

# Web Services - Common Threats and Resolutions

Ameet Paranjape and Kevin Driver

## Abstract

This paper identifies common security threats and vulnerabilities in the current web services implementation, various protocols and solutions to fix these problems, issues the current protocols do not fix, and suggestions to remedy them or to provide additional improvements.

## Introduction

Web services are applications that exchange information over a network. More specifically, web services can be defined as a technology for publishing, identifying, and calling services in a network of interacting computer nodes using eXtensible Markup Language (XML) applications in order to map to programs, objects, or databases [Holgersson, 1; Newcomer, 2].

There is a basic set of XML-based standards and technologies used to transport and transform data, enabling web services information exchange. They are XML, Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description, Discovery, and Integration (UDDI). XML, the basic foundation on which web services are built, provides a language for defining data and how to process it. WSDL, an XML based technology, defines web services interfaces, data and message types, interaction patterns, and protocol mappings. SOAP, a collection of XML-based technologies, defines a process for web services communication and provides a serialization format for transporting XML documents over a network. UDDI is a web services registry and discovery mechanism used for storing and categorizing information and retrieving pointers to web services interfaces [Newcomer, 16].

A web services environment contains at least one provider and one consumer; in addition there may be a registry displaying the web service and providing information to potential consumers. Figure 1 shows the basic interactions between the web service elements and the communication standard used.

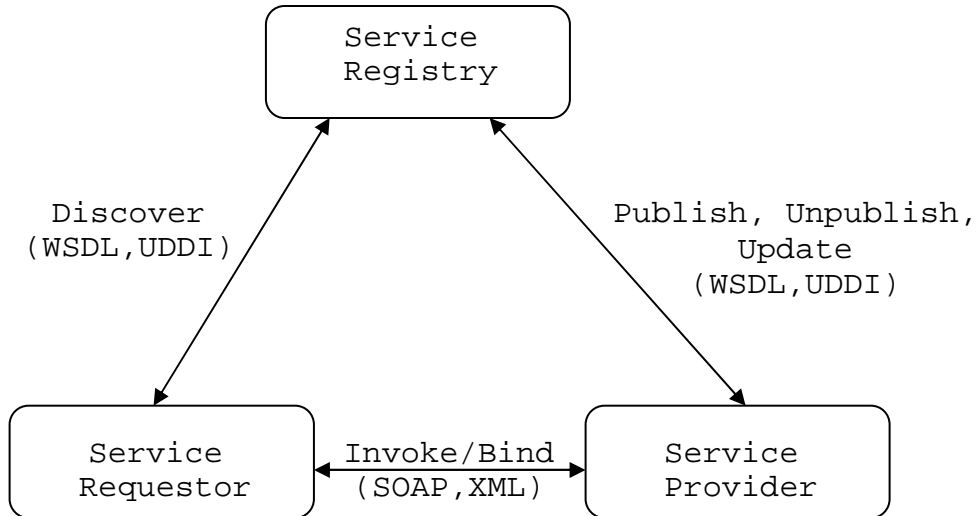


Figure 1: Web Service roles and operations [Holgersson, 2]

As shown in Figure 1, the service provider publishes a web service in a web services registry using WSDL and UDDI. A web service requestor, the node that wants to use the web service, searches the registry also using WSDL and UDDI. If a suitable web service is found, the requestor binds to it using SOAP and XML.

By design the web services infrastructure is meant to easily share information. The system works on a common set of standards, i.e. XML, SOAP, to communicate and as shown in the example above; the web service openly displays services and explicitly describes how to communicate with itself. This "openness," while beneficial for discoverability and interoperability, also makes web services incredibly vulnerable to attack.

### **Fundamental Web Services Security Requirements**

Now that some background on the Web Services technology has been established, it is appropriate to discuss fundamental security

requirements on this technology. In Applied Cryptography, Schneier gives four basic requirements of cryptography (including encryption and digital signatures): confidentiality, authentication, integrity, and nonrepudiation [Schneier, 2]. For use with Web Services, others add integrity and authorization to the list of security necessities [Holgersson, 3; Pfleeger, 5]. How can these high-level ideals be extrapolated into a practical description of fundamental web services requirements?

- Transport Security (confidentiality, authentication)
- Persistent Security (confidentiality, authentication)
- Unauthorized Access Protection
- Integrity Assurance; Freedom from Tampering/Corruption
- Commitment/Nonrepudiation

*Transport Security* addresses the need for securing information exchanged whilst in transit physically over the wire. During this process it is important for the confidentiality and authentication to be layered so only those with appropriate access can retrieve the data. In the context of web services, as the data passes through many "nodes" for processing, only data appropriate for that specific node will be accessed. Holgersson and Soderstrom affirm this need for a solution different from the traditional transport layer security such as TLS and SSL [Holgersson, 4].

*Persistent Security* calls for the securing of the XML documents that make up web services data exchange when they are kept in persistent storage before and/or after traversing the network. The same requirements of layering as described in transport security are desired here as well.

*Unauthorized Access Protection* alludes to the use of public key infrastructure (PKI) technologies, and indeed this is how such protection is accomplished. The use of tokens as well as public/private key pairs will come into play here.

*Integrity Assurance; Freedom from Tampering/Corruption* is accomplished through the use of digital signatures. These qualities

guarantee that information arrives in the same condition in which it was sent.

*Commitment/Nonrepudiation* is also innate in the use of digital signatures. The ability for parties to back out of or deny transactions and/or messages is highly undesirable.

This first-generation set of requirements serves to ensure the usefulness of web services as a technology in providing basic assurance and security. Just how this is accomplished is examined next.

### **Fundamental Web Services Security Requirements Addressed**

After their inception, web services were quickly challenged to stack up to the fundamentals of information security as annotated above. In examining each one, key technologies employed in web services today are defined and should be highlighted. The WS-Security specification presents several of these [3]. Examining the building blocks of the WS-Security specification is useful in understanding how web services are built and communicate.

The primary technologies of WS-Security are XML Signature and XML Encryption [1][2]. According to the working group document, the XML Signature specification was designed for "representing the signature of Web resources and portions of protocol messages (anything referencable by a URI) and procedures for computing and verifying such signatures" [1]. Restated, this standard provides for the signing and verifying of (XML) documents, whole or in part. Signatures are used to determine if a message arrived exactly as it was when the signature was generated, revealing the identity of the signer, and demonstrating that the message was from him or her (assuming the signing key is secure). As Holgersson and Soderstrom state, this implies integrity, authentication, as well as non-repudiation. This specification also provides for the transport and persistent cases listed above. The XML Encryption specification states a corresponding goal: "to develop a process for encrypting/decrypting digital content (including XML documents and portions thereof) and an XML syntax used to represent the (1) encrypted content and (2) information that enables an intended recipient to decrypt it" [2]. This standard provides for encrypting

and decrypting (XML) documents, whole or in part, as well as specifying the means to do so; confidentiality of the document and/or its parts is established here as well [5]. Similar to XML Signatures, XML Encryption also handles the transport and persistent storage cases. These two technologies coupled with WS-Security tokens for supplemental authorization round out the WS-Security specification [3]. The WS-Security tokens provide protection against unauthorized access by providing some low-level identity within the system as a "trusted party."

The specification document outlines how to utilize these components in the headers of (and throughout) XML documents to achieve the fundamental security goals as they have been stated in the previous section. Clearly, the fundamental requirements have been addressed thoroughly by the WS-Security specification.

### **Fine-Grained Web Services Security Requirements**

The fundamental security requirements discussed earlier are, in some ways, a list of the bare necessities of secure systems or communications. However, once these necessities are established, it is possible to construct a level of more finely grained security requirements which go beyond what has been examined thus far. The following list enumerates several such requirements:

- Role-Based Access/Permissions
- Local and Global Identity Provisioning
- Identity Provisioning and Accompanying Trust across Domains
- Security Spanning Sessions (Multiple Messages)
- Message Assurance

*Role-Based Access/Permissions* ensure(s) intuitive distinctions can be made within a system, which apply to certain "classes" of users with varying levels of permissions to core system functions. For example, perhaps a message could contain maintenance instructions for the application itself. These instructions may be less intrusive compared to those that cause the application to restart or suspend

itself, so they require fewer permission checks in order to pass through.

*Local and Global Identity Provisioning* implies some concept of "identity" within the users of the web service. The identity may only need to be managed and verified locally, or it may need to be verified through some sort of "chain of trust."

*Identity Provisioning and Accompanying Trust across Domains* speaks to the challenges of e-commerce. Often, identities in use on one domain are desirable to be used to purchase an item on a subsidiary or partner domain. In this case, some means of maintaining an identity across such a domain shift is needed.

*Security Spanning Sessions (Multiple Messages)* is a requirement which, if implemented, alleviates the need for reestablishing identity with each message. Rather, there can be some notion of a session across these web service communications. This session could persist or be ended in order to provide some cohesiveness and transaction capabilities to the WS technologies.

*Message Assurance* encompasses any guarantee a provider or consumer may wish to place on messages that it sends. Such guarantees would include at-most-once, at-least-once, exactly once, or in-order message delivery. If such guarantees can be made about messages, then less error and/or redundancy needs to be dealt with in the systems providing or exploiting the web service.

### **Fine-Grained Web Services Security Requirements Addressed**

The WS-Security specification alone was not enough to thwart all web services security threats. For organizations to fully adopt the web services protocol as their web communication infrastructure other standards had to be created to meet the granular security requirements from above. Using WS-Security as a foundation, new web services specifications such as WS-Policy, WS-Trust, WS-Federation, and WS-ReliableMessaging were created. Details of these standards are outlined below:

*WS-Policy* is used to specify the conditions of an interaction between two web services endpoints [10]. The assertions behind this policy define how data is sent (i.e. authentication scheme and

transport protocol selection) as well as service selection and usage (i.e. privacy policy, quality of service (QoS) characteristics) requirements [10]. Typically, the web service provider exposes a policy to convey the conditions under which it provides a service. The requester uses the policy information to decide whether to use the policy or not. If the requestor chooses to use the specified policy, it must follow the specified conditions. This is an example of *Role-Based Access/Permissions* because the requestor needs to follow the provider's specified policy before it obtains the proper "class" level to communicate with the provider.

WS-Trust states a web service can require an incoming message to "prove a set of claims (name, key, permissions, etc.)" before any action is taken [11]. The claims of a web service provider are set in WS-Policy. If a message arrives without its proof of claims, the web service provider will reject the message. The complete process is as follows: When a web service provider receives a message from a requestor with security tokens encrypted with WS-Security, the provider checks if the token's claims comply with the policy, verifies the attributes of a requestor through signature checks, and verifies the issuers of the security tokens are trusted with the claims they have made. All checks must pass before any action is taken. This process ensures the requestor's identity and message data can be trusted which is an example of *Local and Global Identity Provisioning*. The same process can be extended to *Identity Provisioning and Accompanying Trust across Domains* with each domain completing the trust check.

*WS-Federation* addresses the challenges brought on by the advent of e-commerce and its use within web services - a means of maintaining an identity across domains. The specification for WS-Federation describes the need for a technology which supplies "mechanisms that enable the decision [of trust] to be based on the declaration (or brokering) of identity, attribute, authentication and authorization assertions between realms" [12]. This is especially valuable to web services, since it frees each service from managing identity individually and it expands upon the premise of WS-Trust.

*WS-SecureConversation* is the web services version of session management. It was created in order to provide a means for messages to maintain some meaningful continuity of identity. The specification describes a marriage of WS-Security and WS-Trust to accomplish this through the establishment (and exchange) of security contexts [13]. A series of messages can be associated with a certain identity and thus become analogous to a user session with that web service.

*WS-ReliableMessaging* fulfills the need for assurance on messages communicated via web services. The goal of this specification is "to identify, track, and manage the reliable delivery of messages between exactly two parties, a source and a destination" which also includes making guarantees about the expected delivery [14]. WS-ReliableMessaging adds to web services the semantics of at-most-once, at-least-once, exactly-once, and in-order delivery, as is desirable in ensuring the reliability of and making guarantees about these messages.

### **Web Service Susceptibilities and Improvements**

Protocols such as WS-Security and the granular protocols above have helped make web services data and data transport more secure and reliable, but the infrastructure is susceptible to further security as well as performance issues. For example, web services have no protocols to defend against a denial of service or timing attack. In addition the performance and scalability of the infrastructure can be improved.

A denial of service (DoS) attack is where the attacker bombards the web service by sending "a great number of messages in short period of time" [Holgersson,7]. When the web service server receives an XML document, the server parses the document to read the instructions for the next course of action. The web service must do this for every message whether it is legitimate or not before rejecting it. Parsing and rejecting an extremely large number of messages is resource intensive, so a user would experience problems accessing web service applications. Even worse, other services dependent on the attacked server could crash or face performance degradation as well.

One solution for this type of attack is to use the tokens provided by the WS-Security protocol. As described above, the WS-Security protocol uses security tokens to identify a requestor and his access rights. When the web service provider receives too many token requests from the same requestor, the web service provider can halt the transaction or issue a warning to the system admin. In addition to counting the number of tokens from a specific requestor, there should be limits on the size of a XML document sent to the web service provider. For example, all XML documents larger than 128MB will be rejected.

Timing attacks use the processing time of various web service actions to expose or gain access to private information. A traditional attack has two goals. The first goal is for the attacker to re-use or replay a message as long as he can get more information. Consider that replaying a certain message might elicit a pseudo-random response. If an attacker is allowed to continue to replay against the system during the lifetime of the message, he may gain some understanding of the algorithm employed. In order to combat this, web services could establish a notion of "max-replays" of a message. The second goal of the attacker is to renew a token or message already in the web service so he can keep attacking the system. To thwart the attacker as he renews tokens, the system should have a fixed end-of-life for all tokens, so after the token's time expires no renewals are allowed - this notion of "lifetime" is utilized in many systems. Additionally, the service could specify the renewal rate of all tokens and refuse to process a token or message if it is renewed immediately before expiration or at a rate greater than the fixed rate.

Yet another untapped potential improvement of web services, which is thus far left unaddressed, is that of performance and scalability. As with any network-reliant application, latency of the network and processing time are significant factors. If the web service is particularly time-critical these two considerations can cause large problems. In other cases, the web service may require top-strength encryption algorithms employing large keys, with the network not acting as a limiting factor. Perhaps a method for adjusting to or compensating for these variables can be developed.

Suppose that two web services are communicating and the network latency, as well as the encryption requirements, are variable - dependant on the type of messages as well as the different services the two hosts may offer. Considering the network-constrained case, it would be desirable for one host to communicate to another something of the form, "I have time-sensitive material, requiring low-levels of protection; please communicate with me using fast algorithms and small keys." In this way, some accounting for network latency is made. Conversely, if data is highly sensitive but without stringent time requirements, a message could be exchanged which says, "Employ the highest level of mutually understood encryption, using strong keys." One might argue that these messages yield information about the data being exchanged, making the more valuable messages obvious. While, this may be true, perhaps the performance/scalability messages could be exchanged in some agreed-upon cipher initially so that the information they may reveal about the forthcoming messages is protected. If web services are able to automatically scale or adjust based on network conditions as well as requirements of the data, this would make web services highly adaptable and even more desirable to be employed in software infrastructures.

## **Conclusion**

Web services technology allows for data exchange between computer nodes using the standards of XML, WSDL, SOAP, and UDDI. In order to exchange data, a web service provider publishes its services and the web service requestor searches for these services. The "openness" between the web service provider and requestor to share service information as well as data poses a significant security risk. To meet some of the very basic security needs of transport security, persistent security, and integrity assurance the WS-Security specification was created. Though it was progress, the WS-Security specification was not enough to thwart specialized security attacks, so specifications such as WS-Policy, WS-Trust, and WS-ReliableMessaging were created to meet more granular security requirements. Together, this collection of specifications has improved web service security dramatically, but there is still room

for improvement. As more organizations adopt web services, protection against denial of service and timing attacks as well as improvements to scalability and performance are needed. Web services define a very new technology, and the improvements and suggestions surveyed and proposed in this paper are just the beginning.

**Works Cited**

- 1) XML Signature WG. W3C 25 July 2006.  
<<http://www.w3.org/Signature/>>
- 2) XML Encryption WG. W3C. 9 November 2005.  
<<http://www.w3.org/Encryption/2001/>>
- 3) Web Services Security: SOAP Message Security 1.0. International Business Machines Corporation, Microsoft Corporation, Inc., VeriSign Inc., et al. 15 March 2004. <<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>>
- 4) Holgersson, Jesper and Soderstrom, Eva. "Web Service Security - Vulnerabilities and Threats within the Context of WS-Security." SIIT2005 Proceedings. Available: <http://scholar.google.com/>.
- 5) Martin Naedele, "Standards for XML and Web Services Security," Computer, vol. 36, no. 4, pp. 96-98, April, 2003.
- 6) Pfleeger, Charles P. Security in Computing. Prentice Hall. 1997.
- 7) Schneier, Bruce. Applied Cryptography. John Wiley & Sons, Inc. 1996.
- 8) Remy, David L. and Rosenberg, Jothy. Securing Web Services with WS-Security. Sams Publishing. 2004
- 9) Newcomer, Eric. Understanding Web Services. Addison-Wesley. 2002
- 10) Web Services Policy Protocol. Mar 2006. BEA Systems Inc., International Business Machines Corporation, Microsoft Corporation, Inc., SAP AG, Sonic Software, and VeriSign Inc. March 2006.  
<<http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-polfram/ws-policy-2006-03-01.pdf>>.
- 11) Web Services Trust Protocol. Mar 2006. BEA Systems Inc., International Business Machines Corporation, Microsoft Corporation, Inc., SAP AG, Sonic Software, and VeriSign Inc. Mark 2006.  
<<http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-trust/ws-trust.pdf>>.
- 12) Web Services Federation Language. BEA Systems Inc., International Business Machines Corporation, Microsoft Corporation, Inc., et al. December 2006.  
<<http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fed/WS-Federation-V1-1B.pdf>>
- 13) Web Services Secure Conversation Language. BEA Systems Inc., International Business Machines Corporation, Microsoft Corporation, Inc., et al. February 2005.  
<<http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-secon/ws-secureconversation.pdf>>
- 14) Web Services Reliable Messaging Protocol. BEA Systems Inc., International Business Machines Corporation, Microsoft Corporation, Inc., et al. February 2005.  
<<http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-rm/ws-reliablemessaging200502.pdf>>