SDN Contribution

Getting Started: Web Services Security (WS-Security)

Applies to:

SAP NetWeaver Web Application Server (WebAS)

Summary

This document provides an introduction to the Web Services Security (WS-Security) standard.

Created on: 2 May 2006

Author Bio

As a Standards Architect with SAP's Industry Standards team, Martin Raepple 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.

To safeguard a Web Services environment against common threats in distributed computing like tampering, eavesdropping or spoofing, 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. Both protocols set up encrypted connections between the client and the server which are suitable for direct point-to-point communication between a client and a single server.

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 core specification (also known as SOAP Message Security) version 1.0 was standardized in April 2004 by OASIS and released together with the WS-

© 2006 SAP AG

Security UsernameToken Profile 1.0 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 latest version 1.1 has been approved in February 2006 by OASIS. It enhances the existing 1.0 specification by providing new security token profiles (e.g. Kerberos Token Profile 1.1, SAML Token Profile 1.1). In addition, the sender of a message can now receive a confirmation that the data it received was generated in response to the message it initiated (Signature Confirmation).

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 partners directly communicating in a 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 by obtaining information about actions to be executed from the SOAP header. This is not possible in the case of a complete encryption using SSL/TLS. At message level, an encryption and signature concept with fine granularity is possible. Here, not the transport channel but the message itself is protected.

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 builds upon existing and approved approaches to distributed systems security such as public key encryption, digital signatures (based on XML Encryption and XML Signature) or X.509 certificates and defines how they should be used with SOAP.

In addition, 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 defines several types of security tokens, such as Username, Binary, and XML tokens.

As an example, the UsernameToken (included as a `<UsernameToken>` child in the WS-Security security header) and its extensions (defined by the accompanying UsernameToken profile) defines how a Web Service consumer can supply its username and optionally a password (or shared secret) to authenticate its 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>
```

The `<BinarySecurityToken>` is another prominent type that defines a format to represent non-XML credentials in a SOAP security header. One such token is the X.509 certificate token 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>
```

The WS-Security specification also 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

© 2006 SAP AG

attacks. A timestamp contains at minimum a <Created> child (see example below) and may define an expiry date using the element <Expired>.

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

Due to its wide adoption in the industry, WS-Security is the ideal candidate to ensure interoperability for secure communication at the message level.

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.

© 2006 SAP AG