Web Services. These semantic descriptions can be used either during semantic service discovery or service selection phase for automatic compatibility verification of the security requirements of a service requester and provider. The security requirements model, ontology classifying existing security services and mechanisms, and a semantic matchmaking method relying on description logics are described in detail. This work is related to several semantic and non-semantic Web Services standards. The relationship to the most relevant of them has been worked out.

1 INTRODUCTION

Web Services enabled business systems can be used by anyone, from anywhere, at any time, and on any type of platform. Semantic Web Services promise a higher degree on automation concerning discovery, invocation, composition, and monitoring of Web Services. Security and trust are very important factors for the success of the Semantic Web. In this work, the security requirements for Semantic Web Services are described in a manner that the security mechanisms based on the existing security standards will be represented in formal logics (to be more precisely in description logics (Baader, F. et al, 2003)). Description logics are directly supported by one of the Web Ontology Language (OWL) dialects OWL-DL (Smith, M. K. et al, 2005). This allows taking into account the security requirements of requesters/providers during the Semantic Web Service discovery or selection phase, i.e., enabling automatic compatibility verification.

For the representation of different security mechanisms we have chosen as the basis the Web Services Security Policy Language (Web Services Policy Language, 2005) that has collected many standard security mechanisms. All security mechanisms are represented as classes in OWL-DL. The security requirements on the security services are described either using these classes directly or by logical combinations of these classes.

At first, we introduce the Web Services Policy Framework (WS-Policy) (Web Services Policy Framework, 2005) and Web Services Security Policy Language (WS-SecurityPolicy) (Web Services Policy Language, 2005). Then, we describe our *basic concept* for realizing security requirements matching.

The classes of security services and the standard security mechanisms are then formally described in an ontology that is used in the expressions of security requirements of communicating parties.

In order to realize automatic matching of security requirements, they must be described in a machine understandable language. WS Policy describes requirements in the form of policies, policy alternatives and assertions. We map the concepts used in WS-Policy to OWL-DL starting from the work described in (Kolovski, V. et al, 2005) and extending it.

Finally, an example demonstrates how to describe security requirements in OWL-DL at the capability-level and how to test the compatibility between the requirements of two communicating parties.

Friesen A. and Feng D. (2006).
A MODEL FOR AUTOMATIC MATCHING OF SECURITY REQUIREMENTS DURING SEMANTIC WEB SERVICE DISCOVERY.
In *Proceedings of WEBIST 2006 - Second International Conference on Web Information Systems and Technologies - Internet Technology / Web Interface and Applications*, pages 387-392
DOI: 10.5220/0001253603870392

2 WEB SERVICES POLICY LANGUAGE

A Policy for a Web Service consists of facts, or assertions, and rules that apply to a particular Web Service. A policy would be used to describe or point to documents describing the owning business, associated products, keywords, taxonomies for the service, security policies, quality of service attributes, etc. A Policy may be used by the overarching concerns: security, quality of service, and management (Web Services Architecture, 2005).

Web Services Policy Framework (WS-Policy) (Web Services Policy Framework, 2005) provides a general purpose model and corresponding syntax to describe the policies of a Web Service. WS-Policy defines a base set of constructs for expressing the capabilities, requirements and general characteristics of entities in a Web Services based system and can be extended by other Web Services specifications. The requirements and capabilities of a policy subject are specified by policy assertions. A policy subject is an entity (e.g., service provider, service requester, message, resource, interaction) with which a policy can be associated. A collection of policy assertions builds a policy alternative. WS-Policy defines a policy as a collection of policy alternatives and offers a normal form for policy expression which is outlined as follows:

```
<wsp:Policy …>
  <wsp:ExactlyOne>
    [<wsp:All>
      [<Assertion …> … <Assertion>]*
    </wsp:All>]*
  </wsp:ExactlyOne>
<wsp:Policy>
```

<wsp:Policy> indicates the beginning of a policy expression. *<wsp:ExactlyOne>* defines a collection of policy alternatives. *<wsp:All>* defines a policy alternative – a collection of policy assertions. The star character * denotes zero or more occurrences.

WS-SecurityPolicy (Web Services Policy Language, 2005) indicates the policy assertions with respect to security features. It defines a base set of assertions that describe how messages are to be secured (e.g., Integrity Assertion, Confidentiality Assertion) and which token types (e.g., X509Token Assertion, KerberosToken Assertion), cryptographic algorithms (e.g., AlgorithmSuite Assertion) and mechanisms should be used.

WS-Policy and WS-SecurityPolicy provide us a normal form for expressing security policies, but are not sufficient for automatic matching because of the lack of semantics.

3 SECURITY MODEL

In this work, we consider a simple Semantic Web Services model with two actors: Service Requester and Service Provider. The main focus of the approach is to capture the security requirements of the requesters and providers on the Semantic Web Services at the capability level not at the message level. We define therefore policy alternatives as various security requirements alternatives that have the following services: Authentication, Authorization, Integrity, Confidentiality and Non-repudiation. These services are chosen, because they are most common and important for security solutions. In the Web Services world, there are many security standards implemented to realize these security services. In the Semantic Web Services world, although there are some security approaches advertised, a conclusive solution is still expected.

Semantic Web Services are technologically seen not totally different from Web Services. Quite the contrary, they are an extension of Web Services. In (Berners-Lee, T., 1998)(Dumbill, E., 2000) Tim Berners-Lee points out that the Semantic web extends the World Wide Web through the use of standards, mark-up languages and related processing tools. Figure 1 illustrates the layered architecture of the Semantic Web.

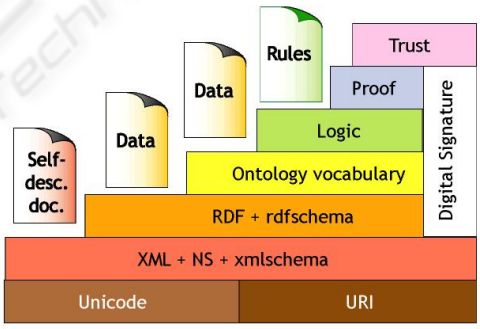

Figure 1: Architecture of Semantic Web.

The first two levels describe the traditional Web. URI enables the navigation of resources, while Unicode enables computer to “read” the content of resources. XML provides a basic format for structured documents but without particular semantics. RDF can describe the resources and their properties and make this information machine understandable, but is very limited for logical description of resources and their relationships (e.g., negation or intersection). RDF Schema declares the existence of properties and can constrain the types of objects they can apply to. The ontology layer offers more meta-information such as transitive property or cardinality of the objects. The logic layer enables describing logic (e.g. union, intersection, predicate

logic and quantification) in order to realize proof by using the inference rules defined on the logic layer.

In the light of this architecture, our approach reuses to a certain degree the security standards and mechanisms of the Web Services. Based on these standards, we define an ontology formalizing the concepts and relationships used to describe security services and requirements. We describe which security standards and mechanisms can be used for each service. These security standards and mechanisms represent the basic components (classes) for describing this ontology. By using the classes defined in this ontology, a service provider can define the security policies for a Web Service and a service requester can also describe security policies of its goal. Furthermore, it is important to distinguish between supported security requirements and necessary security requirements. A supported requirement of an entity (a service provider or a service requester) means that the entity supports all mechanisms applied to realizing the services specified in this requirement. A necessary requirement of an entity means that the entity requires some special mechanisms that its potential communicating party must support. These specially required mechanisms must be also supported by the entity itself. Hence, necessary security requirement is a subset of supported requirement. The policy matching takes place between necessary requirements on one side and supported requirements on the other side. The following example demonstrates the necessity of the differentiation between supported requirements and necessary requirements. If a service requester supports signature algorithms RSA and DSA, while a service provider supports signature algorithms DSA and ECDSA. Without the differentiation between supported requirements and necessary requirements, their security requirements should be compatible, because both of them support at least one algorithm DSA. In the case that the service provider provides a certificate with a ECDSA key, the service requester can not verify signatures created using this key, because it doesn't support ECDSA. This differentiation realizes more granularities for expression of security requirements. Figure 2 illustrates this idea.

In the following, it will be showed which security standards and mechanisms can be used for describing security services.

Authentication: Authentication can be realized by using security token, e.g., KerberosToken or X509Token. Furthermore, the communicating parties can also be authenticated by using their digital signatures or only by using a random value depending on the used token type.

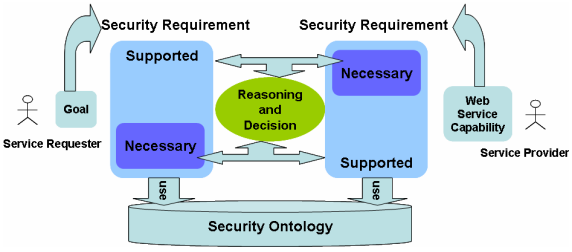

Figure 2: The basic concept.

Integrity: Using KerberosToken or X509Token combined with digital signature can provide data integrity protection.

Confidentiality: Several Encryption algorithms support confidentiality.

Non-Repudiation: Non-Repudiation can be guarantied by using digital signature and timestamp or/and a random value.

Authorization is currently a subject for further work. There are some considerations about authorization such as using RelToken (Rights Expression Language) (Web Services Security, 2005), XACMLToken (eXtensible Access Control Markup Language) (eXtensible Access Control Markup Language, 2005), SAML (Security Assertion Markup Language) (Web Services Security, 2003), and the credential based access control suggested in (Agarwal, S. et al, 2004).

Figure 3 illustrates the simplified class diagram of the concepts (taxonomy) used to specify security requirements in the security model.

4 REPRESENTING WEB SERVICE SECURITY REQUIREMENTS IN OWL-DL

The previously indicated assertions defined in WS-SecurityPolicy are only for expressing security constraints and capabilities and suffer from a lack of formal semantics. The intent of this work is to realize the automatic matching of available security requirements between communicating parties which must be described with machine understandable metadata. A taxonomy describing security concepts has been defined in the last section. By using the basic elements – the classes defined in this taxonomy, WS-SecurityPolicy can be described in a machine understandable language.

An approach of mapping the WS-Policy language into the description logic fragment of the Web Ontology Language (OWL-DL) has been proposed in (Kolovski, V. et al, 2005). An investigation about ontology based specification of Web services policies has been described in

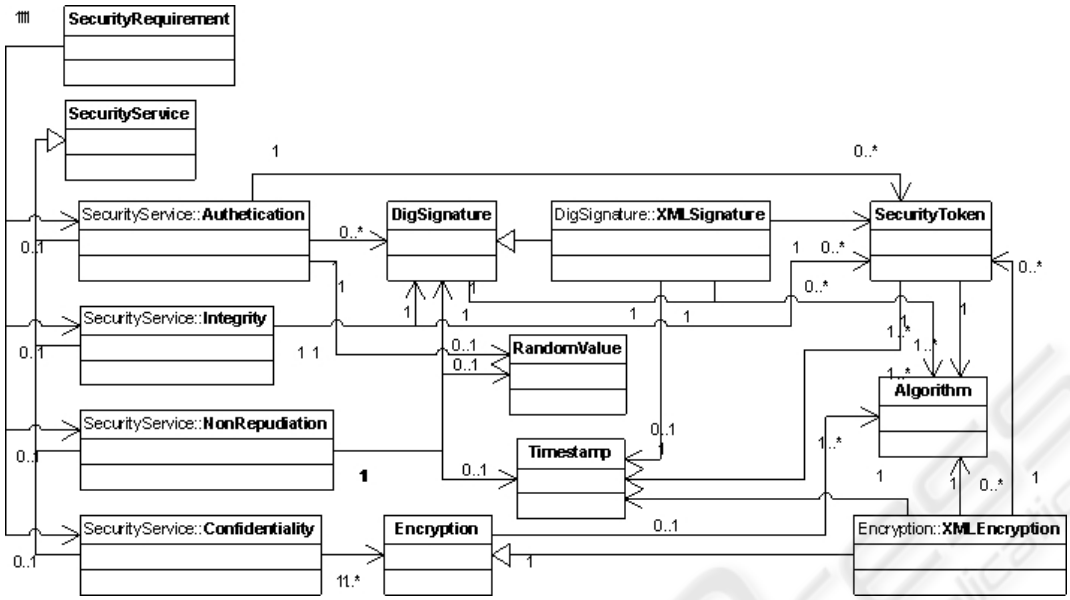

Figure 3: Class diagram of security services (simplified).

(Grimm, S. et al, 2004). We reuse parts of these approaches and extend them in order to describe Web Services security requirements.

WS-Policy involves policy assertions and combinations of assertions. Therefore, by describing the assertions as atomic propositions and the combinations of the assertions by conjunction/ disjunction, it is possible to map the policy language constructs into logic. This mapping defines a clear semantics for the WS-Policy structures.

wsp:ExactlyOne means that at least one of the alternatives in the policy must be supported by a service requester, so that this policy can be supported by the requester. However, the requester can only apply exactly one valid policy alternative.

wsp:All means all of the policy assertions mentioned in this policy alternative must be supported by a requester. Thus, it is a logic conjunction and can be expressed as an OWL intersection.

The normal form of WS-Policy can be represented in OWL as follows:

$$Policy \equiv UnionOf\ (PolicyAlternative_1, \ldots PolicyAlternative_n)\ (n \geq 0)$$
$$PolicyAlternative_i \equiv IntersectionOf\ (Assertion_1, \ldots, Assertion_m)\ (0 \leq i \leq n, m \geq 0)$$

In our proposal, the mapping is extended to supply description of security requirements as policy alternatives. Security policy can be treated as a policy that is represented by the union of various security requirements. Each of them can consist of five previously mentioned security services, which can be described as policy assertions. However, in the future the model can be extended to support additional security services. A formal description of a security requirement in DL is as follows:

$$SecurityRequirement \sqsubseteq$$
$$((\leq 1\ hasAuthentication.Authentication)$$
$$\sqcap (\leq 1\ hasAuthorization.Authorization)$$
$$\sqcap (\leq 1\ hasIntegrity.Integrity)$$
$$\sqcap (\leq 1\ hasConfidentiality.\ Confidentiality)$$
$$\sqcap (\leq 1\ hasNonRepudiation.\ NonRepudiation))$$

At first, a security requirement contains each security service exactly one time, if the requirement supports it. The specification of a specific requirement (specialization of *SecurityRequirement*) is then represented in this case with "=1". Each security service described in this definition as assertion can be represented as a logical expression of various security means or mechanisms. How to describe assertions in DL will be shown below. Secondly, in the case that a party can not support one or more of these five services, its security requirement must not contain these services. Thus "≤1" is applied to the common specification instead of "=1". In the specification of this requirement, the unsupported security service must be represented as "⊥" – the bottom concept. In OWL-DL this concept is described with *owl:Nothing*. Finally, it is also possible that a specialization of *SecurityRequirement* can support all of the restrictions defined for a

specific security service. That is, this requirement can support all kinds of security mechanisms defined for that security service. In this case the specialization of the *SecurityRequirement* does not define further constraints on the security service.

Each assertion can be mapped directly into a general class of OWL-DL or a class with restriction that is described by using object properties and other classes.

For the matching of two policies, we define two policies as compatible, if $Policy_1 \sqcap Policy_2$ is satisfiable.

The following example illustrates an authentication requirement. The compatibility test of the requirements of service requester and service provider is also outlined.

In this example, authentication is realized by using XML Signature (XML-Signature Syntax and Processing, 2005). For simplified description, except signature algorithms (only asymmetric algorithms), the other algorithms such as digest algorithms, canonization algorithms and transform algorithms usually also needed for XML Signature are not further specified.

First, we define the supported security requirements of service requester and service provider.

$PolicyRequirementProviderSupported \sqsubseteq ((\exists\ hasAuthentication.AuthProvider_1) \sqcap SecurityRequirement)$

$PolicyRequirementRequesterSupported \sqsubseteq ((\exists\ hasAuthentication.AuthRequester_1) \sqcap SecurityRequirement)$

They both support authentication services realized by XML Signature, whereat the provider supports $XMLSig_2$, and the requester supports $XMLSig_1$.

$AuthProvider_1 \sqsubseteq ((=1\ hasDigitalSignature.XMLSig_2) \sqcap Authentication)$

$AuthRequester_1 \sqsubseteq ((=1\ hasDigitalSignature.XMLSig_1) \sqcap Authentication)$

$XMLSig_1$ uses algorithms *ECDSA* or *RSA* for ciphering, while $XMLSig_2$ supports only algorithms *RSA* or *DSA*.

$XMLSig_1 \sqsubseteq ((=1\ hasCipher.(RSA \sqcup ECDSA)) \sqcap (XMLSignature))$

$XMLSig_2 \sqsubseteq ((=1\ hasCipher.(RSA \sqcup DSA) \sqcap (XMLSignature))$

The service provider has a certificate signed with DSA, while the service requester has a certificate signed with RSA. In this case, the service provider requires signature algorithm DSA as his necessary requirement, while the service requester requires RSA as his necessary requirement. The necessary requirements must be a subset of supported requirements.

$XMLSigNecessary_1 \sqsubseteq ((=1\ hasCipher.RSA\ (XMLSignature))$

$XMLSigNeccessary_2 \sqsubseteq ((=1\ hasCipher.DSA) \sqcap (XMLSignature))$

Based on the definition of compatibility of policies as indicated above and the differentiation of supported requirement and necessary requirement, we define two security requirements as compatible, if:

- ProviderSupported ⊓ RequesterNecessary is satisfiable, and
- ProviderNecessary ⊓ RequesterSupported is satisfiable.

$XMLSig_1 \sqcap XMLSigNecessary_2$ is not satisfiable. Therefore, the security requirements of the provider and the requester are not compatible.

5 CONCLUSIONS AND FUTHER RESEARCH DIRECTIONS

In this work, a model for describing Web Services security requirements at the capability level was built. This model illustrates our basic idea for realization of automatic matching of security requirements in the Semantic Web Services world, and has been prototypically implemented and tested using Protégé (The Protégé Ontology Editor and Knowledge Acquisition System, 2005) with OWL plug-in and Racer (Racer System, 2005). Furthermore, a user friendly interface has been implemented, with which the security requirements can be reasonably described and matched.

The subjects of further work are summarized as follows:

At this time, there are no broadly-adopted specifications for web services security. In this work, many web services security recommendations and standards of W3C and OASIS were treated as basics for the ontology described in this paper. However, it enables the developer to extend this

ontology with more security mechanisms according to more security requirements.

As described previously, this model will be extended with authorization capability.

The model described in this paper is a simplified model that contains two communicating parties: service requester and service provider, and the composed services are not considered within the scope of this paper, which can be involved in the further implementation.

The matching of security requirements is at the capability level of web services. Finally, we proposed ontology for modelling security requirements and capabilities of security services, which can be treated as basis to describe security services as part of QoS in OWL-S (OWL Web Ontology Language for Services (OWL-S), 2005) in future work.

REFERENCES

Agarwal, S., Sprick, B., and Wortmann S. (2004). *Credential Based Access Control for Semantic Web Services.* Retrieved November 20, 2005, from University of Karlsruhe, Institute of Applied Informatics and Formal Description Methods (AIFB) web site: http://www.aifb.uni-karlsruhe.de/WBS/sag/papers/Agarwal_Sprick_Wortmann-CredentialBasedAccessControlForSemanticWebServices-AAAI_SS_SWS-04.pdf.

Baader, F., Calvanese, D., McGuinnes, D., Nardi, D., and Patel-Schneider, P. (2003). *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Berners-Lee, T. (1998, September). Semantic Web Road map. Retrieved November 25, 2005, from http://www.w3.org/DesignIssues/Semantic.html.

Dumbill, E. (2000, December 6). Berners-Lee and the Semantic Web Vision. Retrieved November 25, 2005, from http://www.xml.com/pub/a/2000/12/xml2000/timbl.html.

eXtensible Access Control Markup Language (XACML) Version 2.0. Oasis standard. (2005, Feb. 1) Retrieved October 12, 2005, from http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

Grimm, S., Lamparter, S., Abecker, A., Agarwal, A., Eberhart, A. (2004). *Ontology based Specification of Web Service Policies.* Retrieved November 20, 2005, from University of Karlsruhe, Institute of Applied Informatics and Formal Description Methods (AIFB) web site: http://www.aifb.uni-karlsruhe.de/WBS/sag/papers/Grimm_Lamparter_Abecker_Agarwal_Eberhart-OntologyBasedSpecificationOfWebServicePolicies-SemanticWebServicesAndDynamicNetworks-Informatik04-04.pdf.

Kolovski, V., Parsia, B., Katz, Y., and Hendler, J. (2005). Representing Web Services Policies in OWL-DL. Retrieved November 24, 2005, from http://www.mindswap.org/papers/2005/Policy-ISWC05.pdf.

OWL Web Ontology Language for Services (OWL-S). November 2, 2004. Retrieved November 25, 2005, from http://www.w3.org/Submission/2004/07/.

Racer system. Retrieved November 27, 2005, from http://www.racer-systems.com/de/index.phtml.

Smith, M. K., Welty, C., McGuinness D. L. (Feb, 2004). OWL Web Ontology Language Guide. W3C Recommendation. Retrieved November 8, 2005, from http://www.w3.org/TR/owl-guide/.

The Protégé Ontology Editor and Knowledge Acquisition System. Retrieved October 12, 2005 from http://protege.stanford.edu.

Web Services Architecture. November 14, 2002. W3C working draft. Retrieved November 25, 2005, from http://www.w3.org/TR/2002/WD-ws-arch-20021114/id2616445.

Web Services Policy Framework (WS-policy). September, 2004. Retrieved September 4, 2005, from http://schemas.xmlsoap.org/ws/2004/09/policy/.

Web Services Security Policy Language (WS-SecurityPolicy).July, 2005. Retrieved September 7, 2005, from http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf.

Web Services Security: Rights Expression Language (REL) Token Profile. December 19, 2004. Oasis Standard. Retrieved October 5, 2005, from http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf.

Web Services Security: SAML Token Profile. February 21, 2003. OASIS Working Draft 06. Retrieved November 14, 2005, from http://www.oasis-open.org/committees/download.php/1048/WSS-SAML-06.pdf.

XML-Signature Syntax and Processing. February, 2002. W3C Recommendation 12. Retrieved November 16, 2005, from http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/Overview.html.