# A Gateway to Web Services Security – Securing SOAP with Proxies

Gerald Brose

Xtradyne Technologies
Schönhauser Allee 6-7, 10119 Berlin, Germany
gerald.brose@xtradyne.com

**Abstract.** Integrating applications and resources using Web Services increases the exposure of critical resources. Consequently, the introduction of Web Services requires that additional effort be spent on assessing the corresponding risks and establishing appropriate security mechanisms. This paper explains the main challenges for securing Web Services and summarizes emerging standards. The most important of these, WS-Security, defines a message-based security model for SOAP that is suitable for achieving end-to-end security in environments with multiple trust domains. We propose one particular, gateway-based approach to implementing Web Services security, and compare it to other approaches.

**Keywords:** Web Services, SOAP message security, security architecture, WS-Security, SAML.

## 1 Introduction

While Web Services are expected to significantly lower the cost of integrating applications and resources — e.g., in *Enterprise Application Integration* (EAI) or Business-to-Business (B2B) settings — there are still many different definitions for the term. An abstract view on Web Services as a component technology, e.g., is the *Web Services Invocation Framework* (WSIF) [DMSW01,KKL03]. For the purposes of discussing security, we take a more narrow approach and simply observe that the kind of Web Services that are used today rely on only two main technical components, viz. SOAP [W3C00] over HTTP as a messaging protocol and the *Web Services Description Language* (WSDL) [W3C01] as declarative language for service interfaces. UDDI [OAS02] is often mentioned as the third main cornerstone of Web Services, but it is not as widely used as the other two, and it is unclear if it will see widespread acceptance on a global scale. We don't consider UDDI here.

The expectation that Web Services will increasingly be used to integrate enterprise resources and applications calls for a thorough investigation of its security implications. Security is an issue because integrating more applications and resources implies an increased degree of exposure to accesses that have not been possible before. This increased exposure is also reflected in an increased risk of damage to these resources. Security is hence a fundamental prerequisite for the acceptance of Web Services in production environments – at least if organizational and trust boundaries have to be crossed.

M. Jeckle and L.-J. Zhang (Eds.): ICWS-Europe 2003, LNCS 2853, pp. 101–108, 2003.
© Springer-Verlag Berlin Heidelberg 2003

However, securing Web Services is a challenge because existing security technology does not apply well to the message-based style of Web Service interactions. First, typical perimeter security mechanisms (firewalls) are effectively bypassed with SOAP over HTTP(S) as soon as HTTP messages are allowed to cross the domain boundary at all. It is obvious that additional security mechanisms are required to provide finer-grained control than this all-or-nothing approach. Second, connection-based security mechanisms such as SSL are not sufficient as they provide only point-to-point protections to messages, i.e., between the connection endpoints directly involved in the transmission of a message. However, Web Services are specifically designed to support settings where messages traverse multiple intermediaries and thus multiple transport connections, so SSL cannot be used to achieve end-to-end security. Since the most widely deployed security measures are not sufficient in a Web Services world, new technology is required.

The remainder of this paper is structured as follows: Section 2 summarizes the most important security specifications that apply to SOAP security. Section 3 discusses the alternatives for implementing SOAP message security according to these specifications and proposes a gateway-based implementation approach, and section 4 presents a concrete example for this approach.

## 2 Specifications and Emerging Standards

The security decisions that need to be made in message-based interactions are the following:

- Where does this message originate from? (authentication)
- May this message be sent to this destination? (authorization)
- Has this message (or parts of it) been modified? (integrity)
- Has this message (or parts of it) been read? (confidentiality)

Note that it is important to carefully distinguish between the current channel over which a message arrives and the actual principal where the message originated [WABL94]. The communication partner at the other end of the transport channel may just be forwarding a message that it received previously and may thus not be the actual origin of the message. With message-based systems, this is somewhat more apparent than in invocation-based distributed systems, where the distinction between a caller and a subject on whose behalf a call is made (delegation) is often subtle and difficult to make.

Also note that the additional security requirement of *availability* is not listed separately above but understood to be covered by access control (authorization). This implies a simple definition of availability as the absence of denial-of-service through excessive service usage by an attacker. We cannot positively guarantee availability in a simple store-and-forward message system, however, because the sender of a message has no means to make sure his message ever reaches the target. Additional application-level protocols would be required here but this does not blend well with the overall Web Services goal of achieving loose coupling between interaction partners. Therefore, availability is not considered a security requirement in its own right here.

The SOAP messaging layer does not offer security mechanisms to support any of the security decisions listed above, nor does the underlying HTTP layer.[1] Existing security technology in the underlying transport layer, i.e., SSL/TLS, does provide means for public key authentication and protection of the confidentiality and integrity of data in transit. As pointed out above, SSL/TLS can only provide security between directly communicating parties, i.e., in a point-to-point fashion. In such a more general scenario with intermediaries and without end-to-end protection, both the sender and the receiver would need to trust that no intermediary in the communication path violates their security requirements, e.g., by storing and later replaying messages. This may be an acceptable assumption in internal EAI projects, but it is not acceptable in B2B applications.

Note that the same problems also exists with other transport protocols for SOAP, such as SMTP. Because the transport protocol layer underneath SOAP does not offer it, end-to-end protection of messages must either be implemented within applications, or it can be provided by middleware above the SOAP layer, as indicated in figure 1.

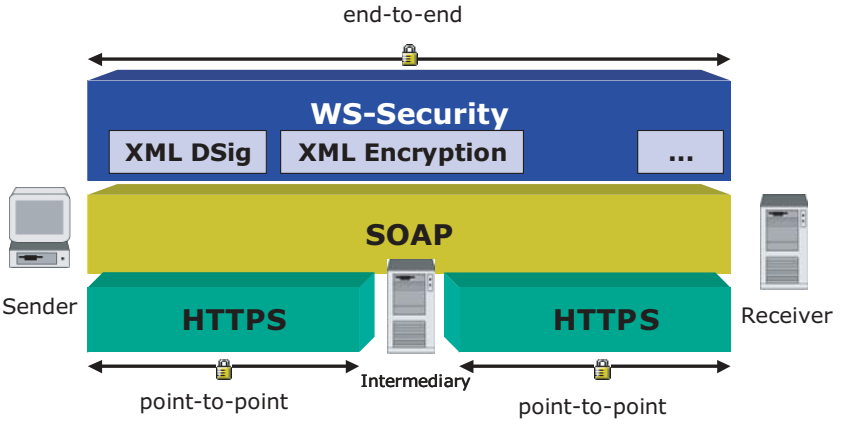



**Fig. 1.** Protocol layers and security.

Because this is a recurring requirement for many applications, the WS-Security [OAS03c] specification proposes a message-based security model for SOAP to address the problems outlined above. Figure 1 illustrates how WS-Security combines with SOAP, HTTP, and SSL, and how it relates to XML Digital Signature and Encryption. The "..." denote extensibility, so other technology or standards can be used with WS-Security. The figure depicts one conceptual SOAP messaging layer on top of one or more HTTP connections between the sender, any intermediaries, and the final receiver of the SOAP message.

The WS-Security specification explicitly addresses end-to-end message security in application environments where messages cross organizational and trust boundaries. Technically, this is achieved through the definition of an XML security header format so that each SOAP message can contain (or reference) the security information about

[1] With the exception of HTTP basic authentication, which supports the transmission of user ID and password.

itself that is required for making security decisions. Security information is expressed as *security tokens*, which assert security claims. The security tokens that may need to accompany a message can include a wide variety of signed or unsigned application-specific, textual and binary data, such as user names, Kerberos tickets, X.509 certificates, encryption and/or signature keys. For example, a message may contain a digital signature over a certain part of the message body and the public key necessary to verify the signature, so that each recipient of the message can determine that the message was not tampered with in transit.

The WS-Security core specification defines only two token formats, a username format and a binary token format, but it is extensible to use additional formats. A prime candidate for security tokens is the the *Security Assertion Markup Language* (SAML) [OAS03a]. SAML expresses any security information in XML syntax as *assertions* by an issuer about a subject. SAML assertions can state that a subject was authenticated (authentication assertion), authorized (authorization assertion), or that a subject possesses a certain attribute (attribute assertion), e.g., a role membership. A standard profile [OAS03b] for SAML assertions as WS-Security tokens is currently being specified at the *Organization for the Advancement of Structured Information Systems* (OASIS). With SAML, it is possible, e.g., to include an authentication statement about the message's origin that was made by one vendor's product, and then use that statement as a basis for making authorization decisions with another vendor's security service.

In addition to the SOAP header element and the security token format, the WS-Security specification defines processing rules for two other security standards, viz. how XML Digital Signature, XML Encryption can be used to protect parts or all of the SOAP message. Finally, WS-Security defines a way of expressing when a recipient should discard stale messages, i.e., the time when a message should no longer be regarded as fresh.

## 3 Architectural Considerations for Implementing Web Services Security

The specifications outlined above do not, of course, imply any specific architectural approaches for possible implementations. A straightforward approach for providing security is to supply application developers with appropriate libraries so that applications can be programmed to perform security controls. As an example, Microsoft provide libraries called *Web Service Enhancements* [Cor] as an extension to its Visual Studio .NET environment. However, this approach places the burden of correctly enforcing security on the application developer rather than on the security infrastructure and security administrators, which requires a significant amount of expertise from developers.

This library approach may be combined with security services provided by the server-side run-time environment, i.e., application server products. The server administrator would configure security policies to be enforced by the application server, which would rely on client applications to provide security information (credentials) in a suitable format for the security infrastructure. In this case, the burden is on the client-side toolkits to appropriately use security libraries, e.g., to create (or obtain form third-party products) and insert SAML assertions that contain authentication statements into outgoing

messages. If an application consists of services deployed on multiple, heterogeneous application servers, these platforms will have to be managed individually, typically using separate management infrastructure and concepts for each vendor's products.

As an elegant and simple-to-deploy approach that does not have these disadvantages, we propose an application-level security gateway ("SOAP security proxy") as an implementation approach. SOAP security proxies can be transparently integrated into existing Web Services infrastructures and do not require changing existing applications. Operating at the application level, they can perform security checks such as XML schema validation and content filtering that are not possible at the transport level. Additionally, SOAP security proxies provide typical application-layer security functions such as role-based access control, and auditing. Based on open standards such as SAML, these components can interoperate with other enterprise security products. Like application-level proxies for other protocols (IIOP, SMTP, HTTP), SOAP security proxies complement rather than replace existing packet filters. Packet filters must be configured to allow traffic to these proxies which they can now trust to be analysed and restricted according to application-level policies.

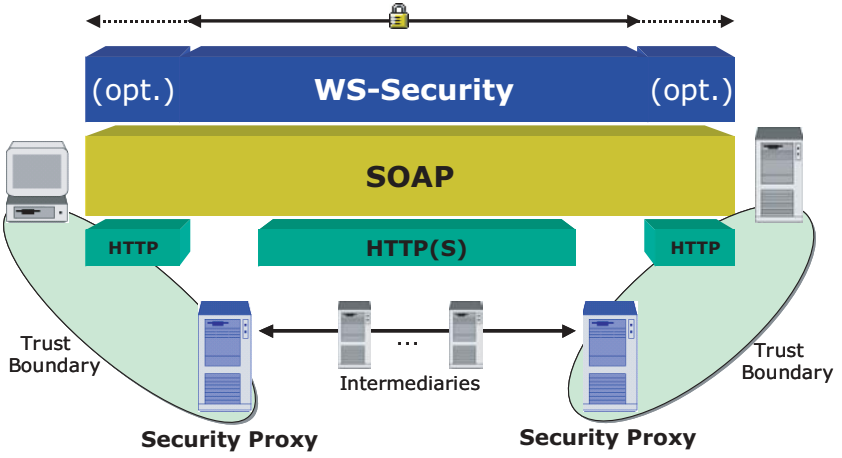



**Fig. 2.** Dual proxy scenario.

The main advantage over library-based solutions as provided by application servers or SOAP toolkits is the reduced administrative overhead. A SOAP security proxy component can be managed as a central component, so the gateway approach thus serves to simplify deployment and integration, and helps unify security management. This advantage becomes more apparent when the number of different server-side platforms is large.

Application clients requiring security management or the distribution of credentials necessitate decentralized management efforts, but this effort can be reduced by the use of a *client-side* SOAP proxy. Such a proxy would be placed within the trust domain of the application clients and can insert security information such as SAML assertions or digital signatures to outgoing messages. This situation is outlined in figure 2.

The advantages of such an approach to the creation of security information and its insertion in messages are most prominent in B2B scenarios where partners want to cooperate without having to trust each other completely. In the example of a manufacturer-supplier scenario, both parties may have large IT infrastructures relying on disparate technologies and standards (programming languages, server architectures, user management, etc.). To align one party's IT with its partner's may become prohibitively expensive if all client-side software has to be modified and new credentials distributed. In this setting, a centralized security gateway that will authenticate clients, map credentials, and add these to outgoing messages can provide significant cost saving benefits because it does not require any software modifications or distribution of credentials.

## 4 Example

As a concrete example for an implementation of the gateway approach outlined above, this section presents Xtradyne's *Web Services Domain Boundary Controller* (Web Services DBC, WS-DBC) [Xtr03]. Figure 3 illustrates the architecture of the implementation.

The Web Services DBC consists of three components, which are typically deployed in separate parts of the network. The proxy component is the actual security enforcement mechanism and would typically be located in the demilitarized zone (DMZ), so that it is directly accessible from an extranet or the Internet. Figure 4 depicts the three main deployment scenarios for a Web Services DBC, viz. intranet, Internet, and federated extranet scenarios.

The proxy performs two main security functions, viz. authentication of messages and authorization (access control). Additionally, the proxy verifies XML digital signatures contained in message headers to validate message integrity, and can perform XML content filtering and XML schema validation to ensure that message content is as expected. This feature can be used to thwart application-level attacks, such as SQL in-

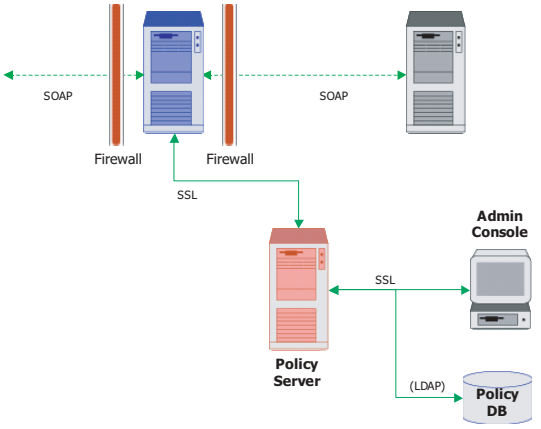



**Fig. 3.** Web Services DBC architecture.

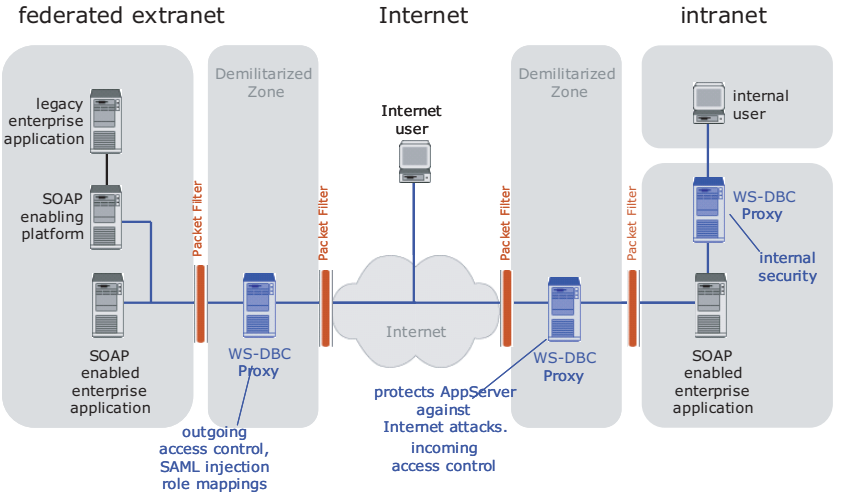



**Fig. 4.** Web Services DBC deployment scenarios.

jection [Anl02]. Finally, the proxy sends out notifications for security-critical events for auditing purposes, and can create SAML assertions that are added to outgoing messages.

To make security decisions, the proxy consults the policy server for policy information. Since the proxy is a performance-critical component, Xtradyne's implementation realizes the proxy as a highly optimized, native C++ implementation for high throughput rather than as a Java servlet that runs embedded in a Web server.

The security policy server is used to configure the proxy component and to hold the security policies. It is therefore a highly sensitive part of the security architecture: if an attacker can compromise the policy server and reconfigure the proxy or manipulate policies, no security can be guaranteed for the resources protected by the proxy. It is hence advisable not to expose the security server in the DMZ but to deploy it in a more restricted network domain.

The security policy server answers policy requests from the proxy and receives event notifications for auditing. User, group, and role information as well as access and audit policies are stored in either flat files on the policy server, or in a remote LDAP repository.

The administration console, finally, is a Java implementation that provides administrators with a powerful and convenient user interface for managing the security infrastructure. It displays configuration and policy information, and shows event notifications. The GUI also supports convenient exposure of secure Web Services through the import of WSDL descriptions. For services imported in this way, the administration console extracts a template security policy from the WSDL description, which then needs to be refined by administrators. The WSDL description is modified during the import to contain the proxy address instead of the original service address, so that client that obtain the WSDL for a secure service from the proxy will automatically receive the "proxified" addressing information.

The administration console relies on a secure CORBA transport (IIOP over SSL) to communicate with the security policy server. The security server uses the same transport to talk to the WS-DBC proxy.

## 5 Summary

This paper summarized the security challenges posed by Web Services and described a number of important emerging standards in field. We discussed the architectural advantages of gateway-based implementations of security services and presented a concrete product example to prove the feasibility of the approach.

## References

[Anl02] Chris Anley. Advanced SQL injection in SQL server applications. Technical report, NGSSoftware Insight Security Research, http://www.ngssoftware.com/papers/advanced_sql_injection.pdf, 2002.

[Cor] Microsoft Corp. Web services enhancements. http://msdn.microsoft.com/webservices/building/wse/default.aspx.

[DMSW01] Matthew J. Duftler, Nirmal K. Mukhi, Aleksander Slominski, and Sanjiva Weerawarana. Web services invocation framework (WSIF). In *OOPSLA 2001 Workshop on Object-Oriented Web Services*, October 2001.

[KKL03] D. König, M. Kloppmann, F. Leymann, G. Pfau, and D. Roller. Web Services Invocation Framework: A step towards virtualizing components. In *Procs. XMIDX 2003*, February 2003.

[OAS02] OASIS. UDDI version 2.04 API specification 1.0. OASIS Committe Spec., http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm, July 2002.

[OAS03a] OASIS. Assertions and protocol for the OASIS Security Assertion Markup Language. Committee Specification, May 2003.

[OAS03b] OASIS. Web services security: SAML token profile. OASIS TC Working Draft 6, February 2003.

[OAS03c] OASIS. Web services security: SOAP message security. OASIS TC Working Draft 12, April 2003.

[W3C00] W3C. Simple object access protocol, version 1.1. W3C Note, http://www.w3.org/TR/SOAP, May 2000.

[W3C01] W3C. Web services description language v1.1. W3C Note, http://www.w3.org/TR/wsdl, March 2001.

[WABL94] Edward Wobber, Martin Abadi, Michael Burrows, and Butler Lampson. Authentication in the Taos operating system. *ACM Transactions on Computer Systems*, 12(1): 3–32, February 1994.

[Xtr03] Xtradyne Technologies. Web Services Domain Boundary Controller. http://www.xtradyne.com/products, 2003.