

Architecting Secure Web Services for eBusinesses through Policies

Mifla Mashood

University of Colombo School of
Computing, Sri Lanka
E-mail: mifla17@gmail.com

Gihan Wikramanayake

University of Colombo School of
Computing, Sri Lanka
E-mail: gnw@ucsc.cmb.ac.lk

Abstract

As Web Services begin to dominate the market of distributed computing, securing the pipelines from intruders is becoming a mission, which cannot be considered as trivial anymore. As businesses adopt web services due to its very attractive features like platform independency, ease of implementation, unprecedented support from major vendors and ability of seamlessly interfacing with legacy systems, they unsuspectingly expose themselves into a zone filled with security loop holes which can pose a great threat to confidential data which might be traveling through the channels using these services.

This paper proposes a comprehensive security solution for securing web services through the use of policies. In an era where web services are building bridges across heterogeneous systems, the need of a security solution, which can seamlessly integrate itself into the existing web service infrastructures of businesses, is becoming more and more apparent. Thus a solution, which can be easily incorporated into the existing infrastructures with minimum cost and effort on the part of the developers and businesses are proposed here.

1. Introduction

1.1 Study Background

Web Services are the next hype which will undoubtedly revolutionize the IT world with the phenomenal support it provides for distributed computing through the use of standardized protocols which enable the seamless integration of heterogeneous systems across different platforms.

A strict definition of Web Services is “encapsulated, loosely coupled and contracted software objects offered via standard protocols” [1]. Essentially, Web Services are application functionality residing on systems that accept requests from other systems locally or across the Internet by means of lightweight, vendor-neutral communications technologies.

Since boundaries of interaction between communicating partners are expected to expand from Intranets to the Internet and as communicating partners are more likely to interact with each other without establishing a business or human relationship, in the foreseeable future more and more interactions are expected to occur from programs to programs rather than from

humans to programs. Therefore, the interaction between communicating partners using Web services is anticipated to be more dynamic and instantaneous. This in turn means that all security requirements such as authentication, access control, non-repudiation, data integrity and privacy must be addressed by the underlying security technology.

2. Why Not SSL?

In this era of IT where SSL [2] is considered as a pioneer in securing communication lines against encroachments of many kinds, the most obvious question anyone would pose is, why not use SSL to secure web services? Although it might seem as a valid argument at the onset, SSL used alone, does not provide a comprehensive security solution for web services. It could expose web services to many undesirable risks, which will be elaborated in details below.

2.1 Pros and Cons of SSL in Terms of Web Services

SSL works just fine for transport-level security but breaks down at the message level. SSL is a point-to-point protocol, that is, at each step the message is decrypted and a new SSL connection is set up. If a message goes through multiple intermediates, which is expected behaviour in Web services, then the message will be exposed to each intermediate, which may be a severe vulnerability. If the requirement is to allow only trusted parties to participate and to secure the data from prying eyes, SSL fits the bill. However if the requirement is to use multiple transport protocols, say HTTP and then JMS, and you cannot trust the point that acts as gateway between HTTP and JMS, then you may need to rethink your security strategy [3].

Certain web service security mechanisms have a cutting edge over SSL due to some reasons. SSL encrypts the entire message, which will be costly in terms of a large message, whereas certain web services specifications supports encrypting selected parts of the message, which could be a crucial fact when it means business. In terms of authorization, while SSL does not provide direct support, there are web service specifications that specifically support authorization, such as the eXtensible Access Control Markup Language (XACML).

Web services security is XML based which is the language of Web services, making it a more congruent fit. It is said that there are times when SSL is an alternative to Web services security and is recommended to be used in conjunction with Web services security in certain cases. However because the message-level security technologies such as XML Digital Signature [4], XML Encryption [5], WS-Security [6] and many others tend to be complex to deploy and hence are yet to get wide adoption, there is a tendency to use SSL as a starting point.

In addition, SSL does not address the need for accountability, i.e. it is difficult to even prove that an SSL session existed once it has been closed. If the application requires the data to be stored on disk before processing, then SSL can't protect against attacks on that data. This could be a significant consideration for certain Web services-based applications [7].

3. Security for Web Services

As it was elaborated in the previous paragraphs SSL alone wouldn't be adequate to ensure security in terms of web services. Thus from this point onwards we will be discussing about a comprehensive security mechanism, which will unleash the powers of Message level security mechanisms while

leveraging it by the use of policies to enable businesses to seamlessly secure their web services.

With the rapid development of various technologies by a variety of vendors, the heterogeneous nature of many systems today is inevitable. 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. Web service deployment is mostly based on the well-known HTTP protocol. 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, although a firewall or proxy is a commonly used security facility. Taking XML's co-standards encrypting and digitally signing into account, arbitrary SOAP calls could be secured with respect to privacy, authentication, non-repudiation, and integrity of the transmitted data. Based on this, the receiver is able to grant authorization to the 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 application code in an extensive manner to incorporate these security mechanisms is outlined here. In this solution which is clearly depicted in Fig. 1, each application and web service will be bound to a XML based policy file, which in turn will handle security for XML based web services in businesses. The policies 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 the corporate security

infrastructure. Therefore, the inclusion of such a facility will disburden applications and even leverage the usage of security mechanisms.

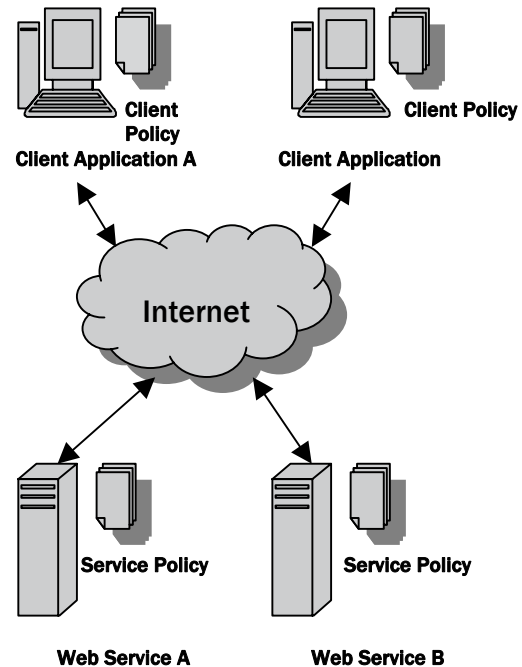


Fig. 1 Proposed solution with Policy Files

The policy files at either end would facilitate to seamlessly secure web services with minimum code change, but the security for the policy files which would be stored at each end as part of this solution still remains unsolved. Thus as part of our solution we would be introducing a "Policy Cipher Tool" to secure the policies at either end.

The remainder of this paper first illustrates a typical infrastructure for most Web services. It also provides the necessary basics of Web services security, which includes many of the XML based security mechanisms, which has been widely adopted so far to secure XML based communication. As the next step the proposed mechanism is elaborated.

3.1 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 dynamically forming an extremely loosely coupled system. In essence this led 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.

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. 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.

Policy files which will form the basis of the proposed solution will extend the SOAP message by the inclusion of one or more of the following XML based security mechanisms to accomplish authentication, data integrity (tamper-proofing), authorization and non-repudiation.

3.2 XML Digital Signatures

XML digital signature, like any other digital signing technology, provides authentication, data integrity (tamper-proofing), and non-repudiation. The project aims to develop XML syntax for representing digital signatures over any data type. The XML digital signature specification also defines procedures for computing and verifying such signatures.

You can sign not only XML data, but also non-XML data. A signature can be either enveloped or enveloping, which means the signature can be either embedded in a document being signed or reside outside the document. XML digital signature also allows multiple signing levels for the same content, thus allowing flexible signing semantics. For example, the same content can be semantically signed, co-signed, witnessed and notarized by different people [8].

3.3 XML Encryption

The W3C is also coordinating XML Encryption [8]. Its goal is to develop XML syntax for representing encrypted data and to establish procedures for encrypting and decrypting such data. Unlike SSL, with XML Encryption, you can encrypt only the data that needs to be encrypted, for example, only the credit card information in a purchase order.

These mechanisms can be built into XML policy files on either end to provide security. The following sections elaborate on how to incorporate these changes by using the WS-Policy [9] specification.

4. Policy Management

Since SOAP messages aren't secure by itself, the need of incorporating different methods to make them secure is becoming more and more evident. A method that would involve least amount of administration or source code modification would be ideal. The Policy files, which are proposed here, are intended to do the same with minimum effort and cost on part of the business, making it a scalable solution for the future.

4.1 Alteration of Outgoing SOAP Messages

Policy files indicated here acts as an intermediary between an application and web service by altering the outgoing

messages/SOAP request before it is sent over the Internet to the requested receiver. The header of the SOAP message can be expanded by the use of security credentials. It is advisable to use an existing standard like WS-Policy for this step to gain an easier interoperability. It is important to keep in mind that although altered by the policy 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 based on the settings specified by the policy. The installation of such policy files into the architecture elaborated in Fig. 1 (above) delegates the task of creating signatures, encrypting etc. to the policy files at either ends [10].

4.2 Processing Incoming SOAP Messages

The policy file of the receiver processes the incoming SOAP message. In addition to the usual steps that are always executed, it authenticates the message using the information that has been added to the header by the policy file of the sender. Thereafter, a check if the sender has permission to use the requested Web service is performed.

4.3 Integration into Existing Infrastructure

One of the main advantages of the proposed solution is the ability of adopting it by using minor code changes on the part of the application and web service. It's recommended to use a standard like WS-Policy proposed by Microsoft in creating these policy files in order to ensure interoperability under all circumstances.

4.4 WS-Policy

In order to successfully integrate with a nontrivial Web service, one must fully understand the service's XML contract along with any additional requirements,

capabilities, and preferences (also referred to as policies). The client needs to know exactly what sort of modifications are to be done on their part to communicate with the service. It also needs to know what security tokens it's capable of processing (such as UsernameToken, Kerberos tickets, or certificates), and which one it prefers. The client must also determine if the service requires signed messages. And if so, it must determine what token type must be used for the digital signatures. Finally, the client must determine when to encrypt the messages, which algorithm to use, and how to exchange a shared key with the service. Without a standard way to convey policies, developers are left to convey them as they always have, i.e. through word-of-mouth and documentation.

Web services infrastructure could be enhanced to understand certain policies and enforce them at runtime. For example, a developer could write a policy stating that a given service requires Kerberos [11] tokens, digital signatures and encryption. Other developers could use the policy information to decide whether they can use the service. Plus, the infrastructure could enforce these requirements without requiring the developer to write a single line of code.

So not only would a policy framework provide an additional description layer, it would also offer developers a more declarative programming model.

WS-Policy provides a flexible and extensible grammar for expressing policies in a machine-readable XML format. The XML representation of a policy is referred to as a policy expression. A policy expression is bound to a policy subject, or in other words, the resource it describes (e.g., a Web service endpoint). The mechanism for associating a policy expression with one or more policy subjects is referred to as a policy attachment.

Policy Expressions

A policy expression is the XML representation of a policy. Using XML to represent policies facilitates interoperability between heterogeneous platforms and Web services infrastructure. WS-Policy provides a normative XML Schema definition that expresses the structure of a policy expression. The WS-Policy schema defines all of the constructs that may be used in a policy expression and it also includes definitions for WS-PolicyAssertions and WS-PolicyAttachment.

```
<wsp:Policy xmlns:wsp="..." xmlns:wsu="..." wsu:Id="..." Name="..." TargetNamespace="...">  
  <!-- policy assertions go here -->  
</wsp:Policy>
```

Policy Assertions

A policy assertion represents an individual preference, requirement, capability or other characteristic, and is the basic building block of a policy expression. A policy assertion is represented by an XML

```
<wsp:Policy xmlns:wsp="..." xmlns:wsu="..." wsu:Id="..." Name="..." TargetNamespace="...">  
  <Assertion wsp:Usage="..." wsp:Preference="..." />  
</wsp:Policy>
```

Some assertions specify requirements and capabilities that will ultimately manifest on the wire, such as a requirement for digital signatures or encryption. These assertions require cooperation between the two parties. Other assertions have no wire manifestation but provide additional information to assist in service selection (e.g., support policies, quality of service policies, etc.). WS-Policy provides a usage qualifier to help distinguish between different types of assertions and

There is also a schema for WS-Security Policy that defines the structure of the security assertions, and WSE 3.0 defines some additional elements that are only used by its infrastructure. The `wsp:Policy` element acts as a container for policy assertions. You can assign a policy expression an ID, a name, or both. The `wsu:Id` attribute assigns the policy expression an ID value in the form of a URI. The `wsp:Policy` element represents a policy expression. According to the schema, `wsp:Policy` has the following basic structure:

element with a well-known name and meaning, typically defined by another specification (like WS-Policy Assertions and WS-Security Policy).

A policy expression simply contains a set of policy assertion elements as illustrated here:

how they're to be processed. You use the `wsp:Usage` attribute to indicate an assertion's usage. The possible values for `wsp:Usage` include Required, Optional, Rejected, Observed, and Ignored (see Table 1). Each policy assertion element must have a `wsp:Usage` value—it can either be declared on the assertion element itself or it can be inherited from an ancestor element. Fig. 2 illustrates its usage.

Value	Meaning
wsp:Required	Applied to the subject and if it does not meet the criteria expressed in the assertion an error will occur.
wsp:Rejected	The assertion is explicitly not supported and, if present, will cause failure.
wsp:Optional	The assertion may be made of the subject, but is not required.
wsp:Observed	The requesters of the service are informed that the policy will be applied.
wsp:Ignored	The assertion is processed but ignored. No action will be taken as a result of it being specified.

TABLE 1 WSP:USAGE VALUES

```

<wsp:Policy wsu:Id="Sign-X.509-Encrypt-Username-4">
  <wssp:Integrity wsp:Usage="wsp:Required">
    <wssp:TokenInfo>
      <wssp:SecurityToken>
        <wssp:TokenType>http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3
        </wssp:TokenType>
        <wssp:TokenIssuer>CN=Root Agency</wssp:TokenIssuer>
        <wssp:Claims>
          <wssp:SubjectName MatchType="wssp:Exact">CN=WSE2QuickStartServer</wssp:SubjectName>
          <wssp:X509Extension OID="2.5.29.14" MatchType="wssp:Exact">bBwPfiTvKp3b6TNDq+14qe58VJQ=
          </wssp:X509Extension>
        </wssp:Claims>
      </wssp:SecurityToken>
    </wssp:TokenInfo>
  </wssp:Integrity>
  <wssp:Confidentiality wsp:Usage="wsp:Required">
    <wssp:KeyInfo>
      <wssp:SecurityToken>
        <wssp:TokenType>http://schemas.xmlsoap.org/ws/2004/04/security/sc/dk</wssp:TokenType>
        <wssp:Claims>
          <wse:Parent>
            <wssp:SecurityToken wse:IdentityToken="true">
              <wssp:TokenType>
                http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken
              </wssp:TokenType>
              <wssp:Claims>
                <wse:Role value="ASKONNARD\User" />
              </wssp:Claims>
            </wssp:SecurityToken>
          </wse:Parent>
        </wssp:Claims>
      </wssp:SecurityToken>
    </wssp:KeyInfo>
  </wssp:Confidentiality>
</wsp:Policy>

```

Fig. 2 Policy Listing

You can define your own policy assertions, but in order for them to be useful, all parties must understand what they mean. Hence, it makes sense to define standard sets of policy assertions that all Web services can natively understand. This is precisely what the WS-PolicyAssertions and WS-SecurityPolicy specifications define for us.

In addition to these generic message assertions, the WS-SecurityPolicy specification defines a set of security-related policy assertions (see Table 2). These assertions allow you to specify the types of security tokens, signature formats, and encryption algorithms supported, required, or rejected by a given subject.

Policy Assertion	Description
wsse:SecurityToken	Specifies a type of security token (defined by WS-Security)
wsse:Integrity	Specifies a signature format (defined by WS-Security)
wsse:Confidentiality	Specifies an encryption format (defined by WS-Security)

TABLE 2 SECURITY POLICY ASSERTIONS DEFINED BY WS-SECURITYPOLICY

A Policy expression, which integrates the security policy assertions, prescribed in Table 2 in order to achieve message authentication, authorization, integrity and confidentiality is listed in Fig. 2 above. The above listing forms the basis of the policy files proposed as a solution in this paper to seamlessly integrate security into existing web services with minimal change.

5. Addressing Policy File Security

By introducing policy files at either end, security can be incorporated into the existing web service infrastructure with minimum effort. But the main drawback here is that the policy files are exposed to online attacks if an intruder manages to gain access to the files at either end. Since policy files would contain information such as the type of tokens used, usernames, passwords etc... penetration at either end would compromise security of the entire solution. Thus we propose a policy cipher tool, which would encrypt files at either end and ensure that they are only readable by the parties involved.

Another issue we came across in encrypting policy files is finding a means by which we can incorporate this security measure with minimum changes on either end. That is basically, the client and the web service should be able to interpret these

encrypted files without much difficulty and code change at either end.

The different types of elements in a policy file have to be also taken into account in its design. A tool which would facilitate all these features is expected to be the final outcome of this research.

6. Conclusion

Although many traditional securing mechanisms such as SSL, firewall, IPSec, PKI based systems exist and are inevitably proposed as a primary solution to securing web services owing to the fact that minimum changes are required on part of the developers, the inherent inabilities which accompanies these mechanisms has given rise to the development of new web service specific security standards which would secure messages in manner that wouldn't compromise the ubiquitous interoperability of this technology. WS-Security by Microsoft has been one such initiative. In conjunction with other standards like WS-Trust, WS-Secure Conversation and WS-Policy, WS-Security provides a firm base for the development of an all encompassed Web Service security architecture in the future. WSE 3.0 by Microsoft has been one of the most effective and widely adopted practical implementation of all these specifications [12].

As indicated in this paper securing Web services is not an impossible task. By the usage of standard approaches like WS-Policy, WS-Policy assertions etc. an easy integration into existing systems or infrastructures can be achieved. Also, since these policy files are based on XML it is possible to allow this add-on and still keep valid XML/SOAP documents. The improvement has been done in two steps. At first XML based policy files are introduced at the web service end points, which in turn ensure that all incoming and outgoing SOAP messages adhere to the policy specifications. As a second step, all applications are attached to policy files.

6.1 Future Work

Although the proposed solution seems to be ideal in seamlessly integrating security into the existing web service infrastructures, it could prove to be a burden on part of the applications, which might require slight modifications in order to incorporate these changes. Thus future work can be channelled through lines where even the applications would be able to transit seamlessly to use these policy files.

References

- [1] ZapThink: The Pros and Cons of WebServices, <http://www.zapthink.com> [visited on 12/11/2004].
- [2] SSL, <http://www.SSL.com>, [visited on 12/11/2004]
- [3] Is SSL enough security for 1st Generation services? <http://www.vordel.com/> [visited on 12/05/2006].
- [4] [XML Signature] – “XML Signature Syntax and Processing,” August 2001.
- [5] [XML-Encrypt] - W3C Working Draft, “XML Encryption Syntax and Processing,” 04 March 2002.
- [6] WS-Security, <http://docs.oasis-open.org/wss/> [visited on 24/03/2006]
- [7] The pros and cons of securing Web services with SSL, <http://searchwebservices.techtarget.com> [visited on 24/05/2006]
- [8] Sang Shin (2003). Secure Web Services, <http://www.javaworld.com/feedback>
- [9] WS-Policy, <http://msdn.microsoft.com/ws/2002/12/Policy/> [visited on 24/03/2006]
- [10] Jothy Rosenberg and David Remy, “Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption”
- [11] [Kerberos]- J. Kohl and C. Neuman, “The Kerberos NT Authentication Service (V5)” RFC 1510, Sep 93
- [12] Brian Nantz and Laurence Moroney, “Expert Web Services Security in the .NET Platform”