Conducting business in today’s world usually requires that a company utilize the Internet for both business-to-customer and business-to-business interactions. Often, the information exchanged in business transactions is mission-critical, market-valued, or confidential; thus, while traversing the Internet, it must be protected from accidental access or deliberate unauthorized control and use. Understanding the mechanics of how WS-Security works and the options it affords in a service-oriented architecture can enable you to make the best selection of security technology to address your requirements for authentications, data integrity, and confidentiality.

The topic of this article is provided in two parts. The first part covers WS-Security features, the relationship between business participants, and the mechanics of how WS-Security capabilities are implemented. The latter part will outline IBM® WebSphere® Application Server support of WS-Security, customers’ business requirements and how they have utilized WS-Security, and other security technologies in their operational solutions.

Introduction

Implications of using security

Design choices and implementations that address security requirements often have an adverse impact on a solution's performance. This is not to imply that all security technologies used in solutions result in slow performance. Rather, you should be aware that web services solutions requiring authentication of business participants, digital signature of message content, and encryption of XML data can have very different performance characteristics based on the technology or method used to secure a solution’s exposed business functions and data.

The security triad covered in this article comprises: (a) authentication, (b) data integrity, and (c) data confidentiality. If you are not an expert on the mechanisms to address these security requirements, I briefly overview the capabilities below before diving into details of how you can implement them.
The security triad

*Authentication* is used to ensure that parties within a business transaction are really who they claim to be; thus proof of identity is required. This proof can be claimed in various ways. One simple example is by presenting a user ID along with a secret password. A more complex example is one that uses an X.509 certificate issued from a trusted Certificate Authority (such as Verisign). The certificate contains identity credentials and has a pair of private and public keys associated with it. The proof of identity presented by a party includes the certificate itself and a separate piece of information that is digitally signed using the certificate's private key. By validating the signed information using the public key associated with the party's certificate, the receiver can authenticate the sender as being the owner of the certificate, thereby validating their identity. When both parties authenticate each other, this is called *mutual authentication* and is often done between a web service consumer and a web service provider.

In order to validate the integrity of business information exchanged in a transaction ensuring that a message's content has not been altered or corrupted during its transmission over the Internet, the data can be digitally signed using security keys. This is the second requirement of the security triad. A common practice is to use the private key of the sender's X.509 certificate to digitally sign the SOAP body of a web service request. Similarly, SOAP header blocks in a request can also be signed to ensure the integrity of information exchanged in a transaction that is outside the scope of the actual business context (for example, message IDs, security tokens). Likewise, a web services response can be digitally signed to ensure data integrity.

The third requirement of the security triad is *confidentiality*. Encryption technology can be used to make the information exchanged in web services requests and responses unreadable. The purpose is to ensure that anyone accessing the data in transit, in memory, or after it has been persisted, would need the appropriate algorithms and security keys to decrypt the data before being able to access the actual information.

Today you can implement all of these security measures using various mechanisms. Based on your specific requirements and business environment, you will be able to make a choice between those that are transport-dependent or those that are specific to SOAP messaging.

Since this article is part of the *Best Practices for web services* series, it will focus primarily on the various ways to leverage WS-Security in your solutions and the performance impact that you should expect.

**General WS-Security model**

The WS-Security specification is in its final approval process within the OASIS standards body and provides mechanisms to address all three of the requirements outlined as the security triad between application end points. With WS-Security, you can selectively implement each of the requirements of the security triad such that one or all of them are addressed in your solution.

An application that requires the services of another application is considered the *Consuming Application* for this article. I refer to the application providing the services as the *Service Provider*. 
The diagram below illustrates this relationship and is the basis for much of the discussion that follows:

**Figure 1 - System environment**

For the scenario described and illustrated in the following sections, the public key of the Service Provider's X.509 certificate must be exchanged with the client and configured into the client's runtime prior to invoking the Service Provider's services. For both the Consuming Application and the Service Provider, the root certificate of the Certificate Authority (such as Verisign) that issued the parties' certificates will need to be imported into the local keystores. The Consuming Application's keystore will have the root certificate of the Service Provider's certificate. Likewise, the Service Provider's keystore will require the root certificate of the Consuming Application's certificate. This is mandatory and allows validation of the digital signatures of the individual certificates that are passed as binary security tokens in the SOAP messages.

A full implementation of all three security requirements would include:

- The senders of the web services request or response
  - Signing the message with the private key of their X.509 certificate
  - Encrypting the message with the public key of the receiver's X.509 certificate to ensure that only they can access the message content

- The receivers of the web services request or response
  - Verifying the signature of the message using the public key of sender to authenticate the sender and to verify integrity of the message
  - Decrypting the message with the private key of their X.509 certificate

**Details of WS-Security XML encryption**

The encryption of data based on the WS-Security specification and the XML Encryption specifications that it references involves:

- A two-phase process using both symmetric and asymmetric algorithms.
• A shared key that is used to encrypt/decrypt the message data using a symmetric algorithm such as Triple DES. Symmetric algorithms are very efficient and work with a single key for both encryption and decryption calculations. WS-Security implementations use a key that is randomly generated. Once the data of the message is encrypted, the key itself is inserted into the message. (Note that the key is also encrypted as described below.)

• Passing the shared key in the SOAP message with the key encrypted/decrypted using an asymmetric algorithms such as RSA-V1.5. Encryption of the shared key is performed differently than the message data with an algorithm that utilizes a pair of keys -- private and public. An X.509 certificate has two keys, one that is private to the owner of the certificate and a second key that is shared with others with whom they are conducting business. Symmetric algorithms are more efficient than asymmetric algorithms; however, they require management of shared keys between the parties and have inherent security risks of their being exposed to others outside of your organization or business partners. Thus, by using only asymmetric algorithms on the random key, WS-Security provides both a relatively efficient solution and one that is easy to manage. In Part Two of this article, I will discuss why I use the word \textit{relatively}.

See the links in the Related topics section for details of XML Encryption.

Real-world WS-Security scenario

The scenario outlined below is one of many possibilities that can be realized with WS-Security. It uses X.509 certificates, a practice common in several of the customer solutions that I will outline in Part Two of this article.

Consuming Application processing of web services request

Typically, the Consuming Application will have a Service Proxy or a JAX-RPC stub component that has been generated from an Integrated Development Environment (such as WebSphere Studio Application Developer) using the Service Provider's WSDL. When a web services invocation is made, the proxy or SOAP runtime on the client system performs the WS-Security functions prior to sending the request.

First, the SOAP message is digitally signed. The SOAP runtime may access a keystore to retrieve security keys and certificates as needed. Depending on the WS-Security support your environment provides, you might be able to sign just the SOAP body, or you might be able to sign individual elements within the body. In addition, SOAP header blocks might be signed. The signature is performed using the private key of the Consuming Application's X.509 certificate. Once the message has been signed, the X.509 certificate itself is included in the SOAP header as a binary security token. The message is encrypted using a symmetric algorithm with a shared key. The key used for the data encryption is encrypted itself using an asymmetric algorithm with the public key associated with the Service Provider's X.509 certificate. Once the message and shared key have been encrypted, a reference to the X.509 certificate of the Service Provider to whom the request is being sent is included in the SOAP header. This is done because the Service Provider might be using multiple certificates.
When a Service Provider receives a web services request, the request is directed to the SOAP processing engine (SOAP runtime) based on the request's URL (published access point for the service). The message data and shared key passed in the request are encrypted, so the first step is to identify the X.509 certificate referenced in the SOAP header and retrieve its private key from a keystore. Once the private key is obtained, the shared key can be decrypted using an asymmetric algorithm. With the shared key in the open, the message data can be decrypted using a symmetric algorithm. With the entire message now in the open, the X.509 certificate passed in the SOAP header can be accessed to retrieve its public key. The message's digital signature is performed with the Consuming Application's public key. As a result of the signature's successful validation, the Service Provider SOAP runtime not only validates the message integrity but also is ensured that the Consuming Application actually signed the message. This process also authenticates the message's origin/sender because only the sender who owns the certificate has access to the private key used to sign the message. Once the message has been decrypted and the signature validates the SOAP runtime calls the web services implementation.
Service Provider processing of web services response

Once the business logic of the service implementation has executed and a response is available, the same WS-Security operations take place for the web services response message. However, the roles of X.509 key pairs are reversed for digital signature and encryption. The Service Provider's SOAP runtime digitally signs the message using the private key of its own X.509 certificate. The certificate is included in the SOAP message, and the message is encrypted using a shared key. The key used for the data encryption could be the same key passed in the original request or another randomly generated key, the latter being more typical. The encryption of the shared key is performed using the public key of the certificate that was passed in the request; thus only the sender of the request who has access to the certificate's private key is the only party that can decrypt the message. Once the message has been signed and encrypted, the Service Provider's SOAP runtime sends the response to the Consuming Application.
Figure 4 - Service Provider processing of response with WS-Security

Consuming Application processing of web services response

The Consuming Application's WS-Security processing of the web services response is very similar to what the Service provider for the request performed.

The Consuming Application receives a web services response with the response directed to the SOAP processing engine (SOAP runtime) based on the original HTTP session. The message data and shared key passed in the response are encrypted. Therefore, the initial step is to retrieve the private key of the certificate associated with the corresponding request to decrypt the shared key using an asymmetric algorithm. With the shared key in the open, the message data can be decrypted using a symmetric algorithm. After the entire message is in the open, the X.509 certificate passed in the SOAP header can be accessed to retrieve its public key. The response message's digital signature is performed with the Service Provider's public key. Following the signature's successful validation, the Consuming Application's SOAP runtime not only validates the message integrity, but also is ensured that the Service Provider actually signed the message. This process also authenticates the message's origin/sender because only the sender who owns the certificate has access to the private key used to sign the message. Once the message has been decrypted and the signature validated, the SOAP runtime forwards the response to the Consuming Application.
Summary

Under the covers, WS-Security is very complex and can be utilized in many different scenarios. The example described above has many aspects that customers today have implemented. Part or all of the scenario has been implemented by customers on WebSphere Application Server platforms, depending on their existing security policies, security infrastructure, or business requirements. This will be discussed in Part Two of this article.

The good news is that WebSphere Application Server support of WS-Security is through a declarative model. Thus, once you have your web services implementations developed and tested, the enablement of WS-Security features is easily accomplished during the deployment phase. As a result, the complexity of the XML Digital Signature and XML Encryption should be of little concern to you as an IT Architect or IT Specialist.
Related topics

- Check out the entire Best Practices column series.
- Web Services Interoperability (WS-I) initial working group draft: Basic Security Profile Scenarios.
- Find more information on WebSphere Application Server's support of Web Services security at the WebSphere InfoCenter.

© Copyright IBM Corporation 2004
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)