MASTER’S THESIS

**Trust-based Security in Web Services**

**Information Systems Institute Distributed Systems Group Technical University of Vienna supervised by Ao.Univ.Prof. Mag. Dr. Schahram Dustdar and Univ.Ass. Dipl.-Ing. Dr.techn. Clemens Kerer by Christian Platzer Samerweg 35 6060 Hall in Tirol**

Vienna, May 2nd 2004

**1 Introduction**

Today more and more companies employ administrators to maintain theirnetwork, adjust user privileges and keep everything up and running. With the growing size of these companies, the amount of work put into these activities grows enormously until a point is reached when a single person can no longer overlook this meshwork of permissions, exceptions and policies. Apart from the resulting overhead to add, delete or adjust permissions, this may also results in some severe security leaks. Wouldn’t it be great if a computer system could decide by itself whether to grant access rights to a requesting user or not?

Another trend arising in the sector of software development is to reuse functionality in the form of Web services. Web services are a platformindependent way to establish communication between two applications connected through a network. Here the maintenance of policies is even more expensive. A distributed application can use many diﬀerent functions and every single one needs to be secured in a proper way. It would be a hard task for a human to adjust all security levels properly because new functions need to be added while others are obsolete and not used any more. SimOﬃce is a security-aware network environment that attacks these problems with a combination of common security standards and a completely new method to judge a user’s intentions. The goal is to create an independent system where decisions are made based on trustworthiness. This way the ’thinking’ is done by the system while a human supervisor can still inﬂuence the judgement in some special cases. The assessment itself tries to imitate human behavior and human trust. Of course this concept is not limited to Web services but can be applied to almost every security-sensitive area in computer networks.

SimOﬃce will simulate a possible environment within an oﬃce where even the coﬀee maker is accessible through a Web service. And what is a harder punishment for bad behavior than a denial of coﬀee!

**1.1 Motivation**

Today, the majority of software companies are implementing tools based around the new standards for Web services. Considering the fast development and the strong commitment of several important software companies like IBM and Microsoft, a wide range of ready-to-use services throughout the entire Web can be expected soon. Google for example, one of the most popular search engines on the web, already oﬀers a Web service for web queries. An implementation based on the provided API is straight forward and requires but basic programming skills.

But Web services available on the Web will not remain the only application for these standards. Modularity and language-independency paved the way for combined applications, using basic Web services even in closed environments like corporate networks. Thus it is quite possible to implement a company’s applications as Web services and make them accessible through published descriptions.The advantages of such an architecture are obvious: Services are completely independent of implementation or operating system and useable from everywhere within the entire network. Developers are encouraged to use the provided functionality without further knowledge of the involved code. Each Web service is registered at a centralized spot which makes service discovery much easier.

Another trend in the development of computer systems is to adapt the human way of thinking, to address security aspects like access rights or judgment of the other parties intent. Combining both, Web services and an intuitive way of access management is the main motivation for this thesis.

**1.2 Problem Deﬁnition**

The main problem is to create a reasonable combination of accessibility and access restrictions.

A federation of Web services must meet some requirements to retain a useful nature: First, access rights to a single Web service must not be ﬁxed if the total number of available services grows too large to maintain it manually. A dynamic technique to adjust access levels automatically is needed. Nevertheless there must be a facility to change access rights manually too. Otherwise an administrator would be unable to customize access restrictions if it is necessary. On the other hand the whole system has to be accessible to every authorized user. What’s a perfect safe system worth if nobody can use it? To meet the requirements above a mechanism is utilized that most people use everyday: Trust. This approach intends to mimic the decisions taken by humans when it comes to judging wether an action of an opposing party is beneﬁcial or not. The goal is to create a federation of Web services where security is assured both by common safety techniques for transport and privacy and a trust-based approach for access control. Creating such a system is not an easy task because systems regulated by trust-based mechanisms tend to be unstable in long-term view. This would result in a complete distrust or the counterpart, a blind trust situation. Neither of this two conditions is suitable for a computer system whose main task is to provide a public service. This thesis will treat the problem of establishing trust relationships and evaluate the capabilities of trust-based access control.

**2 The Notion of the Web Service**

In the deﬁnition of a Web service is given as: ”any process that can be integrated into external systems through valid XML documents over Internet protocols”. This deﬁnition outlines the general idea Web services are built for. Unlike services in general, Web services are based on speciﬁcations for data transfer, method invocation and publishing. This is often misunderstood and when a Web service is mentioned it sometimes refers to a general service provided on the Web, like the weather forecast on a Web page for example. The weather forecast is a service and provides its functionality for a variety of users but unless it comprises an interface to communicate with other applications via SOAP (see Section 2.4) it is no Web service by deﬁnition. Web services can be seen as software components with an interface to communicate with other software components. They have a certain functionality that is available through a special kind of Remote Procedure Call. In fact they even evolved from traditional Remote Procedure Calls. The diﬀerence lies in the interface and the method for transportation. Furthermore Web services can not be viewed or used with an ordinary browser. They require a uniﬁed form of messaging embedded in a XML document.

**3 HTTP**

Found everywhere on the Internet, HTTP (HyperText Transfer Protocol) is a ubiquitous protocol for data connections between Web browsers and servers. This protocol is the current standard for transferring HTML documents, although it is designed to be extensible to almost any document format like XML for example. HTTP Version 1.1 is documented in RFC 2068 [34]. It operates over TCP connections, usually to port 80, though any other port can be used. After a successful connection, the client transmits a request message to the server, which sends a reply message back. The simplest HTTP message is ”GET url”, to which the server replies by sending the named document. If the document doesn’t exist, the server may send an HTML-encoded message stating this. This form of communication represents a typical request/response mechanism. A client sends a request for a speciﬁc document to the server and waits for a response. If the server does not respond with the requested document it is up to the client to wait for the timeout and request the same document again. This loosely coupled type of communication is very common in client-server architectures.

In addition to GET requests, clients can also send HEAD and POST requests, of which POSTs are the most important. POSTs are used for HTML forms and other operations that require the client to transmit a block of data to the server. After sending the header and the blank line, the client transmits the data. This way Web services utilize the HTTP protocol to transmit both Data payload and service request to a Web service. Now it is time to explain how the transmitted data looks like.

**4 XML**

XML is an abbreviation for Extensible Markup Language [44]. It is designed to describe data and improve the functionality of the Web by providing more ﬂexible and adaptable ways of information representation. It is called extensible because its format is not ﬁxed like HTML. Instead, XML is a metalanguage which lets you design your own customized markup languages. A markup is a mechanism to specify structures within a document, whereas the way to add markup to a document is deﬁned by the XML speciﬁcation. But unlike HTML, XML does not specify semantics or a set of tags. There is no prescribed method for rendering XML documents, so semantics will be deﬁned by the application using it or by style sheets.

It is important to specify the character set to avoid misinterpretation of the provided data. The next line describes the root element of the document. Elements are one way to store data in an XML document.

More and more applications make use of XML to store information because of its beneﬁts. Some of them are:

• The structure is well-deﬁned and can be passed between diﬀerent computer systems which would otherwise be unable to communicate.

• Data payload is encapsuled in tags and therefore readable by human viewers.

• Due to their textual nature, XML-Files are platform-independent. These advantages made XML the perfect format to communicate between Web services. To ensure a platform and language independent use for every Web service, SOAP was developed. It is an XML application with deﬁned elements and a predeﬁned structure. The following section will treat SOAP in detail.

**5 Publishing and Finding Web Services**

With SOAP, a communication between Web services is possible and structured and each participant knows how to send or receive the corresponding SOAP Message. The ﬁnal step to complete the communication architecture of Web services is to deﬁne how to access a service once it is implemented. This is where the Web Service Description Language (WSDL) steps in. WSDL describes services as collections of network endpoints, or ports. Again it is an XML document with a deﬁned grammar where the abstract deﬁnition of endpoints and messages is separated from their concrete network deployment or data format bindings. WSDL documents use the following elements to describe a Web service:

• Types: A container for data type deﬁnitions. In contrast to SOAP, WSDL can deﬁne types using some type system (such as XSD).

• Message: A deﬁnition of the data being passed in a single RPC.

• Operation: A description of an action (method) supported by the service.

• Port Type: A set of operations supported by one or more endpoints.

• Binding: A concrete data format speciﬁcation for a particular port type.

• Port: A single endpoint deﬁned as a combination of a binding and the network address where it can be found.

• Service: A collection of related endpoints.

**References**

[1] T. Dimitrakos. A Service-Oriented Trust Management Framework. In Trust, Reputation, and Security: Theories and Practice, pages 53–72. Rino Falcone, Suzanne Barber, Larry Korba and Munindar Singh, 2003.

[2] J. Dunn. Trust and political agency, 2000.

[3] IBM.Web Services Federation Language (WS-Federation), http://www.ibm.com/developerworks/library/ws-fed/, 2003.

[4] T. Beth, M. Borcherding, and B. Klein. Validation of trust in open networks, 2000.

[5]P.F.Pires, M.R.Benevides, andM.Mattoso. BuildingReliableWebServices Compositions. In Web, Web-Services, and Database Systems, pages 59–72. Akmal B. Chaudrin, Mario Jeckle, Erhard Rahm and Rainer Unland, 2002.

[6] L. Eschenauer, V. D. Gligor, and J. Baras. On trust establishment in mobile ad-hoc networks, 2002.

[7] Center for Education and Research in Information Assurance and Security (CERIAS). Formalizing Trust, Fraud, and Vulnerability, http://www.cs.purdue.edu/homes/bb/NSFtrust.html, 2003.

[8] Google Inc. Google Web Apis Developers Kit, http://www.google.com/apis/, 2004.

[9] D. Gambetta. Can We Trust Trust?, chapter 13, pages 213–237. Basil Blackwell, 1988. Reprinted in electronic edition from Department of Sociology, University of Oxford.

[10] IBM. Web Services Trust Language (WS-Trust), http://www.ibm.com/developerworks/library/ws-trust/index.html, 2002.

[11]S. Overhagen and P. Thomas. WS-Speciﬁcation: Specifying Web Services Using UDDI Improvements. In Web, Web-Services, and Database Systems, pages 100–110. Akmal B. Chaudrin, Mario Jeckle, Erhard Rahm and Rainer Unland, 2002

[12] N. Shankar and W. A. Arbaugh. On trust for ubiquitous computing.

[13] IBM. Web Services Security (WS-Security), http://www106.ibm.com/developerworks/webservices/library/ws-secure/, 2003.