Enterprise Security With XML and Web Services

Konstantin Beznosov
konstantin.beznosov@quadrasis.com
www.beznosov.net/konstantin

Quadrâsis
September 17, 2002
Konstantin Beznosov

- Ph.D. “Engineering Access Control for Distributed Enterprise Applications”
- Security standards
  - CORBA Security
  - “Resource Access Decision” (RAD) Facility
  - “Security Domain Membership Management”
  - Extensible Access Control Language (XACML)
- Security Architect
  - with Baptist Health, Concept 5, Quadrasis (HICAM)
  - Architecture, design and implementation of enterprise security solutions and products using CORBA, EJB, COM+, .NET
It Covers

1. Principles of Securing Web Services
   - general requirements and solutions for implementing authentication, data protection, and access control for web services
   - XML encryption and signature, as well as WS-Security

2. Security Mechanisms for Web Services based on Java and .NET
   - building blocks for securing Java and .NET web services
     - authentication, data protection, access control, and audit

3. Planning, Building and Administering Secure Web Service Systems
   - points to pay attention to when designing and developing enterprise solutions based on web services
     - integrating security of perimeter, middle and back-end tiers used in web services systems, providing common security environment to heterogeneous applications, and propagation of security context through various intermediates along the passage of SOAP requests.

◆ Coffee break 10:30—11:00
It Does Not Cover

- XML & SOAP details
- SOAP
- Security of CORBA, .NET, COM(+), EJB
Disclaimer

- The technology is new ⇒ I reserve the right
  - to be wrong and incomplete
  - to not know all the answers
- The tutorial is
  - limited in time and
  - does not cover everything on the subject
- It’s my personal view
- This is not a “how-to” cook book
Assumptions

- You are familiar with
  - XML
  - SOAP
  - Web Services
  - Security for distributed systems
Part I
Principles of Securing Web Services
Briefly About Web Services
What's a Web Service System?

SOAP

Web Service System
Promise of Web Services

- Interoperability across lines of business and enterprises
  - Regardless of platform, programming language and operating system

- End-to-end exchange of data
  - Without custom integration

- Loosely-coupled integration across applications
  - Using Simple Object Access Protocol (SOAP) and Extensible Markup Language (XML)
Features

XML-based messaging interface to computing resources that is accessible via Internet standard protocols

- WS help *intranet* (business units) and *extranet* (business partners) *applications* to communicate
- SOAP – format for WS communications
  - Defined in XML
  - Supports RPC as well as document exchange
  - Stateless
  - No RPC predefined RPC semantics
  - Can be sent over various carriers: HTTP, FTP, SMTP, … postal service, e.g., Swiss Post
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/06/soap-envelope">
    <env:Header>
        <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
            <n:priority>1</n:priority>
            <n:expires>2001-06-22T14:00:00-05:00</n:expires>
        </n:alertcontrol>
    </env:Header>
    <env:Body>
        <m:alert xmlns:m="http://example.org/alert">
            <m:msg>Pick up Mary at school at 2pm</m:msg>
        </m:alert>
    </env:Body>
</env:Envelope>
Typical Web Service Environment

- **Client Tier**
  - Application Client
  - Browser
  - Java Program

- **Presentation Tier**
  - Web Servers

- **Component Tier**
  - Web Services
  - J2EE
  - CORBA
  - ASP.NET
  - COM

- **Back-Office Tier**
  - Mainframes
  - Databases
Security Obstacles to Web Services

◆ Web Services have a huge problem
  – They are too open
  – Requests via HTTP pass through firewalls

◆ Critical corporate information exposure risks
  – Unauthorized access to valuable resources
    ◦ Patient records, Credit card numbers, Manufacturing designs
  – Unauthorized use of service
  – Tampering with or bringing down the service

◆ Protecting Web Services
  – Securing XML documents
  – Securing remote procedure calls (RPC)

◆ Implementation obstacles
  – Bridging to back-office databases
  – Cross-domain Single Sign-On (SSO)
  – Different companies use different security products

◆ Companies may not open up corporate networks
  – Need to implement proper countermeasures
Web Services and Security
WS Security Building Blocks

SOAP Security
WSDL Security
UDDI Security
XML Security
WS-Security
Web Servers Security
Middleware Security
Mainframe Security
Database Security
Network Security
Directories
Firewalls
IDS
Physical Security
SOAP Message Vulnerabilities

Message related risks
- Ill-formed message potentially damaging
- Modification & eavesdropping in transit
- Sent from non-trusted sources

Services related risks
- Access by unauthenticated users
- Use of unauthorized services
- Unaccountable use of service

Interoperability risks
- Vulnerabilities from lack of interoperability with currently deployed security products
- Insecure application-to-application message exchange
Web Usage Scenario - Security

Trader’s System

SOAP Ultimate Sender

Notary System

SOAP Intermediary

Buy 5000 shares MSFT @$XX/share

Trading System

SOAP Ultimate Receiver

Bought 5000 shares MSFT @$YY/share
Securing SOAP: Requirements

◆ Inspection of Messages
  – Message Validation
  – Message Integrity
  – Message Origin

◆ Integration with Enterprise Security
  – Authentication
  – Authorization
  – Audit
  – Administration
  – Interoperability

Encryption of SOAP Message Is NOT Sufficient
Changes in the Security Picture

- WS open enterprise resources to outside world
- New security responsibilities due to mixing lines of business:
  - Outsourcing credit card authorization service
  - Cross-selling and customer relationship management
  - Supply chain-management
- Risk must be assessed and managed across a collection of organizations
- Interactions are more complex and take place among diverse environments
Simple Example of Securing a Web Service
Hello, bcustomer1

Product Category: All

Blue Milk
Description: It looks blue from outside
Price: Store ID: 0

Your Shopping Cart
<table>
<thead>
<tr>
<th>Qty</th>
<th>Product</th>
<th>Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Blue Milk</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Username: 
Password: 
Login

Recalculate  Checkout  Clear Cart
Simple Security Solution

- Web browser
- ePortal.aspx
- IIS
- ePortal.com or Internet customers

- ePortal.aspx
- ASP.NET
- HTTP POST
- ePortal.com

- SOAP server
- ASP.NET
- HTTP Basic Authentication
- SSL

- DCOM
- Accounts
- eBusiness.com

- Middle Tier Server
- COM+

Username: bcustomer1
Password: **********
Login  Logout
Limitations

- Only homogenous solutions
- Relies on Web server security – risky
- End-to-end all-or-nothing encryption with SSL
- Password-based authentication
- Unrestricted delegation

- ePortal is trusted to pass passwords from clients to eBusiness
- OS account for each customer
- Coarse-grain authorization
- Weak accountability
Web Services Security: Advanced Principles
**Authentication**

Requirements: Authenticate initiator, Authenticate intermediaries

Connection-oriented

- Types
  - Password
  - Challenge-response

Implementation Options

- OS
- Web server
- Token
- Web SSO
  - Commercial Products
  - Liberty Project
  - Microsoft Passport
- Client-server SSO

Message-oriented

- Digital signatures
  - XML signatures or S/MIME
  - Require PKI
  - Unclear intent of signature in a SOAP message

- SAML assertions
Data Protection

Requirements: confidentiality, integrity, authenticity

Connection-oriented
- Protects only in transit
  Implementation options
  - SSL
  - IPSec
  Features
  + Easy to implement
  + Comodity
  - All or nothing
  - No protection from intermediaries

Message-oriented
- Protects in transit and in storage
  Implementation options
  - XML Encrypt/Sign
  - S/MIME
  Features
  - Hard to implement
  - Very new
  + Could be fine grain
  + Protection from intermediaries
## Access Control

**Requirements:** fine grain, complex principals

### Consideration

- Semantics of binding user attributes with the context
- Multiples requests in one message
- Choice of authorization mechanisms is constrained by authentication ones

### Options

- Operating system
- Web server
- Application server
- Application
XML Security
XML Encryption

◆ Functionality

- Encrypt all or part of an XML message
- Separation of encryption information from encrypted data

```xml
<EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
    Type='http://www.w3.org/2001/04/xmlenc#Content'>
    <EncryptionMethod Algorithm='http://www.w3.org/2001/04/xmlenc#3des-cbc'/>
    <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
        <ds:KeyName>John Smith</ds:KeyName>
    </ds:KeyInfo>
    <CipherData>
        <CipherValue>A23B45C56</CipherValue>
    </CipherData>
</EncryptedData>
```
XML Signature

- Apply to all or part of a document
- Contains: references to signed portions, canonicalization algorithm, hashing and signing algorithm IDs, public key of the signer.
- Multiple signatures with different characteristics over the same content

```xml
<Signature Id="MySignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
    <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126"/>
      <Transforms>
        <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      </Transforms>
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk="</DigestValue>
  </Reference>
</SignedInfo>
<SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
<KeyInfo>
  <KeyValue>
    <DSAKeyValue>
      <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
    </DSAKeyValue>
  </KeyValue>
</KeyInfo>
</Signature>
```
Security of SOAP Messages
Gaps

- Signature and Encryption specifications are for XML not SOAP
  - Format and location of security information in SOAP message
  - Targeting specific actors
  - Support for multiple security operations
- Passing security-related client information
  - Authentication
  - Attributes
WS-Security

- Message integrity and message confidentiality
- Compliance with XML Signature and XML Encryption
- Encoding for binary security tokens
  - Set of related claims (assertions) about a subject
  - X.509 certificates
  - Kerberos tickets
  - Encrypted keys
SOAP Message with WS-Security

```xml
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2001/12/soap-envelope"
              xmlns:sig="http://www.w3.org/2000/09/xmldsig#"
              xmlns:enc="http://www.w3.org/2001/04/xmlenc#">
  <env:Header>
    <sec:Security>
      <sec:actor="http://www.w3.org/2001/12/soap-envelope/actor/next"
              sec:mustUnderstand="true">
        <sig:Signature>
          ...
        </sig:Signature>
        <sec:BinarySecurityToken
          ValueType="sec:X509v3"
          EncodingType="sec:Base64Binary"
          Id="X509Token">
          MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh5i...
        </sec:BinarySecurityToken>
      </sec:Security>
    </env:Header>
    <env:Body>
      <enc:EncryptedData>
        ...
      </enc:EncryptedData>
    </env:Body>
  </env:Envelope>
```
Web Services Security Roadmap

- Joint IBM and Microsoft White Paper
- Goal: “enable customers to easily build interoperable solutions using heterogeneous systems”
- Initial specifications:
  - WS-Security
  - WS-Trust
  - WS-Policy
  - WS-Privacy
  - WS-Federation
- Follow-On Specifications:
  - WS-SecureConversation
  - WS-Federation
  - WS-Authorization
Current Status

Less Mature

More Mature

- W3C XML Signature is a W3C recommendation
- W3C XML Encryption is a candidate recommendation
- IBM/Microsoft Web Services Security Roadmap is published
- WS-Security published and moved to OASIS
Part II

Security Mechanisms for Web Services

based on

Java and .NET
Options for Building MS WS

1. Publish COM+ component as SOAP Endpoint
   - Only Windows.NET and XP Pro
   - Limitations on what COM+ components could be published
   - Might be not 100% interoperable with other SOAP implementations

2. Use CLR remoting over SOAP/HTTP
   - Supports (non-interoperable) passing object references
   - Supports client and server-activated objects
   - Can be hosted by IIS
   - Vague on client authentication and channel protection, unless IIS security is used

3. Generate COM Wrapper
   - Good way to reuse existing COM components
   - No support for custom types
   - No .NET framework in the picture

4. Use ASP.NET Mechanisms
   - Claimed to be interoperable with other SOAP-compliant web services
   - Leverages .NET, ASP.NET and IIS security mechanisms
ASP.NET Web Services Security
Ways to Do Authentication

- **IIS**
  - Basic (over SSL)
  - Digest
  - Integrated Windows (NTLM, Kerberos)
    - Only for homogenous IE/Windows clients
  - X.509 Certificates over SSL
- **ASP.NET**
  - ASP.NET authentication services
  - Custom HTTP authentication modules
- **SOAP**
  - Credentials in the SOAP header
    - Application-specific
    - According to MS WS-Security Language
    - SAML identity assertion
ASP.NET Custom HTTP Modules

**Advantages**
- Allows custom authentication schemes
- Allows decoupling (HTTP) transport from SOAP
- Makes application security-unaware
- Supports CLR authorization

**Disadvantages**
- Couples client and server
Message Authenticity, Integrity, Confidentiality Protection

◆ IIS
  ∟ HTTP/SSL –
    – Mature commodity
    – Only point-to-point (i.e. no end-to-end)

◆ SOAP message
  ∟ XML encryption and digital signature as defined by W3C[5]
  ∟ WS-Security language as defined by Microsoft[6]
  ∟ End-to-end
  ∟ Very new
  ∟ (almost) no tool support
Different Ways to Control Access

- IIS (Web Server)
  - Restrictions based on client IP address or DNS name
- Port restrictions based on client IP address or DNS name
- Windows OS ACLs
- ASP.NET “URL authorization”
- ASP.NET HTTP Modules
- .NET CLR
  - Up to per method level on a class per user ids or roles
  - Supports CLR identities
Audit: IIS

Extended Logging Properties

General Properties

New Log Time Period:
- Hourly
- Daily
- Weekly
- Monthly
- Unlimited file size
- When file size reaches:
  - 13 MB

Use local time for file naming and rollover

Log file directory:
\%SystemRoot%\System32\LogFiles

Log file name: IIS.log

Extended Properties:

- Time (time)
- Service Name (systemname)
- Protocol Status (sc-status)
- Win32 Status (sc-win32-status)
- Errors (sc-errors)
- Bytes Sent (sc-bytes)
- Bytes Received (sc-bytes)
- Server Name (sc-computername)
- Server Port (sc-port)
- Method (sc-method)
- URI Stem (sc-uri-stem)
- URI Query (sc-uri-query)
- User Name (cs-username)
- Client IP Address (ci-ip)

Extended Logging Options

OK Cancel Apply Help

Apply Cancel Help OK
Audit: .NET

```csharp
using System.Diagnostics;
using System.Threading;

class LoggingSample{

    public static void Main(){
        // Create the source, if it does not already exist.
        if(!EventLog.SourceExists("MySource")){
            EventLog.CreateEventSource("MySource", "MyNewLog");
            Console.WriteLine("Creating EventSource");
        }

        // Create an EventLog instance and assign its source.
        EventLog myLog = new EventLog();
        myLog.Source = "MySource";

        // Write an informational entry to the event log.
        myLog.WriteEntry("Writing to event log.");
    }
}
```
Summary

- Four ways to build XML web services using .NET
- ASP.NET web services are built on top of
  - ASP.NET
  - .NET
  - IIS
  - Windows
- ASP.NET web services have mechanisms for
  - Authentication
  - Data protection
  - Audit
  - Service continuity
- Several routes to protect your .NET Web services
Java Web Services Security
What’s Coming

- The J2EE 1.4 platform = the J2EE 1.3 platform + WS Pack and more

- Web Services Pack:
  - Java APIs for XML
  - Tomcat “JSP™ container”
  - A set of standard JSP tag libraries
  - JavaServer Faces™ UI Kit for building complete web-based UI (JSR-127)
Java APIs for XML

- Java API for XML Messaging (JAXM)
- Java API for XML Processing (JAXP)
- Java API for XML Registries (JAXR)
- Java API for XML-based RPC (JAX-RPC)
- Java API for XML Data Binding (JAXB)
JAX-RPC Architecture

- JAX-RPC Client
- JAX-RPC Client-side Runtime System
- Generated Proxy Code
- Container

- J2EE Server
- JAX-RPC Server-side Runtime System
- Container

- WSDL
- WSDL -> Java
- WSDL <-> Java
EJB Run Time Security

Client address space (JVM)
- EJB object stub
- Caller Identity

Container address space (JVM)
- EJB object
- Enterprise Bean instance
- AccessControlEntries
- Bean Identity
- Container
- EJB server
Upcoming J2EE APIs

- Compiling XML schemas into Java classes
- Sending and receiving SOAP messages
- Making XML-based RPCs
- Handling WSDL files
- Exchanging SAML assertions
Part III
Planning, Building and Administering
Secure Web Service Systems
What Security Means To You

- Examine business-level drivers for security
  - Cross-company security agreements
- Determine level of trustworthiness
  - Your architecture and its implementation are as secure as you want it
Traditional TCB

- Tamper-proof
- Nonbypassable
- Small enough to be thoroughly analyzed
“Distributed TCB”

- Tamper-proof
- Nonbypassable
- Large and difficult to be analyzed
Recommended Approach

- Consistent with TCB principles
- Simplifies the analysis

- Leave security to experts
  - Security COTS integration vs. do-it-yourself
    - More thoroughly tested by other customers
    - More careful about common development mistakes

- Follow good architectural and policy design principles
- Plan for evolution and manageability
  - Have a security framework
Security Architecture Principles

◆ Trust no one
  ■ Don’t’ make your firewall the only point of enforcement
  ■ View Web Services collections as mutually suspicious islands

◆ Enable interoperability
  ■ Use vendor-neutral standards (WS-Security, SAML)

◆ Modularize security
  ■ “Push” security down – security unaware applications
  ■ Insulate applications from security functionality with stable APIs
Security Policy Principles

- Authentication: balance cost against threat
  - SSO
- Authorization: application-driven
  - Use the business of the application to drive authorization settings
- Accountability: audit early, not often
  - “pop” audit into/near the application
- Security administration: collections and hierarchies for scale
Enterprise Application Security Integration (EASI) Framework
Patchwork of Security - Vulnerabilities

Multiple Protocols

Perimeter Security
First line of defense: Protection against External Hackers
- Firewalls/VPNs
- Cryptography
- Web Single Sign-on (SSO)
- Intrusion detection

Mid-tier Security
Second line of defense: Protection against Insider Attacks
- Component-based security
- Cryptography
- Entitlements servers

Multiple Platforms

Back-office Security
Third line of defense: Protection of Back-end Servers
- Mainframe security
- Database security

Multiple Application Servers

Multiple Security Technologies

Multiple Tiers of Security
EASI Solutions Consist of

1. Security framework
2. Hardware and software products for securing applications
3. Integration “modules” to plug applications and security products into the framework
EASI Framework Architecture

Enterprise Security Integration Framework

Security APIs

- Standard Security APIs
- Custom Security APIs
- Vendor Security APIs

Core Security Services

- Authentication
- Authorization
- Cryptography
- Accountability
- Security Administration

Authentication
- Products

Authorization
- Products

Cryptography
- Products

Accountability
- Products

Security Administration
- Products

Framework Security Facilities

- Profile Manager
- Security Association
- Proxy Services
Specific Example of EASI: Quadrasis

Client Tier
- Application Client
- Browser

Presentation Tier
- Web Servers

Component Tier
- Web Services
- Application Servers
- Back-Office Tier
  - Mainframes
  - Databases

EASI Application Environment Adapters
- Authentication API
- Authorization API
- Audit API
- Cryptography API

EASI Executive

Security Services
- Authentication Services
- Authorization Services
- Cryptography Services
- Accountability Services
- Security Administration Services

Security Management
- Security Administration
- Security Configuration
- Security Policy
EASI Pros and Cons

- Common security infrastructure shared across the enterprise
- Decoupling applications from products
- Well defined boundary between business and security logic
- No need to implement everything at once

- Complex due to generality
- Performance and scalability constraints
- Significant initial effort in designing and building it
- Has to be politically accepted in many different “parties” of organization
- Semantic mismatch among security products makes their “swapping” hard
Administering Secure Web Services
How WSS Administration Different

◆ As any other middleware
  ■ Scaling with collections and hierarchies
    – User attributes
    – Domains
    – Permission collections
  ■ Effective authorization models
    – RBAC policies
    – Rule-based policies

◆ Data protection
Role-based Access Control

- Users
- Roles
- Sessions

- Role Hierarchy
- User-to-role assignment
- Session roles

- Permissions
- Operations
- Objects

- Permission-to-role assignment
Role Hierarchy for eBusiness

- member
- visitor
- customer
- staff
RBAC Gotchas

- Engineering of roles in large systems is hard
  - More roles $\implies$ less effective administration
  - Fewer roles $\implies$ unnecessary permissions are granted
  - Exception cases $\implies$ superfluous roles
- Roles tend to be used for everything
  - “18-year-or-older-reader”
- Roles are assigned statically
Selecting Factors for RBAC


- Support role hierarchies
- Dynamic determination of roles
- Rule for exception cases and for other attributes
- Integrated with the organizational workflow to track changes in user roles
Data Protection

- Connection-oriented
  - (relatively) Easy to administer

- Message-oriented
  - Can be fine grain \(\Rightarrow\) administration nonscalable
  - No reasonable administration models
  - Becomes very complex in non-trivial cases
  - Stay away if you can
Example
Functional Security Requirements

**Common**
- Limit visitor access
- Secure exchange with business partners
- Grant members more access
- Eliminate administration of new customers
- Protect the accounts of each individual
- Administrator control of critical functions
- Restrict administrators’ abilities

**ePortal.com**
- Eliminate administration of new customers

**eBusiness.com**
- Protect the accounts of each individual
- Administrator control of critical functions
- Restrict administrators’ abilities
# EASI Framework for ePortal.com

## ePortal.com Enterprise Application Security Integration Framework

<table>
<thead>
<tr>
<th>Security APIs</th>
<th>ASP.NET, COM+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom Self-Registration</td>
<td></td>
</tr>
<tr>
<td>Web SSO, SAML Service</td>
<td></td>
</tr>
</tbody>
</table>

### Core Security Services

- **Authentication**
- **Authorization**
- **Cryptography**
- **Accountability**
- **Security Administration**

### Security Facilities

- **Active Directory Service**
- **WS-Security/SAML Service**

### Additional Security Features

- Firewall
- Intrusion Detection System
- Web SSO
- COM+
- Windows 2000
- SSL
- Custom Self-Registration Module
EASI Framework for eBusiness.com

eBusiness.com Enterprise Application Security Integration Framework

Security APIs
- JAAS, EJB
- Oracle Security, SAML Service

Core Security Services
- Authentication
- Authorization
- Cryptography
- Accountability
- Security Administration

Framework Security Facilities
- iPlanet Directory Service
- WS-Security/ SAML Service

Security APIs
- Firewall
- iPlanet
- WebLogic
- SSL
- Oracle
- Attribute Mapping
Security Gotchas at the System Architecture Level

◆ Scaling
  ■ Distribute requests over multiple security policy servers
  ■ Central administration
  ■ Administration delegation

◆ Performance – “No free lunch”
  ■ Encryption algorithms
  ■ Underlying transport
  ■ Policy granularity
  ■ Caching
For More Information

Appendix
What Information Security Means

Protection
- Authorization
- Accountability
- Service Continuity
- Data Protection
- Non-Repudiation
- Audit
- Availability
- Disaster Recovery
- Access Control
- Design Assurance
- Development Assurance
- Operational Assurance

Assurance

Authentication

Cryptography
# Comprehensive Message Security

A secured SOAP message includes the following features:

- **SOAP Header**
  - **WS-Security**
  - **SAML Token**
  - **XML Signature, DSIG**
  - **X.509 Certificate**

- **SOAP Body**
  - **XML Encryption**
  - **RPC Method Authorization**
  - **Audit**
  - **XML Schema Verification**

## Security Features

<table>
<thead>
<tr>
<th>Security Feature</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAP Header</td>
<td>- Attaches signature, encryption, security tokens to SOAP messages</td>
</tr>
<tr>
<td>WS-Security</td>
<td>- Authenticates initiator of SOAP request</td>
</tr>
<tr>
<td>SAML Token</td>
<td>- Enables role based authorization</td>
</tr>
<tr>
<td></td>
<td>- Time-limited</td>
</tr>
<tr>
<td></td>
<td>- Interoperable</td>
</tr>
<tr>
<td>XML Signature, DSIG</td>
<td>- Multiple signed areas of header and body</td>
</tr>
<tr>
<td></td>
<td>- Integrity protection via PKI based cryptography</td>
</tr>
<tr>
<td></td>
<td>- Prevents tampering</td>
</tr>
<tr>
<td>X.509 Certificate</td>
<td>- Encryption and signature verification</td>
</tr>
<tr>
<td>(or other security token)</td>
<td></td>
</tr>
<tr>
<td>SOAP Body</td>
<td>- Multiple encrypted areas of body</td>
</tr>
<tr>
<td></td>
<td>- Prevents disclosure</td>
</tr>
<tr>
<td>RPC Method Authorization</td>
<td>- Prevents unauthorized call to methods</td>
</tr>
<tr>
<td>SOAP Message</td>
<td>- Validates against XML schema</td>
</tr>
<tr>
<td>XML Schema Verification</td>
<td>- End-to-end tracing, Method access</td>
</tr>
<tr>
<td>Audit</td>
<td></td>
</tr>
</tbody>
</table>

**Secured SOAP Message**

```xml
<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>
    <WS-Security>
      <SAML Token/>
    </WS-Security>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body/>
</SOAP-ENV:Envelope>
```