Using Corporate Firewalls for Web Services Trust

Ingo Melzer and Mario Jeckle

DaimlerChrysler AG
P. O. Box 2360, D-89013 Ulm, Germany
paper@ingo-melzer.de, mario@jeckle.de
http://www.ingo-melzer.de, http://www.jeckle.de

Abstract. Web Services allow collaboration of different systems independent of their operating system or the programming languages being deployed. They enable also the integration of legacy systems and grant access to external partners to those systems. Therefore, it is necessary to implement protection against misuse. Such an addition should be achieved transparently for users and developers. This paper illustrates how to integrate important security features into corporate firewalls or proxies.

1 Introduction

The variety of today's computer systems is a real burden when collaboration is desired or even required. An encouraging approach are Web Services which offer a great possibility for companies and institutions to allow others to access important information to enable a better cooperation.

However, at the same time, the introduction of Web Services for accessing possibly critical business systems may offer other users of the Internet the possibility to gain access to those legacy systems. Most of the time, Web Service deployment is based on the well known HTTP protocol as transport media. Content of this type is normally not inspected or filtered by most firewalls at all. The efficiency of a firewall is therefore significantly reduced. Also, SOAP, the lightweight XML-based protocol of Web Services, does not come with any security features. Though, a firewall or proxy is a commonly used security facility.

However, this does not constitute SOAP based Web Services as a general security hole. SOAP is not secure or insecure — security is simply not its job! According to SOAP's underlying philosophy the application has to take care of this topic. Taken XML's co-standard for digitally signing into account, arbitrary SOAP calls could be secured with respect to authentication, non-reputation, and integrity of the transmitted data. Based on this, the receiver is able to grant authorization to system's access. Since the creation of a digitally signed message requires modifications to the message itself by adding security information, the application creating the SOAP payload is required to be modified as well.

M. Jeckle and L.-J. Zhang (Eds.): ICWS-Europe 2003, LNCS 2853, pp. 79–87, 2003.
© Springer-Verlag Berlin Heidelberg 2003

An alternative to changing numerous business systems to introduce SOAP security is presented here. For secured and closed environments like today's intranets behind corporate firewalls, it could be an option to concentrate the handling of digital signatures at a single point within the network's structure. The machine devoted to this task could sign all outbound Web Service calls and vice versa check the signatures of all signed inbound calls as part of corporate's security infrastructure. Therefore, the inclusion of such a facility will disburden applications and even leverage the usage of digital signatures. As an option, such a facility could reside on a firewall or proxy machine.

Such an add-on to the proxies can be a service which signs all outgoing SOAP calls with a signature owned by the company. On the other side, the firewall of the other partner will block all SOAP calls which do not have an appropriate signature. The key issue of authorization based on authentication can therefore be eased significantly whenever a call passes an enterprise's boundary.

This paper first illustrates a typical infrastructure for most Web Services. It also provides the necessary basics of web services security and digital XML signatures. Based on this, the most important steps for implementing such a proxy for web services, which will be called *Signing Proxy*, to achieve a federated or corporate trust are shown.

2 Web Services Infrastructure

The technical basis of the Web Service philosophy is grounded on the idea of enabling various systems to exchange structured information in a decentralized, distributed environment. In essence this lead to the definition of lightweight protocols for synchronous remote procedure calls as well as asynchronous document exchange using XML encoding via well-known Internet protocols such as HTTP.

After some introductory approaches which were popularized under the name XML-RPC [4] the seminal SOAP[1] protocol which has recently been standardized by the World Wide Web Consortium [5,6] establishes a transport-protocol agnostic framework for Web Services that could be extended by the user on the basis of XML techniques.

The SOAP protocol consists of two integral parts: A messaging framework defining how to encode and send messages. And an extensibility model for extending this framework by its user.

First a brief introduction of the messaging framework is given before showing value of the extensibility mechanisms to accomplish the goals defined above.

Technically speaking SOAP resides in the protocol stack above a physical wire protocol such as HTTP, FTP, or TCP. Although, the specification does not limit SOAP to HTTP-based transfers this protocol binding is currently the most prominent one and is widely used for Web Service access. But it should be noted that the approach introduced by this paper is designed to operate completely independent of the chosen transport protocol and resides solely on the SOAP layer.

[1] At the time of it's definition the acronym stood for *Simple Object Access Protocol*. In the standardized version SOAP is no longer an acronym.

All application data intended to be sent over a network using the SOAP protocol must be transferred into an XML representation. To accomplish this SOAP defined two message encoding styles. Therefore the specification introduces rules for encoding arbitrary graphs into XML. Most prominent specialization of this approach is the *RPC style* introduced by the specification itself which allows the exchange of messages that map conveniently to definitions and invocations of method and procedure calls in commonly used programming languages. As introduced before SOAP is by nature protocol agnostic and can be deployed for message exchange using a variety of underlying protocols. Therefore a formal set of rules for carrying a SOAP message within or on top of another protocol needs to be defined for every respective transport protocol. This is done by the official SOAP specification for HTTP as well as SMTP.

Inside the SOAP protocol the classical pattern of a message body carrying the payload and an encapsulating envelope containing some descriptive data and metainformation is retained. Additionally SOAP allows the extension of the header content by the use of XML elements not defined by the SOAP specification itself. For distinguishing these elements from those predefined by the specification the user has to take care that they are located in a different XML namespace.

The example below shows a SOAP message (the body's content is omitted for brevity) together with the transport protocol specific data necessary when using the HTTP binding. Additionally a user defined header residing in a non-W3C and thus non normative namespace is shown as part of the SOAP `Header` element.

```
POST /axis/theService/ HTTP/1.1 Content-Type: text/xml;
charset=utf-8 Accept: application/soap+xml
Host: 10.0.0.1:8080
Content-Length: nnn
<?xml version="1.0" ?>
<env:Envelope
 xmlns:env="http://www.w3.org/2002/06/soap-envelope">
 <env:Header
    env:mustUnderstand="true"
    env:role="http://10.0.0.99/FWProxy">
  <sp:SigningRequest
      xmlns:sp="urn:xmlns:daimlerchrysler.com:research:sp">
    <sp:Path>env:Body</sp:Path>
    <sp:Username>JohnDoe</sp:Username>
    <sp:Token>SGVsbG8gd29ybGQh</sp:Token>
  </sp:SigningRequest>
 </env:Header>
 <env:Body>
    //some XML encoded payload here
 </env:Body>
</env:Envelope>
```

In contrast to the payload which is intended to be sent to the receiver of the SOAP message clearly identified by HTTP's `Host` header, SOAP headers may or may not be created for processing by the ultimate receiver. Specifically they are only processed by machines identified by the predefined `role` attribute. By doing so the extension framework offers the possibility of partly processing a message along its path from the sender to the ultimate receiver. These intermediate processing steps could fulfill arbitrary task ranging from problem oriented ones like reformatting, preprocessing, or even fulfilling parts of the requests to more infrastructural services such as filtering, caching, or transaction handling. In all cases the presence of a node capable of (specification compliant) processing of a SOAP message is prescribed. This is especially true since an intermediary addressed by the `role` attribute is required to remove the processed header after executing the requested task. Additionally the specification distinguishes between headers optionally to be processed (e.g. caching) and those which are interspersed to trigger necessary message behavior. The latter ones must additionally be equipped with the attribute `mustUnderstand`. If a header addressed to an intermediary flagged by this attribute cannot be processed, the SOAP node is forced to raise an exception and resubmit the message to the sender. Thus it is ensured that all headers mandatory to be processed are consumed by the respective addressees and removed afterwards.

3 Security for Web Services

Three of the most important issues when dealing with security are message integrity, authorization, and authentication. The Signing Proxy presented in Section 4 on page 84 addresses these requirements. Another important security issue is privacy. It is ignored in this paper, because methods to obtain privacy cannot be applied transparently for the users and do not keep valid XML/SOAP documents, which causes some more problems to be solved. Instead of integrating privacy functionality into the Signing Proxy, the use of appropriate transport layer methods is assumed in this paper.

SOAP does not address security issues; it has not been designed to do this. The job has been left to other (mostly XML based) standards. This section briefly introduces the most important standards necessary for building the Signing Proxy which provide more secure SOAP calls.

3.1 XML Signature

RFC3275 defines in [1] how a digital document can be signed and how the result can be represented in XML. Whereas this specification is not limited to XML documents, it is still especially useful for XML documents like SOAP messages, because the result is also an XML document. Using XPath, it is possible to sign only certain parts of an XML file.

A typical Web Service example using XML Signature is shown is Figure 1 on page 83. For simplicity reasons, the firewalls are simplified on both sides.

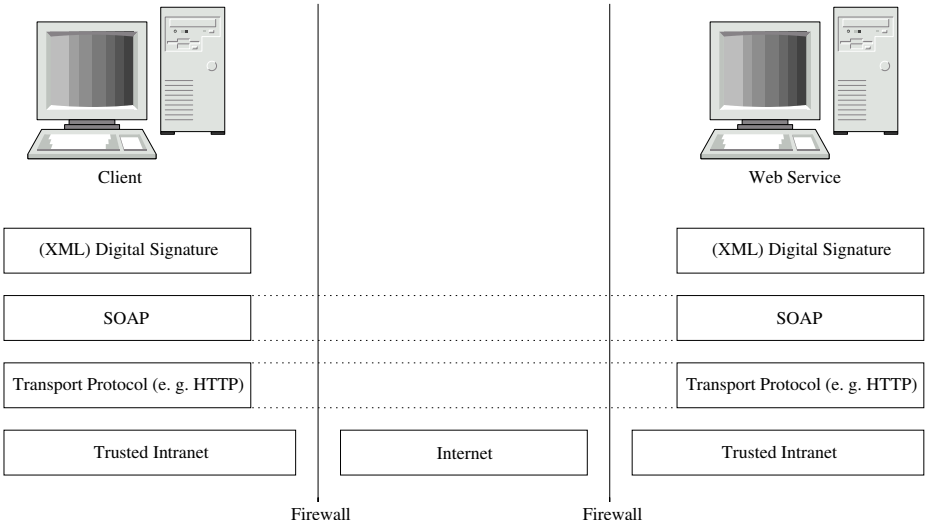


Fig. 1. Web Service Infrastructure

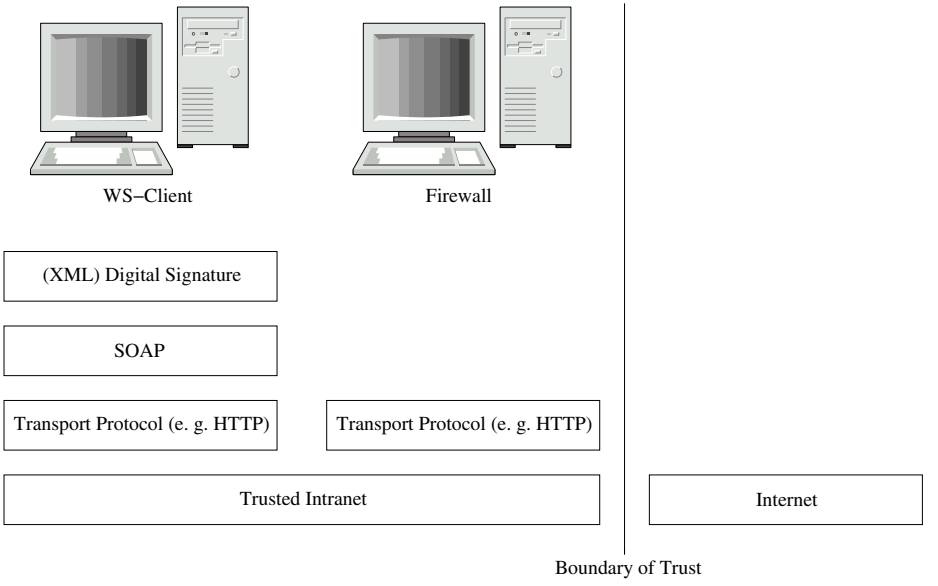


Fig. 2. Web Service Infrastructure including Firewall

Firewalls which parse HTTP are more and more frequently used. This changes both, the Web Service and client side. The according change is shown for one side in Figure 2 on page 83.

3.2 Security Assertion Markup Language

The security assertion markup language, short SAML, is a specification for exchanging security credentials. A specification can be found at
`http://www.oasis-open.org/committees/security/`.
SAML defines an XML vocabulary which allows to specify and share assertions about security relevant topics like authentication and authorization.

3.3 Web Services Security (WS-Security)

WS-Security is an extension of SOAP to add security relevant information into the header of each message. It is a combination of XML Signature and SAML. A specification can be found at
`http://www.ibm.com/developerworks/library/ws-secure/`

The following listing shows a possible extension of a SOAP header which contains a username and a password as defined in the WS-Security standard:

```
<soap:Header>
   <wsse:Security>
      <wsse:UsernameToken>
         <wsse:Username>Ingo</wsse:Username>
         <wsse:Password>oiraM</wsse:Password>
      </wsse:UsernameToken>
   </wsse:Security>
</soap:Header>
```

4 Security Proxy for Web Services

Due to the fact that SOAP does not care about security, it is necessary to develop solutions which add required security features. An ideal solution hides the additional effort such that users and even developers of Web Services do not even notice the additions. This also means that an integration into existing systems in possible without changing a single Web Service. The solution described in this section reaches these goals by means of a SOAP security proxy and a SOAP firewall.

4.1 Intercepting Outgoing SOAP Messages

Proxies are commonly used in an HTTP environment. They reduce network traffic by caching techniques and allow to control Web usage by blocking specific sites or pages with explicit content. Given richer and thus stricter settings a firewall may intercept also content which is operated on a higher level than plain transport protocols like SOAP messages are.

Such a SOAP proxy acts as an intermediary receiving the SOAP request before it is sent over the Internet to the requesting receiver. The header of the SOAP message can be expanded by security credentials. It is advisorable

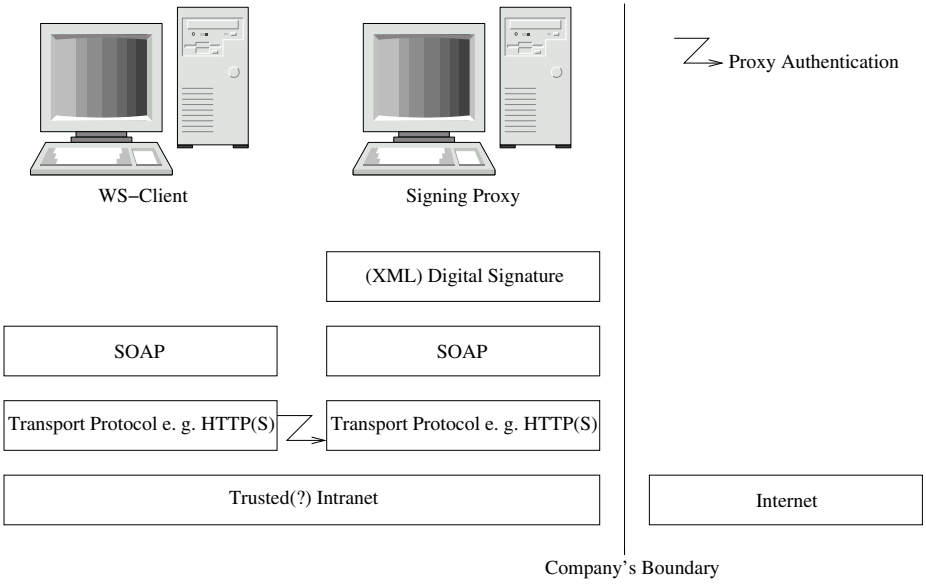


Fig. 3. Signing Proxy inserted into Web Service Infrastructure

to use an existing standard like WS-Security for this step to gain an easier interoperability. The message is thereafter sent to the next destination within the message path.

It is important to keep in mind that the outgoing call is still a valid SOAP message. The receiver can use the additional information, but it is also valid to simply ignore it.

The installation of such a signing proxy into the architecture shown in Figure 2 on page 83 moves the responsibility of creating signatures to the proxy service as shown in Figure 3 on page 85.

4.2 Processing Incoming SOAP Messages

The firewall of the receiver intercepts the incoming SOAP message. In addition to the usual steps which are always executed, it authenticates the message using the information which has been added to the header by the proxy of the sender. Thereafter, a check if the sender has permission to use the requested Web Service is performed. The security credentials could be removed to make the whole process completely transparent for both sides, but this is not necessary since they do not bother the receiver and might still be useful.

4.3 Integration into Existing Infrastructure

One of the main advantages of the presented approach is the small number of changes which have to be made by users and administrators. The users only have to change their proxy settings to contact the signing proxy which contacts

the regular proxy. The administrator should modify the firewall rules to enforce valid signatures in the SOAP header.
But by introducing the singing proxy all applications are enabled to participate instantly in a secured environment not requiring any changes to the application code. On the contrary, security code can be moved from many application layers (i.e. from the application layer of many applications) to one single location. Hence also the amount of auditing and maintenance could be centralized and thus lowered.
As a consequence of this the proposed approach could be transparently integrated into existing processes without requiring changes to the workflow or even the supporting business applications. Hence the Signing Proxy forms a plug-in solution which could easily be deployed at existing sites and be expanded to further partners.

Since the inner nature of the Signing Proxy is hidden by the usage of a standard communication infrastructure (i.e. HTTP and SOAP) interoperability problems are conceptually impossible.

Since currently most practical Web Service applications are concentrated to closed and trusted communities (e.g. B2B communication) the communication path connecting the application with the Signing Proxy is always part of a trusted intranet and can thus be regarded as secured a priori.

Due to the fact that most calls can be regarded to be symmetric, a SOAP message is sent to the Web Service and thereafter a SOAP message is sent back to the caller, those changes have to be made on both sides.

In a bigger environment, it can be useful to install a policy server to get a single point of administration for each intranet.

5 Summary

Securing web services is an important and not impossible task. Due to the fact that this issue has been left out of the SOAP specification, it is important to apply security steps which are appropriate to the given environment.

This paper demonstrates how to develop a proxy service to automatically add security features to web services without saddling the users with this burden.

In addition, the given approach only uses open standards like WS-Security or SAML. This allows an easy integration into existing systems or infrastructures. Also, since these formats are, like SOAP, XML based, it is possible to allow this add-on and still keeping valid XML/SOAP documents.

Advantages.
- Platform-independent (high interoperability)
- transparent integration – even into existing systems
- smooth integration into existing processes
- uses only open standards
- plug-in security
- central administration of security policies
- ideal for B2B (trust)

References

1. D. Eastlake, J. Reagle, D. Solo: XML-Signature Syntax and Processing, RFC3275, March 2002 http://www.ietf.org/rfc/rfc3275.txt
2. Roy Fielding, et al.: Hypertext Transfer Protocol – HTTP/1.1, RFC2616, June 1999 http://www.ietf.org/rfc/rfc2616.txt
3. IBM, Microsoft, Verisign: Web Services Security (WS-Security) Version 1.0, April 2002
   http://www-106.ibm.com/developerworks/webservices/library/ws-secure/
4. Winer, D.: XML-RPC Specification, available electronically (1999)
   http://www.xmlrpc.com/spec
5. Gudgin, M., Hadley, M., Mendelson, N., Moreau, J.-J., Nielsen H. F. (eds.): SOAP Version 1.2 Part 1: Messaging Framework, W3C Proposed Recommendation (2003)
6. Gudgin, M., Hadley, M., Mendelson, N., Moreau, J.-J., Nielsen H. F. (eds.): SOAP Version 1.2 Part 2: Adjuncts, W3C Proposed Recommendation (2003)