Identity, Security, and XML Web Services

The Importance of Interoperable Security Credentials

Jorgen **Thelin** <Jorgen.Thelin@capeclear.com>

Abstract

The use of security credentials and the concepts of single-sign-on and "identity" will play a big part in Web Service products as the technology matures and developers start writing true enterprise-grade line-of-business applications. The emerging XML security standards such as SAML are reviewed, along with the various "identity" standards such as Passport and Liberty, to provide an overview of the evolution of Web Service platform products to support these. This paper examines just how "identity aware" Web Service implementations need to be, and the value a Web Services platform can add in masking developers from the complexity in this area. Lessons are drawn from the experience of using EJB security technology for real-world security scenarios.

Table of Contents

1. Introduction

This paper looks at the related concepts of identity and credentials, and how XML-based approaches are are now being used to provide increased interoperability among security systems. The various specifications and standards in this area are examined, and examples provided of how XML format security credentials are being used with SOAP and Web Services to allow end-to-end flow of identity and authentication information.

2. What is Identity?

A definition from Cambridge Dictionaries Online:

Identity [noun]	*Who a person is, or the qualities of a person or group which make them different from others.*

Source: http://dictionary.cambridge.org/define.asp?key=identity*1+0

At its most basic then, the concept of "identity" is about:

- Who you are
- How you prove who you are
- What that allows you to do

We will now explore these aspects in turn, and see how they apply to XML Web Services, and demonstrate the importance of standardized and interoperable security credentials.

2.1. Identity - Who are you?

An identity equates to a particular subject or principal

For example: Joe Bloggs - who lives at the address: 123 My Street, Your Town

A subject usually equates to a single person, but could also be a group, corporation, or even something like an automated software agent component.

The main thing is that there must be some way to distinguish one subject from another - there must be some differences otherwise we are actually dealing with just a single subject!

2.2. Identity - Proof of identity

The next question is how do you prove who you are? What credentials can you present to prove your identity?

In real life, this is usually through some official documents such as:

- Driving License
- Passport

In computing terms, a user also has a set of security credentials. Some common examples include:

• username + password	The password acts as a "shared secret" that is only known to the receiver and that one user. Possession of that password proves the user is who they say they are. Unfortunately most users tend to use insecure or easily guessable passwords, or divulge their passwords through "social engineering" attacks.
• X509 certificates	X509 certificates are part of the Internet standard public / private key cryptography system. A certificate is digitally signed by an issuing authority and binds a public key to a particular subject. Possession of that subject's private key to match the public key in the certificate proves the identity asserted by the certificate.

It is technically possible to create fake security credentials in computing systems just as much as in real life - and in both cases various steps (both technical and procedural) are taken to make it hard to do this, and make it easier to detect the fakes.

2.3. Identity - Permissions

The third question is what does this identity prove about us? What does this identity allow us to do?

Some real life examples illustrate the concept here:

- Passports
  - Holding a UK passport proves someone is a UK Citizen
  - Being a UK Citizen allows that person to live and work in the UK
  - Being a UK Citizen also allows that person to live and work in any other country in the European Union
  - Losing their passport does not stop someone being a UK Citizen; it just makes it harder to prove they are a citizen.
- Driving Licenses
  - A standard driving license shows someone is allowed to drive a car
  - Someone is not allowed to drive a Heavy Goods Vehicle unless they hold a HGV Driving License

The permissions and entitlements for an identity is ultimately determined by the set of credentials that were presented to assert that identity.

Permissions and credentials are then used to make policy enforcement decisions, such as:

- Is a person allowed to drive a Heavy Goods Vehicle?
- Is a person allowed to work in the UK?

3. Web Services and Identity

3.1. Does this affect Web Services?

Security and Identity is a fundamental requirement of any real-world deployment of a Web Services application

Ultimately all security policy decisions are based on the caller's identity, so the concepts just discussed will become increasingly important to Web Services developers and users.

The challenge is to how to represent and prove a caller's identity in an open and **interoperable** way.

3.2. Security and Identity Considerations for Web Services Applications

There are several security and identity considerations that need to be borne in mind for any Web Services application:

- Caller Authentication
- Caller Authorization
- Caller Attributes

3.2.1. Authentication

Web Services authentication is about mechanisms to answer the following questions:

- Who is the caller?

- How did they prove their identity?
- Do we trust the source of these credentials?

3.2.2. Authorization

Web Services authorization is about answering the following questions:

- What is the caller allowed to do?

3.2.3. Attributes

Caller identity attributes are concerned with answers to the following questions:

- What other facts or unique attributes do we know about the caller? For example, e-mail address, display preferences, department, employee number
- How do we use this attribute information in the application? For example, customizing the data returned to the caller based on their display preferences

4. Interoperable XML Security and Identity

4.1. Requirements for Interoperable Security and Identity

To achieve interoperable security and identity, web services require standardized mechanisms of doing all of the following:

- **Representing security credential data in XML**
  - There needs to be a standardized format for representing security credentials and tokens using XML, to allow the widest interoperability.
  - For example, this can be done through the Security Assertions Markup Language (SAML) specification [SAML], or using the encoding rules for binary security tokens defined in the WS_Security specification [WSSE].
- **Obtaining credential data or security tickets**
  - For true interoperability, there needs to be a standardized mechanism to obtain security credentials or tickets in the first place. This is usually done by using a WSDL and SOAP call to an interface on a security authority service such as a single-sign-on product.
  - For example, single-sign-on services such as Microsoft Passport [.NET Passport] Liberty Alliance [Liberty Alliance] specifications can issue session ticket tokens valid for the defined period.
- **Transport credential data in a SOAP message**
  - There needs to be a way of packaging the credential data or security tokens into a SOAP message in an agreed manner, so that both client and server can find, decode and process the credentials in the necessary manner.
  - For example, SOAP header fields defined in the ws-security specification [WSSE]

4.2. Types of Security Tokens

The WS-Security specification [WSSE] defines the following types of security tokens:

- **Unsigned security tokens**
  - Username
- **Signed security tokens**
  - X.509 certificates (binary)
  - Kerberos tickets (binary)
- **XML security tokens**
  - Any XML token, such as SAML
  - Usually self verifying / signed

This leads to two main types of interaction when security credentials are being used with XML SOAP messages:

- Using non self-validating credentials - See Section 4.3.1
- Using self-validating credentials - See Section 4.3.2

The main difference between these two is whether the receiver of the XML message can verify and validate the security credentials in the SOAP message without referring back to the authority service that issued the credentials.

4.3. Types of XML Security Interaction Dialogues

4.3.1. Typical XML Security Dialogue - Non Self-Validating Credentials

The receiver of the credentials needs to query the security service to validate the credentials.

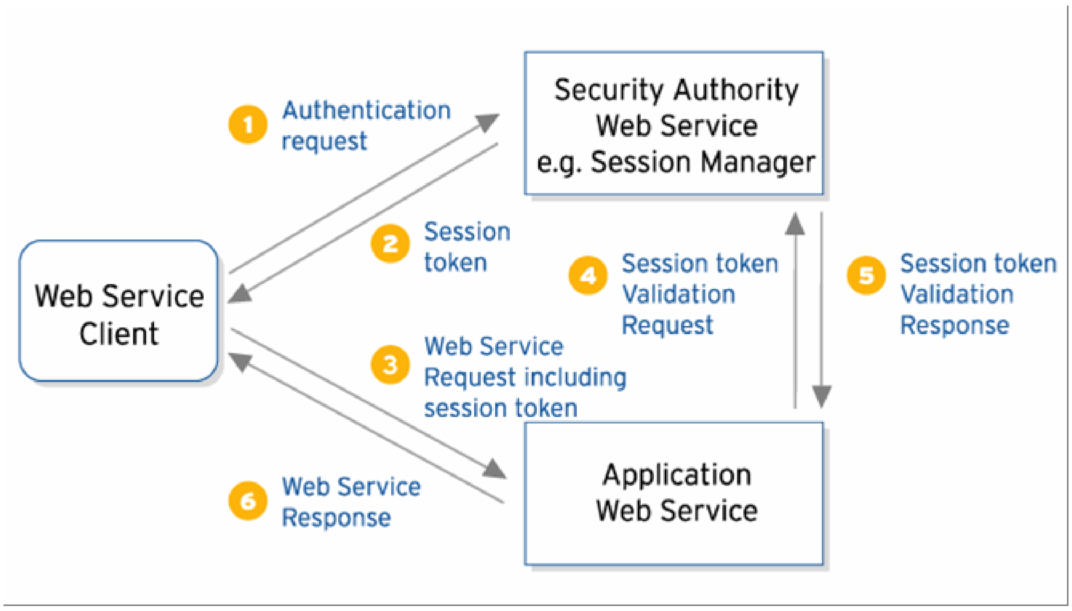
Security Authority Web Service e.g. Session Manager
1 Authentication request
2 Session token
Web Service Client
4 Session token Validation Request
5 Session token Validation Response
3 Web Service Request including session token
6 Web Service Response
Application Web Service

Figure 1. Typical XML Security Dialogue - Non Self-Validating Credentials

4.3.2. Typical XML Security Dialogue - Self-Validating Credentials

The receiver of the credentials can validate those credentials themselves, and does not need to query the security service to check the credentials.

Self-validating credentials are usually achieved by the authority service which issued the security token digitally signing the credentials using their private key, and the receiver of the credentials then verifying using the issuing authority's public key which it already trusts.

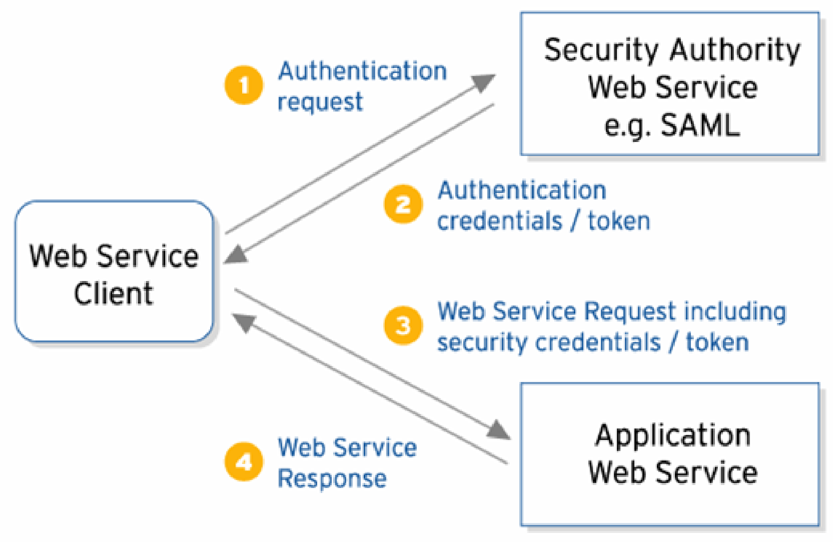
Security Authority Web Service e.g. SAML
1 Authentication request
2 Authentication credentials / token
Web Service Client
3 Web Service Request including security credentials / token
4 Web Service Response
Application Web Service

Figure 2. Typical XML Security Dialogue - Self-Validating Credentials

5. SAML v1.0

SAML is an XML-based framework for exchanging security information. It provides a standardized format for encoding credentials using XML. It is a published standard specification defined by the Organization for the Advancement of Structured Information Standards (OASIS) organization.

The SAML specification defines:

- How to represent security credentials ("Assertions" in SAML parlance) using XML
- An XML message exchange protocol for querying a SAML Authority service

The SAML specification DOES NOT define:

- How to obtail security credentials ("Assertions") in the first place

5.1. SAML Assertion Types

There are three types of assertions defined in the SAML specification [SAML]

- **SAML Authentication Assertions**
  - The results of an authentication action performed on a subject by a SAML authority

    Statements of the form: "An issuing authority asserts that subject S was authenticated by means M at time T"
- **SAML Attribute Assertions**
  - Attribute information about a subject
  - Statements of the form: "An issuing authority asserts that subject S is associated with attributes A, B and C with values 'a', 'b' and 'c'"
- **SAML Authorization Assertions**
  - Authorization permissions that apply to a subject with respect to a specified resource
  - Statements of the form: "An issuing authority decides whether to grant the request by subject S for access type A to resource R given evidence E"

6. Examples of Security Credentials in a SOAP Message

The following sections provide examples of the main type of security credentials carried in SOAP message:

- Username token - Section 6.1

  X509 binary certificate - Section 6.2

  SAML authentication assertion - Section 6.3

6.1. A Username Token in a WS-Security SOAP Header

```
<SOAP:Envelope xmlns:SOAP="...">
  <SOAP:Header>

    <wsse:Security
      xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">

      <wsse:UsernameToken>
        <wsse:Username>jthelin</wsse:Username>
        <wsse:Password Type="wsse:PasswordDigest" >XYZabc123</wsse:Password>
      </wsse:UsernameToken>

      ...
    </wsse:Security>

  </SOAP:Header>

  <SOAP:Body Id="MsgBody">
    <!-- SOAP Body data -->
  </SOAP:Body>
</SOAP:Envelope>
```

This example shows a SOAP message with a WS-Security header containing a username / password token. The token is for the user 'jthelin' in this example, and the base64-encoded copy of the SHA1 digest hash value of the password is included in the header too. To confirm these credentials are valid, the receiver of this message must querying the security authority for this user to confirm whether the enclosed password digest is correct for this user.

6.2. Binary X509 Certificate in a WS-Security SOAP Header

```
<SOAP:Envelope xmlns:SOAP="...">
  <SOAP:Header>

    <wsse:Security
      xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">

      <wsse:BinarySecurityToken  Id="X509Token"
         wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"
         ValueType="wsse:X509v3"
         EncodingType="wsse:Base64Binary"
      >
         MIIEZzCCA9CgAwIBAgIQEmtJZc0...
      </wsse:BinarySecurityToken>

      <ds:Signature
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

        <ds:SignedInfo> ... </ds:SignedInfo>
        <ds:SignatureValue> ... </ds:SignatureValue>

        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#X509Token" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>

      ...
    </wsse:Security>

  </SOAP:Header>

  <SOAP:Body Id="MsgBody">
    <!-- SOAP Body data -->
  </SOAP:Body>
</SOAP:Envelope>
```

This example shows an X.509 digital certificate being included in the WS-Security SOAP header as a binary token. The token has been signed in accordance with the XML Digital Signature specification [XML Signature], and using the key information from the X.509 certificate to calculate the signature hash.

6.3. A SAML Assertion in a WS-Security SOAP Header

```
<SOAP:Envelope xmlns:SOAP="...">
  <SOAP:Header>

    <wsse:Security
      xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">

      <saml:Assertion
        xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
        MajorVersion="1"
        MinorVersion="0"
        AssertionID="SecurityToken-mc375268"
        Issuer="mycompany"
        IssueInstant="2002-07-23T11:32:05.6228146-07:00" >
        ...
      </saml:Assertion>

      <ds:Signature
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

        <ds:SignedInfo> ... </ds:SignedInfo>
        <ds:SignatureValue> ... </ds:SignatureValue>

        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <saml:AssertionIDReference>
              SecurityToken-mc375268
            </saml:AssertionIDReference>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>

      ...
    </wsse:Security>

  </SOAP:Header>

  <SOAP:Body Id="MsgBody">
    <!-- SOAP Body data -->
  </SOAP:Body>
</SOAP:Envelope>
```

This example shows a SAML assertion being included in the WS-Security SOAP header as an XML token. The SAML token has been signed in accordance with the XML Digital Signature specification [XML Signature], and using the key information from the SAML assertion to calculate the signature hash.

7. Single-sign-on Services

Single-sign-on services provide a single point of logon and authentication. They also provide a standardized way to obtain suitable credentials to prove an authenticated identity.

The main contenders currently seem to be:

- Liberty Alliance [Liberty Alliance]

Microsoft Passport [.NET Passport]

Proprietary security products such as Netegrity SiteMinder which are added direct SAML interfaces

WS-Trust - A new specification to provide a standardized interface[WS-Trust]

This still remains an area needing real standardization, and is an area that Web Services Interoperability Organization (WS-I)[WS-I] is now turning its attention to through it's Web Services Basic Security Profile.

7.1. Liberty Alliance

The Liberty Alliance Project [Liberty Alliance] is a cross-industry group aiming to establish an open standard for federated network identity

The Liberty specification v1.0 has two main facets:

- Single sign-on
- Identity federation

7.2. Microsoft .NET Passport

Microsoft .NET Passport [.NET Passport] is a suite of Web-based services that makes using the Internet and purchasing online easier and faster for users.

Microsoft has previously declared plans to upgrade the current Passport facilities to:

- Provide an XML interface
- Support federation
- Use Kerberos v5 as the underlying mechanism for securely exchanging credentials

7.3. WS-Trust

WS-Trust [WS-Trust] is a proposed specification in the WS-Security family to provide a standardized way to acquire security tokens from an authority service.

WS-Trust is still very early in the standardization process, but is the most likely candidate for a common interface to security authorities for the future.

8. Identity-awareness in Web Services

Question: Do web services themselves need to be identity-aware?

Answer: Not really, in most cases.

A mature web services platform product such as CapeClear 4 Server [CC4 Server] can handle almost all the "boilerplate" work of authentication and enforcement of access control lists. This is typically done through configuring interceptors or plugins on a deployed web service to implement the required security policy.

Most standard authentication and authorization functions are best done in a uniform manner by the platform, rather than being implemented on an application-by-application basis. Interceptor plugins allow this to be done as a deployment policy decision, rather than as an implementation decision.

A Web Service application only needs to be Identity-aware if it needs to make use of user-specific attributes asserted for the caller. For example, reading the delivery address from the user's MS Passport record.

9. Security in the Ultimate Web Services Platform

The ultimate goal will be to have **declarative** security functions for web services just like we already have for Enterprise Java Beans (EJB) systems now.

So, we want to get to the situation of being able to make declarative statement of a web service applications security requirements in the following areas:

- Permitted authentication realms / single-sign-on services
- Required transport security attributes (for example, "Callers must use encrypted / SSL connections")
- Required message security attributes (for example, "Messages must be digitally signed")
- Role-based access control lists applied at the granularity of the operation / method call.

There are some moves afoot to add some of this declarative security information into WSDL files, but there are no real standards for this at the moment. Watch this space though.

This places control of security with application **administrators** rather than developers, and allows a uniform way to apply a security policy to enterprise web services.

10. Summary

"Identity" is one of the fundamental concepts underpinning all Web Service security mechanisms, and will become increasingly important as more enterprise web services get deployed. Having a standard XML-based serialized form of credentials is vital for true end-to-end interoperability.

Standardization of specifications for credential exchange and single-sign-on using XML and SOAP are still incomplete, so widespread interoperability is not yet possible. However, this is such an important area that it is the next thing that WS-I are developing an interoperability profile for.

It is best to use a mature Web Services runtime platform such as Cape Clear Server to handle most "boilerplate" security tasks such as enforcing authentication and authorization requirements.

Bibliography

[CC4 Server] Cape Clear 4 Server - Enterprise-grade Web Services Integration Platform from Cape Clear Software Inc. http://www.capeclear.com/products/server/

[Kerberos] J. Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5)," RFC 1510, September 1993. http://www.ietf.org/rfc/rfc1510.txt

[Liberty Alliance] Liberty Alliance Project http://www.projectliberty.org/

[.NET Passport] Microsoft .NET Passport http://www.passport.com/

[SAML] SAML - Security Assertions Markup Language v1.0, OASIS XML-Based Security Services Technical Committee, 05-Nov-2002 http://www.oasis-open.org/committees/security/#documents/

[SOAP] W3C Note, "SOAP: Simple Object Access Protocol 1.1", 08-May-2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508

[WS-I] WS-I - The Web Services Interoperability Organization http://www.ws-i.org/

[WSSE] WS-Security specification v1.0, 05-Apr-2002 http://schemas.xmlsoap.org/specs/ws-security/ws-security.htm

[WSSE-XML] WS-Security Profile for XML-based Tokens v1.0, 28-Aug-2002 http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-security-xml-tokens.asp

[WS-Trust] WS-Trust specification v1.0, 18-Dec-2002 http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-trust.asp

[XML Signature] W3C Recommendation, "XML Signature Syntax and Processing", 12 February 2002. http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/

[XML Encryption] W3C Recommendation, "XML Encryption Syntax and Processing" 10 December 2002. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/

Glossary

EJB	Enterprise Java Beans
OASIS	Organization for the Advancement of Structured Information Standards
SAML	Security Assertions Markup Language
WS-I	Web Services Interoperability Organization

Biography

Jorgen **Thelin**
Chief Scientist
Cape Clear Software Inc.
London
United Kingdom
Jorgen.Thelin@capeclear.com

Jorgen Thelin is the Chief Scientist at Cape Clear Software Inc, where he focuses on Web Services technology and standards. Previously, he has worked on a number of major Java-based projects for blue-chip companies such as Reuters, J.P.Morgan, and BSkyB before moving to Orbware to design and build the Orcas EJB server. He holds a Computing Science degree from Stirling University, Scotland, and an MBA from Warwick Business School, England. He has been using the Java programming language since early 1996, and is a Sun Certified Java Programmer, Developer, and Architect.