Web Service Security Overview, analysis and challenges

El Houssain BEN MESSAOUD and Ouafaa DIOURI
Université Mohammed V Agdal
Ecole Mohammedia des Ingénieurs
Laboratoire SIR
Avenue Ibn Sina B.P. 765 Agdal Rabat Maroc

Abstract

The web services (WS) technology became the reference architecture during the last few years for the integration of heterogeneous systems. As it is nowadays critical for business to make applications communicate over the internet. WS has take essential position for building and integrating e-business applications and to allow Information system technologies to communicate in an interoperable manner. New WS extra standards have been developed through the cooperation of several standardization organizations. This technology has also some limits specially security issues. We will try throughout this paper to provide an overview and an analysis of the standards in the field of web services security as well as to analyze the issues that are not yet addressed.

Treating Web Services security means treating aspects like authentication, authorization, integrity and confidentiality and how they can be guaranteed in WS architecture. Also an overview related standards called WS-Security, including how they combine to address security pains especially in a business to business Web Services scenario.

Keywords: Web Services, SOAP, WS Security, authentication, threats, policies.

1 Introduction

Forty years ago, computers began to be connected to the Internet and data transfer among computers was already common. Since then, Internet has evolved to form a huge information space, in which users can move transparently from one machine to another. In the field of application programs, a similar development is ongoing. Distributed computing has been used as long as there have been computer networks. But at present, distributed applications are increasingly viewed and constructed as one vast computing medium. Applications which interact between different machines to provide orchestrated services have now been deployed on a large scale. This evolution is allowed by new protocols built upon HTTP that are designed to enable interaction between programs [1].
Systems composed by loosely coupled, dynamically bound elements are much more flexible and have therefore better chances to dominate the next generation of information systems [2]. These distributed pieces of software are called web services and are deployed and used by many companies to integrate those information systems.
Despite these advantages, web services technologies, faces security limitations because of regular threat risks. This can decrease trust in information exchange based on this technology and compromise wide adoption in critical business applications.

The promised interoperability of web services also introduce security concerns that do not exist in traditional distributed messaging techniques like RMI and CORBA. This is because the SOAP based XML messages can bypass easily traditional firewalls and this could lead to gain access to sensitive systems for non authorized users just using the interfaces provided by the WSDL files for service description for example.

The object of this paper is to explain the principles of web services architecture. It presents the concepts, standards, and the required infrastructure. It discusses and analyzes the limitations of this architecture from the security view giving also light on the challenges surrounding this aspect related to this technology.

1.1 Organization of the document

The paper is organized as follows:

- Chapter 1 Enterprise architecture: introduces the different types of architectures in modern computing that motivate how and why remote services are emerging and what are the possible limits and problems.
- Chapter 2 Web service : provides an overview of the techniques for implementing web services as well as a listing of several WS-* specifications

- Chapter 3 Web service security: gives an overview about vulnerabilities of the web services applications. Then detailed the models and standards of WS security stack, how they interact and which protocols are used to achieve security requirements
- Chapter 4 Analysis : presents a personal analysis about the presents standards from different point of views
- Chapter 5 Challenges and opportunities: is the conclusion of this paper based on the analysis presented below.

2 Enterprise Architecture

Application architecture is an essential instrument for an application development cycle. The degree of abstraction used in the documentation of application architecture could vary. While some provide only highly abstract physical and logical representations of the technical patterns, others include more detail, such as common data models, communication flow diagrams and aspects of infrastructure [1].

In larger IT infrastructures, we need to define a high level architecture. These specifications will help to control and manage IT infrastructure when numerous, -disparate application architectures co exist and sometimes even integrate. In such a heterogeneous context, the underlying hosting platforms must be able to meet complex demands. Further, enterprise architectures often contain a long-term vision of how the organization plans to evolve its technology and environments [1] [6].

2.1 Client-Server Architecture

The “client-server architecture” refers to an environment in which the client and server had each particular functions as well as different implementation. This architecture is composed by multiple fat clients where each of them needed to connect to a database on a central server. Client-side software hosted the essential part of the processing, including all presentation-related and most data access logic [1].

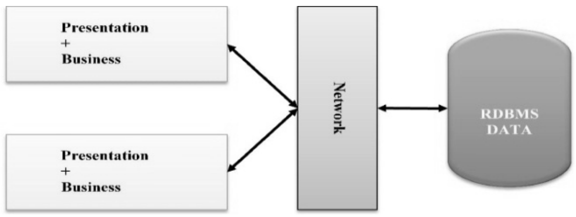

Figure 1 Two -tier client-server architecture

2.2 Distributed Internet Architecture

Regarding the lack of flexibility and the costs of the two-tier client server architecture, Component-based applications became popular. The multi-tier client-server applications as shown on Figure 2 divide the monolithic client executable into components designed to different degrees of compliance with object orientation. Applications can be deployed more easily when the application logic is located in numerous components because the logic is essentially centralized on servers. Sharing and managing pools of database connections by server-side components located on special applications servers reduces concurrent usage on the database server. These improvements brought also disadvantages with them: higher complexity and more costly development and administration processes [1].

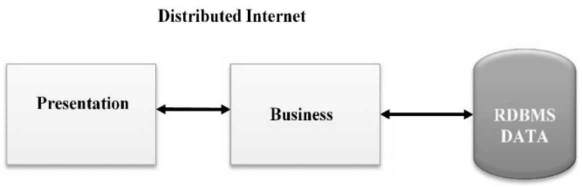

Figure 2 Multi-tier client-server architecture

Moreover, the client-server remote procedure call (RPC) has partially replaced the client-server database connections. RPC technologies like CORBA and DCOM enabled remote communications between components distributed between clients and servers. Problems appeared which were similar to those implied by client-server architectures, such as resources and persistent connections management. Additionally, the maintenance effort had to be increased due to the middleware layer. Servers and transaction monitors needed much attention in large environments.

2.3 Web Services Architecture

“Web Services, is considered a universal client/server architecture that allows disparate systems to communicate with each other without using proprietary client libraries”. The client and the server could be in heterogeneous technologies [3].

Web Services systems enable a high level of decoupling as well as dynamic binding of services. Such systems are composed by services which contain description and messages. Services are found by applications using service discovery [2]. The Web Services architecture is particularly adapted for e-business architectures.

2.4 Service-Oriented Architecture (SOA)

SOA presents a new method to create distributed applications where basic services can be published, discovered and bound together so as to build more complex composed services representing greater added value. Applications interact with services through an interface endpoint and not at the implementation level. Thus, applications become more flexible due to their ability to interact with any implementation of a contract [4]. The implementation of a SOA platform is based generally on Web service technology.

3 Web Services

3.1 Web Services definition

We can find several complementary definitions of Web Services. Some of them are:

“A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, and then waits for a corresponding XML response. Because all communication is in XML, web services are not tied to any operating system or programming language--Java can talk with Perl; Windows applications can talk with Unix applications” [5].

“Web Services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or Web based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML” [7].

“A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards” [8].

Thus, web services are platform-independent, based on XML messages and internet protocols. The idea is to distribute services over the Internet and to make them available for clients. These services can be implemented with any language.

3.2 Web services characteristics & Benefits

From [4] and [8] we can derive the following characteristics of web services:

XML-based: Web Services rely on XML for data representation and transportation. The use of XML avoids any network, operating system or platform binding.

Loose coupling: There is no direct link between a web service and its users. Alterations of the WS interface do not deteriorate the user’s capability of interacting with the service. Implementing a loosely coupled architecture facilitates software system management and helps the integration of different systems.

Adaptability: Ability to be work in a synchronous or asynchronous manner: In synchronous scenario, the client sends his request and then waits for the response without being able to execute other operations during this period. In contrast, asynchronous scenario allows clients to request a service and in parallel execute other operations without waiting for the result (“fire and forget” model).

Reusability: A Web service is a component which is remotely accessed using HTTP. Web Services provide a means to make a pre-existing code reusable and available through Internet.

Interoperability: Web Services enable the share of data and the communication between heterogeneous applications. For example, .NET applications can interact with Java web services and vice versa. Thus, application integration becomes platform and technology independent.

Standardized Protocol: Web Services uses industry standard protocols for the communication. All the four layers (Service Transport, XML Messaging, Service Description and Service Discovery layers) use well defined protocols in the Web Services protocol stack. This offers to organizations a reduction in their costs of applications integration with a best quality.

Automatic Discovery: Web Services automatic discovery mechanism allows businesses to easily find the Service Providers and retrieve web service descriptions that have been previously published. Client can query, based on search criteria, the service registry for web service matching his needs.

3.3 Web Services Architecture

Web services architecture is composed of three major components:

Service provider: It encapsulates the implementation of the service and makes it available on the Internet for consumers.

Service requestor: The consumer of the web service. It invokes an existing web service by opening a network connection and sending an XML-SOAP request containing the right parameters based on the description of the needed service and the address of the service provider.

Service registry: It is a centralized directory of services where providers or developers can publish new services or find existing ones.

The next figure gives a logical view of web services by illustrating the relationship between the web services components and operations. First, the web service provider publishes its web services with the discovery component. Next, the web service consumer looks for desired web services using the registry of the discovery component. Finally, the web services client invokes the web services by using the information obtained from the discovery component.

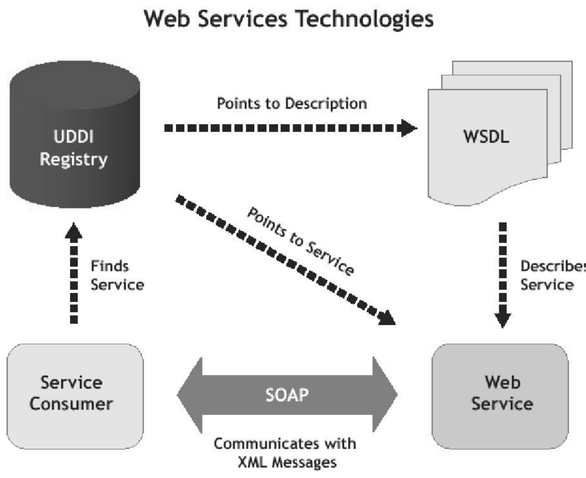

Figure 3 (taken from [9])

3.4 The Web Service Protocol Stack

Web services are built by using various related technologies. Figure 4 illustrates the stack of standards on which web services are generally based on.

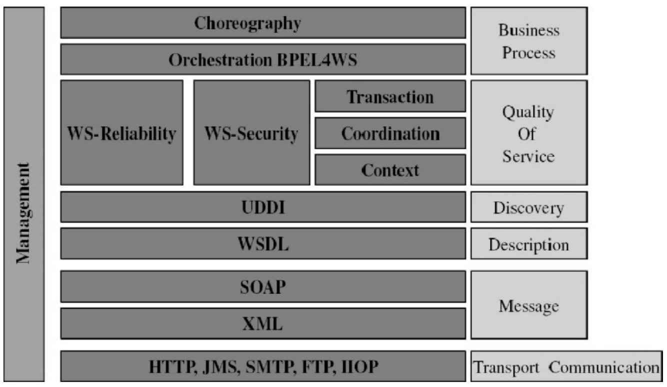

Figure 4 The Web Services technology stack (inspired from [4])

Service transport: The service transport layer delivers messages between applications. This layer usually implements hypertext transfer protocol (HTTP), Simple Mail Transfer Protocol (SMTP) or file transfer protocol (FTP).

XML messaging: This layer is responsible for encoding messages in a common XML format so that messages can be understood at both parties. This layer includes XML-RPC and SOAP [8].

Simple object access protocol (SOAP): SOAP is a simple XML-based messaging protocol responsible for transferring data between different web services. SOAP allows communication among interacting web services by implementing a request/response model [4].

Service description WSDL: The purpose of this layer is to define the public interface of a specific web service and its description. WSDL is based on XML [8].

Service discovery: The service discovery layer registers services into a common repository and provides an easy publish/find mechanism. This layer is often implemented via Universal Description, Discovery, and Integration (UDDI) [8].

Service orchestration: The service orchestration layer is in charge of the execution logic of web services based applications by determining their control flows (e.g. conditional, sequential, parallel and exceptional execution). This layer enables enterprises to define and realize complex business processes [4].

More layers can compose the full stack of web services like for example specifying quality of services (QoS) aspects.

To resume, a web service implementation is created using a specific programming language. This service is published using his WSDL interface. This service can be invoked by a consumer “client” using this interface. Web services are presented to clients as a set of methods that provide business logic on behalf of the provider. Web services must be deployed on a server container to be available for consumers as an online resource. The application developers have not to care about creating or parsing SOAP messages. That task is performed by the web service’s APIs runtime system. Web service can work over heterogeneous platforms. For example, a Java-based Web Service built and deployed on IBM AIX operating system can be accessed from Visual Basic program which runs on Windows.

4 Web service security

Security became an indispensable requirement for computer systems, ensuring that, access and information sharing occur without damaging the systems and that its information will not be exposed to malicious users. The Main security properties are:

- **Confidentiality**: ensures that information will be readable only by authorized users.
- **Integrity**: ensures that information cannot be changed, accidentally or intentionally, for users who do not have this right.
- **Authenticity:** it ensures that the user is communicating is really who he claims to be.
- **Non-repudiation**: ensures that the user cannot deny his involvement in the occurrence of a transaction.
- **Availability**: ensures that legitimate users have access to information and resources all the time.

We will try in this chapter to first list and discuss the major vulnerabilities of web services. There classification and then speak about security standards that give the ability to prevent the attacks caused by these vulnerabilities.

4.1 Vulnerabilities

Service Web Services are an integral part of next generation Web applications. The development and use of these services is growing at an incredible rate, and so too security issues surrounding them. Both providers and consumers need to assess Web Services’ security.

In this section, we present the terminology for WS vulnerabilities, their classification, and give detail of the major known attacks with some countermeasures.

4.1.1 Some definitions

When thinking about security, it is helpful to think in terms of assets, threats, vulnerabilities, and attacks [11].

- **Asset**. An asset is something related to your application that is worth protecting. Sensitive data, intellectual property, and access to critical operations are all assets. For example, user credit card numbers are an asset worth protecting in your application.
- **Threat**. A threat is any potential occurrence, malicious or otherwise, that could harm an asset. In other words, a threat is any bad thing that can happen to your assets.
- **Vulnerability**. Vulnerability is a weakness that makes a threat possible. This may be because of poor design, configuration mistakes, or inappropriate and insecure coding techniques. Weak input validation is an example of an application layer vulnerability, which can result in input attacks.
- **Attack**. An attack is an action that exploits vulnerability or enacts a threat. Examples of attacks include sending malicious input to an application, or flooding a network in an attempt to deny service.

To summarize, a threat is a potential event that can adversely affect an asset, whereas a successful attack exploits vulnerabilities in your system. The attacks can be divided to multiple categories as shown in the next sections.

4.1.2 Web service risk factors

SOAP web services have two main risk factors [14]:

- Distributed systems risks: Risks to the services themselves similar to risks that exist in web applications and component applications, such as malicious input attacks like SQL Injection. These risks arise from being distributed on a network. Network firewalls (which examine only a packet’s header) are largely blind to web services risks due to the fact that web services are deployed on commonly available open ports.

- Message risks: Risks to the document and data that is exchanged among the service requesters and providers. The document may participate in a multi-hop transaction or be subject to inspection by a variety of intermediaries, each operating in different security zones, including separate policy, geographic, technical, and organizational domains. The message's payload may also, of course, contain sensitive data.

4.1.3 Web Services Attack Classification

In this section we briefly overview the list of web services attack classifications. As the open architecture of web service present multiple attack surfaces in every layer. They can be classified in some categories:

- Message Alteration
- Non Authorized access
- Spoofing
- Denial of Service
- Replays attacks

Table 1 : Web services security threat framework

Webservice Layer	**Attacks and threats**	**Category**
Transit layer	Transit snifing/spoofing	Spoofing
	WS-Routing security concernes	Message Alteration
	Replay attacks	Replays attacks
Engine layer	Buffer overflow	Denial of service
	XM parsing attacks (complex/recursive)	Denial of service
	Large payload	Denial of service
Deployment layer	Fault code leaks	Non Authorized access
	Permissions and access issues	Non Authorized access
	Poor policies	Non Authorized access
User code layer	Parameters tampering	Message Alteration
	WSDL probing	Message Alteration
	SQL/XPATH injection	Message Alteration
	Virus/spyware/malware injection	Message Alteration
	Brute force	Non Authorized access
	Data type mismatch	Message Alteration
	Session tampering	Replays attacks
	Authorization violation	Non Authorized access

4.1.4 Example of vulnerabilities

Here we present a list of some security issues in the domain of Web Services. The list does not claim to be complete; it merely is a selection of the most impressive attacks with example of each category [14]:

4.1.4.1 Message Alteration

These threats affect message integrity. An attacker will modify parts message. For example, an attacker may insert extra information into a message. The attacks may affect message header and/or body parts.
An attacker may also affect message integrity by manipulating its attachments. For example, an attacker may delete an attachment, or insert an attachment into a message.

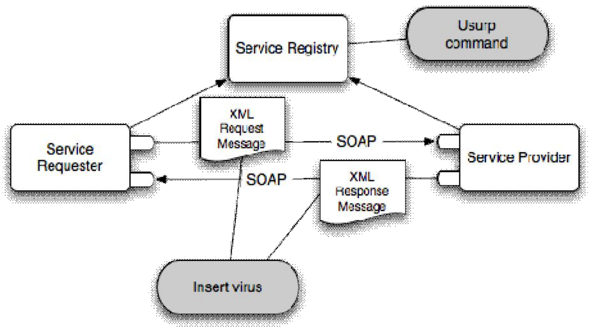

Figure 5: Message alteration attack

XML injection is a good example:

4.1.4.1.1 XML Injection

An XML Injection attack tries to modify the XML structure of a SOAP message by inserting content containing XML tags. Such attacks are possible if the special characters "<" and ">" are not escaped appropriately.
At the Web Service server side, this content is regarded as part of the SOAP message structure and can lead to undesired behavior.
Example:
<SOAP-ENV:Body>
<SOAPSDK4:MethodName xmlns:SOAPSDK4="http://urltoapp/…">
<SOAPSDK4:username>administrator</SOAPSDK4:username>
<SOAPSDK4:password>' ***OR '1'='1***</SOAPSDK4:password>*
</SOAP-ENV:Body>

*Here the parameters in the SOAP envelope have been injected with SQL to bypass authentication by always returning true (I.e SELECT * from UserTable where username='administrator' and password='' OR '1'='1';*

As the SOAP message obviously violates the Web Service schema, it should be rejected.

An important step in detecting such attacks is a strict schema validation on the SOAP message, including data type validation.

4.1.4.2 Non authorized access

4.1.4.2.1 Man-in-the-middle

In this kind of assault it is possible for an attacker to compromise a SOAP intermediary and then intercepts messages between the web service requester and the receiver. The parties will think that they are communicating with each other. The attacker may just have access to the messages or may modify them. Mutual authentication techniques can be used to alleviate the threats of this attack.

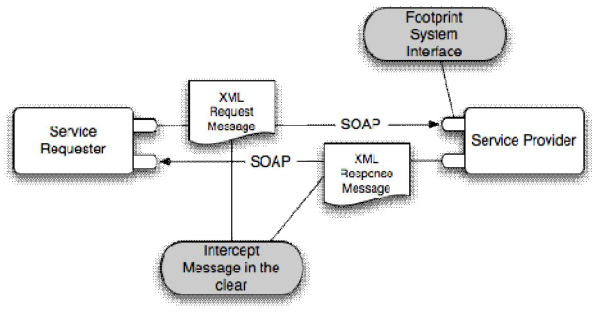

Figure 6: Man-in-the middle attack

4.1.4.3 Spoofing

Spoofing is a complex attack that exploits trust relationships. The attacker assumes the identity of a trusted entity in order to sabotage the security of the target entity. Usually, spoofing is used as a technique to launch other form of attacks such as forged messages. Strong authentication techniques are needed to defend against such attacks.

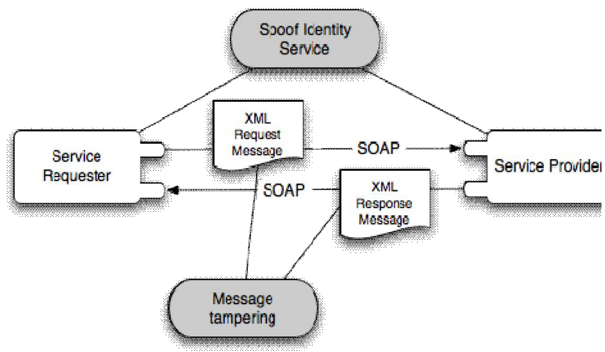

Figure 7: Spoofing attack

Metadata Spoofing attack is a good example:

4.1.4.3.1 Metadata Spoofing

Web Service client retrieves all information regarding a Web Service invocation (i.e. message format, network location, security requirements etc.) from the metadata documents provided by the Web Service server. Currently, this metadata usually is distributed using communication protocols like HTTP or mail. WSDL Spoofing is the modification of the network endpoints and the references to security policies. A modified endpoint enables the attacker to easily establish a man-in-the-middle attack for eavesdropping or data modification. If additionally a spoofed security policy with lower or no security requirements is used, such attacks are possible despite the use of WS-Security. The solution is that all metadata documents must be carefully checked for authenticity.

4.1.4.4 Denial of Service

Denial of service (DoS) attacks focus on preventing legitimate users of a service from the ability to use the service. Such attacks target at eliminating a service's availability by exhausting the resources of the service's host system, like memory, processing resources or network bandwidth. In the webservice world, it is based generally on XML parsing that is expensive (Extremely large / complex XML documents, deeply nested tags…). This can lead to create extremely large memory footprints or exhaust CPU treatment capacity.

4.1.4.4.1 Oversize Payload

One classic way to perform such a Resource Exhaustion attack is to query a service using a very large request message. This is called an Oversize Payload attack. It is quite easy to perform, due to the high memory consumption of XML processing. The total memory usage caused by processing one SOAP message is much higher than just the message size. An obvious countermeasure against Oversize Payload attacks consists in restriction of the total buffer size for incoming SOAP messages.

Example: A Web Service was attacked using a large SOAP message document, which consisted of a long list of elements considered as parameter values of the Web Service operation:

```
<Envelope>
<Body>
<getArrayLength>
<item>x</item>
<item>x</item>
<item>x</item>
...
</getArrayLength>
</Body>
</Envelope>
```

The SOAP message had a total size of approx. 1.8 MB. The message processing induced a full CPU load for more than one minute and an additional memory usage of more than 50 MB. Enlarging the message to approx. 1.9 MB could resulte in an out-of-memory exception.

4.1.4.5 *Replay Attacks*

In this attack an intruder intercepts a message and then replays it back to a targeted agent. Appropriate authentication techniques coupled with techniques such as time stamp and sequence numbering the messages can defend against replay attacks.

4.1.4.5.1 Session replay

An attacker steals messages off of the network and replays them in order to steal a user's session to accomplish authorized operations for this user by an authorized manner.

4.2 Web services security

To secure Web services, a range of XML-based security mechanisms are needed to solve problems related to authentication, role-based access control, distributed security policy enforcement, message layer security that accommodate the presence of intermediaries. This is a principal condition to make Web services widely adopted, since no company wants to risk exposing their applications and business flows with no damage. Standardization organizations are proposing specifications in order to make these services more secure as traditional Security techniques doesn’t provide security against Application level communication as they works on the Lower levels of the OSI stack of message transfer specially on transport layer. The most important standards are:

- XML Encryption
- XML Signature
- WS-Security
- WS-Policy
- WS-Security Policy
- WS Trust

Here we will present first the security requirements for the web service architecture, the different security approaches and models, give the detail of the security standards and protocols and summarize all of that.

4.2.1 Web Services Security Requirements

There are many security challenges for adopting Web services. The objective is to create an environment, where message level transactions can be conducted securely in an end-to-end fashion during transit and data storage. The requirements for providing end-to-end security for Web services are summarized in [15]:

Tab 2: Web service security requirements

Requirement	Explanation
Authentication	Authentication is needed in order to verify the identities of the requester and provider agents. In some cases, the use of mutual authentication may be needed since the participants may not necessarily be directly connected by a single hop. Several methods can be used to authenticate services (can be combined) including: passwords, certificates, Lightweight Directory Access Protocol (LDAP), Kerberos, and Public Key Infrastructure (PKI)
Authorization	Authorization is needed in order to control access to resources. Once authenticated, authorization mechanisms control the requester access to the requested resources on the system
Data Integrity and Data Confidentiality	Data integrity techniques ensure that information has not been altered, or modified during transmission without detection. Data confidentiality ensures that the data is only accessible by the intended parties. Data encryption and digital signature techniques are used for this purpose. It must be verified in End-to-End manner
Non-Repudiation	It is a security service that protects a party to a transaction against false denial of the occurrence of that transaction by another party. It used to resolve probable disagreement.
Audit Trails	Audit trails are needed in order to trace user access and behavior. They can ensure system integrity through verification. It is often not possible to prevent the violation of obligations. Instead, if an audit guard detects a policy violation, some form of retribution or remediation must be enacted.
Distributed Security Policies	The architecture must be able to provide a security policy and enforce it across heterogeneous platforms with varying constrains and privileges

4.2.2 Web services Security Model

The most used Web service security model is described next. It helps to achieve securing the integrity and confidentiality of the messages and for ensuring that the service provider acts only on requests in messages that express the claims required by his policy and all of that as an end-to-end security mechanism.

The model requires that a Web service can demand that an incoming message prove a set of claims (e.g., name, key, permission, role, etc.). The service may ignore or reject any message arriving without having the required claims. The required claims and related information are called policy.

A requester can send messages with proof of the required claims by associating security tokens with the messages in order to prove that their sender has the claim to demand the action. When a requester does not have the required claims, the requester or a component on its behalf can try to obtain the necessary claims by contacting other Web services called "Security Token Services". These components may in turn require their own set of claims. Security token services broker trust between different trust domains [13].

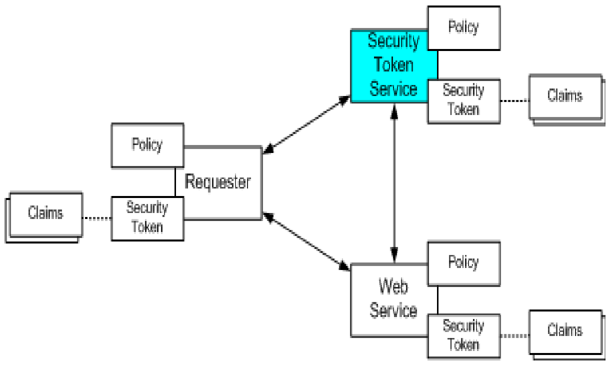

Figure 8: Web Service Security Model

4.2.3 Protocols and standards

Concretely, Web service security specification includes a message security model that provides the basis for the other security specifications. Layered on this, WS-Security propose a policy layer which includes a Web service endpoint policy (WS-Policy), a trust model (WS-Trust), and a privacy model (WS-Privacy). Together these initial specifications provide the foundation upon which secure interoperable Web services across trust domains are established. For example, WS-Security describes how to attach signature and encryption headers to SOAP messages. It also describes how to attach security tokens to messages. Next we will see how all that works [15] [14].

4.2.3.1 WS-Security

Developed at OASIS, this standard defines a SOAP extension providing quality of protection through message integrity, message confidentiality, and message authentication.

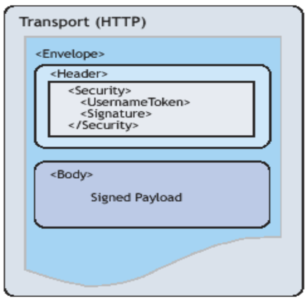

Figure 9 : WSS integration in a SOAP Message

It provides a general mechanism to associate security-tokens with messages (UsernameToken, BinarySecureToken, XML Tokens), describes how to encode binary security tokens in messages and includes enhancements to SOAP to provide quality of protection mechanisms. Additionally, it also describes how to include opaque encrypted keys.

The WSS specification defines an end to end security framework that provides support for intermediary security processing. Message integrity is provided by using XML Signature in conjunction with security tokens to ensure that messages are transmitted without modifications. Message confidentiality is granted by using XML Encryption in conjunction with security tokens to keep portions of SOAP messages confidential.
The WS-Security header is denoted in the XML message as

```
<wsse:Security>
...
</wsse:Security>
```

WS-Security allows the service requester or provider to encrypt and sign parts of a given message. This allows for a flexible integration where sensitive data may be encrypted and signed, but because message-level security is not an all or nothing proposition, the expense and complexity of these security mechanisms may be limited to specific message parts. The WS-Security header contains timestamp, encryption, digital signature, and security token data to provide message security services [14]. These notions are exposed here:

4.2.3.1.1 Timestamp

The timestamp is included in the header for the service provider to evaluate the length of time since the claims (for example, authentication claims) were made in the message and when the message is read by the service provider. In an asynchronous system such as an enterprise service bus or more elaborate SOA orchestrations, significant time may elapse between the time a message is generated by a service and the time it reaches the implementation consumer. One of the main uses of the WS-Security message timestamp is to introduce some entropy in the message to protect against replay attacks.
The timestamp also allows the service to stamp an expiration date on the message's claims so that the service provider knows to accept claims only within a given time parameter. For example, when authorizing a payment on a credit card, a payment system may hold a sum of money against a credit card for a period of time; if the transaction is not completed within the given time, a new authorization may need to be generated.
The timestamp is represented in the XML message in the WS-Security header [14]:

```
<wsse:Security>
       <wsu:Timestamp>
         <wsu:Created >2006-08-
       09T06:12:03Z</wsu:Created>
         <wsu:Expires >2006-08-
       09T08:12:03Z</wsu:Expires>
       </wsu:Timestamp>
</wsse:Security>
```

4.2.3.1.2 WS-Security token types

WS-Security headers may contain three different types of security tokens: username, binary, and XML tokens [14]:

- *Username token:* The username token is the most basic type of security token in WS-Security. The username token is a simple XML description of the username the service claims to represent. The basic username token is unsigned, making it a weak assurance option for protecting messages. The username token is made stronger by signing it as part of the message and by adding a password, either in the form of a plaintext password (which would be a poor choice for messages passed over any communications channel that is not highly secure, or arguably even over channels that are considered to be very secure) or as a password digest.
- *Binary token:* X.509 digital certificates and Kerberos tickets are binary security tokens that are encoded as binary and represented in XML documents passed between the services. These token types allow security architects to integrate their existing identity and access management systems, such as PKI, LDAP, and Active Directory, into their webservices applications. While X.509 and Kerberos can provide higher assurance than username tokens, they do add complexity to applications. The balance that the software security architect must seek is evaluating the number of systems that are to be integrated that already use security credentials from X.509 and Kerberos systems.
- *XML security tokens (SAML)*: WS-Security and SAML both provide some similar solutions in web services security, and in some cases may be used instead of each other. WS-Security is able to leverage SAML as an XML security token type. SAML's security model uses *assertions* that are mediated between an assertion producer and assertion consumer, which is conceptually similar to what the WS-* model calls *claims*. WS-Security provides the framework to bind SAML tokens to SOAP messages.

4.2.3.2 *XML Digital Signatures*

It specifies a process for generating and validating digital signatures expressed in XML, ensuring the integrity and authenticity in XML documents. XML signatures are designed for use in XML transactions. It is a standard that was jointly developed by W3C and the IETF (RFC 2807, RFC 3275). The standard defines a schema for capturing the result of a digital signature operation applied to arbitrary data and its processing. XML signatures add authentication, data integrity, and support for non-repudiation to the signed data.
XML Signature has the ability to sign only specific portions of the XML tree rather than the complete document. This flexibility can ensure the integrity of certain portions of an XML document, while leaving open the possibility for other portions of the document to change.

4.2.3.3 *XML Encryption*

This standard specifies a process for encrypting data and representing the result in XML such that it is only discernable to the intended recipients and opaque to all others. The data may be arbitrary data (including an XML document), an XML element, or XML element content. The result of encrypting data is an XML Encryption element which contains or references the cipher data.
It provides end-to-end security for applications that need to exchange data in XML format in a secure way, without concern that they can have their contents revealed and misused by non authorized parties.

4.2.3.4 *SAML*

SAML is an OASIS standard. It means Extensible Markup Language standard (XML) that supports Single Sign On. It defines a standardized XML format for credential and security assertion data SAML can be used in business-to-business and business-to-consumer transactions. There are three basic SAML components: assertions, protocol, and binding. Assertions can be one of three types: authentication, attribute, and authorization. Authentication assertion validates the identity of the user. The attribute assertion contains specific information about the user. While, the authorization assertion identifies what the user is authorized to do.
The protocol defines how SAML request and receives assertions. There are several available binding for SAML. There are bindings that define how SAML message exchanges are mapped to SOAP, HTTP, SMTP and FTP among others. So the authentication and authorization information can be moved around systems within or between organizations SAML is platform-independent and language independent. A key objective of SAML is to allow organizations to exchange date regardless of the security system they use [16].

4.2.3.5 *WS-Policy*

The WS-Policy Framework defines a general purpose model and corresponding syntax to describe and communicate Web services policies to allow Service consumers can discover the information they need to know to be able to access services from a Service Provider.

WS Security policy is an extension of WS-Policy dedicated to describe the security requirements. It defines a model and syntax to describe and communicate security policy assertions within a larger Policy Framework covers assertions for security tokens, data integrity, confidentiality, visibility, security headers and the age of a message.

4.2.3.6 *WS-Secure Conversation*

It Defines mechanisms for establishing and sharing security contexts, and deriving keys from security contexts, to enable a secure conversation by allowing the creation of sessions where several SOAP messages can be exchanged without theneed to renew the authentication and authorization foreach (Holgersson and Soderstrom, 2005). This standard is built on top of the WS-Security and WS-Policy models to provide secure communication between services on optimizing resource use. For example, a signature may be checked to establish the context, and that context is set for either a period of time or an amount of messages.
The initialization process for WS-SecureConversation creates a SecurityContextToken (SCT), which may be created through WS-Trust. The SCT is passed with each subsequent message, as opposed to passing a normal security token with each message that must be independently checked. The lifespan for an SCT typically specifies a number of messages or a time span.

4.2.3.7 *WS-Trust*

Defines how trust relationships are established, allowing Web services interoperate safely. This can be accomplished using the secure messaging mechanisms of WS-Security to define additional primitives and extensions for the issuance, exchange and validation of security tokens. WS-Trust also enables the exchange of credentials within different trust domains.

In WS-Trust, a Security Token Server (STS) is used to handle Request Security Token RST calls. The security tokens supported by WS-Trust are the same tokens supported by WS-Security: Username, X.509, Kerberos, and SAML. In addition, since the tokens are represented in XML, the message can contain proprietary and homegrown security tokens such as session cookies and mainframe tokens.

4.2.3.8 *XACML*

XACML is an Extensible Markup Language standard (XML) based technology, developed by OASIS for writing access control polices for disparate devices and applications.
XACML includes an access control language and request/response language that let developers write policies that determine what users can access on a network or over the Web. XACML can be used to connect disparate access control policy engines.

4.2.4 WS security standards - the big picture

The WS-Security specification determines the use of XML Encryption and XML Signature in SOAP to secure communication. It is used either as an alternative or an extension to using HTTPS to secure the message exchanges. It covers two types of mechanisms

- Communication Protection mechanisms: like XML encryption and XML signature for the integrity and confidentiality of the exchange, and also timestamping protecting from replay attacks.
- Access Control Mechanisms: for example using SAML, which defines how identity, attribute, and authorization assertions should be exchanged among participating services in a secure and interoperable way and also XACML providing a complete authorization engine.

Layers like WS Trust and WS security Policy are here to define standard manner to describe security constrains and distribute security information between heterogeneous and trusted domains.

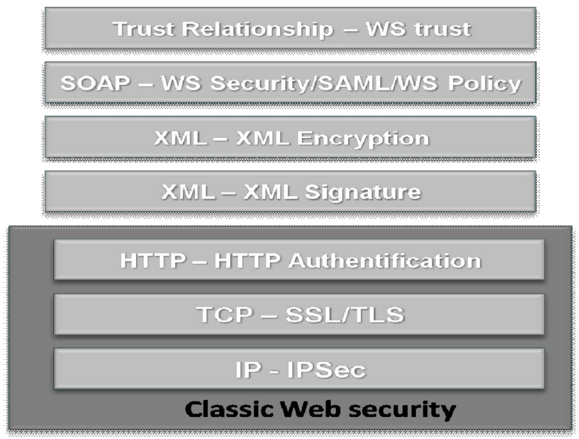
Trust Relationship – WS trust
SOAP – WS Security/SAML/WS Policy
XML – XML Encryption
XML – XML Signature
HTTP – HTTP Authentification
TCP – SSL/TLS
IP - IPSec
Classic Web security

Figure 10: WS security stack

This table, present the security requirement covered by each standard cited below:

Table 3: WS security perimeter

Security requirement	Standard
Integrity and non repudiation	XML Signature
Confidentiality	XML Encryption
Authentication and identity federation	SAML, WS Trust
Authorization	XACML
Expression of security requirements	WS Policy
Security context among transactions	WS-SecureConversation

5 Analysis

As seen in the previous chapters, WS-Security provides some important security services for web services and allows non intrusive integration on existing systems. The WS-Security standard does not address other issues related to security infrastructure like policy storing and key management which must be set up separately. Also based on our analysis for the current standards, we will describe in this section some lacks we fined to address all the pains and the issues related to web service security.

5.1 Channel security Vs Message security

Implementing WS security (message security mechanism) becomes a necessity as it’s proven regarding the arguments bellow that Channel security is no longer enough:

- "Point-to-point" security: Any communication with multiple "hops" requires establishing separate channels (and trusts) between each communicating node along the way. Trust transitivity is not guaranteed, as trusts between node pairs {A,B} and {B,C} do not automatically imply {A,C} trust relationship.
- Lack of flexibility: Although Channel security technologies are used in Web services security, they are not sufficient for providing end-to-end security, as Web services require more granularities. In general, Web services needs complex interactions that can include the routing of messages between and across various trust domains.
- Lack of interoperability: Not using standard message security technology implies that applications have to utilize proprietary mechanisms for transmitting credentials, over the secure channel. This can lead to altering the clients/servers and prevent forming automatic B2B service integration.

Unfortunately, the reality is that there still a lot of Web Services that are protected by some form of channel security mechanism, which alone might suffice for a simple internal application. However, in a B2B exchange and depending on the sensitivity of the data, a combined protection would work better for each specific case.

5.2 Complexity

WS Security standard aims to provide tools for message-level communication protection, whereas each message represents an isolated piece of information, carrying enough security data to verify message properties, such as: authenticity, integrity, freshness, and to initiate decryption of any encrypted message parts [17]. This manner is very complex regarding to the traditional channel security, which methodically applies pre-negotiated security context to the whole stream. We note that this type of service could be provided by WS-SecureConversation implementation, but this standard is steel not enough mature.

From the architectural view, the WS-Security standard was conceived as a message-level toolkit for securely delivering data for higher level protocols. Those protocols rely on the transmitted tokens to implement access control policies, token exchange, and other types of protection. However, taken alone, the WSS standard does not mandate any specific security properties, and poorly designed application can lead to subtle security vulnerabilities and hard to detect problems as WS-Security is not ready to use out of the box like SSL. Developers need when using WS-Security to determine when to sign and encrypt, as well as decide on a token. They need also to decide on which order these operations will be processed.

5.3 No Audit standard

In compliance with some regulation constraints, audit must be implemented on sensitive web service systems. Audits are also used when reconstructing the chain of events that led to a certain problem. Unfortunately, no standard auditing framework is proposed to the webservice stack till now and so these systems must be developed individually. Additionally there is not an “out of the box” way in most web services implementations to correlate service requests and responses. This is a critical lack in some business cases [14].

5.4 Bigger attack surface of WS-Security

Analyzing WS architecture and comparing it with SSL let us to assure that the attack surface of WS-Security is much bigger than that of SSL as with message-oriented security; we need to have messages before you can do anything. That's not the case with SSL, where the attacker gets less to play with. WS-Security acts as a target-rich environment that is open for attack. In contrast, SSL with client certificates keeps users out of the message details and metadata unless authenticated.

5.5 Lack of standard token and credentials validation

WS-Security standard rely on the transmitted tokens carrying credentials to implement access control policies, However, the verification mechanics of those credentials are completely at the Web Service's discretion (example: taking the supplied username and password's hash and checking it against the backend user store, or extracting subject name from the X.509 certificate used for signing the message, verifying the certificate chain and looking up the user in its store). At the moment, there are no requirements or standards which would guide how it should be done. All depends on the application implementation and the quality of the design provided by the developers.

5.6 XML Encryption problems

Using WS-Security on Web Services could introduces in some cases, new problems concerning service availability specially when using XML encryption standard supposing providing confidentiality to sensible data. XML Encryption can also mask message content from being inspected correctly. As this encrypted content can contain an intended attack like Oversize Payload, Coercive Parsing or XML Injection. This kind of attacks is unfortunately hard to detect as analyzing the message structure by lunching schema validation needs decryption first. In this case, here are two possibilities on how a targeted system may be affected. If decryption is done after message validation, the malicious message content may pass the message validation. If decryption is done before message validation, the system resources could be exhausted during message decryption because of the XML and cryptographic processing. Thus, even if a system is able to counter the unencrypted attack, obfuscated attacks may affect a target system anyway and its availability compromised. [12] This kind of attacks is complex to generate but must be considered by application architects.

5.7 Brokered Authentication problem

The client and the service provider do not attempt to authenticate each other directly. They use an intermediary security token as proof of successful authentication. The client attaches this token to the request and the service uses this token to authenticate the client. That validates the client's identity and then processes the message [10]. This manner makes the availability of Web services depending of the availability of the identity providers. These components must provide High availability guarantee when used in B2B scenarios.

5.8 Adoption in Internet scenarios

In general WS security protocols are efficient in communication between two trusted parties with an established security association. These protocols are not designed for protecting in an internet scenarios where anonymous consumers could introduces security vulnerabilities as the ability to establish and maintain security policy agreements and security data, such as user credentials, with potentially unknown customers is not firmly established. Consequently, we can say that the infrastructure is not yet sufficient for secure public internet transactions. Securing these transactions depends first on risk analysis and a tradeoff with the cost/effort that is required to implement custom solutions.

5.9 Performances

Using WS-Security implies using signing and encryption. Those operations are costly in matter of resources (CPU and Memory) they can cut application throughput between 5 percent and 50 percent. A solution for this overhead could be the use of a dedicated hardware call XML Firewalls.

Those hardware systems provide performance as they allow real-time processing of huge documents. But they cannot always be used as an optimal solution as this quality comes with a price and also they cannot be easily integrated with the already existing back-end software infrastructure.

6 Conclusion & Challenges

In this paper we have described the different distributed architectures and specially the most used architecture actually when integrating heterogeneous information systems which is web service.

We have described the nature and characteristics of web services and have presented their advantages.

We have seen that web services constitute a sort of automated services which communicate via Internet and rely on open Internet-based standards.

We have seen also that Web services are invoked using messages instead of APIs or file formats. This works due to the independence of the service interface through the WSDL standard from the implementation.

Nevertheless, web services are also subject to some limitations which include low performance, weak transaction management facilities but more critical, an immature and incomplete security framework although the presence of the WS Security framework that we presented and give a critical analysis.

This analysis revealed some points to treat end some progress to do in order to get a sufficient trust level when managing sensitive and valuable business transactions. It is important de note that Web Service security represents a key requirement for today's distributed interconnected electronic world. To date, the problem of security has been investigated very much in the context of standardization efforts; these efforts, however, been concentrated in adapting existing security techniques, such as encryption, for use in Web Services. The standards have also focused on addressing the problem of security interoperability through the development of standard formats for security assertions, tokens and credentials. These standards are grouped under the WS-Security framework which is not inventing any new techniques but they are providing a way how to use the existing technologies with SOAP to secure the communication of web services.

In the previous sections we demonstrate that the usage of WS-Security does not automatically ensures full security for Web Services. As shown before, WS-Security defines mechanisms for enabling integrity and confidentiality for Web Service messages. However, several issues have not yet been addressed while some of the vulnerabilities are caused by implementation weaknesses and exploit protocol lacks for example, WS-Security does not define any direct countermeasures against attacks like Denial-of-Service. WS Security stack remain in all cases the best starting point as it is well reviewed by industry experts and regularly updated.

Throughout our analysis we recommend a number of considerations to build a secure architecture enhanced by defense-in-depth precautions:

Strong access control mechanism: A well-known protective mechanism for service availability is access control. Access control restricts access to the service to trusted users. Additionally, access control enables accountability, allowing excluding the attacker. WS-Security defines security tokens for authentication, which can be used for access control systems. Of course, access control cannot fully eliminate the threat of attacks. First of all, even trusted communication partners can intentionally or unintentionally execute attacks.

Solution design based on deployment scenario: depending on the scenario constraints security component must be well chosen. For example, due to the fact that authentication needs a key infrastructure; it is not applicable in B2C relationships, as there is no wide-spread key infrastructure among private users.

Service Assurance is critical: Even if the messages are secured in transit, many of the depending on services are not on our direct control, from the security view. Even for services in our control, there is still the possibility of insider threat. It's therefore critical to establish mechanisms for detecting security violations through auditing and the use of intrusion detection tools, along with policies and procedures for recovery and response when problems are detected. The need for good security practices at the network, host, application, personnel, processes, and physical layers is also fundamental.

Safe implementation and configuration: It is important to realize from the beginning that no security standard by itself is going to provide security to the message exchanges, it is the installed implementations and the technical configuration, which will be assure the required security rules. For example, either requiring digital signature if a weak key size is configured to be used the system could remain vulnerable.

Heterogeneity improves Security: One security benefit from web services is its interoperability throughout several platforms. It means that a given application may use many different technologies, which may reduce the impact of an attack that exploits a specific vulnerability in a particular software or hardware platform. A heterogeneous system can be more resilient in the face of attacks if it is designed to provide diverse and redundant means for assuring continuity of essential services. A Well designed application must provide an abstraction layer in front of the services so this can protect it against cascade failure preventing the access to the underlying technologies.

Regular security policy review: Web services standards do not create effective policy (though WS-Policy and WS-SecurityPolicy are used to express policy) the creation and coordination of security policies are the responsibility and obligation of the participating organizations [14]. A collaborative review of their security policies by participating organizations can help to enhance the global security level and resolve incompatibilities.

To resume, WS-Security is one of the important building blocks for fending attacks but has to be applied carefully. The solutions for the problems cited below must be oriented on these directions:

- Prevention through the use of pre verified security policy templates which group together best practices for protecting incoming and outgoing SOAP message and a mechanism for regular review by experts
- Detection and reaction: through the emergence of a standard in WS Audit that can provide an Audit record aiming to prove what happened in case of a violation and giving the possibility to lunch compensation actions in reaction of the consequences of these violations.

7 References

[1] Erl, Thomas. Service-Oriented Architecture, Concepts, Technology, and Design. s.l. : Prentice Hall Indiana, 2006. 0-13-185858-0.
[2] Gottschalk,Karl. “Web Services Architecture Overview”. 2000. http://www.ibm.com/developerworks/webservices/library/w-ovr/.
[3] Myerson, Judith. Web Services Architectures.
[4] Papazoglou, Michael. Web Services: Principles and Technology. s.l. : Prentic Hall, 2008.
[5] Cerami, Ethan. Top Ten FAQs for Web Services. 2002. http://webservices.xml.com/lpt/a/1130.
[6] ABDALDHEM AL BRESHNE, PATRIK FUHRER, JACQUES PASQUIER. Web Services Technologies: State of the Art Definitions, Standards, Case Study. September 2009
[7] IBM. WebSphere Business Integration Adapters. http://publib.boulder.ibm.com/infocenter/wbihelp/v6rxmx/index.jsp?topic=/com.ibm.wbia_adapters.doc/doc/webservices/webservices17.htm.
[8] Point, Tutorials. What are Web serivces. http://www.tutorialspoint.com/webservices/what_are_web_services.htm.
[9] David, M. Rubin. Intro to Web Services. 2002. http://www.softstarinc. com/Methodology/Softstar%20Web%20Services%20Presentation.ppt.
[10] D.Shravani1 P.Radhika2 Dr.P.Suresh Varma3 Dr.D.Sravan Kumar4 M.Upendra Kumar Architecting Secure Service Oriented Web Services, ACEEE Int. J. on Communication, Vol. 01, No. 03, Dec 2010
[11] “Security Fundamentals for Web Services : Chapter 1” http://msdn.microsoft.com/en-us/library/ff648318.aspx
[12] Meiko Jensen, Nils Gruschka and Ralph Herkenhoener. A survey of attacks on web services. Computer Science - Research and Development (CSRD), Volume 24, Number 4, pages 185-197, Nov.2009.
[13] IBM, Microsoft. “Security in a Web Services World: A Proposed Architecture and Roadmap” April 7, 2002, http://msdn.microsoft.com/en-us/library/ms977312.aspx
[14] Gunnar Peterson and Howard F. Lipson "Security Concepts, Challenges, and Design Considerations for Web Services Integration" https://buildsecurityin.us-cert.gov/articles/best-practices/assembly-integration-and-evolution--security-concept-challenge-and-design-considerations-web-services-integration Published: December 14, 2006
[15] “Web Services Architecture” http://www.w3.org/TR/ws-arch/ W3C Working Group Note 11 February 2004
[16] Jorgen Thelin. "XML Security Standards: Current and Emerging Specifications attempting to provide standardization of XML security infrastructure” Cape Clear Software Inc. 2003
[17] “Web Services”. OWASP. https://www.owasp.org/index.php/Web_Services