A Signing Proxy for Web Services Security

Ingo Melzer
DaimlerChrysler AG
paper@ingo-melzer.de

Mario Jeckle
FH Furtwangen
mario@jeckle.de

Abstract: Web Services offer a way for very different systems to collaborate independent of the the used programming language or the involved operating systems. Their basis is the XML-based SOAP protocol which can be used over any protocol which is able to transport a byte steam. Due to the fact that Web Services do not depend on any operating system and there is no burden of a underlying paradigm, they are ideal for the integration of even completely inhomogeneous systems. However, SOAP does not (and does not have to) deal with security issues, which is nevertheless important for the involved systems. This paper describes an addon for existing Internet proxies to achieve user and developer transparent security features for Web Services. This approach allows corporate firewalls to handle authentication. A first step is to add corporate signatures to all outgoing SOAP messages to enable a corporate trust relationship. A second improvement is to use proxy authentication as defined in RFC 2616 and RFC 2617 to add personal signatures assuming that the proxy has access to some key management system.

1 Introduction

Today's computer systems are extremely inhomogeneous which can be a real burden when collaboration is desired or even required. A possible solution to this problem are Web Services which offer a great possibility for companies and institutions to implement general interfaces to their heterogeneous systems. This allow others, humans or computer systems, 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 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-standards encrypting and digitally signing into account, arbitrary SOAP calls could be secured with respect to privacy, authentication, non-reputation, and integrity of the trans-

mitted data. Based on this, the receiver is able to grant authorization to system's access. Since the creation of a secured message requires modifications to the message itself by adding security information, the application creating the SOAP payload is required to be modified as well.

An alternative to changing numerous business systems to introduce security at this protocol layer is presented here. For reasonable secured and closed environments like today's intranets behind corporate firewalls, it is possible to concentrate the handling of digital signatures at a single point within the network's structure. The machine devoted to this task could sign and encrypt if desired all outbound Web Service calls and vice versa decrypt and check the 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 security mechanisms. 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 or encrypts selective message contents to be readable only for the desired reader. On the other side, the firewall of the other partner will block all SOAP calls which do not have an appropriate signature or cannot be decrypted successfully. 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.

Secondly, this step will be improved to add personal signatures. For this goal, it is necessary that the proxy is able to identify each user which can be achieved by using proxy authentication as described in RFC 2616 and RFC 2617 [FGM$^+$99, FHBL$^+$99]. If a key management system is installed and the proxy has access to this system, it can add a second signature to the SOAP header which belongs to the calling user.

Additionally, using the XML co-standard XML-Encryption, it is possible to add privacy to the Web Services application by encrypting the outgoing messages. However, due to the fact that it is very difficult to do this transparently for all users and arbitrary receivers, encryption plays only a minor role in this paper.

2 Infrastructure

2.1 Web Services

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 dynamically forming a extremely loosely coupled system. In essence this lead to the definition of lightweight platform-independent 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 [Wi99] the SOAP[1] protocol which has recently been standardized by the World Wide Web Consortium [W3C03a, W3C03b] 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. Firstly, 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.

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 defines 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 complete SOAP message accompanied 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" ?>
```

[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.

```
<env:Envelope
 xmlns:env="http://www.w3.org/2002/06/soap-envelope">
 <env:Header>
    <ns1:DeliveryNotification
    env:mustUnderstand="true"
    env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
       xmlns:ns1="urn:xmlns:daimlerchrysler.com:research">
       <ns1:SendTo URI="MailTo:john.doe@daimlerchrysler.com"/>
    </n1:DeliveryNotification>
 </env:Header>
 <env:Body>
    <ns2:QuoteRequest>
       <ns2:ID>7492653</ns2:ID>
       <ns2:Amount>10000</ns2:Amount>
       <ns2:DeliverTo>DaimlerChrysler ...</ns2:DeliverTo>
       <!-- more details omitted for brevity ... -->
    </ns2:QuoteRequest>
 </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.

2.2 HTTP Proxy Authentication

At least if personal information like a personal signature should be inserted by the security proxy for Web Services of 4 on page 9, the proxy has to be able to identify its users. The easiest solution for this requirement is HTTP proxy authentication as described in RFC 2616 and RFC 2717.

Proxy authentication is can be used for authenticating users to proxies, proxies to proxies, or proxies to origin servers by use of the Proxy-Authenticate and Proxy-Authorization headers. If authentication information is missing, the contacted proxy must issue a 407 response code which stands for “Proxy Authentication Required”. The client has to reissue his request with an additional proxy authentication header.

3 Security for Web Services

As it is deployed, nowadays security is rather a vertical issues spanning all protocol layers than an insulated issue. Since SOAP is designed to serve as a true protocol providing services to applications residing on top of it and concurrently using services supplied by lower layer protocols security has to be also addressed at this layer. Due to the protocol independent design of SOAP, this layer cannot rely on security mechanisms such as SSL/TLS, SHTTP or S-MIME probably provided by underlying layers, even this limitation may be irrelevant in some practical applications.

Furthermore, existing security provided by other protocols are often limited to certain message exchange patterns like synchronous communication (e. g. SSL) and are therefore not compatible with SOAP's assumption of autonomy from underlying protocols.

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 9 addresses these requirements. Another important security issue is privacy. It is not fully covered in this paper, because methods to obtain privacy cannot be applied transparently for the users and do not keep a validating XML/SOAP documents, which causes some more problems to be solved. However, if encryption between the outgoing proxy and the firewall of the receiver is desired, it is possible (after an initial corporate key exchange) to build a system like the basic signing proxy for encryption which results in a real security proxy.

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.

However, due to the fact that the additional security starts at the proxy and the intranet is usually not completely trustworthy, additional steps for this part of the route are necessary. The easiest solution is usually to access the proxy using SSL. There are no nodes between the client and the proxy which have to be able to read the traffic, so a point to point encryption is a good solution.

3.1 XML Encryption

[IDS02] defines a process for encrypting data which is represented as an XML document under consideration of XML's structural constraints formulated by [W3C00, W3C01]. These structuring principles are respected in a twofolded manner. First, XML Encryption ensures that the outcome of an encryption process is still a well-formed XML document. Whereas validity with respect to a XML schema or a document type definition has to be sacrificed for security's sake. Basically, it is not possible to retain schema validity for encrypted contents. An attempt to represent encrypted content adhering to the type constraints of the original type would significantly curb the possibilities to lexically represent the encrypted data.

Second, XML Encryption is designed to take XML's nature as data format into account. This is done by allowing the partial encryption of an XML which offers to encryption only parts of an XML tree, i. e. single elements.

Furthermore, the standard itself does not define or even recommend specific methods for encrypting XML encoded data. The application of particular algorithms and types of obfuscation such as asymmetric or symmetric cryptography is completely discounted by the specification itself.

Rather than defining ways of securing XML content, XML encryption establishes an open and extensible framework to apply cryptographic methods to a XML document to assure privacy. By doing so, arbitrary algorithms of unrestrained types can be uses to secure the content.

This is guaranteed by a number of metadata elements which describe the usage of encryption for interoperability's sake without revealing trustworthy data.

The example below shows the XML document presented in section 2.1 with encryption applied solely to the SOAP body. The encrypted content of the respective subelements is stored in the `CipherValue` element.

```
<?xml version="1.0" ?>
<env:Envelope
 xmlns:env="http://www.w3.org/2002/06/soap-envelope"
 xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
 <env:Header>
    <ns1:DeliveryNotification
    env:mustUnderstand="true"
    env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver">
       xmlns:ns1="urn:xmlns:daimlerchrysler.com:research">
       <ns1:SendTo URI="MailTo:john.doe@daimlerchrysler.com"/>
    </n1:DeliveryNotification>
    <wsse:Security>
        <xenc:ReferenceList>
            <xenc:DataReference URI="#bodyID"/>
        </xenc:ReferenceList>
    </wsse:Security>
 </env:Header>
 <env:Body>
    <xenc:EncryptedData Id="bodyID">
        <ds:KeyInfo>
            <ds:KeyName>CN=John Doe, C=DE</ds:KeyName>
        </ds:KeyInfo>
        <xenc:CipherData>
            <xenc:CipherValue>dGhlIHF1aWNrIGJyb3duIGZv...</xenc:CipherValue>
        </xenc:CipherData>
    </xenc:EncryptedData>
 </env:Body>
</env:Envelope>
```

3.2 XML Signature

RFC 3275 defines in [ERS02] 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, because the result is also an XML document. Using XPath, it is possible to locate and identify only certain parts of an XML file to be signed. This is essentially the same mechanism that was discussed for XML encryption to allow partial application of cryptography.

Since the singing process can be traced back to an application of encryption, the framework

is quite similar to the one introduced before. Essentially, signing is done by applying asymmetric encryption of a message digest which results from the application a one way hash function to the parts to be signed.

The example below again shows the document presented in section 2.1 with a digital signature applied to the body (i.e. the element and all its children) of the SOAP message.

```
<?xml version="1.0" ?>
<env:Envelope
 xmlns:env="http://www.w3.org/2002/06/soap-envelope">
 <env:Header>
    <ns1:DeliveryNotification
    env:mustUnderstand="true"
    env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
       xmlns:ns1="urn:xmlns:daimlerchrysler.com:research">
       <ns1:SendTo URI="MailTo:john.doe@daimlerchrysler.com"/>
    </n1:DeliveryNotification>
    <wsse:Security
        xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
        <wsse:UsernameToken Id="MyID">
            <wsse:Username>John</wsse:Username>
        </wsse:UsernameToken>
        <ds:Signature>
            <ds:SignedInfo>
               <ds:CanonicalizationMethod
                  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
               <ds:SignatureMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
               <ds:Reference URI="#MsgBody">
                  <ds:DigestMethod
                      Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                      <ds:DigestValue>LyLsF0Pi4wPU...</ds:DigestValue>
               </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>DJbchm5gK...</ds:SignatureValue>
            <ds:KeyInfo>
                <wsse:SecurityTokenReference>
                    <wsse:Reference URI="#MyID"/>
                </wsse:SecurityTokenReference>
            </ds:KeyInfo>
        </ds:Signature>
    </wsse:Security>
 </env:Header>
 <env:Body Id="MsgBody">
    <ns2:QuoteRequest>
       <ns2:ID>7492653</ns2:ID>
       <ns2:Amount>10000</ns2:Amount>
       <ns2:DeliverTo>DaimlerChrysler ...</ns2:DeliverTo>
       <!-- more details omitted for brevity ... -->
    </ns2:QuoteRequest>
 </env:Body>
</env:Envelope>
```

It should be noted that since encryption and digital signatures serve different purposes w.r.t. security both can be used in conjunction. Particularly, the both standardized frameworks are designed to be interoperable.

A typical architecture for digital signatures to a Web Service is shown is Figure 1 on page 8. For simplicity reasons, the firewalls are simplified on both sides.

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 8.

3.3 Web Services Security (WS-Security)

Since SOAP adds an additional layer of semantics on top of XML signatures and encrypted content cannot be applied in a naive manner. Particularly with regard to encryption this

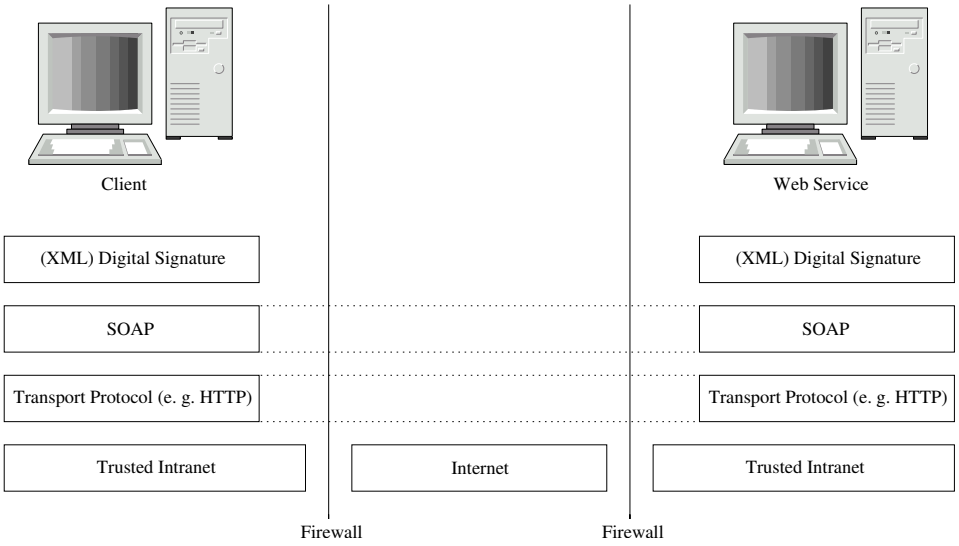

Figure 1: Web Service Infrastructure

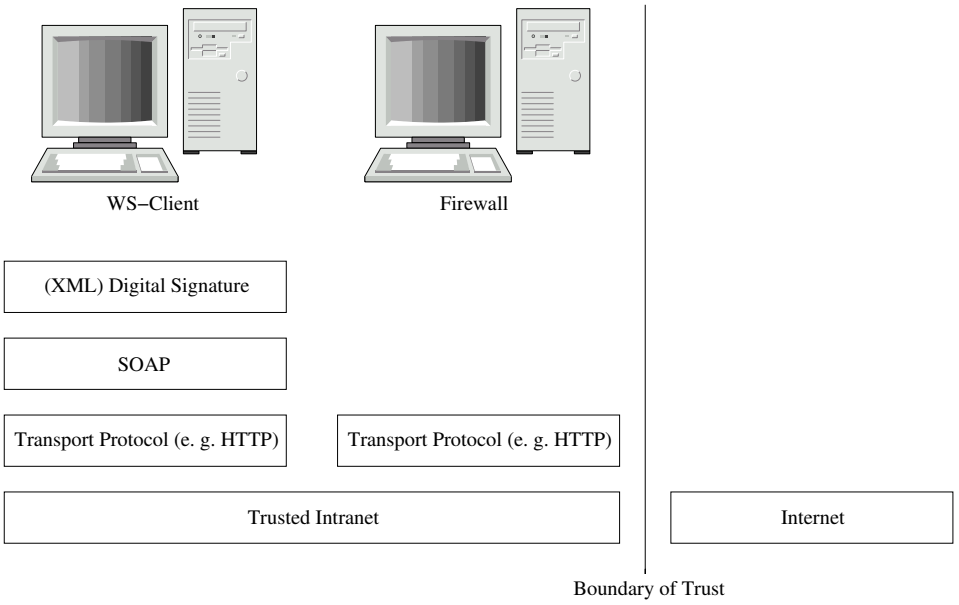

Figure 2: Web Service Infrastructure including Firewall

means only parts parts of the SOAP message which are not required to be readable by intermediaries are allowed to be encrypted. Otherwise either every possible intermediary which could be arbitrarily any machine on the Web must be in the position to decrypt the message in order to process the headers or the headers cannot be processed at al. On the other hand the usage of security mechanisms create some descriptive meta information which has to expressed appropriately.

In order to facilitate interoperability of SOAP and the security mechanisms discussed before the standard WS-Security defines how to apply digital signatures to and encrypt the content of SOAP documents.

WS-Security, [ADLH+02], is designed as an extension of SOAP to add security relevant information into the header of each message. It is a combination of XML Signature and XML Encryption. A specification can be found at
`http://www.ibm.com/developerworks/library/ws-secure/`.

By its very nature WS-Security does not define additional security mechanisms or message semantics. It solely describes syntactic constraints how to place the information related to security inside a SOAP message.

Both examples given in the previous sections are coded according to principles set out by these standard.

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 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.

If the intranet is not completely trustworthy, it is possible to apply a point to point encryption between the Web Services client and the proxy using SSL. This does not cause problems, because there should not be any intermediaries between those two.

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.

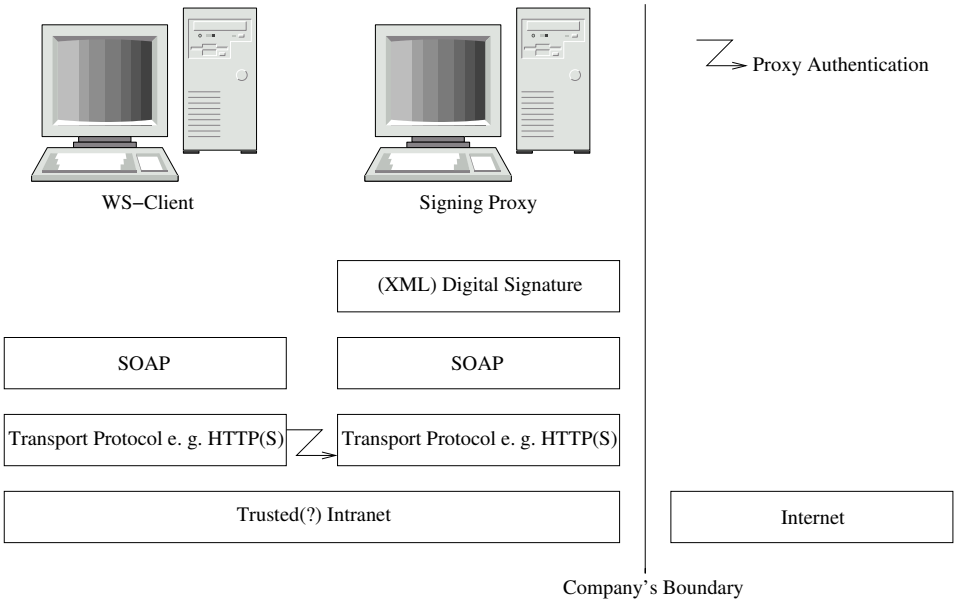

Figure 3: Signing Proxy inserted into Web Service Infrastructure

The installation of such a signing proxy into the architecture shown in Figure 2 on page 8 moves the responsibility of creating signatures to the proxy service as shown in Figure 3 on page 10.

Assuming that the signing proxy has access to some key management system or even a public key infrastructure, shore PKI, this system can be further improved. Using proxy authentication as described in RFC 2616 and RFC 2617 [FGM+99, FHBL+99], the proxy is able to identify the individual users and also add their personal signatures.

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. Proxy authentication has to be enabled if personal signatures are desired.

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. If the intranet cannot be trusted, the situation changes only very little — SSL can be used and the proxy is accessed using HTTPS.

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.

As mentioned earlier, in addition to corporate trust, it is possible to also add personal signatures. There are two prerequisites for this step. Firstly, the proxy has to be able to identity the different users which requires the use of proxy authentication. The other requirement for adding personal signatures is the use of a expanded key management system like a PKI which can be accessed by the proxy and grants access to the users private keys. This is a dangerous requirement because the confidentiality of private keys is extremely important and faults in this area can destroy people's trust in the whole system.

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.

The improvement has been done in two steps. Firstly, the proxy is enhanced to add a corporate signature to all outgoing calls proving that the origin lies within the company. This is a rather simple step, because the say private key can be used for all outgoing messages.

The other improvement has been to also add personal signatures. For this step, it has been necessary to identify all users which has been done using proxy authentication. If the proxy is then able to access a PKI, the personal signatures can also be added to the SOAP header.

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)
- also usable on personal basis (assuming proxy authentication and PKI)

References

[ADLH+02] Atkinson, B., Della-Libera, G., Hada, S., Hondo, M., und Hallam-Baker, P.: *Web Services Security (WS-Security) Version 1.0*. April 2002.

[ERS02] Eastlake, D., Reagle, J., und Solo, D.: *RFC 3275: XML-Signature Syntax and Processing*. March 2002.

[FGM+99] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., und Berners-Lee, T.: *RFC 2616: Hypertext Transfer Protocol — HTTP 1.1*. June 1999.

[FHBL+99] Franks, J., Hallam-Baker, P., Lawrence, S., Leach, P., Luotonen, A., und Stewart, L.: *RFC 2617: HTTP Authentication: Basic and Digest Access Authentication*. June 1999.

[IDS02] Imamura, T., Dillaway, B., und Simon, E.: *XML Encryption Syntax and Processing*. W3C — World Wide Web Consortium. December 2002.

[W3C00] W3C — World Wide Web Consortium: *Extensible Markup Language (XML) 1.0 (Second Edition)*. October 2000.

[W3C01] W3C — World Wide Web Consortium: *XML Information Set*. October 2001.

[W3C03a] W3C — World Wide Web Consortium: *SOAP Version 1.2 Part 1: Messaging Framework*. June 2003.

[W3C03b] W3C — World Wide Web Consortium: *SOAP Version 1.2 Part 2: Adjuncts*. June 2003.

[Wi99] Winer, D.: *XML-RPC Specification*. 1999.