Security in the Web Services Framework

Chen Li and Claus Pahl

Dublin City University
School of Computing
Dublin 9
Ireland

Abstract

The Web Services Framework provides techniques to enable the application-to-application use of the Web. It has the potential of becoming the core of a new Web-based middleware platform, providing interoperability between computational services using Web- and Internet-technologies. Security is of course of major importance in this context. We introduce here extensions to two major building blocks of the Web Services Framework – the Web Services Description Language WSDL and the Universal Description, Discovery, and Integration Service UDDI. We add description mechanisms and matching techniques that support the retrieval of Web Services from repositories.

1 Introduction

Researchers and practitioners have started work on a Web-based middleware platform with the *Web Services Framework* (WSF) at its centre [6]. The WSF provides a services description notation (Web Services Description Language WSDL), a repository facility (Universal Description, Discovery, and Integration Service UDDI), and a communications protocol for service invocations (Simple Object Access Protocol SOAP). All components are based on the eXtensible Markup Language XML.

Even though security on the Internet is paramount and must, consequently, be a major aspect for a *Web-based middleware architecture*, this has not been sufficiently addressed in the WSF context. Three security aspects in relation to the WSF can be identified:

- Describing security requirements and constraints for the application of Web services.
- Retrieving Web services from repositories that match client security requirements.
- Implementing security requirements when services are invoked across the Web.

We will focus on the first two aspects [3]. With respect to the third aspect more work on architectural issues is needed, which goes beyond the scope here.

Our starting point will be a security object notion, which will lead to a simple extension of the WSDL. This introduces our key concepts for representing various security issues in form of security objects. In a second step, we will introduce an ontological framework for Web services security description and matching. This can either replace the extended WSDL or support UDDI-based matching and retrieval features. This aspect will draw heavily on another central Web technology for the future, the *Semantic Web* [7].

2 Description of Web Services Security

A *Web service* is essentially a Web interface to a coherent set of operations. WSDL is an *interface definition language* (IDL) that addresses the structural and functional description of Web services. The design objective is the application-to-application use of the Web platform, providing interoperability for distributed heterogeneous software systems. Port types are the central abstraction concept to capture the connection points of the Web communications infrastructure.

The starting point for this investigation into Web services security is the description of security requirements for Web services applications. Our objectives are to find a coherent representation of security requirements in a WSDL-style and to integrate the security aspects into WSDL as a simple language extension.

2.1 Permission and Obligation Rules

Security is about protecting assets. In the Web services context data and computational services are assets under consideration [5]. *Security aspects* that apply to the assets are:

- *Confidentiality* – the prevention of unauthorised disclosure of data.
- *Integrity* – the prevention of unauthorised modification of data.
- *Availability* – the prevention of withholding of data or services from authorised users.
- *Authentication* – the proven identification of users in a computer system.
- *Accountability* – the provision of activity logs recording all user activity.

Security policies for computer security are usually formulated in terms of access control policies [2]. These are usually expressed in terms of *triples* (S,A,O) consisting of a subject S, an activity A, and an object O. We shall called these triples *security rules*. We can use these rules to express the relevant security aspects in this context. A closer look at the security aspects shows their relationship to the elements of the security rules, (subjects, activities, objects): confidentiality: O; integrity: O; authenticity: S; availability O,A; accountability: S,A,O.

We define a Web services security language based on security rules which addresses two different types of security: computer security and communications security. *Computer security* is essentially *access control* within a computer system. *Communications security* is about providing a *secure logical connection* between two agents. The security aspects such as confidentiality, integrity, or authentication are relevant for both contexts. *Security rules* – in form of triples – are the central notational concept to express security requirements in our approach. We distinguish two forms for networked environments.

- Communications security is a connection-oriented security focus. A *requirements rule* expresses *obligations* of the client and of the server. Such a rule expresses the necessary security-relevant preparations for the use of a service, or security measures needed after the service execution (securing return data). Sample rules are (*Server*, *authenticates*, *Client*) or (*Client*, *encrypts*, *Data*). The activities *authenticates* and *encrypts* are associated with authentication and confidentiality, respectively.
- Computer security is a node-oriented security focus. A *permission rule* expresses *capabilities* of the client at the server side. Such a rule expresses restrictions on the

actual service usage, e.g. which operations at which service object are allowed to be executed. An example is (*Client*, *operation*, *Service*) which means that a *Client* is allowed to use *operation* at *Service*.

We have introduced a language that expresses security policies in terms of activities that prevent the violation of security properties. This more operational specification of security is in contrast to declarative security specifications in terms of the security aspect to be guaranteed. Here, where an agreement between client and provider has to be achieved in terms of the actual security techniques to be used, an operational form is more adequate. However, a mapping of these activities to the abstract properties they aim to guarantee shall be assumed. The abstract activities still need to be mapped to concrete techniques, e.g. authentication through either usernames/passwords or digital certificates.

Even though this is not our objective here, a declarative specification of security aspects could be facilitated in terms of abstract security predicates, which directly encode one of the security aspects. This approach is more appropriate if general reasoning about security properties is envisaged.

2.2 sWSSL

Our approach to expressing security for WSDL is to introduce *security objects*. Security objects are security specifications associated with subjects. A security object consists of several (S,A,O)-security rules for communication and computer security for one particular subject S. We integrate security object into WSDL descriptions, creating an extension called the *simple Web Services Security Language sWSSL.*

```
<secObj    name=″..″  subject=″..″  >
    <rule  name=″..″
           type=″permission″   activity=″..″   object=″..″ />
    ...
    <rule  name=″..″
           type=″requirement″  activity=″..″   object=″..″ />
    ...
</secObj>
```

This is a security object template in XML-format. Several permission or requirements rules can be specified for one subject in a security object.

Ports are the connection points of Web services. Security specifications have to relate to the port type specifications. In order to allow specific security requirements to be attached to for example individual message elements, but also to apply a policy for an entire service, we allow security objects to be associated with different elements such as port types, operations and messages. We combine this with a notion of *scope*. A security object is valid for all XML elements subordinated to the element under question, except when it is superseded by a local one.

```
<portType     name="BankingService     security="tns:BankSec”>
   <operation name="AccBalance"        security=”tns:BalSec” >
       <input  message="tns:User"/>
       <output message="tns:AccBalResp"/>
   </operation>
   <operation name="FundsTransfer">
       <input  message="tns:User"      security=”tns:MsgSec”/>
       <input  message="tns:TransfReq" security=”tns:MsgSec”/>
   </operation>
</portType>
```

This excerpt from an online banking service refers to different security objects (referred to by name) used on different levels. `BankSec` is the security object for operation `FundsTransfer`, except when it is superseded by `MsgSec` for the input elements.

Similar to WSDL service bindings that specify the location and communications protocol to be used in the actual service invocation, security bindings have to be provided that implement the required security policy. This needs to address communication and access control. The focus of this paper is description and matching of security specifications. However, we will address security bindings later on briefly.

3 Web Services Security Description and Reasoning

The language sWSSL is a simple extension of WSDL. This approach would allow existing tools and infrastructures to be used and would make it easy to understand for the WSF community. However, a more advanced alternative to sWSSL shall be introduced. If reasoning about security properties is envisaged, the Web-based representation of security knowledge is the starting point. We propose an *ontological approach* to *Web services security description* and *reasoning*.

RDF – the Resource Description Framework [7] – is the basic Web technology for *knowledge representation*. RDF enables the specification of knowledge in form of triples – similar to our representation of security rules. RDF, however, is a general framework, allowing us to describe any concept in terms of its relationships to other (more basic) concepts. We talk about subjects, properties, and objects in the RDF. There is a richer framework building up upon RDF, which is more suitable for *ontology definitions*. DAML+OIL is a Web-based ontology language [1]. Besides a richer set of primitives it also has a strong grounding in logics, which provides reasoning support.

3.1 Security Knowledge Description

Since we have already used a triples-format for security specifications, these can easily be rewritten in terms of RDF- or DAML+OIL-triples. *Triples* are basic descriptional entities in RDF. A *class* of *subject elements* is described in terms of *properties* in relation to another *class* of *object elements*. The essential extension that we need here is to express *combinations of security rules*. Ontology languages provide *concept combinators* corresponding to set-theoretic operators such as intersection or union (or alternatively their logical counterparts conjunction and disjunction). The concept definitions in terms of triples can be seen as forms of quantified logical expressions.

```
<daml:Class>
   <daml:Restriction>
      <daml:onProperty  rdf:resource="#Client"/>
      <daml:toClass>
         <daml:intersectionOf rdf:parseType="daml:collection">
            <daml:Restriction>
               <daml:onProperty rdf:resource="#authenticates"/>
               <daml:hasClass   rdf:resource="#Server"/>
            </daml:Restriction>
            <daml:Restriction>
               <daml:onProperty rdf:resource="#operation"/>
               <daml:hasClass   rdf:resource="#Service"/>
            </daml:Restriction>
         </daml:unionOf>
      </daml:toClass>
   </daml:Restriction>
</daml:Class>
```

This DAML+OIL example describes a composite rule consisting of two abstract rules, (`Client`, `authenticates`, `Server`) and (`Client`, `operation`, `Service`), for a given subject. The two rules are combined using conjunction (`intersection`).

The idea of reformulating security objects in terms of RDF/DAML+OIL can even be applied to WSDL as a whole, describing a full WSDL specification in terms of RDF [4]. An example of WSDL in RDF is to express the structural constraints, e.g. *(portType, consistsOf, message)*. Another example is to capture extensions of WSDL by syntactical and semantical (functional) specifications, such as pre- and postconditions, e.g. *(service, preCond, Cond)* or *(service, hasSyntax, signature).*

3.2 Security Requirements Matching

Subsumption is the central reasoning construct for ontologies. Concepts in ontologies represent classes of elements. Subsumption is the subclass or inclusion relationship on concept classes. We will base our matching approach for security specifications on subsumption. Note, that this is a symbolic representation of security properties and matching. The validation or verification of those properties against the communications and the actual service implementations, or an intruder analysis, is here not aimed at.

Matching of possibly composite security rules shall be defined based on subsumption.

- *Requirements rule*. The client must at least satisfy the provider security specification, i.e. must be better. Usually, server requirements are expressed using conjunctions. A provider might require *confidentiality(data)* and a client might guarantee *confidentiality(data) and integrity(data)* – we have expressed this as *predicates P* applied to *objects O*, i.e. *P(O)*, for some given subject. This is the case if the client specification is stronger, i.e. *confidentiality(data) and integrity(data)* is included or subsumed in *confidentiality(data).*

- *Permission rule*. The client must not exceed the provider specification, i.e. must not access more than allowed. Usually, server permissions are expressed in a disjunctive form. A provider might allow *operation1 or operation2* at *service* and a client might require *operation1* at *service*, i.e. the client request is included in the permissions.

For both rules, the subsumption (inclusion) relationship is the key criterion. However, server specifications are usually either conjunctive or disjunctive.

4 Conclusions

Securing the Web Services platform is crucial for its success as a middleware platform for distributed software applications. The major contributions of this paper are, firstly, a uniform format to describe communications and computer security through security objects, and, secondly, coherent integrations of this format into WSDL as a proper extension and also into an ontology framework for security knowledge representation and reasoning. A straightforward extension of WSDL is possible through security objects, but in order to exploit the full potential of the Web, using RDF and the Semantic Web technologies provides additional benefits for description, matching and reasoning.

A Web services description includes a bindings section, which associates an abstract IDL-style description with a concrete service location and an interaction protocol. This form of binding and execution support for service implementations – i.e. remote procedure call (RPC) support – is a mature mechanism. Similar binding support for security mechanisms to implement the security requirements is less mature. Since service port types are connection points between the network and the service, *firewall capabilities* are required here. For permission rules, a security manager or security proxy needs to implement access control aspects. For requirements rules, a Web server needs to secure the communication (e.g. SSL-based). This aspect requires further work and shall be addressed in the future.

References

[1] DAML Initiative. *DAML+OIL Ontology Markup*. http://www.daml.org. 2001.

[2] D. Gollmann. *Computer Security*. John Wiley and Sons. 1999.

[3] Chen Li. *A Framework for Web Services Security*. M.Sc. Dissertation. Dublin City University. 2003. (forthcoming).

[4] C. Pahl. An Ontology for Software Component Matching. *Proc. FASE'03 Fundamental Approaches to Software Engineering*. Springer-Verlag, LNCS-Series. 2003.

[5] C.P. Pfleeger. *Security in Computing* (2nd Ed). Prentice Hall. 1997.

[6] World Wide Web Consortium. *Web Services Framework.* http://www.w3.org/2002/ws. 2002.

[7] World Wide Web Consortium. *Semantic Web Activity*. http://www.w3.org/sw. 2002.