Global Journal of Computer Science and Technology
Network, Web & Security
Volume 12 Issue 17 Version 1.0 Year 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Secured Web Services Specifications

By Sudeep Mukherjee & Dr. Rizwan Beg
Integral University Lucknow, India

Abstract - The proliferation of XML based web services in the IT industry not only gives rise to opportunities but challenges too. Namely the challenges of security and a standard way of maintaining it across domains and organisational boundaries. OASIS, W3C and other organisations have done some great work in bringing about this synergy. What I look in this paper are some of the more popular standards in vogue today and clubbed under WS-* specification. I will try to give an overview of various frameworks and protocols being used to keep web-services secure. Some of the major protocols looked into are WS-Security, SAML, WS-Federation, WS-Trust, XML-Encryption and Signature. This paper will give you a brief introduction to impact of using WS-* on time complexity due to the extra load of encrypting and certificates. Windows communication foundation (WCF) is one of the best designed toolset for this though WCF is not the topic of discussion in this paper.

Keywords : soa; web-service; ws-security; ws-trust; ws-federation; xml; soap.

GJCST-E Classification : D.4.6

Secured Web Services Specifications

Sudeep Mukherjee [α] & Dr. Rizwan Beg [σ]

Abstract - The proliferation of XML based web services in the IT industry not only gives rise to opportunities but challenges too. Namely the challenges of security and a standard way of maintaining it across domains and organisational boundaries. OASIS, W3C and other organisations have done some great work in bringing about this synergy. What I look in this paper are some of the more popular standards in vogue today and clubbed under WS-* specification. I will try to give an overview of various frameworks and protocols being used to keep web-services secure. Some of the major protocols looked into are WS-Security, SAML, WS-Federation, WS-Trust, XML-Encryption and Signature. This paper will give you a brief introduction to impact of using WS-* on time complexity due to the extra load of encrypting and certificates. Windows communication foundation (WCF) is one of the best designed toolset for this though WCF is not the topic of discussion in this paper.

Keywords : soa; web-service; ws-security; ws-trust; ws-federation; xml; soap.

I. Introduction

Dependable and secure computing intends to provide services with a high degree of availability(A), reliability, safety, integrity (I), maintainability, and confidentiality (C)[1]. Old fashioned Human-to-Machine interaction is a forgotten story on World Wide Web. Increasingly we see that application-to-application interaction is running our internet. Therefore it is not surprising when we humans interact with the web majority of work is done by these software agents communicating with other computer systems requesting service and getting the desired result in response.

This has radically changed the efficiency as well as customer satisfaction for online business houses, so much so that many business models have no to minimal human intervention. As such new technologies, protocols and frameworks have flooded the market. Yet this great leap has a very dark side to it too. The expansion of application-application messaging infrastructure has attracted old and new attackers who are bent upon destroying or breaking this system for financial gains.

E-commerce application are the favourite hunting ground for attackers who would like nothing more than to get their hands on the sensitive back end data like, customer profile, cards, addresses etc. In the years gone by most of the companies had an easy option just install a firewall or intrusion detection software this kept their domain and data safe but as mentioned earlier with the new concept of application-to-application cross domain communication firewalls have become defunct to a large extent. Firewalls isolate an organization's network system but allow two TCP ports remain open - port 80 for HTTP and port 447 for HTTPS[2].

These ports are used for communication to send and receive Web pages. This deadly combination of easy access and human readable data is a goldmine for attackers. Irrespective of the level of SOA integration security should be one of the top priorities of any organisation. Every computer based organization must revisit their security strategy for facing new security challenges posed by Web Services. Some of these issues are

- Legacy applications work on the concept that authentication alone can filter out the unwanted attackers unfortunately this assumption in new internet infrastructure is grossly mistaken. These applications do not have the where withals to face the new age attackers.
- Most organisations to save cost have used the strategy to keep their core application the same and expose them to the World Wide Web through a layer of web-services, this causes an immediate security hole and more often than not the business logic is compromised.
- Validation checks are kept on the client-side UI, this is not the approved way of doing business in a SOA based architecture
- As mentioned earlier firewalls or packet-filters at the network level are incapable of detecting malicious behaviour of XML/SOAP based attackers.

Transport Layer Security (TLS), is the most popular tool used to secure web-based data through authentication and encryption. Unfortunately in the case of SOA because TLS works between two endpoints it has no way of protecting multiple points or intermediaries. SOAP requires protection of its messages as it is passed through a chain of intermediaries, this is the inherent nature that makes Web-Services most vulnerable.

As security solution on a transport layer, the TLS couldn't provide flexibility for message transmitting, such as encrypted different elements of the message by different key, in which recipients could only read parts of the message about him.[3]

Author α : Department of Computer Science & Engineering, Integral University Lucknow, India. E-mail : sudeep.integral@gmail.com
Author σ : HOD Department of Computer Science & Engineering, Integral University Lucknow, India. E-mail : rizwanbeg@gmail.com

Because of their nature (loosely coupled connections) and their use of open access (mainly HTTP), SOA infrastructures implemented by web services add a new set of requirements to the security landscape. Web services security requirements also involve credential mediation (exchanging security tokens in a trusted environment), and service capabilities and constraints (defining what a web service can do, under what circumstances).[4]

Let's look at some of the ways to keep Web-Services secure. This paper tries to enumerate few of the security tools that have been introduced by the industry which make Web Services more secure. The first major aspect that I will look into is Authentication.

II. Authentication

Authentication is needed to protect resources and control the access to these resources. If SOA concepts are to be implemented then the authentication procedure should be seamless between different entities and the user should not be asked to login more than once.

Service-to-service authentication is possible using variety of methods like HTTP-based to SSL certificate based. If we look into the SOAP message then the new protocols gives us an added option of passing tokens along with the SOAP request. Mostly the HTTP and SSL based authentication is transparent to the Web service while SOAP-based token protocols require interaction between Web services.

Web services that use tokens for authentication are best served by the OASIS WS-Security standard. Currently five token types are defined. These are the Username Token, X.509 token, the SAML token, Kerberos token, and the Rights Expression Language (REL) token. When a service provider attempts to access a remote Web service, it has the option to send an authentication token, impersonating the user within a WS-Security message.

1) Username Token

2) X.509 Certificate Token

An X.509 certificate specifies a binding between a public key and a set of attributes that includes (at least) a subject name, issuer name, serial number and validity interval. This binding may be subject to subsequent revocation advertised by mechanisms that include issuance of CRLs, OCSP tokens or mechanisms that are outside the X.509 framework, such as XKMS. An X.509 certificate may be used to validate a public key that may be used to authenticate a SOAP message or to identify the public key with a SOAP message that has been encrypted.[5]

3) The Rights Expression Language (REL) Token

4) SAML Token

Security Assertion Markup Language (SAML) is an XML standard for exchanging authentication and authorization data between entities SAML is a product of the *OASIS* Security Services Technical Committee.[6]

A SAML specification defines

- Assertions: It basically defines the three A's i.e. Attribute, Authentication and Authorization data.
- Protocol: This defines the main elements taking place in the Web-Service Request/Response standard and they help in packaging assertions.
- Bindings: Clearly lays out the way to map SAML Protocols on all the other messaging and communication protocols.
- Profiles: Defines the combination of bindings, assertions and protocols to support a particular use case.

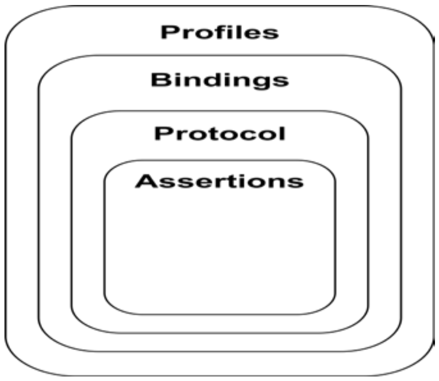

Figure 1 : SAML Structure

An assertion contains a packet of security information

```
<saml:Assertion…>
  …
</saml:Assertion>
```

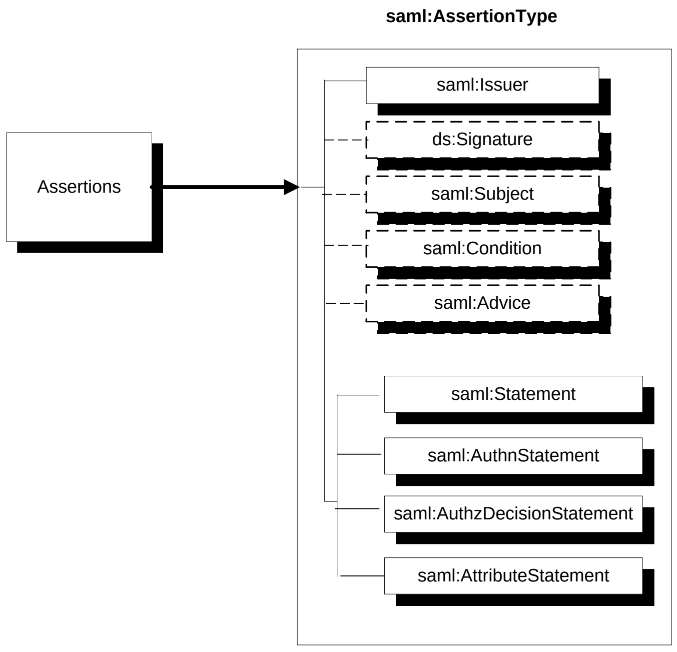

Figure 2 : SAML Assertion Element

For authentication between different organisations and their web services the best method is to use Identity and Trust based federated authentication mechanism. This sort of large scale model is well supported by WS-Trust and WS-Federation specifications developed to support WS-Security.

III. Direct Authentication

If a web-client needs to use a web-service, then the web-service will require the client to authenticate itself so as to enforce further authorization and auditing controls. Following are some of the ways this can be achieved

- A client may present either a shared secret or a password to the web-service for authentication.
- An identity server helps the web-service to validate credentials of the client.
- If the Web-service is a simple one then it may not need any authentication.
- Both the Web-service and the consumer trust one another and do not need any other kind credential management.

In all of the above cases we can use direct authentication scheme under which the Web service acts as an authentication service and the client presents its credentials for validation directly to the web-service. This credential will include some kind of shared secret and will be checked against an identity store.

IV. Brokered Authentication

Let's look at the following few scenarios which will explain the need for Brokered Authentication

The client may be using different services, i.e. more than one service.

No Trust between the service and client.

The Identity server and Web-Service have no trust.

Brokered authentication is used by the Web service to validate the credentials presented by the client. It does not need a direct relationship between client and service. This process has the following players

Client. The principal agent who initiates a request to the Web-service.

Web-Service. Web service that provides a response based on proper authentication.

Trusted Broker. By issuing a token to the client it issues a promise that the client is safe and authorized to use the Web-Service

Identity store. The server which holds the credentials for every authorized client on the domain.

Figure 4, describes these steps which will make things clearer

1) The client requests authentication from authentication broker.
2) The broker communicates with the identity store and validates the client.
3) The Broker on successful authentication responds with a security token. This token has a predefined time to live and during its lifetime it can be used by the client to authenticate itself and request any service from the server.
4) A request is submitted to the service in this request security token from the previous step is attached.
5) The service validates the security token and authenticates the incoming request as genuine and authorized.
6) If the token is validated successfully then the server responds to the client as requested in the 4th step.

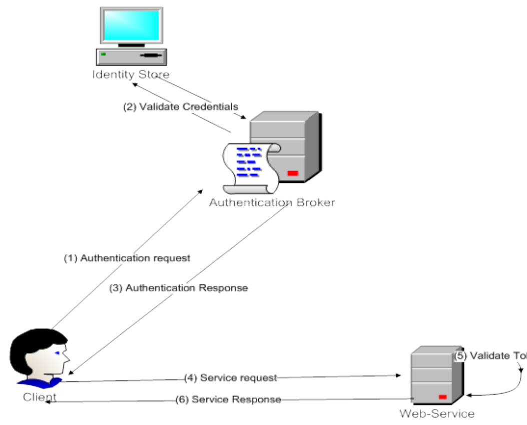

Figure 3 : Brokered authentication

Authentication brokers can be of different kind but essentially all perform the same task. Three popular authentication brokers are:

1. X.509 PKI
2. Kerberos protocol
3. Web Service Security Token Service (STS)

V. X.509 Pki

In this process we use X.509 certificates given by a certificate authority using in a public key infrastructure for verification of the credentials presented by client application.

When a Web-service receives the message, it uses the public key, to validate the signature.

Following players are involved in this method of authentication

Certificate authority. It is an authentication broker that is delegated with the task of generating and forwarding X.509 certificates.

Client. The principal agent who initiates a request to the Web-service and needs to authenticate itself with the token provided by the broker.

Web-Service. Web service that provides a response based on proper authentication.

In order to ensure a consistent processing model across all the token types supported by WSS: SOAP Message Security, the **<wsse: Security Token Reference>** element SHALL be used to specify all references to X.509 token types in signature or encryption elements that comply with this profile.[5]

VI. Brokered Authentication: Kerberos

Kerberos protocol can be used for authentication. It will work like a broker between the client and the server. The client sends a request to the broker for a ticket. The broker returns a service ticket and session key used to create a Kerberos security token. The security token carries the service ticket and a special piece of data called *authenticator.* This is encrypted by using the session key. The client can now send the Kerberos security token with its request to the Web service.

On receipt of this token, web-service extracts encrypted ticket. Then it uses its own service key to decrypt the service ticket. This session key from the service ticket is used to decrypt the authenticator and authenticate the client.

Figure 5 summarizes the following steps

1. Client in its request attaches a Ticket Granting Ticket and forwards it to the KDC.
2. In response to the above request KDC generates a session key and service ticket. It contains data for client authorisation and the new session key. To protect the ticket KDC uses the Public Key of Web-service to encrypt it. On arrival of the response Client decrypts the session key and the authenticator is encrypted with this session key. The new Security Token in this case includes the authenticator and the ticket sent by the KDC.
3. The client is sends a request with this security token attached to the Web-service.
4. On receipt of such a message the Web-service uses its private key to decrypt the Service Ticket. This ticket contains the session key which is used to decrypt the authenticator. After the security token is validated, the Web-service is now ready to respond to the client.

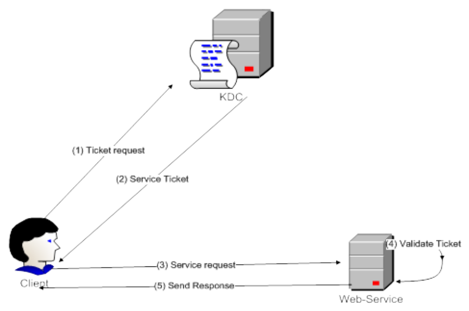

Figure 4 : Kerberos Brokered Authentication

VII. Brokered Authentication: WS-Trust & WS-Federation

Let's assume that many different clients have their authentication implemented on different platforms. We are required to bring seamless interoperability among these different clients. Only way to implement this is through the concept of Identity management and trust implementation.

Another use-case which ways in favour of these two is that sometimes organisations need security tokens which can be extended to support extra security. Thus we require a way that broker can be flexible enough to adjust to the needs of the user. This too can be achieved by Identity management.

Most of the commercial domains are protected by well entrenched firewalls thus one of the requirements for security can be that the security tokens must easily traverse or pass through these firewalls using the ports that re standard. As discussed earlier single sign on facility can be a by-product of this kind of security structure.

a) *Identity Management*

With SOA security stack Identity management has a very broad spectrum and it covers everything from documents, information, identity-related events etc. All of these can be used to confirm the identity of the client and authenticate him at the entry point of our SOA implementation. Under the security architecture of SOA an entity's identity is the basis for trust as well as authorisation.

b) *Identity Management Architectures*

There are three major identity architectures available for use in Web services:

Isolated identity management.

Federated identity management.

Centralized identity management.

c) *Usage of Identity Management with Web Services*

According to Axel Buecker and Heather Hinton, successful cross-organizational Web services require a way for providers to securely identify and provide services to authorized requesters and a way for requesters to securely invoke Web services with the necessary credentials.[7]

Without a proper identity framework things can become complicated in the Web-services environment. Let's say an organization Sudeep Inc. uses X.509 certificates to identify Web services and clients, while another company XYZ Ltd. has Kerberos tickets for identification. If a client from XYZ Ltd sends a request to a web-service under the domain of Sudeep Inc we will have a big issue as the client will have Kerberos ticket attached with the message while the web-service requires a X509 certificate. Even though both client and web-service are genuine yet the communication will fail. To make life easier for cross-domain and organisation communication we use Identity management frameworks. This enables the Web servers to safely identify each other. Irrespective of what kind of security apparatus is being used individually. According to

Buecker and Hinton, organizations need only develop a single set of Web services to facilitate Web service identity management across organizational boundaries:

Trust services.
Authentication and validation services.
Identity and attribute mapping services.
User lifecycle management services.
Authorization services.

d) Federated identity management

If you need to establish a business in a distributed environment then the first requirement is to manage identities in a federated way.

A federation is a set of organizations that establish trust relationships with respect to the identity information the federated identity information that is considered valid. A federated identity management system (IdM) provides a group of organizations that collaborate with mechanisms for managing and gaining access to user identity information and other resources across organizational boundaries[8].

This simplifies identity and credential management for the SOA as a whole, but requires individual services to be aware of and trust assertions from one another. In a single enterprise- wide SOA, it may not be difficult for providers to trust one another, but they may be less willing to trust assertions when the SOA includes providers from different organizations. A requester in the SOA may make a request to a provider and supply an arbitrary assertion to gain access. In identity federation, it is important to develop organizational policies appropriate for the types of data that traverse the SOA.

This benefits the user as well because they don't have to remember different credentials for different services of the same organisation. A single credential authenticates them for every service. This in turn increases the user experience. Many IdM systems use cookies to make user information available to servers. State information is stored at the client, which sends the cookie to the server the next time the user accesses that server. Like session and trust tickets, cookies can be valid only for the session during which they were issued or can persist beyond the session's end. A persistent cookie is typically written to a file on the browser's hard drive if its lifetime hasn't elapsed when the browser is shut down and therefore can be used for a longer period of time.[8]

Different Roles in Federated Identity Management Framework.

The two major roles are the identity provider and service provider or the Web-Service.

1) *Identity provider*
2) *Service provider*

e) Trust management

Trust is contract between two parties which entails them to believe the claims made by each other. Trust management is the process of or model for creating relationships amongst the different entities in an organisation, domains or systems. This infrastructure is created by cryptographic methods

Creating Trust amongst Web-Services

If a signed SAML or WS-Security message cannot be guaranteed to be trustworthy during communication between remote clients, then neither is of any use to anybody. Originally SAML had direct trust relationship but now it has brokered trust and community trust model too.

Direct trust relationships are the simplest of all because each entity has a copy of others public key and uses it to authenticate the communicating partner. It may be simple but not at all scalable.

Direct trust relationship has been enhanced and named as Brokered trust model. Under this scheme when two pairs are communicating with each other it is not necessary for them to share their public keys instead they exchange each other keys with the usage of Trusted Third Party. This model scales better as compared to the Direct Pairwise model.

Public Key Infrastructure is central to the third model called community trust model. Under this scheme trust is established through an external PK

I. This model is as simple as direct Pairwise model yet more scalable than the brokered trust model.

Trust Federation Frameworks

Let's look at some of the frameworks now being used to provide trust framework for web-services. It is up to the company to decide which framework is best suited for its unique needs.

SAML

SAML, developed by the Security Services Technical Committee of the Organization for the Advancement of Structured Information Standards (OASIS), is an XML-based framework for communicating user authentication, entitlement, and attribute information. As its name suggests, SAML allows business entities to make assertions regarding the identity, attributes, and entitlements of a subject (an entity that is often a human user) to other entities, such as a partner company or another enterprise application.[9]

WS-Trust

This protocol was developed and proposed by IBM, Microsoft, RSA, Verisign, BEA, and several other vendors. Stated goal of all these vendors was to create a federation system which has its root in SOAP and WSDL but uses the WS-Security extensions too.

WS-Trust addresses these issues by:

- Defining a request/response protocol
  - o Client sends RequestSecurityToken
  - o Client receives Request Security Token Response
- Introducing a Security Token Service (STS)

Brokered authentication with STS involves the following participants:

- Client. The client accesses the Web service.
- STS. The STS is the Web service that authenticates clients by validating credentials that are presented by a client.
- Service. The service is the Web service that requires authentication of a client prior to authorizing the client.[10]

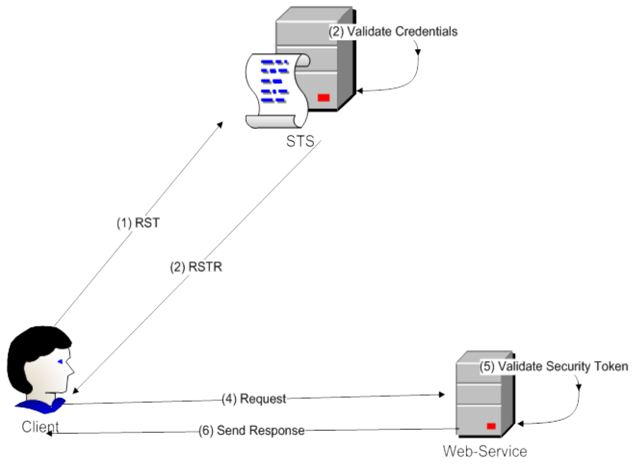

Figure 5 : STS token issuance and request/response

WS-Federation

The goal of federation is to allow security principal identities and attributes to be shared across trust boundaries according to established policies. The policies dictate, among other things, formats and options, as well as trusts and privacy/sharing requirements. In the context of web services the goal is to allow these identities and attributes to be brokered from identity and security token issuers to services and other relying parties without requiring user intervention (unless specified by the underlying policies). This process involves the sharing of federation metadata which describes information about federated services, policies describing common communication requirements, and brokering of trust and tokens via security token exchange (issuances, validation, etc.).

Federations must support a wide variety of configurations and environments. This framework leverages the WS-* specifications to create an evolutionary federation path allowing services to use only what they need and leverage existing infrastructures and investments. Federations can exist within organizations and companies as well as across organizations and companies. They can also be ad-hoc collections of principals that choose to participate in a community. [11]

Requestor: A programmatic agent for obtaining information or service

Subject: The entity on whose behalf a Request or operates

Claims: Statements made about a subject.

Security Token: A data structure for expressing collections of claims

Security Token Service (STS): A Web service that provides issuance and management of security tokens.

Identity Provider (IP): An entity, typically a trusted third party authority that provides claims about a set of Subjects

IP/STS: STS operated by an IP to issue claims using tokens

Relying Party (RP): An entity that provides information or services to Requestors based on claims they present

Target Service: A web service (or application) operated by an RP

RP/STS: STS operated by a RP to issue claims using tokens[12]

WS-Federation expands on WS-Trust by providing various protocols by which STSs (interchangeably called Identity Providers in WS-Federation), requesters, and providers can interact with one another to allow Web services to trust each other across organizational boundaries. Each organization is a separate *trust realm*. WS-Federation allows Web services to communicate between multiple trust realms. Additionally, WS-Federation provides two profiles for how requesters interact with providers and STSs: the active requester profile and the passive requester profile. The passive requester profile details how messages should be passed between a requester Web browser, the provider, the Identity Providers (IPs) and STSs of both organizations so that WS-Federation can be used within the context of Web applications, providing users with a single sign-on experience. The active requester profile details how requesters should interact with the provider and the IP/STSs to access a provider in another trust realm. [13]

VIII. Soap Message Security -Xml Security

SOAP is the base on which communication through message interchange structure is built. SOAP itself is a XML based protocol. Unfortunately the data in the SOAP message are vulnerable to confidentiality and integrity threats. We all know that TLS provides end-to end security, unfortunately SOAP headers can be modified by intermediary nodes leaving a big security hole. To further make matter worse are the different protocols of different networks which may be the intermediary node. If we implement security on the XML level it helps in improving source integrity and confidentiality.

As used in computer security, cryptography provides the following processes:[14]

- ✓ Encrypting converts plaintext (that is, data in normal, readable form) into cipher text, which conceals the

Global Journal of Computer Science and Technology (E) Volume XII Issue XVII Version I Year 2012

meaning of the data to any unauthorized recipient. Encrypting is also called enciphering. Most cryptographic systems combine two elements:

- ➢ An algorithm that specifies the mathematical steps needed to encrypt the data.
- ➢ A cryptographic key (a string of numbers or characters), or keys. The algorithm uses the key to select one relationship between plaintext and cipher text out of the many possible relationships the algorithm provides. The selected relationship determines the composition of the algorithm's result.
- ✓ Decrypting converts cipher text back into plaintext. Decrypting is also called deciphering.
- ✓ Hashing uses a one-way (irreversible) calculation to condense a long message into a compact bit string called a message digest.
- ✓ Generating a digital signature involves encrypting a message digest with a private key to create the electronic equivalent of a handwritten signature. You can use a digital signature to verify the identity of the signer and to ensure that nothing has altered the signed document since it was signed.

WS-Security provides much flexibility, marrying SOAP messaging with multiple security standards and technologies: The standard extends the SOAP Header to provide security information for secure messaging, it leverages lower-level standards such as XML Signature and XML Encryption, it is extensible to support multiple token formats for identity and authorization, and it supports multiple trust models for sharing security contexts.[15]

WS-Security gives us the added benefit of using multiple encryptions within the same SOAP message, thus various parts of the same SOAP message can be encrypted for different receivers (SOAP intermediaries). In the same way an intermediary can add its own additional signature to the message. This has the effect of providing integrity protection for newly added header. XML Signature and XML Encryption are the first line of defence in XML and Web services security. Therefore both are well supported in available products and development API's.

WS-Security has the added advantage of providing a mechanism for avoiding replay attacks (i.e., timestamps) and a way to add security tokens to the message being communicated. A drawback of WS-Security is that it has no concept of session, and it is focused on securing a single SOAP message or a single SOAP request/response exchange. Where we require security for multiple message exchanges, WS-Secure Conversation is the protocol that can be used to maintain a security context.

IX. XML-Signature

Signatures are used to verify message origin and integrity. Signatures are also used by message producers to demonstrate knowledge of the key, typically from a third party, used to confirm the claims in a security token and thus to bind their identity (and any other claims occurring in the security token) to the messages they create.[16]

XML Signature is the specification that defines a standard interoperable format for representing digital signatures in XML. It lays out a method that shows us a way to efficiently apply signature to primarily XML messages. It can be used on binary files too but that is not the focus of this paper

XML Signature gives us a practical and flexible signature mechanism. Yet coders must think about the threat perception of their application and should keep in mind few of these points

A. What is Signed is Secure
B. Only data item should be signed
C. Signature for external references too

X. XML Encryption

XML Encryption has been designed to provide SOAP/XML documents confidentiality by encrypting them completely or partially. Both XML Encryption and XML Signature are similar standards. Even encryption is not only limited to XML but it can be used for binary data too. This standard demands support for both Triple-DES and AES-128/256.

XI. Impact Of Security on Time Complexity

Though security is now an inevitable part of modern communication yet we must be prepared to sacrifice a bit on time complexity. In this section I have tried to bring out some differences between secured and unsecured web-service. The web-service used are technically very simple, they have separate functions for primitive data types and a simple class. We will be calling the web-service from a client application in three scenarios.

1. First scenario will be when client and server applications are on the same machine
2. Second scenario will be when client and server are on different machines but in the same domain.
3. Third scenario will be with the server application on a shared host and will be accessed by the client over the internet.

The factors which I have taken into account are as follows.

- Reliable Messaging
- Security
- Concurrent Clients

The result variable has been kept simple. I will measure the output by comparing the Response Time which is the total time elapsed from the point client makes a request to the time client receives a response. The experiment uses simple data types as messages and to even out any kind of noise or disturbance each experiment has been replicated 100 times. The need for replication arose due to the variance in Response Time between two identical requests. The reason for such a variance is manifold, it could be due to network latency, client machine lag or server load. To summarize the data I have calculated Average Response Time, which tends to reduce the effect of noise on the data. It may be argued that the output variable should have been more complex or dependent on other factors but this paper doesn't deal with it. Following table summarizes the data.

Firstly we find that when web-services do not use 'Reliable Messaging' and 'Security' the average response time is always lower. Unfortunately in many scenarios not using security is not an option.

Secondly looking at time complexity I would say that using only Reliable Messaging is much better than using security. If we are forced to use both types of security then performance is surely to take a severe hit as represented in the data table.

Third point which becomes apparent is that if number of concurrent user increases the performance decreases. This is true for most software's but in web-services we must understand that this becomes more critical as web-services are meant for an environment where number of users are not fixed and more often than not internet will be the communication medium.

The experiment quite clearly brings out the problem with using security on web services. Unfortunately there are no other alternatives at present. Therefore if we require security then we must be ready to sacrifice response time. My experiment is not definitive due to the fact I have ignored all error mechanisms and 'data noise'. Noise tends to pollute the result. Plus I have used simple messages instead of complex messages.

XII. Conclusion

The standards I have looked into are not the only one in existence there are many more and depending on your needs you might have to look at few of them.

Web services have this great ability to produce extremely inter-operable systems and loosely coupled architecture. It is and will give every organisation a high degree of flexibility in their solution architecture. If we are to use this inherent flexible nature to its full, it is essential to understand the security threats lurking in the shadows and devise methods to minimize or eradicate these threats. Fortunately most of the vendors have jumped on this bandwagon thus we see so many security standards are being designed. Many of these vendors like IBM, Microsoft, Google etc are churning out API's and tools to support these security standards. We as developers are fortunate to be spoilt for choices.

If we look at the most widely used languages of web like C#, VB.NET and each one of them already have API's in place to tackle security issues. Unfortunately the specifications are standard but the API's are not. Thus there are quite a few differences in all these competing vendor specific implementation.

Next problem is the user apathy toward implementation of these new standards. Robust security for Web Services is mandatory and most of the technology, standards protocols etc are already in place. The problem lies in mapping this existing security technology to XML and SOAP, and it is neither easy nor short. This mapping is of non-trivial nature and you need a deep understanding of the specifications as well as the language. If either of the knowledge is missing you will end up implementing faulty security for your services.

Once the user and the vendor are on the same wave length we will see the world of secured and completely seamless interoperable world of web services that we all want become reality. Use of a defined set of interfaces, along with centralized identity and access control policies, will reduce the risk of user access to unrelated resources. Running computing services in isolated domains, providing default encryption of data in motion and at rest, and controlling data through virtual storage have all become activities that can improve accountability and reduce the loss of data. In addition, automated provisioning and reclamation of hardened run-time images can reduce the attack surface and improve forensics.[17]

References Références Referencias

1. R. A. Lutz Lowis, "Vulnerability Analysis in SOA-Based Business Processes," presented at the IEEE TRANSACTIONS ON SERVICES COMPUTING, 2011.
2. "The XML Security Gap: It's Bigger than you think," White Paper2003.
3. K. Hongzhao, "A Study on the Security Mechanism for Web Services," presented at the World Congress on Engineering and Computer Science, San Francisco, USA, 2010.
4. V. Jain. (2009 20/09/2012). Securing Web Services and Service- Oriented Architectures with Oracle Web Services Manager 11g [White Paper].
5. OASIS, "Web Services Security X.509 Certificate Token Profile 1.1," in *Introduction*, ed: OASIS Open, 2006.
6. T. H. Nathan Klingenstein, Hal Lockhart,Scott Cantor, Anil Saldhana. (2012 08/09/2012). *OASIS Security Services (SAML) TC*. Available: https://

www.oasis-open.org/committees/tc_home. php? wg_abbrev=security

7. P. A. Axel Buecker, Neil Readshaw, "Federated Identity and Trust Management," ed. USA: IBM, 2008.
8. S. ABHILASHA BHARGAV, ANNA C.SQUICCIARINI, ELISA BERTINO, "Trust Negotiation in Identity Management," presented at the IEEE SECURITY & PRIVACY, 2007.
9. OASIS, "SAML V2.0 Executive Overview," in *Committee Draft*, ed: OASIS, 2005.
10. Microsoft. (2005, 29/10/2012). *Brokered Authentication: Security Token Service (STS)*. Available: http://msdn.microsoft.com/en-us/ library/ ff650503.aspx
11. OASIS, "Proposed Charter: OASIS Web Services Federation (WSFED) Technical Committee," in *WS-Federation Specification Overview*, ed, 2007.
12. OASIS, "WS-Federation 1.1 Overview," ed. OASIS WSFED: OASIS, 2007.
13. T. W. Anoop Singhal, Karen Scarfone, "Guide to Secure Web Services," vol. 800-95, ed. USA: National Institute of Standards and Technology, 2007.
14. C. A. Nigel Williams, Arnaud Desprets,Tommy Joergensen,James O'Grady, "Securing CICS Web Services," in *Securing CICS Web Services* vol. First Edition, 1 ed: IBM, 2008, p. 70.
15. B. L. Mike Rosen, Kevin T. Smith,Marc J. Balcer, *Service-Oriented Architecture and Design Strategies*. Indianapolis: Wiley Publishing, Inc., 2008.
16. OASIS, "Web Services Security:SOAP Message Security 1.1," in *Message Security Model*, ed: OASIS, 2006, p. 13.
17. IBM. (2009, IBM Point of View: Security and Cloud Computing. *Cloud computing White paper,* 19. Available: ibm.com/cloudcomputing

This page is intentionally left blank