Web Services Security Interoperability with SAP NetWeaver and Microsoft .NET (Part I)

Applies To:

SAP NetWeaver Web Application Server Java 6.40 SP 13, SAP NetWeaver Developer Studio 2.0.12, Microsoft .NET 1.1 and Microsoft Web Services Enhancements 2.0 Service Pack 3, Microsoft Visual Studio .NET 2003

Summary

This two-part article series provides an in-depth analysis of the interoperability issues regarding SAP's and Microsoft's Web Services Security (WS-Security) implementation. Based on a real-world B2B scenario from the retail industry, the first part assesses the security risks and recommends the appropriate security configuration from the various options provided by WS-Security. The second part explains the configuration settings and shows code samples to secure the communication based on the requirements identified in the first article. It also considers the requirements from the Basic Security Profile (BSP) 1.0 Working Group Draft released by the Web Services Interoperability (WS-I) Organization and looks at the interoperability issues when using the declarative security features in WSE.

By: Martin Raepple

Company: SAP AG

Date: September 16, 2005

Table of Contents

Part I

Part 1 introduces the sample scenario used throughout this two-part article series to demonstrate interoperability using Web Services Security (WS-Security) between SAP NetWeaver Web Application Server Java 6.40 and Microsoft .NET 1.1. Following a risk assessment of the scenario, this article will explain the standards-based security measures available on both platforms to secure Web Services-based communication and provide guidance to choose an appropriate level of protection according to the identified security requirements. Part 2 will show how to implement the security policy and provide guidelines for developing secure, interoperable applications based on J2EE, .NET and the WS-I Basic Security Profile. This article assumes that you have basic understanding of network security, XML and Web Services. [1] provides an excellent introduction to these fundamentals.

Overview of the sample scenario

Holiday Corp. is a wholesaler for cosmetic supplies, specializing on suntan lotions. The company provides a Web-based customer portal where approved retailers can place orders online over the Internet. To optimize the inventory management and keep the procurement process lean, Holiday Corp. decides to outsource the procurement planning to its supplier Lotion Inc. Instead of the wholesaler placing the purchase orders when the stock levels fall on or below a preset warning level, this task is performed by the supplier. To provide this service, also known as Vendor Managed Inventory (VMI), it is necessary for the supplier to be able to trace the stock using the sales data from the wholesaler. A close relationship is a prerequisite for Lotion Inc. to be in total control of the timing and size of the orders being placed. Figure 1 shows the main steps in the VMI process: First, the sales and inventory data is sent from the wholesaler to the supplier. There, replenishment planning for Holiday Corp. is carried out. When inventory stocks drop below a certain level, Lotion Inc. generates a purchase order and sends it to the wholesaler (step 2). In Holiday Corp.'s retail system, an order is generated based on the message received. The order number that is generated is sent in response to the supplier. To finalize the procurement process, the wholesaler must confirm the purchase order before the production of the goods and shipping activities can begin (step 3). Along with the confirmation the backorder level for the related items in the wholesaler's inventory system is increased by the ordered quantity.

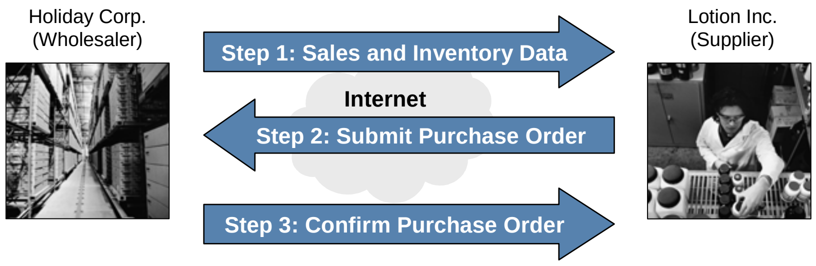

Holiday Corp. (Wholesaler)
Step 1: Sales and Inventory Data
Internet
Step 2: Submit Purchase Order
Step 3: Confirm Purchase Order
Lotion Inc. (Supplier)

Figure 1 VMI process in the sample scenario

From a technology perspective, Holiday Corp. requires stable standards to meet the interoperability requirements of its business partners. In this sample scenario [2], Holiday Corp.'s strategic platform for all current and future business applications is SAP NetWeaver, whereas Lotion Inc. chose Microsoft .NET for its development and runtime environments. Even though there are a number of ways to achieve interoperability and secure communication between these two platforms, this article will investigate how well Web Services meet these requirements by building Web Service interfaces for Holiday Corp.'s J2EE-based inventory management solution using SAP NetWeaver Developer Studio and Microsoft Visual Studio .NET to provide standard Web Service interfaces for Lotion Inc.'s procurement planning application.

Risk Assessment of the VMI process

Security is perceived as a very important issue for business applications, especially when they exchange sensitive data over non-secure networks like the Internet. Uncovering security vulnerabilities in a productive operational environment always hurts an organization's revenues as well as its reputation. Therefore, a risk assessment should be among the first steps in your design process of a new system. It not only helps to uncover potential security exposures before someone else does, but also provides a solid foundation for identifying appropriate and cost-effective countermeasures.

This section will introduce a systematic approach for conducting a risk assessment in 3 steps. However, a full assessment for the VMI scenario is beyond the scope of this article since it would also consider aspects such as host security or physical security. Hence the focus will be on the network security issues regarding the communication between the wholesaler and the supplier.

Step 1: Identify protected resources

This step will answer the question "What needs to be protected?" In general, protected resources include things like network services, customer databases or payment information. For the VMI scenario, the messages sent over the Internet between the wholesaler and the supplier need to be protected to ensure the undisturbed operation of the VMI process. These are:

- Product Activity Messages sent periodically by the wholesaler with the sales and inventory data update for the supplier to enable replenishment calculations.
- Purchase Order Message sent by the supplier and used by the wholesaler to replenish its inventory.

- Purchase Order Confirmation Message sent by the wholesaler to acknowledge the order, confirm the contract and start the delivery process of the ordered goods.

Step 2: Threat analysis

A key aspect of information security is to preserve the confidentiality, integrity, availability[1] and accountability (synonymous with non-repudiation) of an organization's information assets. Loss of one or more of these attributes can threaten the continued operation of business processes and result in serious business damage. Keeping these security goals in mind, the threat analysis below provides an example of assessing the impact of exploiting a vulnerability based on the working knowledge of typical attacks in network security and the resources identified in the first step of the risk assessment:

- Tampering: An attacker intercepts the message transfer and modifies the data. Neither the wholesaler nor the supplier will recognize that the data has been modified and will accept it. This attack is a threat for all messages exchanged between the wholesaler and supplier.

- Eavesdropping: The unauthorized intercepting and reading of sensitive information. Potential targets are the Product Activity Messages and Purchase Orders since they contain confidential information which could be of value to any of Holiday Corp.'s competitors.

- Spoofing: The attempt to gain access to a system by using a false identity. Spoofing can be accomplished at the network layer (e.g. by using a false IP address to bypass a packet filter) or application layer (e.g. by using a stolen user name and password to login to a Web application). For the Web Services-based communication in the VMI scenario, unauthorized submission of any of the messages exchanged in the process could result in serious damage. One could further differentiate between the required levels of protection against spoofing based on the legal liability associated with the submission of the message. Confirmation of the purchase order in the last step of the VMI scenario obviously requires strong authentication whereas moderate protection for the preceding messages seems to be sufficient.

- Repudiation: The sender's or receiver's false denial of having created or received a message. As the purchase order confirmation in the VMI scenario triggers the production and delivery of the goods in the procurement process, an individual transaction may include literally millions of dollars of potential exposure. Therefore, preventing the wholesaler from denying that they committed an order is of great importance and represents a high risk for the supplier.

- Message Replay: An attacker records a message and replays it to the Web service. Even without further manipulation of the data, replay of any of the messages exchanged in the VMI process could result in a serious disruption of the process or monetary losses. As an example, the recurrent submission of Product Activity Messages could result in false replenishment calculations.

---

[1] Preserving availability is out the scope of this article.

Step 3: Risk Analysis

The table below provides a ranking based on a qualitative measure for the risks identified in the previous step. This ranking takes into consideration the analyzed impact of a successful attack as well as the likelihood of the vulnerability being exploited by an attacker. The supplier's false denial of having received a product activity notification is neither very likely nor would it result in serious damage and thus can be ranked low. Since the replenishment process strongly relies on the trustworthiness of the purchase order confirmation, risks associated with either unauthorized modification, submission or repudiation have a high ranking.

Resource / Threat	Product Activity Notification	Purchase Order Submit	Purchase Order Confirmation
Tampering	**M**	**M**	**H**
Eavesdropping	**H**	**H**	
Spoofing	**M**	**M**	**H**
Repudiation	**L**	**L**	**H**
Message Replay	**M**	**M**	**M**

Legend		
	L	- Low Risk
	M	- Medium Risk
	H	- High Risk

The risk ranking is correlated with the protection level each resource requires and can be mapped to the strength of the countermeasures provided by Web Services Security which will be discussed in the next section.

Web Services Security (WS-Security)

To safeguard a Web Services environment against the described threats, mechanisms are available at different layers of the network stack. At the transport level, mature security protocols like Secure Sockets Layer (SSL) / TLS (Transport Layer Security) do provide means for certificate-based authentication and encryption to cover risks associated with confidentiality and integrity of data in transit. At the (SOAP) message level, the Web Services Security (WS-Security) standard provides a flexible framework to implement a wide variety of security models. The latest core specification (also known as SOAP Message Security) version 1.0 [3] was standardized in March 2004 and released together with the WS-Security UsernameToken Profile 1.0 [4] and the WS-Security X.509 Certificate Token Profile 1.0. Today, many vendors in the industry provide stable implementations of WS-Security 1.0 and its profiles. The work on the version 1.1 of the specification is under way and is expected to be finishing later this year.

Motivation for WS-Security in the VMI scenario

SAP NetWeaver Web Application Server (Web AS) Java 6.40 and Microsoft .NET provide support for all of the above standards. Being spoilt for choice in the mechanisms to choose, one should carefully evaluate the use of either message or transport level security mechanisms or even the combination of both. SSL only provides a secure channel between directly communicating partners (point-to-point) at the network. In other words, SSL protects the messages only while in transit but offers no security for (XML) data in storage. Therefore, WS-Security should be used in complex scenarios where end-to-end protection is required and both the sender and receiver do not have a trusted relationship to any intermediary that might be involved. Since there is no third party or intermediary in the VMI scenario, the value gained from WS-Security seems not enough to justify the additional development and configuration effort required for message level security compared to the relatively simple setup of a SSL channel between the wholesaler and the supplier. But looking back at the risk assessment, there are two key requirements that SSL does not address sufficiently:

- All messages must be authenticated by the Web Services as coming from a set of entities. These entities might be a technical user in the case of an auto-generated Product Activity notification for the supplier or the real employee acting on behalf of the wholesaler when committing a new purchase order. Since SSL has no knowledge about the application context, WS-Security can provide better support for this requirement as it operates at the application layer. Furthermore, the authentication mechanisms supported by WS-Security are closely integrated with each of the platform's native authentication and authorization infrastructure, namely the User Management Engine (UME) in SAP NetWeaver Web AS and the Windows User Administration respectively.

- Non-repudiation has been identified as an important requirement in the context of the confirmation step in the process. To be more precise, the transaction must be strongly bound to the person and the message to ensure that it cannot be denied afterwards. While SSL can be configured providing a client credentials through the use of mutual authentication so that both the wholesaler and the supplier have to present a valid digital certificate, this does not completely cover the requirements necessary for non-repudiation in the VMI scenario. Even though the digital signature legislation makes it possible in many countries to legally bind a digital signature to a transaction, other trustworthy evidence is required for legal enforceability like an audit trail. Again, this can only be achieved at the application layer where the actual end entity (the organization confirming the order) and application context (e.g. the Purchase Order ID) is available and no intermediary (even on the same machine, such as a secure Web Server) can decrypt or alter the message content before it is forwarded to its final destination.

Security mechanisms in WS-Security

WS-Security addresses end-to-end security through the definition of a security header format for SOAP messages that is positioned within the SOAP header and is designated by the opening and closing `<Security>` block. The specification basically provides a framework to use existing and approved approaches to distributed systems security such as public key encryption, digital signatures or X.509 certificates and defines how they should be used with SOAP.

XML Signature

In order to protect a message against unauthorized modification and non repudiation, WS-Security utilizes XML Signature [2] to digitally sign XML documents (or some part thereof) in a two stage process: First, the content to be signed is digested using a secure hash function like the Secure Hash Algorithm (SHA) which creates a small and unique value (digest) for the original content to facilitate faster processing. Next, the digest is encrypted using the private key of an asymmetric encryption algorithm key pair (e.g. RSA). The resulting signature is inserted in the WS-Security security header as a `<Signature>` element which contains the following child element:

- `<SignedInfo>`: Since message digest algorithms treat XML data as octet streams, it is important to define a canonical form of the content to be signed in order to make sure that logically equivalent XML documents which could have slight textual differences (like the order of attributes, encoding of line delimiters etc.) produce the same digest at the sender and receiver. Failure to remove these differences will result in slightly different serialized versions of the data and the algorithm that verifies the digital signature will fail although it should pass. The `<CanonicalizationMethod>` child element of `<SignedInfo>` contains the identifier of the canonicalization algorithm as specified by the W3C [2] that the XML parsers at both sides use to get a consistent binary representation of the complete `<SignedInfo>` element before the ultimate signature is generated according to the algorithm defined by another child element with the name `<SignatureMethod>`. Finally, `<SignedInfo>` contains a collection of one or more `<Reference>` elements that are pointers to the data (within the document or externally) being signed. Each reference can specify its own canonicalization algorithm in the `<Transform>` element to be performed before the digest is calculated for the corresponding document part as well as the digest algorithm (`<DigestMethod>`) itself. The actual digest value is stored in the `<DigestValue>` child element of the `<Reference>`.

- `<KeyInfo>`: The `<KeyInfo>` element references the public key sent along with the signed message needed by the receiver to verify the signature. Although XML Signature allows many ways to express this reference, the Basic Security Profile [9] (BSP, see below) mandates the use of a single `<SecurityTokenReference>` element to point to a X.509 certificate (token). Even though the OASIS WS-Security TC has not yet defined a standard algorithm to derive a cryptographic key from a shared secret like a password, some implementations and the BSP also allow referencing a UsernameToken (see below).

- `<SignatureValue>`: Contains the encrypted result over the `<SignedInfo>` element based on algorithms indicated by the `<SignatureMethod>` element.

XML Encryption

WS-Security uses the W3C Recommendation XML Encryption [7] standard to allow encryption of any combination of a message's body or header elements. To identify and represent encrypted data in an SOAP message, one or more `<EncryptedKey>` elements in the header provide the information about the cryptographic key(s) in the `<KeyInfo>` child element being used to encrypt the data. When using an asymmetric encryption algorithm like RSA (as specified by the `<EncryptionMethod>` child of `<EncryptedKey>`) for example, `<KeyInfo>` points to the recipient's X.509 public key certificate inside a `<SecurityTokenReference>`. Additionally, `<EncryptedKey>` contains a `<ReferenceList>` child with a sequence of `<DataReference>` and `<KeyReference>` elements which take a URI to refer to the protected portions of the message. The actual encrypted data is wrapped in an `<EncryptedData>` element that replaces the original (unencrypted) message data.

UsernameToken

WS-Security describes a general-purpose mechanism for associating messages with certain privileges or credentials owned by the message sender (e.g. a login name or a cryptographic key), also called a security token. The core specification [2] defines three types of security tokens: Username, Binary, and XML tokens.

One such token format is the UsernameToken (included as a `<UsernameToken>` child in the WS-Security security header) and its extensions (defined by the accompanying UsernameToken profile [4]) that define how a Web Service consumer can supply its username and optionally a password (or shared secret) to authenticate that identity to the web service provider. If included in the token as the `<Password>` element, the password can be of type pain text or digest. In addition, an optional `<Nonce>` child containing a secure (meaning unpredictable) random one time value and a creation timestamp (`<Created>`) can be added to prevent a replay attack using the same token again by any unauthorized 3rd party. The example below shows a UsernameToken with a plain text password as specified by the `Type`-attribute of the `<Password>` element:

```
<wsse:UsernameToken>
  <wsse:Username>alice</wsse:Username>
  <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">2secret4u</wsse:Password>
</wsse:UsernameToken>
```

X.509 Certificate Token

The `<BinarySecurityToken>` in the WS-Security core specification [2] defines a token format to represent non-XML credentials in a SOAP security header. One such token is the X.509 certificate token [5] used to carry the binary data of a public key certificate according to the X.509 standard. The `ValueType` attribute must be specified according to the content of the token. The example below represents a X.509 Certificate Token.

```
<wsse:BinarySecurityToken xmlns:wsu=… wsu:Id="sap-1„ ValueType=
"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-soap-message-security-
1.0#Base64Binary">MIICG…</wsse:BinarySecurityToken>
```

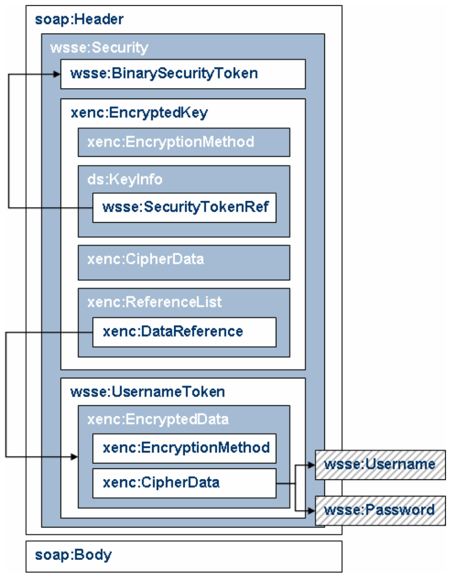
Timestamp

The WS-Security specification [2] defines the `<Timestamp>` security header element to provide a mechanism for expressing the creation and expiration times of a message and thus help thwart replay attacks. A timestamp contains at minimum a `<Created>` child (see example below) and may define an expire date using the element `<Expired>`.

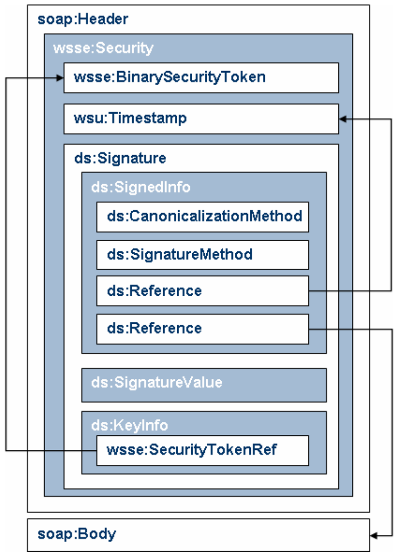
```
<wsu:Timestamp xmlns:wsu=… >
  <wsu:Created>2005-08-16T19:39:11Z</wsu:Created>
</wsu:Timestamp>
```

SOAP message examples using WS-Security

Now that we had a brief overview about the different security mechanisms in WS-Security, let us look a two example messages from a structural perspective:



- The first example uses XML Encryption to encrypt a UsernameToken. Within the WS-Security header, the message contains the public key certificate (`<wsse:BinarySecurityToken>`) that is used to encrypt the actual encryption key which can be found in the `<xenc:CipherData>` child of the `<xenc:EncryptedKey>` element. The `<xenc:ReferenceList>` contains one `<xenc:DataReference>` element that points to the encrypted data portion inside the UsernameToken.



- In the second example, a signature is attached to the message. Again, a public key certificate (`<wsse:BinarySecurityToken>`) is provided by the sender to let the receiver of the message verify the signature based on the algorithms specified in the `<ds:SignedInfo>/<ds:SignatureMethod>` element. `<ds:SignedInfo>` also contains two `<Reference>` elements, one pointing to the `<Timestamp>` in the WS-Security header, the other to the message body. Both references have a digest value which are signed (encrypted) using the private key that belongs to the sender's public key certificate. The actual signature is stored in the `<ds:SignatureValue>` child of `<ds:SignedInfo>.`

Choosing appropriate countermeasures

From the list of security mechanisms supported by WS-Security, one can derive a relationship to the threats discussed in the previous section and assign them a value which represents the protection level. Due to the strong cryptographic nature of secure message digests and asymmetric key algorithms, digital signatures mean a high protection level against message tampering, whereas encryption as it stands can only support mediocre protection. It should be noted that any assessment of protection levels is always scenario specific and basically reflects an organizations overall security policy. A multinational bank might not accept UsernameTokens at all as an appropriate means to authenticate Web Service consumers, whereas the same mechanism might be rated H for an intranet procurement solution for office supplies.

Security mechanism (WSS) / Threat	XML Encryption	XML Signature	Username Token	X.509 Token	Timestamp
Tampering	**M**	**H**			
Eavesdropping	**H**				
Spoofing			**M**	**H ****	
Repudiation		**M/H ***			
Message Replay					**M/H *****

Legend		
	L	- Low Protection
	M	- Medium Protection
	H	- High Protection
	<empty>	- Not applicable

* M if UsernameToken used for signatue, H in the case of a strong cryptographic key
** H if signed by trusted authority
*** M if unsigned, H if signed

Based on the table above, one can easily map the protection level each security mechanism provides with the quantified risk ranking from the last step of the risk assessment. Given a medium risk for spoofing a product activity notification for example, a UsernameToken will fulfill the authentication requirements best in terms of protection and costs. Mitigating the high risk of spoofing in the final confirmation step through the use of X.509 certificates seems reasonable since it is a stronger authentication mechanism compared to a username and a password[2]. The table below shows the mapping result between the resources and chosen security mechanisms for the VMI scenario:

[2] For very high risks, technology by itself might not fulfill an organizations need for risk mitigation. In these cases, other options like insurance should be taken into consideration.

Security mechanism (WSS) / Resource	XML Encryption	XML Signature	Username Token	X.509 Token	Timestamp
Product Activity Notification	x		x		x
Purchase Order Submit	x		x		x
Purchase Order Confirmation		x		x	x

Web Services Security Interoperability

Web Service protocols like WS-Security define a rich but also complex framework in terms of additional SOAP header elements and processing rules. Even though the core specification and additional profiles try to be as accurate as possible, much effort is needed to achieve a common understanding among different implementations - and thus interpretations - of a standard in order provide interoperability. For WS-Security, this not only applies to the SOAP layer, but also to the exchange of user credentials which means that each specified token must provide an interoperable format and semantic. By bringing this to mind, it is not surprising that additional clarification on the specification is needed to further promote WS-Security interoperability across platforms, operating systems and programming languages.

Web Services Interoperability Organization (WS-I)

The above statement is exactly the mission of WS-I, an open industry organization governed by SAP, IBM, Microsoft and others. Specifically, WS-I currently has six working groups, mainly focusing on interoperability profiles which are named groups of Web Services specifications at a specific version level, along with conventions about how they should be used together for best interoperability. To date, WS-I has released the Basic Profile 1.1 (SOAP 1.1, WSDL 1.1, UDDI 2.0 and others), Attachment Profile 1.0 (extends the Basic Profile to address the W3C Note on SOAP Messages with Attachments, SwA), and Simple SOAP Binding Profile 1.0 (extends the Basic Profile to address HTTP 1.1). To demonstrate interoperability across the different platforms, WS-I delivered a specification for the so called Supply Chain Management (SCM) Sample Application which is implemented by different vendors as a demonstration environment and working example for WS-I profile conformance. SAP is chair of the group working on Sample Applications. In addition, developers can download a set of testing tools to automate conformance testing of a Web Service implementation to the WS-I profiles.

Basic Security Profile (BSP)

For security related interoperability guidelines, work on the Basic Security Profile (BSP) 1.0 is underway. This WS-I interoperability profile is dealing with transport level security (SSL 3.0 / TLS 1.0) and WS-Security 1.0[3] at the message level (core specification [2], token profiles [4] [5]). Similar to the other profiles, the BSP focuses on improving interoperability by strengthening requirements when possible and reducing some of the

[3] The upcoming version 1.1 of WS-Security will be covered by the next release of the BSP.

flexibility by limiting several options to one. The following interoperability recommendation (each identified by the prefix ‘R’ and a unique id) is an example taken from the current BSP 1.0 Working Group Draft that limits the number of algorithms for digital signatures to the two most widely-implemented and interoperable (HMAC and RSA), using RFC2119 language (e.g., MUST, MAY, SHOULD) to indicate the nature of the requirement and its impact on conformance:

R5421 Any ds:SignatureMethod/@Algorithm element in a SIGNATURE MUST have a value of "http://www.w3.org/2000/09/xmldsig#hmac-sha1" or "http://www.w3.org/2000/09/xmldsig#rsa-sha1"

In addition to the interoperability recommendations, the BSP makes a number of security recommendations (identified by the prefix ‘C’ and a unique id) intended to improve security. These recommendations are informal only and have no impact on conformance.

C4211 Any SECURITY_TOKEN named wsse:UsernameToken that contains a wsu:Created element and a wsse:Password element with a Type attribute value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText" SHOULD be referenced by a ds:Reference in a SIGNATURE in order to prevent replay.

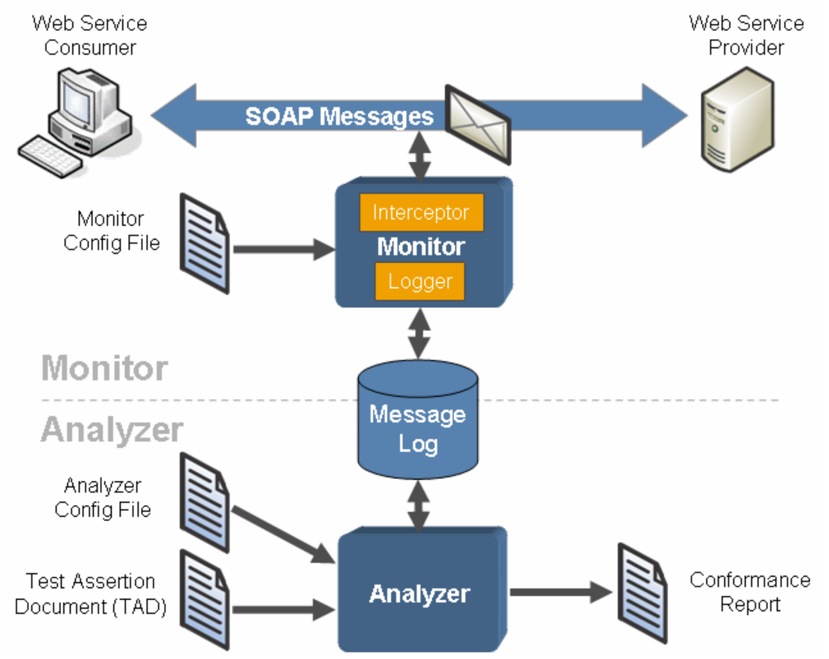
Above security recommendation C4211 for example suggests to protect against replay of Username tokens which were sent without a nonce by binding them to the signature of the sender.

Although recommendations like C4211 help to master the fairly complex WS-Security standard, they are not aimed to supply best practices for common business use cases in order to help developers or administrators reduce security risks from wrong security configurations. Since this work is currently not on the radar of any of the standard’s bodies, most of the vendor’s design-time tools and platforms provide so called policy templates which group best practices for protecting incoming and outgoing SOAP messages. In the second part of this article, we will use policy templates in SAP NetWeaver Developer Studio to configure the different identities, keys and protocols in a more transparent manner.

WS-I Test Tools

The WS-I Test Tools consist of

- a monitor to unobtrusively log any message sent between a Web Service consumer and provider. Using a man-in-the-middle approach, the monitor acts as a transparent proxy in the communication path and forwards any request to the actual service provider as well as the response to the original consumer.
- an analyzer to determine the profile conformance of the implementation based on the logged messages, the Web Service description (WSDL) and any UDDI entries for the Web service if they reference a WSDL-based service description.



Along with every profile, WS-I delivers so called Test Assertion Documents (TAD) that defines an XML-based conformance rule syntax for the Analyzer to automate profile testing.

The problem of the current tools architecture when using them for BSP testing is that the BSP test assertions are limited to examining the message without unraveling the signatures and encryptions. To conduct a detailed verification of secure messages, the analyzer would have to act as a man-in-the-middle with full access to the cryptographic keys. Work on a modified Test Tools suite for the BSP is currently underway.

Conclusion

Due to its wide adoption in the industry, WS-Security is the ideal candidate to ensure interoperability for secure communication at the message level. To derive a meaningful, that is secure and reasonable selection of countermeasures from the security models provided by WS-Security, a risk assessment should be conducted by the security team in an organization. Building further on this foundation, the next article in this series will provide a step-by-step guide to implementing the secure and interoperable solution for the VMI scenario.

References

[1] Dipak Chopra, *Security for SOA and Web Services*, SDN, December 2004, https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/com.sap.km.cm.highlightedcontent?DocumentUri=/library/security/Security+for+SOA+and+Web+Services.article

[2] Downloadable ZIP-Archive of the Vendor Managed Inventory (VMI) application code, SDN, https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/docs/library/uuid/2eb2a071-0601-0010-a182-d90dc3b1c11f

[3] OASIS, *Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)*, March 2004, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0

[4] OASIS, *Web Services Security: UsernameToken Profile 1.0*, March 2004, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0

[5] OASIS, *Web Services Security: X.509 Certificate Token Profile*, March 2004, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0

[6] W3C, *XML Signature Syntax and Processing*, W3C Recommendation 12 February 2002, http://www.w3.org/TR/xmldsig-core

[7] W3C, *XML Encryption Syntax and Processing*, W3C Recommendation 10 December 2002, http://www.w3.org/TR/xmlenc-core/

[8] W3C, *Exclusive XML Canonicalization Version 1.0*, W3C Recommendation 18 July 2002, http://www.w3.org/TR/xml-exc-c14n

[9] WS-I, *Basic Security Profile Version 1.0*, Working Group Draft, June 2005, http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html

Author Bio

Martin Raepple has practiced as an Information Technology professional for over 10 years and has experience in applying technology in a wide range of industries including telecommunications, financial services, manufacturing and transport. As a Standards Architect with SAP's Industry Standards team, Martin works in the area of standardization and interoperability testing of new Web Services technologies, focusing on message security and identity management. Martin is a frequent speaker at conferences and author of books and articles relating to information security, integration middleware and J2EE development.