Web Service Security

Malek Adjouadi[1], Armando Barreto[1], Dinesh Tenneti[1], Md Raoof[1], Rama Krishna.K[1], Yixis Cejas[1], Norris Milton II[2], Scott Graham[1], Naphtali Rishe[1]

Florida International University[1] *University of Illinois at Chicago[2]* *Florida Memorial University[3]*

Abstract

Authentication is the process of making sure that the person who is requesting a web service is really the person that they claim to be. This is done by requiring the user to provide a set of credentials. In return, they will receive a security token that can be used to access the server. The credentials usually take the form of a user id and password. On the other hand, the security token that is returned is usually more conceptual than physical. It can take the form of a cookie placed on their browser, a session id stored on the server or an actual string of characters.

Architects and developers responsible for Web service security have a considerable number of options available. These options are further complicated by the fact that different projects and different organizations have different security requirements. This paper proposes a scheme for taking these requirements into consideration when proposing secure web service access methods.

1. Introduction

Web Services are self-contained, modular applications that can be described, published, located, and invoked over a network, generally, the World Wide Web. Below we have a definition from IBM and Microsoft

"Web service acts as application components whose functionality and interfaces are exposed to potential users through *the application of existing and emerging Web technology standards including XML, SOAP, WSDL, and HTTP. In contrast to Web sites, browser-based interactions or platform-dependent technologies, Web services are services offered computer-to-computer, via defined formats and protocols, in a platform-independent and language-neutral manner."*

Stages of Web service Development and usage

1) Service Provider develops service and a document interface
2) Publishes the service in Universal Description and Discovery and Integration(UDDI)
3) Service requestor/subscriber finds a service provider in UDDI
4) Then the UDDI informs the requestor/subscriber the list of services and services provided.
5) The requestor/subscriber develops an application to the service
6) The service subscriber/requestor request the service provider for the service
7) The service provider delivers the service to the requestor/subscriber.

Web Services generally uses the HTTP and SSL ports (TCP ports 80 and 443, respectively) in order to pass through firewalls. Most firewalls are unable to distinguish Web Services traffic, traveling over HTTP and SSL ports, from Web browser traffic. With some firewalls it is possible to block Web Services traffic altogether, but not possible to set up different rules for separate Web Services

Service-Oriented Architecture: Publish, Find, and Bind. UDDI, WSDL, and SOAP— enable an SOA to run on the open Internet. The main advantage of SOAP is its loose coupling. SOAP can either be used for Remote Procedure Calls or for messaging between applications. We have to choose a firewall that supports SOAP filtering. A firewall should be capable of blocking SOAP messages based on endpoint and based on the payload of the SOAP message, validated against an XML Schema.

UDDI is used to publish services to the entire world, allowing anybody to connect to the UDDI registry, find a Web Service published, and bind to it. This clearly does not match real-world business scenarios, in which business is rarely done between complete strangers (—especially in the absence of credential checks such as credit worthiness). But this view does UDDI a disservice. That is why UDDI sometimes gets a bad reputation.

Most uses of UDDI involve cases where the transacting parties already know each other. It provides a way to publish information about newly available Web Services, or to provide information about a Web Service whose interface has changed. UDDI is also useful inside an organization (that is, "behind the firewall") in order to publish information used by internal applications to bind to each other using SOAP.

Web services security: It is about making a process (not an entity) secure

Web services application Security is broader than we might expect because these services cut across many business models (like Ecommerce sites, Supply chain Management, cross selling and Customer management).

Security is always seen as just protecting (or dealing with) Networks, Os, Cryptography and firewalls, but where as in web services since there is an increase in the amount of access to the data. So there is a change in the risk levels associated with Web Services. Therefore we need new kinds of architectures to secure Web service applications where Risk Management holds the key and Information Security has to be taken care.

Table 1 tells us that the blocks of security stands on flow down through the Communication layers and not horizontally

Table 1 General layers of security & Web services layers of security Table OSI Layers

Layer Name	**Example of Technology that uses this Layer.**	**Web Services Technology**
Application	E-mail, directory services	HTTP, SMTP, SOAP
Presentation	Encrypted data, compressed data POP/SMTP	Encrypted data, Compressed data
Session	POP/25, SSL	POP/25, SSL
Transport	TCP, UDP	TCP, UDP
Network	Packets IP, ARP	IP Packets
Data Link	PPP, 802.11	PPP, 802.11, etc.
Physical	ADSL, ATM	ADSL, ATM, etc.

Security standards that are fundamental for multi-tier end to end applications are Authentication, Authorization, Cryptography, Accountability, and Security Administration

2. XML Security:

WS-Security defines how security tokens are contained in SOAP messages, and how XML Security specifications are used to encrypt and sign these tokens, as well as how to sign and encrypt other parts of a SOAP message.

The different types of XML Securities used are

1) XML signature,
2) XML Encryption,
3) SAML (Security Assertion Markup language),
4) XACML
5) XKMS (XML Key Management Specification)
6) XACML (extensible Access Control Markup Language.)It is used in WS-Authorization.

2.1 SAML

SAML is one the important XML type security provided. SAML explains how security assertions may be expressed in XML format. SAML is used at application level to support authentication and across domains. SAML specification concentrates on perimeter Web SSO. However, the broader problems that it is designed to solve pertain to end-to-end security, company-to-company security, SSO, and privacy. SAML addresses a specific, important aspect of these problems. One of the primary problems that SAML solves is defining a standard format for passing authentication, attribute, and/or authorization proof from process to process. As long as the different models understand SAML, we can have interoperability between security models. SAML's open platform and language-neutral constructs make it acceptable to most security models.

2.2 Public Key Algorithms used

Public Key Algorithms are divided into two general approaches

Digital Signatures: RSA, Digital Signature Algorithm (DSA) or Elliptic Curve DSA (ECDSA)

Encryption: RSA, Diffie-Hellman (DH) or Elliptic Curve Diffie-Hellman (ECDH), and here each have different operation

3. Web services Security Architectures

WS-Security, by contrast, is primarily for securing SOAP messages. WS-Security is arguably the most important Web Services security specification, because it explains how XML security relates to SOAP, and will underpin many later specifications.

3.1 WS-Security

WS-Security explains how security information is contained in SOAP messages. It provides a means of how to encrypt information (using XML Encryption and XML Signature) and for enclosing security Tokens to the SOAP Message to represent the claims.

From security conversation- ->WS-Security defines the use of security tokens within SOAP messages, when combined with XML Signature and XML Encryption, to provide proof of possession and confidentiality of the claims that these tokens encapsulate. When a SOAP message is received, the security token is evaluated and

checked against a security policy. However, this process must be repeated for each incoming SOAP message. This is obviously a performance issue, because there is no concept of a session for a group of SOAP messages.

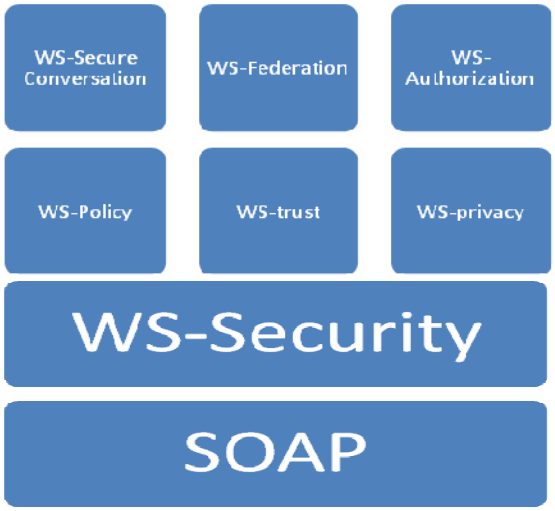

Figure 1. IBM and Microsoft Web Services security specifications

3.2 WS-Policy

WS-Policy allows organizations that are exposing Web Services to specify the security requirements of their Web Services {algorithms for encryption and digital signatures, privacy attributes (such as which parameters must be encrypted)}, and how this information may be bound to a Web Service.

It allows organizations initiating a SOAP exchange to discover what type of security tokens are understood at the target, in the same way that WSDL describes a target Web Service.

For example, one organization may only consume Kerberos tickets while another organization may only understand X.509 certificates.

3.3 WS-Trust

It is built on WS-Security to both provide a framework for requesting and issuing security tokens and broker trust (directly/indirectly) for establishing a Trust in Web Services Environment.

WS-Security layer is used to transfer the Tokens making use of XML Signature and XML Encryption (to have Integrity and confidentiality).

3.4 WS-Privacy

It uses a combination of WS-Security, WS-Policy, and WS-Trust for communication privacy policies, which are stated by the organization that deploy Web services, and it sees that the incoming SOAP request contains claims that the sender accepts (conforms) to these Privacy policies.
The Privacy Policies are given to the WS-Security which encapsulates these into Security Tokens which can be verified.

WS-trust is used to evaluate the privacy claims that are encapsulated using WS-Security within SOAP messages against user preferences and organization policy

3.5 WS-SecureConversation

WS-Secure Conversation uses asymmetric encryption and is built over WS-Security and WS-trust to exchange content securely (claims about security attributes and related data) across messages, and for deriving keys (session keys, derived keys, and per-message keys) from established security contexts .It is designed for the SOAP Message layer where message may traverse a variety of transports and intermediaries not all of whom can be trusted.

3.6 WS-Federation

How to support federation (that is, how to make dissimilar security systems interoperate) Like WS-SecureConversation, WS-Federation also builds upon the specifications that underpin it. It explains how federated trust scenarios may be constructed using WS-Security, WS-Policy, WS-Trust, and WS-SecureConversation. "Federation" in this case involves brokering between different security specifications—for example, communication between a party who understands Kerberos and another party who understands X.509 digital certificates to allow an end user to authenticate to one party, but then use a Web Service exposed by the other party. WS-Policy and WS-Trust are used to determine which tokens are consumed, and how to apply for tokens from a security token issuance service. WS-Federation acts at a layer above WS-Policy and WS-Trust, indicating how trust relationships are managed.

3.7 WS-Authorization

This specification has a number of overlaps with XACML. WS-Authorization describes how access policies for a Web Service are specified and managed. This specification is flexible and extensible with respect to both authorization format and authorization language. It

supports both ACL-based authorization and RBAC-based authorization.

3.7.1 SOAP Security. SOAP is a unidirectional, XML-based protocol for passing information. (As of draft version 1.2, SOAP is no longer an acronym.). SOAP messages can be combined to implement request/response processes, or even more sophisticated interactions (and intermediary nodes).

4. Core Web Service Security Patterns

4.1 Authentication patterns

Authentication is the process of identifying an individual using the credentials of that individual. Authentication allows Web services to verify the identity of entities within the SOA. For direct authentication, traditional X.509 and its XML counterpart, the XML Key Management Specification (XKMS), provide mechanisms for identifying users and individual Web services within a SOA. Brokered trust mechanisms, such as WS-Trust and SAML, allow Web services to rely on trusted third parties to perform authentication.

4.1.1 Authenticating the user. A Web service security configuration should specify authentication policies that define how the user credential (or authentication data) is to be retrieved as well as how it is to be verified. In the Web service security model, the log-in configuration should not only address authentication of immediate clients who submit the request directly to the service, but also address indirect clients i.e. an end client's identity may be part of the request, which can traverse through many intermediaries.

4.1.2 Asserting the user identity. The identity of the user must be securely associated with the context of execution to be used in the downstream requests. To do so, the authentication handler asserts the identity of the user within the request.

4.1.3 Service-to-Service Authentication. Service-to-service authentication can be performed using a variety of methods from HTTP-based token authentication to SSL/TLS-certificate based authentication, or by passing tokens along with the SOAP request.

4.1.4 WS-Security for Authentication. Authentication in WS-Security is performed by including *claims* in the WS-Security header of a SOAP message. Claims provide information about the identity of the SOAP message sender, which can then be used to determine whether or not the sender is authorized to access the resources requested. Claims can be any form of security token: an X.509 certificate, a username/password pair, a Kerberos ticket, or an XML security token (such as a SAML assertion). Claims can either be endorsed or unendorsed.

4.1.4.1 Direct Authentication. When both the client and service participate in a trust relationship that allows them to exchange and validate credentials including passwords, direct authentication can be performed. Direct authentication requires the presentation of credentials, which are typically a user name and password. Direct authentication is used where the Web service acts as an authentication service to validate credentials from the client.

4.1.4.2 Brokered Authentication. Brokered Authentication is used where the Web service validates the credentials presented by the client, without the need for a direct relationship between the two parties. An authentication broker that both parties trust independently issues a security token to the client. The client can then present credentials, including the security token, to the Web service. This same token could be used to access all services within an organization.

4.1.4.3 X.509. X.509 certificates can be used across organizational boundaries. Most X.509 implementations, such as SSL, exchange a symmetric session key that is used for encryption. Signatures created using X.509 certificates can be mapped to a particular participant in a conversation, assuming both participants have unique certificates.

4.1.4.4 Kerberos Token. The Kerberos protocol is used to authenticate clients within a domain. Cross-domain trusts can be established but are typically limited within an organization. Kerberos tokens can be used for impersonation and delegation.

4.1.4.5 Custom Security Token. Custom security tokens can be used for session based operations. It supports confidentiality and data origin authentication at the message layer only. Windows impersonation or delegation is not supported.

4.2 Message Protection Patterns

Message Protection prevents sensitive data be intercepted, viewed and/or modified for malevolent purposes while messages are sent and received between Web Services and Clients using Internet Protocols. Message protection comprises three main categories: Data Integrity, Data Origin Authentication, and Data Confidentiality. Data integrity is the verification that a message has not been changed in transit, Data Origin Authentication supports the ability to identify and validate

the origin of a message, and Data Confidentiality is the encrypting of message data so that unauthorized people cannot view the contents of the messages.

Microsoft has developed patterns and practices for Data Confidentiality and Data Origin Authentication, but there is no pattern for Data Integrity. Data Integrity is included in the patterns "Implementing Transport and Message Layer Security" and allows the verification that a message has not been changed in transit.

4.2.1 Data Confidentiality. Encryption can be used to protect sensitive data that is contained in a message. Unencrypted data, known as plaintext, is converted into encrypted data known as ciphertext. An algorithm and a cryptographic key are used to encrypt the data. Ciphertext or encrypting data is then converted back to plaintext at its destination.

To apply Data Confidentiality: first, the data is encrypted, the sender converts plaintext to ciphertext, leaving it unintelligible to parties other than the expected recipient. Second, the data is decrypted by the desired recipient by converting it back to plaintext.

Two types of cryptography can be used to provide data confidentiality: symmetric and asymmetric. With symmetric cryptography, both the sender and recipient share a key that is used to perform both encryption and decryption. Before communication can occur, the sender and the recipient must exchange a shared secret key. Symmetric cryptography is commonly used to perform encryption.

With asymmetric cryptography or public key cryptography, the sender encrypts data with one key, and the recipient uses a different key to decrypt cipher text. The encryption key and its matching decryption key are often referred to as a public/private key pair.

In addition to providing encryption, we can use public key cryptography to provide digital signatures.

The public key of the recipient is used to encrypt data. It can be openly distributed to those who want to encrypt a message to the recipient. The private key of the recipient is used to decrypt messages, and only the recipient must be able to access it.

4.2.2 Data Origin Authentication. Using Data Origin Authentication enables the recipient to verify that messages have not been altered with in transit (data integrity) and that they originate from the expected sender (authenticity).

We can use digital signatures to provide evidence that a client has performed a particular action that is related to data. A digital signature is a mechanism to capture a client's association to data. With Proof-of-possession a client demonstrates knowledge of either a shared secret or a private key to support client authentication.

Proof-of-possession using a shared secret can be established using the actual shared secret, such as a user's password.

Two types of signatures can be used to sign a message: symmetric and asymmetric.

A symmetric signature is created by using a shared secret to sign and verify the message. A symmetric signature is commonly known as a Message Authentication Code or MAC. A MAC is created by calculating a checksum with the message content and the shared secret and can be verified only by a party that has both the shared secret and the original message content that was used to create the MAC.

4.3 Implementing Transport and Message layer Security

Implementing transport and message layer security demonstrates the implementation of "Authentication Patterns" and "message Protection Patterns" .Here we have two sections one is Message layer security and Transport layer Security and some times we choose between these layers based upon the protection scope, support for protocols, and flexibility provided.

We have different types of implementations in both layers of security which are given below, and we choose one of them according to the specifications

4.3.1 Message layer Security

1) Direct Authentication with Username Token
2) Message Layer Security with Kerberos,
3) Message Layer Security with X.509 Certificates)

4.3.2 Transport layer security. There is already lot of information available about securing Transport layer

1) Brokered Authentication Using Windows Integrated Security on IIS
2) Transport Layer Data Confidentiality Using HTTPS, Transport Layer Security Using HTTP Basic over HTTPS,
3) Transport Layer Security Using X.509 Certificates and HTTPS.
4) Transport Layer Security with Kerberos and IPSec on Windows Server 2003

4.4 Resource Access pattern

Authentication, authorization, and auditing, along with other environmental and operational requirements, should combine to influence the security solution that you use to help secure access to resources of particular Web service or multiple interacting Web services.

Authentication credentials and what protocol was used for protecting the data from directly being accessed by using authentication as and when needed. Decide whether

we need client's identity/credentials for accessing the resource which may be located on the same domain (or even may be the same computer) or another domain (here domains mean security domains), and does the web service using connection poling a resource sharing Technique. We use Trusted Subsystem, Impersonation & Constrained Delegation for different Security considerations.

4.5 Service Boundary Protection Patterns

When Message Protection Patterns are not enough to protect sensitive data we need to provide additional protection at the service's boundary to protect Web services, to ensure that when a Web service operation fails we do not disclose confidential information in the SOAP Fault that is returned, and to prevent somebody to intercept a message and replay it forcing a Web service operation to execute more than one time. Service boundary protection comprises Message Replay Protection, Message Validator, and Exception Shielding.

4.5.1 Message Replay Protection. We can store a unique identifier for incoming messages, and use message replay detection to identify and reject messages that match an entry in the replay detection cache.Web services can implement message replay detection by having the client signing the message, then sending the signed message to the recipient. When the message is received, the service verifies the client's signature and the message timestamp to ensure that the message contents have not been altered during transit. The Web service can also compares the message timestamp to its own current clock value. Finally, the service checks the replay cache for the SignatureValue field. If the SignatureValue is already in the cache, the message is rejected as a duplicate. If the message signature is not in the cache, the message signature and timestamp are added to the cache.
The Web service must only accept messages that are younger than the messages that have already been removed from the cache to ensure that a hacker will not be able to replay a message that has been already deleted from the replay cache.

4.5.2 Message Validator. For the message validation mechanism the client sends a request message to the service; the service validates the message comparing the size of the request against the maximum allowable size that is specified for request messages. Also, if the message is signed, verifies the signature to ensure that the message has not been altered within transit, checking that the message matches a predefined schema, with acceptable data types and ranges of values. Since malicious content can appear in the SOAP message or in the message payload, both are checked. Finally, if the request passes all the validation checks that are performed by the message validator, the service processes the message and sends a response to the client.

4.5.3 Exception Shielding. Exception details may contain hints that a hacker can use to take advantage of resources used by the system. Detailed fault messages can disclose information about the Web Service or resources accessed by the Web Service code that threw the exception. If an unhandled exception is thrown by the Web Service, sensitive information, such as connection strings, server names, SQL queries, XPath commands, stack traces, and data schemas can be obtained for hackers.
Information related to anticipated exceptions needs to be returned to the client. In cases where an exception is expected, an error message that does not contain sensitive information can be returned to the client. A service may provide information about the cause of the fault, where the information is not considered a security risk.

Exceptions that occur within a Web service should be analyzed during troubleshooting. Information within an exception can be used by monitoring tools to automatically notify system administrators when an exception occurs, can be used by application developers to diagnose exceptions that occur within the logic of the service or with resources that the service is dependent on.

The Exception Shielding pattern is used to clean unsafe exceptions, replacing them with exceptions that are safe by design. It returns only exceptions that have been cleaned or exceptions that are safe by design. Exceptions that are safe by design do not contain sensitive information about the Web Service in the exception message.

In the exception shielding process the client submits a request to the service. The service attempts to process the request and throws an exception. The exception can be safe or unsafe by design. Exception shielding logic processes the exception. If the exception type is safe by design, it is already considered cleaned and is returned to the client as is. If the exception is unsafe, the exception is replaced with an exception that is safe by design, which is returned to the client. The service wraps the exception in a SOAP fault and returns it to the client.

4.6 Service Deployment Patterns

Service Deployment patterns are used in web services in situations where there is need for services in a private network be available to external applications without exposing resources in the private network, which can be achieved by designing a intermediary web service that acts as a perimeter service router, that can provide an external interface for internal web services using which messages from external applications are routed to the appropriate Web service on the private network.

The functionality of the perimeter service router is described in the following steps:

1. **The external application sends a request message** to the service's external interface on the perimeter service router. This "hides" the internal endpoint address by accepting requests through an external endpoint address.
2. **The perimeter service router forwards the request message to the service** to the appropriate endpoint address that will route the request to the appropriate service request based on the specific address where the request was sent.
3. **The service sends a response** and performs security checks, such as authentication, and then processes the request; the service may send a response back to the external application.
4. **The perimeter service router forwards the response to the external application** if the server sends a response in Step 3.

Usage of the Perimeter Service Router pattern includes the following:

- Security can be maintained at the perimeter service router.
- Servers that host internal Web services can be taken offline for maintenance without affecting the external interface, which can be accomplished by configuring the perimeter service router to start routing messages to a backup server
- The perimeter service router represents a single point of entry for external clients. This allows it to be extended to support additional operations that external clients require. These requirements could include:

- **Protocol Transition**: External clients are authenticated using X.509 certificates, or custom authentication and then is validated against a database and later it can be transitioned into an internal protocol, such as the Kerberos version 5 protocols to access internal Web services.
- **Message Validation**: Before sending request messages from the external clients to an internal service validation is done. To detect tampering message signatures are also validated.
- **Replay Detection**: Duplicate requests that are sent to the interface are rejected by performing a check in the cache of requests that the perimeter service router keeps track of.
- **Auditing**: For accounting or auditing purposes all activities may need to be attributed to a specific user or organization.

5. Conclusion

In this paper we study various security protocols for ensuring security in accessing the web services over the internet .We can use each of these above mentioned protocols for the specific purposes as needed by the clients or the users. Depending upon the environment and the level of security needed by the clients we use different protocols as per requirements. Protocols are applied at various levels from lower level to the highly complicated organizational level, per the needs. Further work has to be done as there are many security considerations and liabilities for each of the protocols. On further work on these security issues we can refine the existing web services enhancements and we achieve the better security that can be applied and handled in a simpler fashion.

6. Acknowledgement

This research was supported in part by NSF grants HRD-0317692, CNS-0220562, CNS-0320956, and CNS-0426125, and NATO grant SST.NR.CLG:G980822.

7. References

[1]Web Service Federal Language
www.ibm.com/developerworks/webservices/library/ws-fed/?S_TACT=105AGX04&S_CMP=LP

[2]SAML http://xml.coverpages.org/saml.html

[3] Web Service Trust
http://www.ibm.com/developerworks/webservices/library/specification/ws-trust/?S_TACT=105AGX04&S_CMP=LP

[4]Web service core specification
www.oasis-pen.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdfI.

[5] Username Token Profile 1.1
www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdfI.

[6] X.509 Token Profile 1.1
www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf

[7] SAML Token profile 1.1
www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf

[8] Kerberos Token Profile 1.1
www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf

[9] Web Services Security: SOAP Message Security V1.0
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[10]Web services Enhancements
http://msdn2.microsoft.com/en-us/webservices/aa740663.aspx

[11] IBM Web Service Security
http://www.ibm.com/developerworks/library/ws-secure/