

WEB SERVICES AND WEB SERVICES SECURITY

Robert J. Boncella
Washburn University
bob.boncella@washburn.edu

ABSTRACT

Web services are self-contained modular applications that provide a computation upon request. These services can be described, published, located, and invoked over a network, generally over the Internet. However, intranets, extranets, and LANs can also be used. as well. In using web services for its information systems needs, a firm may open access to its information assets. This action can become an attractive target for malicious hackers, industrial espionage, and fraud. The assurance of security of web services is necessary for a firm to be willing to adopt the web services technology as a means of running its information systems

KEYWORDS: web services, ws-security, web service security

I. INTRODUCTION

The purpose of this article is to provide a foundation for understanding the need and techniques of web services security. To provide this foundation, we present a review of the concept of web services and its related ideas. After discussing the security requirements for web services, we describe the techniques that provide these security requirements.

II. WEB SERVICES

DEFINITION AND IMPLICATIONS OF WEB SERVICES

A definition of web services is

Web services are self-contained modular applications that provide a computation upon request".

These services can be described, published, located, and invoked over a network, generally the Internet. Intranets, extranets, and LANs can also be used. Most often the World Wide Web (WWW) will be the means for the request of services. Furthermore web services are independent of the underlying systems providing the computation.

The implications of this paradigm of computing cannot be underestimated. Potentially, the web services architecture can provide universal interoperability of software applications. Every software application in the world could interact with every other software application in the world. This interaction is independent of geographical location, system hardware, operating system, and programming languages of the software application.

The web services paradigm of computing could provide a de facto standard for any particular type of computation. For example, any number of computing systems can provide the verification of a

customer's shipping address. If each of these systems provide this service publicly upon request that service becomes a commodity. As a commodity it will allow for the substitutability of services. It is possible for a firm to develop an information system for its use based entirely on the computational services provided by the web services paradigm. Such an information system can be developed at least cost because of the substitutability of services provided by the de facto standardization of computation. This method of development also impacts the cost of maintaining the information system. The firm is not responsible for the process that provides the service. The service provider is, and hence is required to maintain of the quality of that service.

Many services that a firm uses in its business operations have the characteristic of web services. For example, a firm that needs to ship an item from New York City to Laramie, Wyoming can use a number of service providers, including UPS, USPS, and FedEx, and DHL. The firm is interested in the results of service not how the provider implements the service.

If a firm is to use the advantages of web services it must trust the security of the web services. To use web services for its information systems needs, a firm must provide access to its information assets. This action can become an attractive target for malicious hackers, industrial espionage, and fraud. The assurance of security of web services is necessary for a firm to be willing to adopt this technology as a means of creating its information systems.

To appreciate the challenges of web services security, the reader needs to understand the architecture.

WEB SERVICES ARCHITECTURE

The foundation of web services is the request/response paradigm. This paradigm involves a client requesting a service via a specific protocol to a service provider. That service provider responds with either the service or a notification of why it cannot provide that service using a specific protocol. An example would be a web browser using the HTTP protocol to request a web page from a web server. The details of this paradigm can be found in [Boncella 2000]. This concept is used to implement the service oriented architecture (SOA) which implements web services. This architecture contains three entities and three operations and is illustrated in Figure 1.

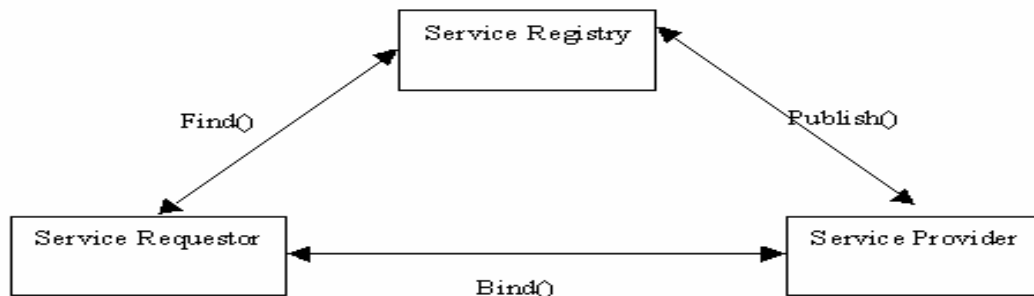


Figure 1. Web Services Architecture

In Figure 1, the service requestor is an application that requires a service (computation) from another application or applications. If the requesting application does not know where this service is located it sends a request to the service registry using the Find() operation.

The service registry is a well know application that responds to a Find() request sent by a service requestor. The service requestor provides search criteria for a particular service with the Find() operation. The service registry returns information about the requested service and where to find the service.

The service providers Publish() (1) the specifications of the services they provide and (2) where the services are provided to the service registry.

If the specifications of a service provider's service match the requirements of a the service requestor, the service requestor uses the location information provided in the service registry to Bind() to the service provider. Once bound together the service requestor and service provider can engage in the request/respond paradigm to complete a computation.

To carry out their intended functions, these entities and operations use a set of specifications and standards that allow uniform message passing between the entities. The following specifications are the components that are used to implement web services computing paradigm.

- A Transport Protocol
- XML 1.0
- XML Schema Part 1: Structures
- XML Schema Part 2: Datatypes
- SOAP 1.1
- WSDL 1.1
- UDDI ver. 2.04 API

A brief description of each specification follows together with references where a more detailed account can be found.

Transport Protocols

A number of transport protocols maybe used to carry out communication between the client and the service provider. The protocol employed depends on the type of communication. If it is a request/response (like a request for a web page) then HTTP will be used. If it is message passing (like an e-mail) then SMTP may be used. For file transfer, FTP maybe employed.

XML 1.0

XML is a tag-oriented language whose tags can be user defined and are used to describe the data contained in the document. The brief overview of XML in Sidebar 1 will help the reader to understand subsequent examples.

XML Schema Part 1: Structures

XML Schema: Structures can be used to define, describe and catalogue XML vocabularies for classes of XML documents. See [Thompson, et.al., 2001] for more details.

XML Schema Part 2: Datatypes

XML Schema: Datatypes can be used to define datatypes in XML vocabularies and documents. See [Biron and Malhotra, 2001] for more details.

SIDEBAR 1**XML 1.0**

An XML document is made up of elements. Elements are denoted by tags. The tags follow a set of grammar rules. These rules are:

- Each start tag must have a corresponding end tag. (unless the element denoted by the tag does not have any sub elements then the start tag ends with ">" rather than ">").
- Attribute values must be enclosed in quotes.
- Some characters in data must be represented by references
- Tags must be properly nested.
- The document must have the XML prologue - `<?xml version="1.0"?>`.

An example of the use of these rules is shown in Appendix I.

SOAP 1.1

Simple Object Access Protocol (SOAP) is a message protocol that enables requests and/or responses to be sent in XML format from client to a server. SOAP defines an envelope that contains a header and a body. The SOAP body contains the payload. The payload contains a request from the client and then, if necessary, the response to the request from the server.

SOAP messages are a sequence of characters that are embedded in the transport protocol being used in the communication between the client and service provider. The SOAP specification is defined at a high level of abstraction so that any operating system and programming language combination can be used to create SOAP responsive programs. SOAP does not specify what data can be transmitted or what function call can be made. It is simply a transport mechanism that allows for computation to be carried out by a service provider.

To understand SOAP, the reader needs to know about

- SOAP grammar,
- SOAP body,
- SOAP header, and
- SOAP error reporting.

These topics are covered in Appendix II. Readers not familiar with these SOAP concepts should read Appendix II before continuing with the main text.

WSDL 1.1

Web Services Description Language is a specification that details how to describe a web service. A WSDL document for a service is an XML document that contains all the information necessary to connect to that service. Any program that wishes to use a particular Web service will use WSDL for that service to determine how to bind to the service.

An WSDL document is divided into two description areas: (1) abstract and (2) concrete. These are sometimes referred to, respectively, as nonfunctional and functional areas. The functional or concrete area provides the protocol-dependent details the user must follow to access the service being provided. The abstract interface provides an application level service description for the user of the service. This separation of the service description into two areas allows the same service (application level) to be provided by several different implementations (protocol use).

Each of these areas has an associate a set of XML elements. There are four XML elements for the abstract area:

- **<wsdl:types>** This element is used to specify the data types used in the service
- **<wsdl:message>** This element is used to provide an abstract, typed data definition sent to and from services.
- **<wsdl:operation>** This element is used specify the name of the service. The operation specified with the tag is allowed three messages -
 - **Input message** - defines the data the service expects
 - **Output message** - defines the data the service send in response
 - **Fault message** - defines error messages that mat be returned by the service.
- **<wsdl:portType>**. This element is used to list all the operations a particular Web service can provide. A portType is a collection of operations that are supported by the Web service.

The three XML elements for the concrete area:

- **<wsdl:binding>**. - This tag has two functions. (1) it will be a link between abstract elements and the concrete elements since one of its attribute is the name of the portType specified by the portType tag. (2) it provides the address of and protocol used by the Web service.
- **<wsdl:port>** - This element specifies the IP address and the port that offers the service.
- **<wsdl:service>** - This element is used as container for all ports that are specified by a WSDL document.

Another WSDL element is used as the root element of a WSDL document. This element is the definitions tag specified as:

```
<wsdl:definitions name= "some service to be specified"
  list of additional attributes>
</wsdl:definitions>
```

The schematic view of a wsdl document is:

```
<?xml version="1.0"?>
<wsdl:definitions ....>
  <wsdl:types>
    ....
  </wsdl:types>
  <wsdl:message>

  </wsdl:message>
  wsdl:message>

  </wsdl:message>
  wsdl:message>
```

```

</wsdl:message>
.
.
.
wsdl:message>

</wsdl:message>
<portType>
....
</portType>
<binding>
  <operation>
    <input>

    </input>
    <output>

    </output>
  </operation>
</binding>
<service>
  <port>

  </port>
</service>
</definitions>

```

UDDI ver. 2.04 API

Universal Description, Discovery, and Integration is a specification of the registry that lists web services that are of interest to a service requestor entity. It uses taxonomies that categorize web services in a way meaningful to clients.

The UDDI specification is analogous to an automated online "phone directory" of Web services. An entity in need of a particular web service would use a registry's search facility to find the service it needs. The registry encodes three types of information about a service.

1. The "white pages" information which includes the service name and contact details.
2. The "yellow pages" information that categorizes the service based on business and service type. The yellow pages information contain classifications based several accepted taxonomies::
 - NACIS industry categorization,
 - UNSPC project and service categorizations, and
 - ISO-3166-2 geographic taxonomies.

In addition external taxonomies can be used so that niche industries can employ their own special classification codes.

3. "Green pages" information which includes technical data about the service.

Registries can be one of three types: public, private, and semiprivate.

Public: A public UDDI Business Registry is a browser accessible registry accessible by the public.

Private: A private UDDI registry is accessible by entities internal to an organization. This registry will be behind the organization's firewall and only accessible via the organization's intranet. This type of registry of Web services can assist in the development of applications that span the enterprise or reuse software developed in an organization's division but provides a computation that can be used in another division.

Semiprivate: a semiprivate registry is one where customers or perhaps trading partners are allowed through the organization's firewall to access the organization's private registry.

The UDDI entries are structured around fundamental tags and include <businessEntity> and <businessService elements:

1. The <businessEntity> tag includes a UUID (Universal Unique ID) that is assigned to each business entity. The additional elements are used to specify the "white pages" and "yellow pages" information associated with this business entity.
2. The <businessService> tag includes all the information necessary to determine if the service is useful for the client.

AN EXAMPLE

The example is illustrated in Figure 2.

Step 1 - IT department B creates service X. It publishes it using UDDI specifications in a public service registry, using the publish() procedure. This site contains the URL of the WSDL document that gives the specifications for service X.

Step 2 - IT department A is creating information system Y and needs service X. They consult the public registry via the find() procedure. They find IT department B's entry for service X.

Step 3 - IT department A uses the URL posted in the public registry to download a copy of the WSDL specification for service X. Using this specification they build a SOAP client to request service X.

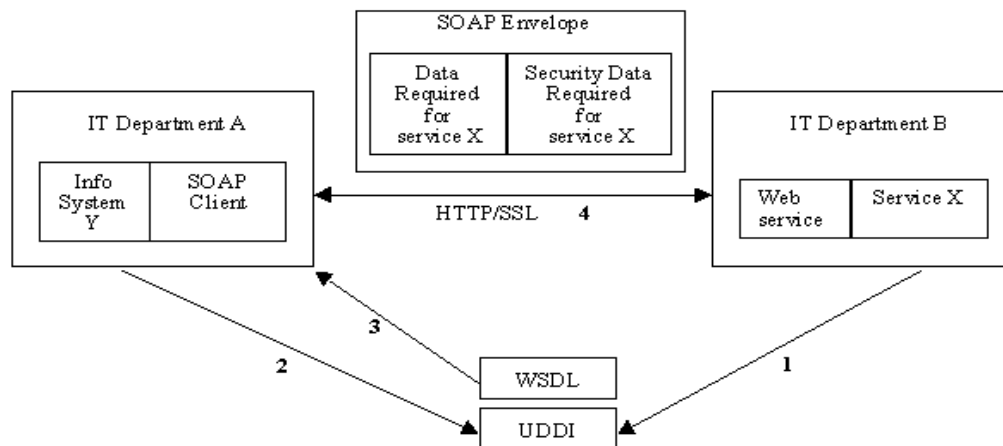


Figure 2. Example

Step 4 - IT department A's SOAP client builds a SOAP envelope that contains the required data for service X and the requisite security data for using service X provided by IT department B.

For a detailed example of how SOAP, WSDL, and UDDI interact the interested see [Cubera , et. al.2002]. Cubera et al.'s article presents an integrated and detailed example that illustrates the dependencies of SOAP, WSDL, and UDDI. In a real business situation, the simple scenario depicted in Figure 2 would require more than three pieces (SOAP, WSDL, & UDDI) of the Web services framework to operate properly. At the very least, an assurance that communications were conducted in a secure environment and that messages were reliably delivered to their destinations would have to be provided.

Currently this kind of security is provided for Internet traffic and distributed computing by technologies such as Secure Multipurpose Internet Mail Extensions (S-MIME), HTTP Secure (HTTPS), Kerberos, X.509 certificates, et.al. Web Services will require different security model in order to assure its security requirements. The reason for this lies in the difference between end-to-end and single-hop usage. Business messages typically originate deep inside one enterprise and go deep inside another. Mechanisms such as Secure Sockets Layer are effective in securing (for confidentiality) a direct connection from one machine to another, but they are of no help if the message has to travel over more than one connection. What is needed is security at the SOAP level.

Researchers are now defining a security model as a set of add-on specifications. For example, the SOAP Security Extensions: Digital Signatures proposal describes how SOAP messages can be digitally signed. Groups are also developing specifications for authentication, confidentiality, and authorization using SOAP. An overview of these techniques and an associated security model are presented in Section III.

III. WEB SERVICES SECURITY

If a firm is going to use web services for its information systems it must be assured that its data will be secure. To understand web services security an overview of information security is provided first. In addition, a glossary is included at the end of this paper for those readers unfamiliar with information security terminology.

INFORMATION SECURITY REQUIREMENTS

The six requirements that define information security are:

- *Confidentiality* - This requirement assures user privacy and prevents the theft of information both in transit and from storage. Symmetric and asymmetric encryption are used to create cipher text for transmission to and from clients and servers and for information held in storage.
- *Integrity* - This requirement assures that information either in transit or in storage was not modified, either intentionally or unintentionally. An encrypted message digest - a digital signature - assures message integrity.
- *Non-repudiation* - This requirement assures that the sender of a message cannot legitimately claim they did not send the message. Digital Certificates and Public Key Infrastructure (PKI) are used to assure non-repudiation.
- *Authentication* - This requirement assures that the sender and receiver are who they claim to be. PKI as well as smartcards and user name/password authentication methods can be used to assure authentication.
- *Authorization* - This requirement assures that an authentication entity can access only those information resources they must have either to request or provide a service. Once authenticated an entity authorization will be determined. Generally an authenticated entity has an associated access control list (ACL)

Availability - This requirement assures that uninterrupted service is provided to authenticated and authorized users. In addition, interruptions of service by denial-of-service attacks are controlled.

Currently SSL (Secure Sockets Layer), PKI (Public Key Infrastructure), and firewalls are able to meet these requirements for conventional web traffic using HTTP [Boncella, 2000 and Boncella, 2003]. However SSL and firewalls are inadequate to assure these requirements for web services.

SSL AND Web Services Security

Figure 3 illustrates the case where a web service is provided indirectly to the user. From a security view, two sets of information security requirements need to be assured. These are referred to as security contexts.

1. The security context that assures information security from the user to the web site,
2. The security context assures information security from the user to the web service provider. This context is referred to as *persistent security*.

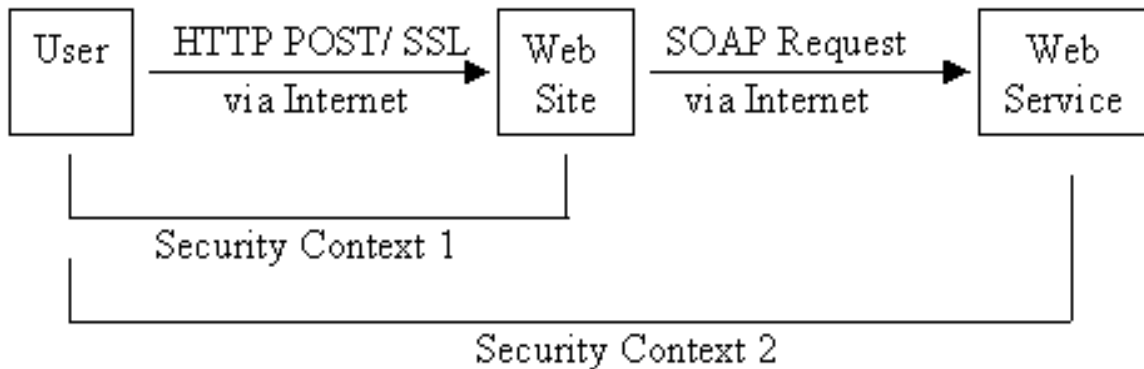


Figure 3. Indirect Web Service

Persistent security requires the security of the SOAP request/response message be assured over more than one client/server connection. The security of the SOAP message extends beyond the first request/response interaction. SSL is inadequate for the persistent security requirement

SSL was not designed to handle this type of process. SSL does encrypt the data stream but it does not provide end-to-end confidentiality. In particular SSL leaves the data exposed between the web site and the application providing the service.

In addition SSL will not be able to assure integrity since it does not provide digital signatures.

Most often, implementations of SSL only ensures one-way authentication - the user is assured of the provider but the provider is not assured of the user. In Figure 3 it is possible that the user is not assured of the service provider since it is an indirect service provider. If authentication cannot be met, the authorization will be suspect.

Finally SSL does not support an end-to-end audit trail from service request to service response.

Firewalls and Web Services Security

The function of a firewall is to restrict the flow of data packets into and out of a computer network. Firewalls perform this function at one or more layers of the OSI model¹.

Firewalls function at several layers of the OSI model.

- A Layer 3 firewall filters packets on the basis of their IP address and/or port addresses.
- A Layer 4 firewall, sometimes call a circuit-level firewall, uses TCP handshaking to determine a session's legitimacy and to prevent session hijacking.
- An application layer firewall filters on basis of application being requested. For example, HTTP/SSL and/or POP/SMTP maybe allowed others refused.

Most SOAP messages are bound to HTTP or SMTP. In most computing systems messages destined for web service (HTTP) and e-mail service(SMTP) are not filtered but allowed to pass into and out of the computing systems. Thus, in practice, SOAP messages bypass firewalls.

To be effective a SOAP level firewall should determine:

- Whether the incoming SOAP request is intended for an available Web Service,
- whether the SOAP request is valid, and
- that the SOAP message contains valid data (edit checks on its type and size).

In essence, to provide firewall-type security for SOAP messages this firewall must be a content filtering firewall

WEB SERVICES SECURITY REQUIREMENTS

The requirements for information security (such as confidentiality and integrity) remain the same for web services. Because of the additional requirement of persistent security, the means by which web services security requirements are assured differs from SSL and firewalls. To assure persistent security, SOAP messages must include information about the message's security requirements.

WEB SERVICES SECURITY TECHNOLOGY

Since SOAP technology is built on XML, a reasonable approach to Web Services would be to assure the information security requirements for an XML message. This section presents an overview of the information security technologies available to assure the security of an XML message,

Confidentiality for Web Services

XML Encryption, a specification produced by the World Wide Web Consortium (W3C), is used to encrypt portions of XML documents. XML Encryption assures confidentiality in the case of any security context beyond a simple HTTP/SSL connection (Figure 2). In the case of a security context ranging over several SOAP intermediaries, portions of the SOAP document can be kept

¹ See CAIS Volume 4, Article 11 [Boncella, 2000] for an overview of the OSI and its layers and their functions.

confidential from any SOAP intermediary en route as the message makes its way from the user to the web service provider and back. Reagle [2001] provides detailed information.

Integrity for Web Services

XML signature is a specification produced jointly by W3C and the Internet Engineering Task Force (IETF). An XML signature is the XML equivalent of a digital signature. It can be used to digitally sign selected portions of an XML document. In particular it can be used to sign data and thereby assure its integrity. An XML signature is used within SOAP messages. Eastlake and Reagle [2001] provide detailed information.

Authentication and Authorization Web Services

A web services user can request services from any number of different service providers. That user should be authenticated on each service provider's system. This authentication can then be used to determine the resources that a user is authorized to access on a particular service provider's system.

In a sequence of requests for service, rather than prompting a user for authentication each time a request for service is made it is desirable to provide a single sign on (SSO) process. In SSO, when the initial web service provider authenticates a user, any subsequent requests generated by that user to other service provider's systems is automatically authenticated on that system and the user's authorization is determined as well.

The two approaches to implementing SSO are:

1. include authentication information for each web service in the initial SOAP message, and
2. maintain a user's authentication list in a central repository.

Security Assertions Markup Language (SAML) and XML Access Control Markup Language (XACML) can work together to implement the first approach. SAML information can be inserted into SOAP messages. This information is about user authentication and authorization as well as information about the user. XACML express access control rules in XML format. For detailed information about SAML see [OASIS, 2004] and for XACML see [OASIS, 2003]. Microsoft's Passport scheme and Sun's Liberty Alliance Project use the centralized repository approach to user authentication.

Nonrepudiation - PKI for Web Services

XML Key Management specification (XKMS) provides PKI services (registering, locating, and validating keys) through XML. This service is provided as SOAP- based web service. Ford, et. al., [2001] give detailed information.

WS-SECURITY

The preceding technology is used to assure the information security requirements for XML messages. What is needed to guarantee information security for Web Services is technology that assures the information security requirements for SOAP messages. The WS-Security specification is designed to assure the security of SOAP messages.

WS-Security is a specification that extends the SOAP specification. The WS-Security specification provides for a number of features. Among these are:

- the use of multiple security tokens for authentication and authorization,
- the use of multiple trust models, multiple signature formats, and multiple encryption technologies, and

- satisfying the requirement of persistent security for SOAP messages - end to end message-level security.

What follows is an overview of this specification and its current status of adoption. Atkinson, et.al.[2002] provide detailed information.

WS-Security Architecture

The WS-Security model is an abstraction of the security services required for assuring the security requirements of SOAP messages. This model separates the functional requirements from their implementation. As a result, different information security technologies can be used within a SOAP message to assure its security. The purpose of the model is to allow mixing of information security technologies that provide the same functionality within the same SOAP message. For example if one web service provider requires a Kerberos ticket for authentication to use its service, the SOAP message can specify the required ticket, If that same SOAP message needs to use the services of another provider that requires X.509 certificate the certificate can also be provided in the SOAP message.

Figure 4 is a diagram of the WS-Security architecture. In April 2004 OASIS (Organization for the Advancement of Structured Information Standards) approved this WS-Security specification. IBM, MICROSOFT, and VeriSign Inc. initially laid out this architecture in April of 2002. Sun Microsystems and many other companies joined the WS-Security effort after Microsoft, IBM and VeriSign submitted the specification to OASIS in June 2002. OASIS formed a WS-Security Technical Committee to develop this specification into a standard.

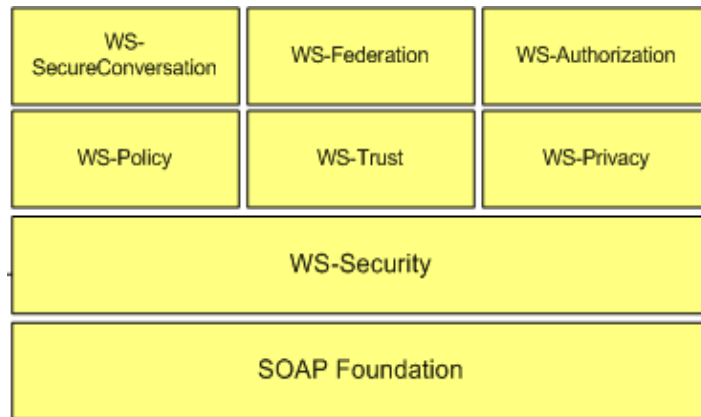


Figure 4. IBM/Microsoft Architecture(Proposed April 2002)

The WS-Security model defines a process by a Web Service that can require a requestor to provide a set of claims (such as authentication, encryption keys, and authorizations.) This arrangement is considered the Web Services security policy. The requestor of the service can provide these claims along with a security token that validates the claims to the Web Service provider. If the requestor does not satisfy the policy requirements of the Web Service provider, the requestor may attempt to acquire the necessary components of the security policy from other Web Service providers.

WS-Security

The WS-Security specification describes the extensions of SOAP messaging that allow for message integrity and message confidentiality. Specifically WS-Security describes how to attach

signature and encryption headers to SOAP messages. It also describes how to attach security tokens to messages. This specification will work with multiple security technologies including PKI, Kerberos, SAML, Basic/Digest, and SSL.

WS-Policy

The WS-Policy specification describes how web services providers may specify the security technology required for an application using their services. The security technology specified would be required security tokens, supported encryption algorithms, privacy rules (which parameters must be encrypted), and digital signature. This information will be in the WSDL document for a service.

WS-Trust

WS-Trust specifies a framework for trust models that enables Web services to interoperate securely. Trust is concerned with assuring both parties engaged in encrypted communication that the key being used for encryption was not compromised. That is, the party using the key is indeed the legitimate owner of the key.

WS-Privacy

The WS-Privacy specification describes how Web service providers and requestors will state their privacy preferences and organizational privacy practice statements. It allows a web service provider to specify its privacy policy in a structured manner and post it on the Web server that is being accessed. The requestor can read this policy and compare it to their privacy preferences. This specification allows for privacy policy exchange and agreement for Web services.

WS-SecureConversation

The WS-Secure Conversation specification describes how to manage and authenticate message exchanges between parties including security context exchange and establishing and deriving session keys. In particular, this specification allows a web service provider and a requestor to establish an agreed-upon security context. WS-SecureConversation may be described as "SSL at the SOAP" level.

WS-Federation

This specification describes how to manage and broker the trust relationships in a heterogeneous federated environment including support for federated identities. An example of this type of federation would be one where a requestor is authenticated by a party who understands X.509 certificates but is allowed to use a web service provided by a party who requires a Kerberos ticket. Both parties recognize each other's authenticated users. This specification provides mechanisms for managing trust relationships between parties.

WS-Authorization:

This specification describes how to manage authorization data and authorization policies. It overlaps with XACML.

Extended Example of WS-Security

An extended example of how the WS-Security specification is used in SOAP headers can be found in Section Five of [Atkinson, B, et. al., 2002]. Atkinson et al's example illustrates the use of a variety in information security technologies that will assure the security of a SOAP message as it passes through multiple Web service providers.

STATUS OF IBM/MICROSOFT ARCHITECTURE

Not all pieces of the IBM/Microsoft architecture are available in September 2004. Table 1 is based upon [MSDN 2004]

Table 1. Status of the Web services Security Model Components

Security Model Component	Status
Web Services	Kerberos Binding - published as a public specification on 19 December 2003. SOAP Message Security was published as an OASIS Standard in March of 2004. UsernameToken Profile 1.0 was published as an OASIS Standard in March of 2004. X.509 Certificate Token Profile was published as an OASIS Standard in March of 2004.
WS-Trust	published as a public specification on 24 May 2004.
WS-SecureConversation	published as a public specification on 24 May 2004.
WS-SecurityPolicy	published as a public specification on 18 December 2002.
WS-Federation	published as public specifications on 8 July 2003

WEB SERVICES SECURITY THREATS

Even if the WS-Security model is adopted in full, potential security threats are introduced by the use of web services. One is the possibility of a SOAP message containing malicious data that would cause the web service application to execute in an unintended mode.

Another is the SOAP message could contain a request for a service that is not advertised as being provided on that site. The unadvertised service could compromise the service provider. SOAP messages easily pass through firewalls.

What is needed are firewalls that filter the content of SOAP messages requesting passage through the firewall. Albrecht [2004] contains more details on this vulnerability.

A final security threat is a denial of service attack on the application or computer system that provides the web service.

IV. SUMMARY

The purpose of this tutorial is to provide a foundation for understanding the need for and techniques of web services security. In addition to describing web services security efforts, the tutorial presents an overview of the architecture of WS-Security and its status and the architecture and components of web services.

Editor's Note: This web services security tutorial is an extended version of the tutorial presented at AMCIS 2004 in New York in August. The article was received on September 9, 2004 and was published on October 1, 2004.

REFERENCES

EDITOR'S NOTE: The following reference list contains the address of World Wide Web pages. Readers who have the ability to access the Web directly from their computer or are reading the

paper on the Web, can gain direct access to these references. Readers are warned, however, that

1. these links existed as of the date of publication but are not guaranteed to be working thereafter.
2. the contents of Web pages may change over time. Where version information is provided in the References, different versions may not contain the information or the conclusions referenced.
3. the authors of the Web pages, not CAIS, are responsible for the accuracy of their content.
4. the author of this article, not CAIS, is responsible for the accuracy of the URL and version information.

- Albrecht, C. (2004) How Clean Is the Future of SOAP?, *Communication of the ACM*, (47)2, February.
- Atkinson, B., Della-Libera, G., Hada, S., Hondo, M., Hallam-Baker, P., Klein, J., LaMacchia, B., Leach, P., Manferdelli, J., Maruyama, H., Nadalin, A., Nagaratnam, N., Prafullchandra, H., Shewchuk, J., Simon, D. (2002) "Web Services Security (WS-Security)", <http://www-106.ibm.com/developerworks/webservices/library/ws-secure/> (current Feb. 22, 2004)
- Biron, P.V. and Malhotra, A. (2001) "XML Schema Part 2: Datatypes" <http://www.w3.org/TR/xmlschema-2/> (current Feb. 22, 2004)
- Boncella, R. (2000) "Web Security for E-Commerce", *Communications of the AIS*, 4, 11, November
- Boncella, R. (2003) *SSL in The Internet Encyclopedia*, Hossein Bidgoli (Editor), New York, New York: J. Wiley
- Christensen, E., Curbera, F., Meredith, G., Weerawarana, S. (2001) "Web Services Description Language (WSDL) 1.1", <http://www.w3.org/TR/wsdl> (current Feb. 22, 2004)
- Curbera, Francisco Duftler, Matthew Khalaf, Rania Nagy, William Mukhi, Nirmal and Weerawarana, Sanjiva (2002) "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI", *IEEE Internet Computing Online*, March–April <http://www.computer.org/internet/v6n2/w2spot.htm> (current September 18, 2004).
- Fallside, D. (2001). "XML Schema Part 0: Primer", <http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/> (current September 6, 2004).
- Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999) "Hypertext Transfer Protocol HTTP/1.1", <http://www.ietf.org/rfc/rfc2616.txt> (current Feb. 22, 2004).
- Eastlake, D., and Reagle, J (2001) "XML Signature", <http://www.w3.org/Signature/> (current Feb. 22, 2004)
- Ford, W., Hallam-Baker, P., Fox, B., Dillaway, B., LaMacchia, B., Epstein, J., Lapp, J., (2001) "XML Key Management Specification (XKMS)", <http://www.w3.org/TR/xkms/> (current Feb. 22, 2004)
- Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., Nielsen H. F. (2003) "SOAP Version 1.2 Part 1: Messaging Framework", <http://www.w3.org/TR/soap/> (current Feb. 22, 2004)
- Kristol, D, and Montulli, L. (2000), "HTTP State Management Mechanism", <http://www.ietf.org/rfc/rfc2965.txt> (current Feb. 22, 2004)
- Mitra, N. (2003) "SOAP Version 1.2 Part 0: Primer", <http://www.w3.org/TR/soap12-part0/>, (current September 6, 2004)
- MSDN (2004) "Web Services Security Specifications Index Page", <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/wssecurspecindex.asp>, (current September 6, 2004)
- OASIS (2001) (Organization for the Advancement of Structured Information Standards), "Universal Description, Discovery and Integration", <http://www.uddi.org/> (current Feb. 22, 2004)

- OASIS (2003), (Organization for the Advancement of Structured Information Standards), "eXtensible Access Control Markup Language", http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml. (current Feb. 22, 2004)
- OASIS (2004), (Organization for the Advancement of Structured Information Standards), "Security Assertion Markup Language (SAML)", http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security (current Feb. 22, 2004)
- Reagle, J. (2001) " XML Encryption", <http://www.w3.org/Encryption/2001/> (current Feb. 22, 2004)
- Thompson, H.S.,Beech, D., Maloney, M., Mendelsohn N. (2001) " XML Schema Part 1: Structures", <http://www.w3.org/TR/xmlschema-1/> (current Feb. 22, 2004)
- W3C (1996) "Extensible Markup Language (XML)", <http://www.w3.org/XML/> (current Feb. 22, 2004)

APPENDIX I. EXAMPLE OF XML RULES

This appendix contains an example illustrates the use of XML rules. This example and its subsequent annotation is based on [Fallside,2001].

```
<?xml version="1.0"?>
<purchaseOrder orderDate="2004-9-20">
  <shipTo country="US">
    <name>Bob Boncella</name>
    <street>723 Walnut Street</street>
    <city>Lawrence</city>
    <state>KS</state>
    <zip>66045</zip>
  </shipTo>
  <billTo country="US">
    <name>Robert J. Boncella</name>
    <street>723 Walnut Street</street>
    <city>Lawrence</city>
    <state>KS</state>
    <zip>66045</zip>
  </billTo>
  <comment>Shipping Address is the same as billing address</comment>
  <items>
    <item partNum="945-785">
      <productName>Sealife Digital Camera</productName>
      <quantity>1</quantity>
      <USPrice>249.95</USPrice>
      <comment> Model DC200</comment>
    </item>
    <item partNum="945-786">
      <productName>Sealife Digital Underwater Housing</productName>
      <quantity>1</quantity>
      <USPrice>125.95</USPrice>
      <shipDate>1999-05-21</shipDate>
    </item>
  </items>
</purchaseOrder>
```

The purchase order consists of a main element, **purchaseOrder**, and the sub-elements **shipTo**, **billTo**, **comment**, and **items**. These sub-elements (except comment) in turn contain other sub-elements, and so on, until a sub-element such as **USPrice** contains a number rather than any subelements. Elements that contain sub-elements or carry attributes are said to have complex types, whereas elements that contain numbers (and strings, and dates, etc.) but do not contain

any sub-elements are said to have simple types. Some elements have attributes; attributes always have simple types.

The complex types in the instance document, and some of the simple types, are defined in the schema for purchase orders. The other simple types are defined as part of XML Schema's repertoire of built-in simple types.

APPENDIX II. DETAILS ABOUT SOAP

SOAP GRAMMAR

A SOAP message consists of three parts: an envelope. Within the envelope are an optional header and a required body. Each of these is delimited with tags. These are listed below.

```
<SOAP _ ENV:Envelope> - Start envelope tag
  <SOAP _ ENV:Header> - Start header tag
    Header Information
  </SOAP _ ENV:Header> - End header tag
  <SOAP _ ENV:Body> - Start body tag
    SOAP Message
  </SOAP _ ENV:Body> - End body tag
</SOAP _ ENV:Envelope> - End envelope tag
```

SOAP BODY

The SOAP body or payload may contain a variety of messages. Normally the payload will be a method (procedure) call to a remote service provider. This message will include the URL of the service provider as well as the parameters required for the method. Occasionally the payload will contain an XML document that is being transferred. Or it may be a response message from a service provider.

The format of the body is determined by whoever created the Web Service. This format is specified by a Web Services Description Document (WSDL) provided by the service provider.

SOAP HEADER

The header is optional in a SOAP message. However it is present it will be the first element in the SOAP envelope. Since the header is not define in the SOAP specification clients and services are free to define it for their use. Most often the header will be used to authenticate the client by enclosing a username and password with the header. The example below illustrates this type of use.

```
<SOAP _ ENV:Header>
  <rjb:authentication xmlns:rjb="http://www.washburn.edu/boncella/auth"
    SOAP-ENV:mustUnderstand="1">
    <loginID>
      zzbonc
    </loginID>
    <password>
      IAMME
    </password>
  </rjb:authentication>
</SOAP _ ENV:Header>
```

The **SOAP-ENV:mustUnderstand** attribute in the above example is used to determine if the service provider can handle the fields in the header. In this case setting the attribute to the value of 1 requires the service provider to handle the authentication process specified in the

<rjb:Authentication> tag. If the service provider does not handle this field then the service provider will report an error to the client.

Another attribute of the header tag is the **SOAP-ENV:actor** or **SOAP-ENV:role** attribute. This attribute is specified in SOAP 1.1. In SOAP 1.2 this attribute is called "role". Regardless of its name the attribute allows the designer of the message to specify who will be the service provider of various parts of the SOAP message body. The message body may specify a complex computation where different service providers must perform the components of the computation. For example in the case where a salesperson has made a sale to a client the payroll department has the procedure to compute the sales commission, the accounts receivable has the procedures to create a billing, and the shipping department has the procedures to send the merchandise. Each of these sets of procedures plays a role (or is an actor) in the transaction of this sale.

SOAP ERROR REPORTING

In addition to the envelope, body, and header tags, the SOAP specification includes tags that allow the service provider to report errors to the client. This is the **fault** tag and its options. The **SOAP-ENV:fault** has four options. These are:

SOAP-ENV:faultcode - a required element that will provide a code indicating the problem. The faultcode tag has four generic faultcodes. These are:

server - These errors indicate An error occurred with the service provider not in the message

client - These errors indicate something is wrong with the message.

versionMismatch - SOAP versions are different between the client and its service providers.

mustUnderstand - error generated when header element cannot be processed by service provider but it is required to be processed.

SOAP-ENV:faultstring - human readable form of the faultcode indicating location and type of error

SOAP-ENV:faultactor - if a chain of services is requested this indicates what service caused the fault.

SOAP-ENV:detail - this will contain values of parameters at the time of the failure.

An example of a fault tag follows.

```
<SOAP-ENV:fault>
  <SOAP-ENV:faultcode>
    Client.Authentication
  </SOAP-ENV:faultcode>
  <SOAP-ENV:faultstring>
    Authentication Failure - loginID unknown
  </SOAP-ENV:faultstring>
  <SOAP-ENV:faultactor>
    http://www.washburn.edu/boncella/auth
  </SOAP-ENV:faultactor>
  <SOAP-ENV:detail>
    <loginID>
      zzbonc
    </loginID>
```

</SOAP-ENV:detail>
</SOAP-ENV:fault>

GLOSSARY OF TERMS

Firewall	An internetwork gateway that restricts data communication traffic to and from one of the connected networks (the one said to be "inside" the firewall) and thus protects that network's system resources against threats from the other network (the one that is said to be "outside" the firewall).																				
Https	When used in the first part of a URL (the part that precedes the colon and specifies an access scheme or protocol), this term specifies the use of HTTP enhanced by a security mechanism, which is usually SSL																				
Kerberos	A system developed at the Massachusetts Institute of Technology that depends on passwords and symmetric cryptography (DES) to implement ticket-based, peer entity authentication service and access control service distributed in a client-server network environment.																				
Public-key Infrastructure (PKI)	A system of CAs (and, optionally, RAs and other supporting servers and agents) that perform some set of certificate management, archive management, key management, and token management functions for a community of users in an application of asymmetric cryptography.																				
Secure/MIME (S/MIME)	Secure/Multipurpose Internet Mail Extensions, an Internet protocol [R2633] to provide encryption and digital signatures for Internet mail messages.																				
Secure Sockets Layer (SSL)	An Internet protocol (originally developed by Netscape Communications, Inc.) that uses connection-oriented end-to-end encryption to provide data confidentiality service and data integrity service for traffic between a client (often a web browser) and a server, and that can optionally provide peer entity authentication between the client and the server.																				
X.509 public-key certificate	<p>A public-key certificate in one of the formats defined by X.509--version 1 (v1), version 2 (v2), or version 3 (v3). (The v1 and v2 designations for an X.509 public-key certificate are disjoint from the v1 and v2 designations for an X.509 CRL, and from the v1 designation for an X.509 attribute certificate.)</p> <p>An X.509 public-key certificate contains a sequence of data items and has a digital signature computed on that sequence. In addition to the signature, all three versions contain items 1 through 7 listed below. Only v2 and v3 certificates may also contain items 8 and 9, and only v3 may contain item 10.</p> <table border="0"> <tr> <td>1. version</td> <td>Identifies v1, v2, or v3.</td> </tr> <tr> <td>2. serialNumber</td> <td>Certificate serial number; an integer assigned by the issuer.</td> </tr> <tr> <td>3. signature</td> <td>OID of algorithm that was used to sign the certificate.</td> </tr> <tr> <td>4. issuer</td> <td>DN of the issuer (the CA who signed).</td> </tr> <tr> <td>5. validity</td> <td>Validity period; a pair of UTCTime values: "not before" and "not after".</td> </tr> <tr> <td>6. subject</td> <td>DN of entity who owns the public key.</td> </tr> <tr> <td>7. subjectPublicKeyInfo</td> <td>Public key value and algorithm OID.</td> </tr> <tr> <td>8. issuerUniquelIdentifier</td> <td>Defined for v2, v3; optional.</td> </tr> <tr> <td>9. subjectUniquelIdentifier</td> <td>Defined for v2, v2; optional.</td> </tr> <tr> <td>10. extensions</td> <td>Defined only for v3; optional</td> </tr> </table>	1. version	Identifies v1, v2, or v3.	2. serialNumber	Certificate serial number; an integer assigned by the issuer.	3. signature	OID of algorithm that was used to sign the certificate.	4. issuer	DN of the issuer (the CA who signed).	5. validity	Validity period; a pair of UTCTime values: "not before" and "not after".	6. subject	DN of entity who owns the public key.	7. subjectPublicKeyInfo	Public key value and algorithm OID.	8. issuerUniquelIdentifier	Defined for v2, v3; optional.	9. subjectUniquelIdentifier	Defined for v2, v2; optional.	10. extensions	Defined only for v3; optional
1. version	Identifies v1, v2, or v3.																				
2. serialNumber	Certificate serial number; an integer assigned by the issuer.																				
3. signature	OID of algorithm that was used to sign the certificate.																				
4. issuer	DN of the issuer (the CA who signed).																				
5. validity	Validity period; a pair of UTCTime values: "not before" and "not after".																				
6. subject	DN of entity who owns the public key.																				
7. subjectPublicKeyInfo	Public key value and algorithm OID.																				
8. issuerUniquelIdentifier	Defined for v2, v3; optional.																				
9. subjectUniquelIdentifier	Defined for v2, v2; optional.																				
10. extensions	Defined only for v3; optional																				

In addition to the elements of the Glossary two links that will aid inquires about Internet security and Web Services terms and concepts.

Internet Security Glossary: <http://www.faqs.org/rfcs/rfc2828.html>

Web Services Glossary: <http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/>

ABOUT THE AUTHOR

Robert J. Boncella is Professor of Computer Information Science at Washburn University, Topeka, KS. Dr. Boncella holds a joint appointment in the Computer Information Sciences Department, where he conducts classes in Data Communications and Computer Networks, and in the School of Business, where he offers instruction on Computer Based Information Systems in the school's MBA program. He holds a Ph.D. and Masters degrees in Computer Science from the University of Kansas and a Master of Arts in Philosophy from The Cleveland State University. He is a member of ACM, AIS, AAAI, and IEEE. His current areas of interest are web based information systems, intelligent agents, and decision making under Uncertainty, and computer security and privacy.

Copyright © 2004 by the Association for Information Systems. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation on the first page. Copyright for components of this work owned by others than the Association for Information Systems must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or fee. Request permission to publish from: AIS Administrative Office, P.O. Box 2712 Atlanta, GA, 30301-2712 Attn: Reprints or via e-mail from ais@aisnet.org



Communications of the Association for Information Systems

ISSN: 1529-3181

EDITOR-IN-CHIEF

Paul Gray

Claremont Graduate University

AIS SENIOR EDITORIAL BOARD

Detmar Straub Vice President Publications Georgia State University	Paul Gray Editor, CAIS Claremont Graduate University	Sirkka Jarvenpaa Editor, JAIS University of Texas at Austin
Edward A. Stohr Editor-at-Large Stevens Inst. of Technology	Blake Ives Editor, Electronic Publications University of Houston	Reagan Ramsower Editor, ISWorld Net Baylor University

CAIS ADVISORY BOARD

Gordon Davis University of Minnesota	Ken Kraemer Univ. of Calif. at Irvine	M.Lynne Markus Bentley College	Richard Mason Southern Methodist Univ.
Jay Nunamaker University of Arizona	Henk Sol Delft University	Ralph Sprague University of Hawaii	Hugh J. Watson University of Georgia

CAIS SENIOR EDITORS

Steve Alter U. of San Francisco	Chris Holland Manchester Bus. School	Jaak Jurison Fordham University	Jerry Luftman Stevens Inst. of Technology
------------------------------------	---	------------------------------------	--

CAIS EDITORIAL BOARD

Tung Bui University of Hawaii	Fred Davis U. of Arkansas, Fayetteville	Candace Deans University of Richmond	Donna Dufner U. of Nebraska -Omaha
Omar El Sawy Univ. of Southern Calif.	Ali Farhoomand University of Hong Kong	Jane Fedorowicz Bentley College	Brent Gallupe Queens University
Robert L. Glass Computing Trends	Sy Goodman Ga. Inst. of Technology	Joze Gricar University of Maribor	Ake Gronlund University of Umea,
Ruth Guthrie California State Univ.	Alan Hevner Univ. of South Florida	Juhani Iivari Univ. of Oulu	Claudia Loebbecke University of Cologne
Munir Mandviwalla Temple University	Sal March Vanderbilt University	Don McCubbrey University of Denver	Emmanuel Monod University of Nantes
John Mooney Pepperdine University	Michael Myers University of Auckland	Seev Neumann Tel Aviv University	Dan Power University of No. Iowa
Ram Ramesh SUNY-Buffalo	Maung Sein Agder University College,	Carol Saunders Univ. of Central Florida	Peter Seddon University of Melbourne
Thompson Teo National U. of Singapore	Doug Vogel City Univ. of Hong Kong	Rolf Wigand Uof Arkansas, Little Rock	Upkar Varshney Georgia State Univ.
Vance Wilson U. Wisconsin, Milwaukee	Peter Wolcott Univ. of Nebraska-Omaha		

DEPARTMENTS

Global Diffusion of the Internet. Editors: Peter Wolcott and Sy Goodman	Information Technology and Systems. Editors: Alan Hevner and Sal March
Papers in French Editor: Emmanuel Monod	Information Systems and Healthcare Editor: Vance Wilson

ADMINISTRATIVE PERSONNEL

Eph McLean AIS, Executive Director Georgia State University	Samantha Spears Subscriptions Manager Georgia State University	Reagan Ramsower Publisher, CAIS Baylor University
---	--	---