A Public Web Services Security Framework Based on Current and Future Usage Scenarios

J. Thelin
Chief Architect, CapeConnect
Cape Clear Software, Inc.

P.J. Murray
Product Manager, CapeConnect
Cape Clear Software, Inc.

Abstract—**This paper discusses the security implications of Web Services and proposes a framework for providing security based on current and future requirements. The framework provides a basis for achieving end-to-end security for Web Services within the pre-existing security environment. Finally, lessons from initial experiences with Web Services security and advice for the future are provided.**

Keywords: **Web Services, XML, Security, Authentication, Authorization, Encryption.**

1 Web Services and Security

Web Services require stronger security than Web sites. They expose functionality (typically business logic) in an open, standardized way. This implies that they are more vulnerable than when business processes were exposed in proprietary ways. This means that security will become an automatic part of any Web Services development. In addition, Web Services will be interwoven with existing applications, so the Web Services security must also accommodate existing security infrastructure. The new Web Services security software and protocols are interesting, but suffer from immaturity, lack of widespread adoption (no critical mass), and lack of technical staff with specific knowledge. The first wave of Web Services, and the products used to build them, have used well-known and accepted security technology (such as access control and authentication) that have been borrowed from the Internet and the World Wide Web. However, Web Services have not reached beyond the requirement for basic security. The objective of this paper is to describe a public framework for Web Services, based on an analysis of current and near-future usage scenarios for Web Services.

1.1 Current Usage Scenarios

There is a long-term vision for Web Services where there will be "millions of Web Services" commercially available to consumers and organizations will use Web Services to expose systems to customers and partners. However, in the meantime, the most immediate use of Web Services is for tactical projects that rely on the technical advantages offered by Web Services:

- Enterprise Application Integration: SOAP can be used to integrate Java and EJBs with logic deployed in other enterprise systems such as CORBA and .NET. The best initial projects for Web Services in organizations often involve the reuse of existing back-end systems – with Web Services used to expose them in a new way. This approach has the added benefit that the focus of the project has been the Web Services rather than developing some new business logic. For internal integration, the security implications for this have tended to depend on factors such as the sensitivity of the internal information being passed around and whether the information ever moves beyond the internal firewall at any point (which can happen if, for example, branch offices are connected over the Internet).
- Exposing back-end logic to multiple types of clients at the same time, such as Visual Basic and Java GUIs. Many projects have an attractive value proposition for using mainstream developers (Visual Basic programmers, for example) to develop the front-end clients while reserving the EJB programmers (a relatively small percentage of the very best software developers) for developing business logic. The security implications for this have tended to depend on factors such as the sensitivity of the internal information being passed around (with authentication and access control

being common security solutions) and whether the information ever moves beyond the internal firewall at any point (which can happen if, for example, branch offices are connected over the Internet) SSL has tended to be used in the cases where deployments have been across a combination of intranets and the Internet.

- Deploying applications across firewalls: SOAP (when HTTP is used as the transport layer) can be used to integrate applications or clients across firewalls. This has been particularly useful for projects deadlines that need to avoid the organizational issues usually involved with firewalls. This also has been useful for projects that involved integrating with business partners with heterogeneous firewall security requirements. The security implications of what is essentially a shortcut are often ignored due to tight deadlines.
- EJB Component Reuse: The UDDI repository can be used by organizations to make their existing business systems available for reuse within their organizations. The value proposition to organizations for such projects is not just the rapid return on investment but also new opportunities. Because this is for internal use, organizations to date have been happy with the various user identification systems in the UDDI registries.
- Web Services technology allows organizations to expose existing (business) logic for reuse in ad-hoc EAI projects. This is done by generating WSDL for existing logic (typically component-based logic such as Java, CORBA, or Enterprise JavaBeans) and registering them in a UDDI registry. An EAI project can then be reduced to looking up the registry for a suitable service. An example is a company implementing the logic for credit card validation once, but making it available for reuse anywhere it is needed. The security implications for such projects have tended to be as varied as the projects.

1.2 Emerging Uses of Web Services

With current Web Services technology, there are still higher-order integration problems that have yet to be fully solved with Web Services, such as data transformation, business logic integration, synchronous/asynchronous issues, and so on. The next generation of Web Services implementations, which will address these higher-order integration problems, will be generally driven by business needs (rather than tactical projects, which are based on Web Services technical advantages). It is no coincidence that these projects generally have considerably great security implications than the initial ad-hoc uses of Web Services.

The emerging uses of Web Services include:

1.2.1 Point-to-point System Integration

Web Services are ideal when 'Lite' internal integration needs exist within an organization. 'Lite' integration is the transfer of data between two or more systems. A typical scenario is when a company's employee information needs to be passed into various downstream applications.

The threshold, however, stands at more complex integration technology: for example, transaction processing, business process automation, and so on. Web Services excels at communicating data, but currently not at operational processing. When composition of business services is required in a single atomic operation with complex workflow, Web Services do not yet provide such mechanisms.

The security implications for such point-to-point integration projects will largely depend on factors such as the sensitivity of the internal information being passed around and whether the information ever moves beyond the internal firewall at any point (which can happen if, for example, branch offices are connected over the Internet).

Simple communication security technology such as SSL is usually sufficient to address the security problems here.

1.2.2 Enterprise Application Integration

Bridging across a complex architecture comprised of multiple systems residing on multiple platforms using different object models based on different programming languages has previously required complex and expensive EAI technology, but Web Services provides a more effective communication technology for this than traditional EAI technology.

However in many instances, Web Services currently lack many of the enterprise features of an EAI solution, especially around process management, transactions, administration, and so on, although this will change over time.

The security implications for such technology integration projects will probably be the most critical technical issue. There are currently no standards for mapping security features across all the different possible technologies being integrated, and this is even true when using established EAI technology to some extent.

Web services platform products are now starting to provide a unifying security layer when integrating disparate technologies by including implementations of all the basic security features such as user authentication, access control, activity auditing and reporting that are required for enterprise applications.

1.2.3 Technology Integration

One of the largest categories of usage scenarios for web services at the moment is about the integration of diverse applications build on various different implementation technologies – i.e. true technology integration. This can involve such simple things are Microsoft VB clients talking to Java EJB systems – something that just 12 months ago was considered virtually impossible to achieve.

Crossing a technology gap such as this usually highlights a corresponding security gap that needs to be addresses also.

So for example, a Microsoft VB (Visual Basic) program will most likely be obtaining user identity information from the Windows ActiveDirectory system and the native NT Authentication scheme, while a Java program this VB program needs to talk to may be using JAAS (Java Authentication and Authorization Services) technology to access an LDAP repository and the EJB (Enterprise JavaBeans) declarative security system to control access.

Web service platforms and security product vendors typically need to address the security gap associated with the technology gap being bridged in one of two ways:

- Use products and technology that can "map" credentials and user information between the different security schemes (e.g. mapping Windows ActiveDirectory credentials to LDAP credentials). This can obviously prove increasingly harder as the number of technologies being used increases. This is where products such as Quadrasis' EASI product can add great value in an organization.
- Provide a unifying security layer in the web services platform that to a large extent can replace the other existing security control mechanisms.

1.2.4 Business Partner Collaboration

Until the introduction of Web Services standards, business partners faced a difficult task to integrate their systems. Solutions were almost always once-off, customer integrations. They were difficult to implement and difficult to maintain. Changes at either partner could easily unravel the entire system. Collaboration between multiple partners was strictly the domain of very large companies.

For example, a yellow-pages site may be created for automotive parts vendors. A parts-provider may thus desire to provide a Web Service to integrate their services into the marketplace through the UDDI registry.

Web Services offer a standards-based way for business partners to collaborate. The usual business and organizational issues will still be the substantive amount of work that is done with a new business partnership. However, a common technology framework ensures that the focus is the business benefits rather than resolving technological integration problems.

The key security requirement here is for standards to exist to avoid the need to implement a custom security solution for each different partner being communicated with, in the same way that the interaction technology has typically converged to SOAP and WSDL.

1.2.5 Composite Business Processes

Once backend services are available in a standardized manner through exposing them with XML Web Services technologies and standards like SOAP and WSDL, it makes the task of reusing these core business services in new applications and new usage scenarios significantly simpler.

New business processes can be created by combining together the existing business process components in innovative and exciting new ways, without having to worry about the traditional technology barriers that have hindered much of this work in the past.

However, this can easily lead to exactly the same sorts of problems with security gaps as found in the Technology Integration usage scenarios unless all the web services being composed utilize the same set of XML security standards. This clearly highlights the importance of mature implementations of standards that have been widely adopted in the industry.

1.2.6 Reducing I.T. Lifecycle Costs

There are a number of factors that make Web Services a better choice than older technologies from the perspective of lifecycle costs:

- Web Services are comparatively cheaper to implement, lowering the investment part of any return-on-investment calculation.
- Web Services are generally quicker to implement (assuming productivity tools like CapeStudio are used). This results in a faster time to market and lower development costs.
- Lower ongoing maintenance and transaction costs. For example, because tools like CapeStudio automatically expose application logic without

coding, changes can be implemented quickly and seamlessly.

The trend towards the web services platform providing the unified security policy enforcement layer also creates considerable cost savings in that using a single security system considerably reduces staff training and operations costs.

1.2.7 I.T. Investment Protection

By allowing the functionality of existing I.T. systems to be published and re-used through SOAP, WSDL and UDDI is considerably more cost effective than re-designing from scratch. Adding a web services interface onto an existing legacy system can provide a new lease of life for the system, and take away much of the immediate pressure to replace highly complex systems immediately.

Using web service technology as the standardized form for publishing and re-using application services also helps to protect future I.T. investment, by providing a degree of separation between the interface definition and the underlying implementation.

The use of web service security standards based on XML similarly provide a level of future proofing as the implementation of this security framework can be changed while still relying on the technology-neutrality of standards based on XML communications.

2 A Public Web Service Security Framework

2.1 Security through Product Generations

This section describes a public framework for Web Services. It is a real-world case study, from a commercial product called CapeConnect.

CapeConnect has always provided security features since it was first released in November 2000. The initial security features provided in the early generations of CapeConnect used well-known security technology already widely used on the Internet and World Wide Web:

- Confidentiality: Existing web technology such as SSL (Secure Socket Layer / Transport Layer Security) can be used to ensure the confidentiality of data in transit.
- Authentication: SOAP request's user credentials are authenticated against an XML data store. The authentication module is also pluggable.
- Authorization: Access control can be applied to Web Services created by CapeConnect.
- HTTP Authentication: HTTP Basic authentication is supported for password protected Web sites.
- Importing of external security credentials: Security credentials are automatically imported from the Web tier without additional development work.
- Single sign-on: A single sign-on service is included in the CapeConnect product.

The feature list is typical of early generation Web Services platform.

The benefits of Secure Sockets Layer (SSL) are:

- SSL is implicitly supported in the SOAP 1.1 specification, in that the transport layer for the SOAP message is HTTP based.
- SSL is a well know and well understood technology, which implies that there are many software developers available to exploit it.
- SSL is widely used in the Internet and World Wide Web

The limitations of using SSL are well understood:

- SSL encryption and decryption is CPU-intensive, thereby reducing transaction handling capacity and hence scalability.
- SSL can only protect data while it is in transit, but not while it is on the host at either end of the connection.
- After the SSL protected SOAP message arrives and is decrypted, it is no longer protected and therefore the contents are vulnerable to unauthorized usage.
- While SSL can assist in providing client credentials through the use of mutual authentication on the connection so that both the client and server have to present a valid PKI certificate from a trusted source, this does not completely cover all the requirements necessary for complete support for non-repudiation.
- SSL can only protect data on a single connection hop, and this degree of protection may lapse where multiple hops are necessary to reach the final processing destination due to such things as protected network topologies.

It became apparent as CapeConnect was deployed into more production environments in major corporations that a public security framework was needed to accommodate a wide range of third-party security products already in use as part of the corporate infrastructure.

The main focus of security enhancements in CapeConnect Four were to:

- Provide more complete support for end-to-end propagation of security credentials throughout the SOAP processing stack within CapeConnect
- Allow custom libraries to be plugged-in at various points along the processing chain to support a range of external security products.

CapeConnect now provides a complete end-to-end security framework flexible enough to allow the easy customization of the product to suit the specific requirements of individual organizations and vertical market partners.

There are several key steps in the security strategy, which are described in later sections:

1) Establishment of user credentials on the client;
2) Transportation of those credentials to the server, and importation of those credentials into the server process;
3) The flow of credentials through the server process to the back end application processes;
4) The application of any Web Service security policy.

2.2 Client Side Credential Establishment

There are two possible methods that security credentials can be established for communication with a CapeConnect server:

- The user can perform a login through the SoapDirect API using the SDLoginManager class. This will result in an authentication call to the CapeConnect security service via SOAP.
- The user can perform a login through the standard JAAS (Java Authentication and Authorization Service) API. Depending on the JAAS configuration on the client, this will result in one of the following:
  - authentication to a "third-party" security product like Kerberos, and so on
  - authentication to the CapeConnect security service by sending of a SOAP call.

The CapeConnect client-side runtime, including the SoapDirect library, can pick up the credentials established in the previous steps, and then will transport them in an appropriate manner within the SOAP message sent to the CapeConnect server.

The client-side transports can use SSL based connections or encrypted SOAP messages using the XML Encryption standard to ensure message confidentiality, and can also use techniques like signing SOAP messages using the XML Digital Signatures standard to ensure message integrity. In the same way, the CapeConnect server can be configured to require some or all of these security measures to be present for messages it receives, so enforcing the level of security desired for a particular application. For example, an application can be configured so that it will only accept SOAP messages sent over a secure SSL connection where a client-side certificate was used.

2.3 CapeConnect Security Service

The CapeConnect security service (know as *ccauthenticate*) provides a means of a clients to authenticate themselves to the CapeConnect server, and obtaining a session ticket that is then used to track their session credentials and enforce access control policies for secure Web Services running in CapeConnect.

A timeout value is applied to all session tickets, and the contents of these tickets are cryptographically secured to prevent "ticket stealing" attacks.

The *ccauthenticate* security scheme can operate in conjunction with, or instead of, the other transport-level security controls such as HTTP / J2EE configuration controls available for the CapeConnect server.

2.3.1 Plug-in API for Authentication

The CapeConnect security service supports an API to plug in an external authentication provider to replace its default scheme.

Standard plug-ins are provided for:

- The default CapeConnect file-based storage of user details
- An LDAP Directory Server
- Any GSS-API based authentication provider.
- Any JAAS-based authentication provider.

2.3.2 Server Side Credential Importation

Once the credentials reach the server process, it is the job of the message listeners to import those credentials and reassert them inside the server process in an appropriate manner.

The message listeners are able to import and accept various types of transport –level and message-level authentication credentials, including the following:

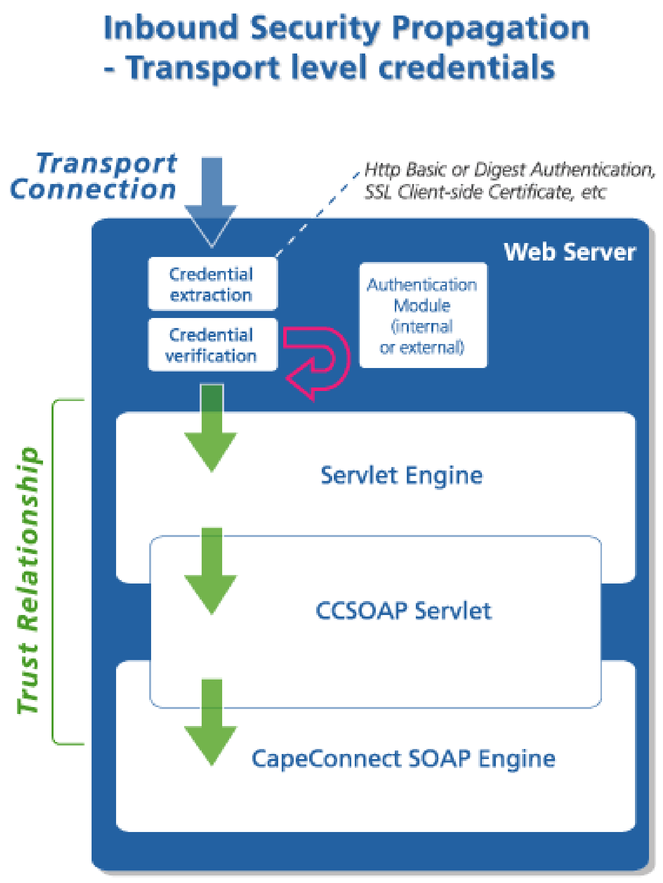
2.3.3 Transport-Level Credentials

There are several different ways that security credentials can be encoded and sent at the transport level:

- HTTP Basic authentication [2]
- HTTP Digest authentication [2]
- J2EE Form-based authentication
- External Web server plug-in – e.g. LDAP plug-in to Tomcat
- SSL credentials as a result of mutual-authentication over an SSL connection where a client-side certificate was available and presented.
- Transport connections using Kerberos tickets for encryption and mutual authentication, probably via the GSS-API in JDK 1.4
- SAML (Security Assertions Markup Language) authentication *HTTP* headers [4]

The transport-level credential import points are illustrated in Figure 1:

Figure 1: Credential import points – Transport Level


Inbound Security Propagation
- Transport level credentials
Transport Connection
Http Basic or Digest Authentication, SSL Client-side Certificate, etc
Web Server
Credential extraction
Credential verification
Authentication Module (internal or external)
Servlet Engine
CCSOAP Servlet
CapeConnect SOAP Engine
Trust Relationship

2.4 Message-Level Credentials

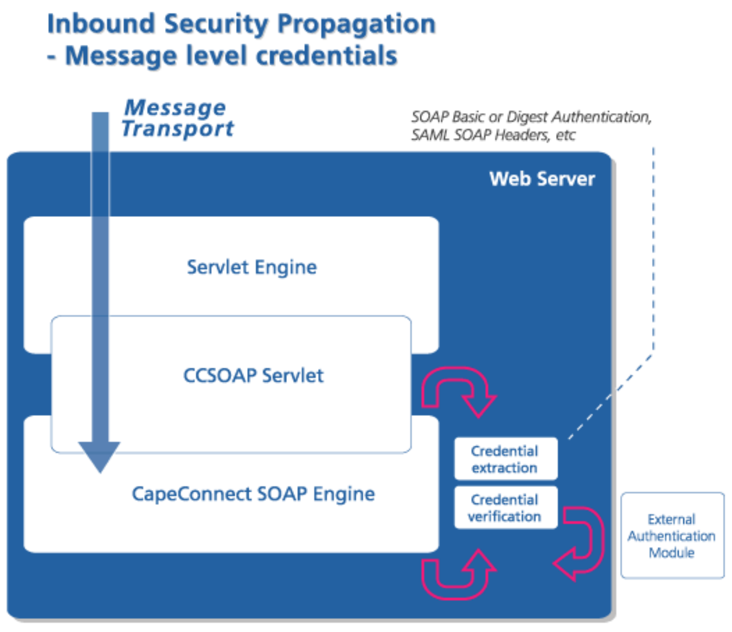
There are several ways that security credentials can be encoded and sent at the message level, and processing of these may either be handled automatically by the SOAP infrastructure, or directly by the application itself.

- Custom credentials sent in the header of a SOAP message and recognized by the runtime platform. This typically requires a custom handler to be plugged in to the CapeConnect engine for the particular credentials format to be handled, such as SAML. Expected credential formats are:
  - CapeConnect authentication service session tickets
  - SAML authentication *SOAP* headers [4]
  - The SOAP Extensions for Basic and Digest Authentication [5]
- For credentials not handled automatically by the runtime platform, a web service application can extract the credentials for itself by accessing the SOAP headers for the request. All mature web service runtime platforms allow this to be done fairly easily. The application is then responsible for decoding and validating the application specific credentials and accepting or rejecting access on that basis. This is the way authorization and access control is handled with UDDI, which is just a regular web service with a well defined XML message format and an application specific security token.

Message-level credentials require a custom handler (either in the infrastructure or in the application itself) to pull the appropriate credential items from the SOAP message header, and then perform whatever import / authentication actions are required to confirm the validity of these credentials.

The message-level credential import points are illustrated in Figure 2:

Figure 2: Credential import points – Message level


Inbound Security Propagation
- Message level credentials
Message Transport
SOAP Basic or Digest Authentication, SAML SOAP Headers, etc
Web Server
Servlet Engine
CCSOAP Servlet
CapeConnect SOAP Engine
Credential extraction
Credential verification
External Authentication Module

2.5 Plug-in API for Credential Importation

This is the API that custom security message handlers will need to follow to be able to support application protocol specific credential handling such as SAML SOAP headers.

These message handlers will be given the contents of the SOAP message, and will need to extract the appropriate credentials from the message and return a suitable java.security.Principal object corresponding to these details.

2.6 Credential Propagation to Call Handlers

Transport-level credentials are associated with the incoming message data as it is taken off the wire. Those credentials are then automatically propagated through the CapeConnect engine to the backend call handlers.

In the process of passing the call invocation and credential details through the SOAP Engine to the backend call handlers, an optional role-based Web Service security policy can be applied for each Web Service application configured with the CapeConnect SOAP Server.

2.6.1 Role-based Security Policy

As part of the routing of the SOAP message to the back end call handlers, a Web Service can be configured in CapeConnect to say whether an access control policy should be applied. If an access policy is present, a role-based authorization check will be performed to confirm that the current caller is in a role that is permitted to perform the required operation. This is very similar to the operation of the EJB declarative security model.

This check is performed at the Web Service tier rather than relying on the underlying distributed component technology (such as EJB or CORBA) to perform this function, so that a uniform security model can be applied to Web Services, and also allowing the option of applying differing access controls depending on whether the call is coming via a Web Service or from an internal call direct to the EJB server.

The authorization check is performed at the level of the specific method / operation being called, to provide very fine grained control of the security policy for any individual web service.

2.6.2 Plug-in API for Authorization

The role-based security service supports an API to plug in an external authorization provider to replace the default scheme.

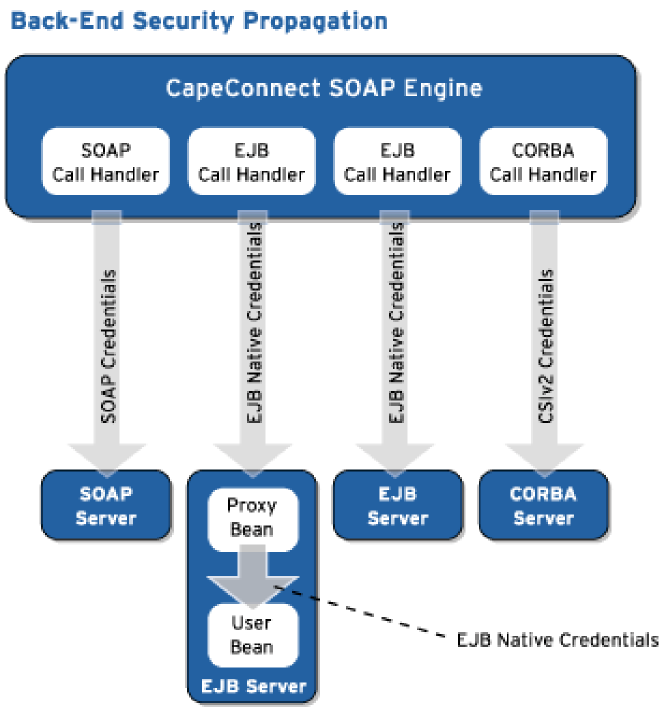
Standard plug-ins are provided for:

- The default CapeConnect file-based storage of user role details
- An LDAP Directory Server to store user attribute properties

2.7 Credential Propagation from Call Handlers to Back-end Systems

There are several options for how credential information can be propagated from the call handlers to the backend application systems, and to a large extent it the method used depends on a particular Application Server. These options are illustrated in Figure 3, and then described in more details following.

Figure 3: Credential propagation



One possibility is to forward the call as a SOAP message to a listener in the app server, and this SOAP message could directly contain any SAML or SOAP Basic/Digest Authentication headers so that the receiving Application Server can import the credentials and re-establish the identity information to be used for the duration of that call. Additionally, the HTTP transport listeners in the Application Server may be able to handle SAML HTTP Headers or HTTP Basic/Digest Authentication headers for automatically import of the credentials by the Application Server.

Another possibility is to forward the SOAP message into a CapeConnect proxy EJB running inside the target app server, and pass the security credentials explicitly as an additional parameter on the call to the proxy bean. The proxy bean requires a means to re-establish the caller credentials inside the target app server, and in some cases, this is likely to require custom code for the proxy bean in each different Application Server.

A further possibility is to establish the caller's credentials to the client library used for communicating with the target Application Server (such as WebLogic's t3 library) in exactly the same way that a regular EJB client would. Again, this is likely to require custom code for the call handler for each Application Server, but the appropriate credential-establishment APIs are likely to be more readily available here than inside the Application Server itself.

As a slightly simplified case of the above, where CapeConnect is running as a J2EE application inside a J2EE app server, then the task of credential propagation from servlet to EJB will occur automatically with no further effort or configuration changes being required. This clearly represents the easiest way to achieve this objective.

Finally, where a standard exists for how security details are sent across a particular transport scheme (such as the CSIv2 for propagation of security credentials across IIOP connections), credentials can be asserted onto that transport level through applying that standard. This will typically be how things will work with EJB 2.0 based J2EE servers, as soon as these are in widespread deployment.

2.8 Credential Propagation from the Gateway to the XML Engine

It must be possible to configure the gateway so it performs an authentication dialog with the XML Engine (*xmlengine)* to establish a secure and trusted channel between the two.

One obvious way to do this is for the gateway to use SSL mutual authentication and an X509 certificate for transport connections. Another way would be through using a GSS-API based connection to provide mutual authentication and encryption support on the link.

In must cases, the gateway acts as an invisible proxy for the *xmlengine*, and is simply concerned with routing the message rather than actually processing it.

Where SSL client-side certificate information was used for authentication, the details of this certificate need to be attached to the SOAP message before it is forwarded to the *xmlengine*. If the gateway and *xmlengine* have established a trust relationship, the *xmlengine* can accept the presented credentials as valid and vouched for.

The SOAP message forwarded by the gateway includes all the SOAP Headers in the original message, subject to SOAP's MustUnderstand rules. This includes SAML assertions passed as SOAP header fields.

2.9 Non-Repudiation

The security framework in CapeConnect Four provides full support for the non-repudiation requirements of any serious enterprise-grade web service.

The use of connections employing client-side certificates is one of the fundamental features of a non-repudiation strategy, and CapeConnect has this support built in as a standard feature.
The other major component of a non-repudiation strategy is the transmission of messages signed with an appropriate digital signature, and this is available on both the client and server side of the connection with CapeConnect.

3 Conclusions

3.1 Lessons from the First Wave

Some key conclusions can be made from the initial security efforts with Web Services:

- Basic generic security is sufficient for internal Web Services projects. The initial technology provided in most Web Services platforms included access control, authentication, and the option of using SSL as the SOAP transport layer.
- As Web Services usage grows, it is necessary to accommodate a wide range of third-party security products already in use as part of the corporate infrastructure.
- A full end-to-end security solution is needed to avoid security gaps.
- Web Services security procedures and requirements, both organizational and technical, have yet to be fully explored. Best practices have yet to be developed.
- The new XML and Web Services security specifications have not yet gained any significant adoption rates in commercial/corporate environments.
- It is still unclear which of the emerging Web Services and XML security specifications will emerge as industry standards. It is therefore necessary to track all the standards and be careful about moving ahead of the market.
- A key practical problem for the new XML and Web Services security specifications is lack of trained staff. This means that many types of Web Services projects where the new technology is ideal do not use it.

3.2 Recommendations for the Future

It can be assumed that Web Services will take a number of years before the full security implications are understood and then resolved. In the meantime, the following actions are recommended:

- Track the usage scenarios within your organization; these will determine the security levels.
- "Proof on concept" projects, rather than full scale commercial projects are recommended.
- It is necessary to have a .NET strategy, because Microsoft will promote its own security products and strategies and will inevitably be successful in acquirement many users.
- Track the new XML and Web Services security standards initiatives and standards, but remember that your organization's existing security infrastructure is probably the key factor.

4 References

[1] Generic Security Services API (GSS-API)
Internet Standards: RFC-2743, RFC-2853
http://www.ietf.org/rfc/rfc2743.txt
http://www.ietf.org/rfc/rfc2853.txt

[2] HTTP Authentication: Basic and Digest Access Authentication
Internet Standard: RFC-2617
http://www.ietf.org/rfc/rfc2617.txt

[3] Java Authentication and Authorization Service (JAAS)
Sun Microsystems, Inc.
http://java.sun.com/products/jaas/
http://java.sun.com/j2se/1.4/docs/guide/security/jaas/JAASRefGuide.html

[4] Security Assertions Mark-up Language (SAML)
OASIS XML-Based Security Services Technical Committee (SSTC)
http://www.oasis-open.org/committees/security/
http://www.oasis-open.org/committees/security/docs/

[5] SOAP Extensions for Basic and Digest Authentication
IETF Internet-Draft: draft-cunnings-salz-soap-auth
http://www.zolera.com/resources/opensrc/i-d/soap-auth.html

[6] XML-Signature Standard
W3C XML-Signature Working Group
http://www.w3.org/Signature/
http://www.w3.org/TR/xmldsig-core/

Editors: Hamid R. Arabnia and Youngsong Mun
ISBN: 1-892512-37-8