Indian J.Sci.Res. 17(2): 312-322, 2018 ISSN: 0976-2876 (Print)
ISSN: 2250-0138(Online)

SECURITY ASPECTS OF REST WEB SERVICES

[1] Amar Pimpalkar, [2]Dr Manish Jivtode
[1,2] Department of Computer Science, Janata Mahavidyalaya, Chandrapur.

Abstract **-** Security is an important factor in every application but client, server and the communication channel in distributed applications (with distributed databases as back end) are even more exposed to outside attacks. When choosing REST service, many factors should be considered. The most important of these factors are confidentiality, Integrity, Authentication, Authorization, Non-repudiation, and Availability. When determining REST service model will be more secure for a particular application, the decision should not be made on the basis of available security features. In this paper we tried best to enlighten the design issues and security issues of REST web service and gave the suggestion to improve design and security issues of REST web services. This paper will observe the fundamental features of the REST web service model. The design will improve scalability, accessibility and flexibility. We suggested some solutions for security aspects such as confidentiality, Integrity, Authentication, Authorization, Non-repudiation, and Availability that affect to a REST service model.

Keywords**:** Web service security, SOAP, REST security model, Scalability, Accessibility, Flexibility, Message security, HTTP, HTTPS

I. Introduction

Web services are open standard (XML, SOAP, HTTP, etc.) based web applications that interact with other web applications for the purpose of exchanging data. Web services can convert existing applications into web applications. Figure 1 below shows exchanging information between two applications.

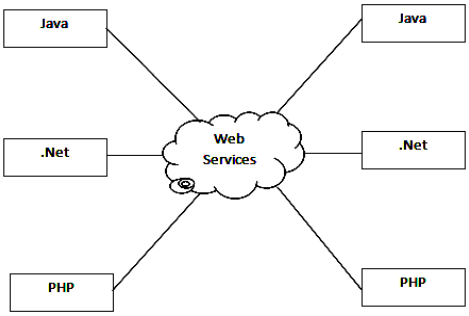

Figure 1: Exchanging Information Between Two Applications

Source: Borrowed from [1]

In the above figure shows, the java, .net or PHP applications communicate with other applications through web service over the network. For example, Java application interacts with Java, .Net and PHP application. So web service is a language independent way of communication. There are mainly two types of services – SOAP web services and REST web services.

SOAP stands for Simple Object Access Protocol. It is a XML-based protocol for accessing web services. SOAP is a W3C recommendation for communication between two applications. SOAP is XML based protocol. It is platform independent and language independent. By using SOAP, it can interact with other programming language applications.

SOAP web services written in any programming language and executed in any platform. SOAP defines its own security known as WS Security.

REST stands for Representational State Transfer. Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induces desirable properties, such as performance, scalability, and modifiability that enable services to work best on the Web. In the REST architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs), links on the Web. The resources are acted upon by using a set of simple, well-defined operations. The REST architectural style constrains architecture to client/server architecture and is designed to use a stateless communication protocol, typically HTTP. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol. REST web service permits different data format such as Plain Text, HTML, XML and JSON. REST requires less bandwidth. JAX-RS is the java API for REST web services.

REST web services inherit security measures from the underlying transport.

II.Service Oriented Architecture (SOA)

A service-oriented architecture is a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some

[1]Corresponding Author

activity. The SOA provides almost infinite flexible and scalable external computing and processing services that not only offer significant cost benefits, but also provide the ability to connect with customers, partners and suppliers.

Figure 2: Service Oriented Architecture (SOA)

Source: Borrowed from [2]

In SOA approach, software resources are different than traditional architecture. It is an application architecture within which all functions are defined as independent services with well-defined invokes able interfaces, which executed in defined sequences to form business processes. SOA is a software architecture that starts with an interface definition and builds the entire application topology as a topology of interfaces, interface implementations and interface calls [9].

A. Service

SOA represents a model in which functionality is decomposed into distinct units called services which can be distributed over network and can be combined together and reused to create business applications. A service can be a simple and independent business function or a business transaction but which may be implemented as composite function transparent to the caller [3]. They are software modules that are accessed by name via an interface, in a request-reply mode. Hence, they provide necessary transparency as well as user-friendliness.

B. The Benefits of SOA

- An SOA developed from existing system, created using existing technologies with component-based approach. It is generates new routes with flexibility.
- SOA stress the development of loosely coupled services as the software construction unit [4].
- SOA is embedded in object-orientation but adds a layer of abstraction. Service-orientation is not a departure from object-orientation, but rather an evolution [5].
- SOA from a technology perspective, but a policy, a practice, and a framework by which it is ensured that right services are provided and consumed [6].
- The development of Web Services using service oriented technology provide heterogeneous network addressable service component for location transparency. SOA new services to customers without worrying about the underlying IT infrastructure.
- SOA provide cost effectiveness by integrating historically separate systems with reduction in cycle times and costs, and reduces risk by improving visibility of operation.

C. SOA implementation

Software architectures built upon well defined characteristics in which loose coupling is the significant one, as it has to deal with complexities and changes continuously. Any architecture need not be strictly dependant on specific technology. SOA is a design principle, which is not tied to any technology but implemented using SOAP, RPC, DCOM, CORBA, Web Services etc. The CORBA is widely used technology and provides a rich development environment but required to learn a new programming model and does not support interoperability and cost-saving application development as it is a tightly coupled architecture [7]. The major advantages of Web Services are loose coupling and interoperability.

D. Role of SOA

- **Reusability:** SOA is reuse of business services. Developers within an enterprise and across enterprises take the code developed for existing business applications, expose it as web services, and then reuse it to meet new business requirements. They run in different operating environments, they are coded in different language. They use different programming interfaces and protocols [8].
- **Interoperability:** The SOA vision of interaction between clients and loosely-coupled services means wide spread interoperability. Web services comprise a maturing set of protocols and technologies that are widely accepted and used, and that are platform, system, and language independent [9].
- **Scalability:** Services in an SOA are loosely coupled; applications that use these services tend to scale easily than applications in a more tightly-coupled environment. An asynchronous service performs its processing without forcing the client to wait for the processing to finish. A synchronous service forces the client to wait. The relatively limited interaction required for a client to communicate with a coarse-grained, asynchronous service, and applications that use these services to scale without putting a heavy communication load on the network [10].
- **Flexibility:** Loosely-coupled services are more flexible than more tightly-coupled applications. In a tightly-coupled architecture, the different components of an application are tightly bound to each other,

sharing semantics, libraries, and often sharing state. This makes it difficult to evolve the application to keep up with changing business requirements. The loosely-coupled, document-based, asynchronous nature of services in an SOA allows applications to be flexible, and easy to evolve with changing requirements [11].

III.WCF services

Windows Communication Foundation (WCF) is a program development platform and runtime system for building, configuring and deploying network-distributed services like Secured Transactions [12]. It is the latest service oriented technology developed by Microsoft. It is unified programming model provided in .Net Framework 3.0. WCF is a combined feature of old Microsoft Framework Technologies like Web Service, Remoting, MSMQ and COM+. WCF provides a common platform for all .NET communication.

WCF services provide better reliability and security compared to ASMX web services. In WCF, there is no need to make much change in code for executing the security model and alter the binding. Small changes in the configuration file will match requirements, and WCF does not support overloading. WCF framework for building service-oriented applications sends data as asynchronous messages from one service endpoint to another. A service endpoint can be a part of a continuously available service hosted by Internet Information Server (IIS), or it is a service hosted in an application. An endpoint is a client of a service that requests data from a service endpoint. The message is as simple as a single character or word sent as XML, or as complex as a stream of binary data.

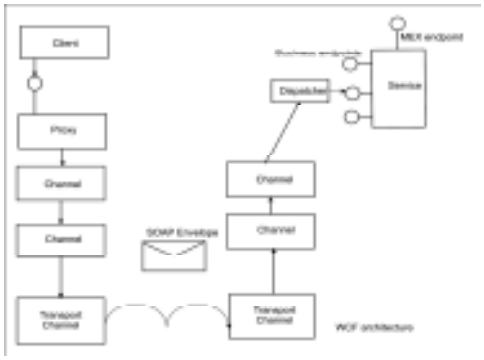

Figure 3: WCF web service architecture

Source: Borrowed from [14]

A. Communication between endpoint of WCF

Windows Communication Foundation (WCF) is a framework for building service-oriented applications (SOA). SOA is taken into account in Windows Communication Foundation and services are in order to communicate and exchange messages in distributed systems. Invoking (calling) methods placed on distinct nodes on WCF services is similar to invoking web methods from web services. Creating a WCF client is even simpler. All that's required is to create a local stand-in for the service which is called a proxy that is connected to a particular endpoint on the required target service, and then invoke the service's operations via the proxy. Creating a proxy requires knowing exactly what contract is exposed by the target endpoint, and then using the contract's content to generate the proxy [15].

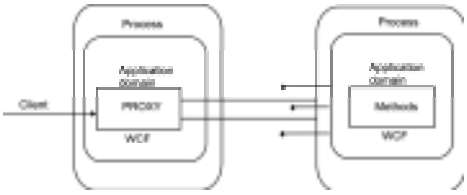

Figure 4:Proxy generation in the server side of WCF

B. Structure of WCF

Web services are part the of remote communication enabled through WCF, with higher degree of flexibility and portability in WCF than through traditional ASMX because WCF is designed, from the ground up, to summarize all of the different distributed programming infrastructures offered by Microsoft [16]. WCF Service is a program that exposes a collection of Endpoints. Each Endpoint is a portal for communicating with the clients ends. Different End points created with a different port address in the configuration file of the client side. All the WCF communications are take place through end point. End point consists of three components as shown in figure.

Figure 5:WCF service end points

(Source: Borrowed from [13]

a) URL structure of an address endpoint

The address uniquely identifies the endpoint and tells potential consumers of the service where it is located. This address consists primarily of a Uniform Resource Identifier (URI), which specifies the location of the endpoint. The address URI for most transports has four parts. The four parts of the URI http://www.example.com:322/mathservice.svc/secureEndpoint can be itemized as follows:

- Scheme: http:
- Machine: www.example.com

- (optional) Port: 322
- Path: /mathservice.svc/secure Endpoint

WCF supports the following transport schemas:

- HTTP
- TCP
- Peer network
- IPC (Inter-Process Communication over named pipes)
- MSMQ

b) Binding endpoint

The binding specifies how to communicate with the endpoint. This includes:

- The transport protocol to use TCP or HTTP
- The encoding to use for the messages that is text or binary
- The necessary security requirements for SSL or SOAP message security

c) Contract endpoint

The contract outlines what functionality the endpoint exposes to the client. A contract specifies:

- What operations can be called by a client?
- The form of the message and the type of input parameters or data required to call the operation.

In WCF, all services expose contracts. The contract is a platform-neutral and standard way of describing what the service does.

WCF web service defines following types of contracts -

- **Service contracts:** Describe which operations the client performs on the service.
- **Data contracts:** Define which data types are passed to and from the service. WCF defines implicit contracts for built-in types such as integer and string, but easily define explicit opt-in data contracts for custom types.
- **Fault contracts:** Define which errors are raised by the service, and how the service handles and propagates errors to its clients.
- **Message contracts:** The service to interact directly with messages. Message contracts typed or un-typed and are useful in interoperability cases and existing message format.
- **Operation contracts:** A WCF Contract is a collection of Operations that specifies endpoint communicates to the outside world. Each operation is a simple message exchange, for one-way or request/reply message exchange. Each contract has a collection of Contract Behaviors that are modules that modify or extend the contract behavior. Behaviors use endpoint behaviors to customize the local behavior of the service endpoint. Endpoint behaviors achieve this by participating in the process of building a WCF runtime. An endpoint behavior is the Listen Uri property; specify a different listening address than the SOAP or Web Services Description Language (WSDL) address.

C. WCF hosting

WCF service is hosted by any managed process running in the operating system. WCF service must be hosted in a Windows process called the host process. A single host process can host multiple services, and the same service type can be hosted in multiple host processes. WCF makes no demand on whether or not the host process is also the client process [18].

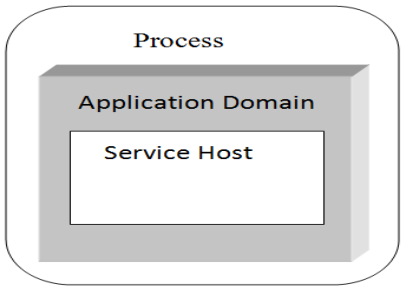

Figure 6:WCF Service Host relationships

(Source: Borrowed from [20])

The mechanisms that are used to host a WCF service are:

- **Internet Information Server (IIS):** Internet information Service provides number of advantages if a service uses HTTP as protocol. It does not require Host code to activate the service, it automatically activates service code.
- **Windows Activation Service:** (WAS) is the new process activation mechanism that ships with IIS. In addition to HTTP based communication, WCF can use WAS to provide message-based activation over other protocols, such as TCP and named pipes.
- **.EXE:** WCF services to be run as executables (.EXE files).
- **COM+:** WCF services to be run as a COM+ application.
- **Self-Hosting:** WCF service hosted as console application, Win Forms or WPF application with graphical UI.
- **Windows Services:** WCF services to be run as a Windows service.

IV. Representational State Transfer

REST is architecture for developing web services. REST architecture uses HTTP or similar protocols, by constraining the interface to a set of well-known standard operations (i.e., GET, PUT, POST, and DELETE for HTTP). REST architecture is designed to show how existing HTTP is enough to build a Web service and to show its scalability.
Roy Fielding, who is also one of the principal authors of HTTP, in his doctoral thesis [13], "Architectural Styles and the Design of Network-Based Software Architectures," defined a set of architectural principles for the Web called REST. REST architectural principles can be used for designing Web service solutions. REST focuses on a system's resources, including how resource states are addressed and transferred over HTTP [17].

The main idea behind REST was to use well-developed HTTP for transferring data between machines, rather than using a protocol that works on top of the HTTP layer for message transfers. An application designed following REST principles would use HTTP to make calls between the machines, rather than relying on complex mechanisms like CORBA (Common Object Request Broker Architecture), RPC (Remote Procedure Call), or SOAP. Therefore, REST applications use HTTP request functions to post data, read data, and delete data, thus using the full functionality of HTTP Create, Read, Update, and Delete (CRUD) operations. REST can run on HTTPS, providing for the secured transmission of data.

The CRUD operations in conjunction with HTTP REST functions are shown -

Table 1 HTTP methods and CRUD action

CRUD operation	REST keywords (HTTP)
READ-read, retrieve	GET
CREATE-create or add new entries	POST
UPDATE-update or edit existing data	PUT
DELETE-delete existing data	DELETE

(Source: Borrowed from [30])

The application of REST principles for web services requires HTTP protocols find it easy to understand and apply REST principles.
The term "Restful" is like "object-oriented," a language, a framework, or an application that may be designed in an object-oriented way, but that does not make its architecture the object-oriented architecture. Restful Web services are emerging as the choice for many of the leading Internet companies to expose their internal data and functionalities as URI identified resources. Some of the advantages of Restful web services include:

- **Light-weight:** Restful web services directly utilize HTTP as the invocation protocol which avoids unnecessary XML markups or extra encapsulation for APIs and input/output. The response is the representation of the resource itself, and does not involve any extra encapsulation or envelopes. As a result, Restful web services are much easier to develop and consume than WSDL/SOAP web services. Additionally, they depends less on vendor software and mechanisms that implements the additional SOAP layer on top of HTTP. Restful web services deliver better performance due to their light-weight nature.
- **Easy-accessibility:** URIs used for identifying Restful web services are shared and passed around to any dedicated service clients or common purpose applications for reuse. The URIs and the representation of resources are self-descriptive and make Restful web services easily accessible. Restful web services have been widely used to build Web 2.0 applications and mashups.
- **Scalability:** The scalability of Restful web services comes from its ability to naturally support caching and parallization / partitioning on URIs. The responses of GET (a side effect free operation) are cached exactly the same as web pages are currently cached in the proxies, gateways and content delivery networks (CDNs). Additionally, Restful web services provide a very

simple and effective way to support load balancing based on URI partitioning. Compared to ad-hoc partitioning of functionalities behind the SOAP interfaces, URI-based partitioning is more generic and flexible, and could be easier to realize.

- **Declarative:** To imperative services from the perspective of operations, Restful web services take a declarative approach and view the applications from the perspective of resources. Being declarative means that Restful web services focus on the description of the resources themselves, rather than describing how the functions are performed. Declarative approach is considered to be a better choice to build flexible, scalable and loosely-coupled SOA systems.

REST as a resource, and in the words of Dr. Roy Fielding, "a resource is anything that is important enough to be referenced as a thing in it". Static web pages do not change if re-requested by the same or a different user. Dynamic Web pages could change and, unlike static ones, are not cacheable. Each resource should have at least one URI

(Uniform Resource Identifier) to represent it. REST refers to it as resource representation. URI is the Web address of the resource, and makes the resource available online. A resource was accessed/referenced by its URI. Addressability is a way of representing a resource online "an addressable application exposes a URI for every piece of information it might conceivably serve". An application should be properly addressed to make it available online. Consider searching for something online with a search engine. The search engine is addressable and HTTP is addressable. If the resource are searching is addressable, the search engine will pick it up.

Statelessness is concept of REST. Statelessness means that all HTTP requests are independent of one another. Every HTTP request from the client should contain all the information for the server, so the server does not have to keep state information from client requests. Defining addressability in terms of statelessness, all possible states of the server are also resources and should be represented by a URI.

A stateless Web service has advantages, such as using load-balanced servers for performance. Since no request to the Web service is dependent on another, and because the client carries all the necessary information, there is no need for coordination among the servers. This eliminates the processing time for servers for the earlier data/process operations. Each state of a resource is called representation. A resource has many states and each state be represented individually. The states of a resource are analogous to the states of matter, as in a liquid, gas, or solid. Consider the Red box i.e. a DVD rental sharp are search for the availability of a DVD at a particular sharp online. The search will represent a DVD as available, will arrive soon, or checked out, so the same resource, the DVD, is represented in many states. This illustrates the importance of the keyword "Representation" in REST.

According to Dr. Roy Fielding, there are two regular perspectives on the process of architectural design - one is to start building everything without any plan, or building from scratch, and later add components to match the requirements. The other is to start building with a plan to implement all requirements. Additional features may be added later making them in accord with existing features and completing them depending on the requirements. REST components communicate by transferring a representation of a resource in a format matching one of an evolving set of standard data types, selected dynamically based on the capabilities or desires of the recipient and the nature of the resources.

The architecture of REST at the server side is as follows:

- URL: Mandatory field to access web services running at the server.
- GET: All the methods of getting data from the server; the formats and interfaces the server supports for accessing the client details.
- POST: All the methods of adding details to the server; all the different interfaces and formats the server supports for adding data to its database.
- PUT: All the methods for updating the data at the serve; different types of interfaces and formats the server supports for adding data to the database.
- DELETE: All the methods for deleting the data at the server. Different types of interfaces and formats the server supports for deleting data in the database.

V. Web services security

Web service security is a fundamental aspect of any enterprise level application and is applied in different systems. This means that web service needs to include features that deal with security risks, including falsification and eavesdropping on data that is being transmitted over communication paths, as well as on spoofing and repudiation.

The following are the core principles that web services are based on:

- Message Orientation
- Protocol Compos ability
- Autonomous Services
- Managed Transparency
- Protocol Based Integration

Web services use SOAP (Simple Object Access Protocol) exclusively for all messaging. SOAP has been to use the lowest common denominator (for data types) to exchange messages between platforms using standard XML based messaging. SOAP message is composed of three sections:

- <envelope>
- <header> and
- <body>

Which form the fundamental building block for all messages? This emphasis on interoperability and standards has fulfilled the growth of web services in the past couple of years [33].

Web Services security has seen the evolution on following

- SAML (Security Access Markup Language) was developed to provide a session based solution for authentication and authorization across multiple platforms. SAML was developed under the guidance of OASIS and ebXML,

- SPML (Service Provisioning Markup Language) was developed predominantly to address provisioning of accounts across disparate systems,
- XACML (eXtensible Access Control Markup Language) is a specification designed to express policies for information access over the internet,

WS-Security (Web Service Security) the standard was developed initially by IBM, Microsoft and VeriSign and now has been passed on to OASIS for further development. WS-Security is fast becoming the default standard for secure conversation using web services. In addition to security it also defines the framework for federation, policy and trust. Security in any application seeks to address two fundamental criterions –

- Confidentiality and
- Authentication / Authorization

The primary aim of confidentiality is to make sure that messages exchanged between requestor and the web service.

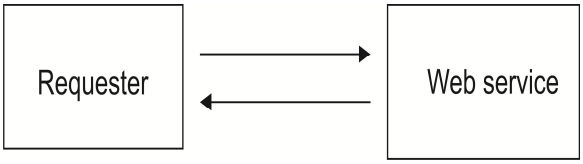

Figure 7:Interaction between Requester and Web service

The communication between the requestor and the web service can be secured using transport level security or message level security by encrypting the messages.

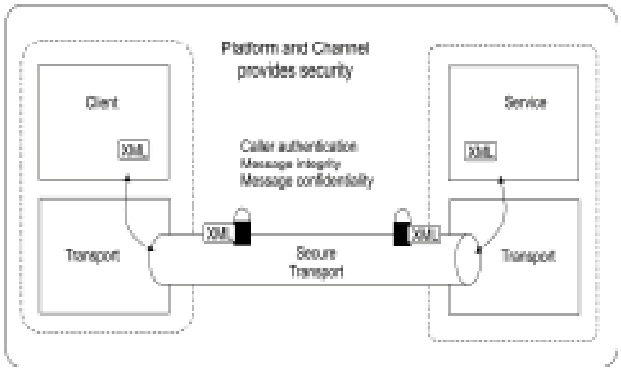

Figure 8 :ransport Level securities (point-to-point configuration)

Transport level security is implemented using HTTP/HTTPS session between the requestor and the web service. The HTTP/HTTPS encrypts all the data between the Requestor and the web service using digital certificate present at the web server. This sort of encryption is sufficient for most application but comes with a severe performance penalty. In any message only some parts of the message are confidential and HTTP/HTTPS does not give requestor the option to selectively encrypt parts of the message. Due to this the requestor has to incur high processor cost to encrypt the message and the web service then has to spend same amount of processor resources to decrypt the messages. There are some hardware solutions available in the market like the SSL Accelerator cards which allow the web services server to offload the task of decrypting the message. The other disadvantage of Transport Level Security is that it only offers point to point security. There is no option to secure the message beyond a server to server connection.

Use transport security in the following scenarios:

- Sending a message directly from application to a WCF service and the message will not be routed through intermediate systems.
- Both the service and the client are located in an intranet.

Using transport security has the following advantages:

- It provides interoperability, meaning that communicating parties do not need to understand WS-Security specifications.
- It may result in better performance.
- Hardware accelerators can be used to further improve the performance.

Using transport security has the following disadvantages:

- Security is applied on a point-to-point basis, with no provision for multiple hops or routing through intermediate application nodes.
- It supports a limited set of credentials and claims compared to message security.
- It is transport-dependent upon the underlying platform, transport mechanism, and security service provider, such as NTLM or Kerberos.

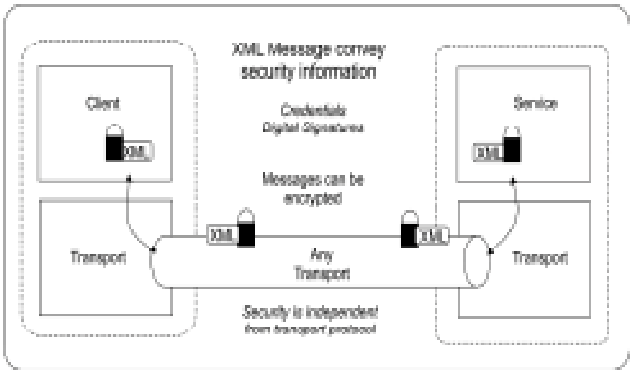

Figure 9:Message Level Security (end-to-end configuration)

Message level security addresses the needs of integrity, message authentication and confidentiality. It also the messages cannot be changed without detection. The messages are kept confidential using XML encryption standard with security tokens. Message integrity is assured by taking the checksum of the message and then sending the hash of the checksum in encrypted form. Due to the open nature of the XML when encrypting the messages standard method must be adopted to represent things like white spaces, line feeds and so on. This standardization is called Canonicalization and when applied produces a consistent result at both the requestor and the web service end.

Use message security in the following scenarios-

- Sending a message to a WCF service and the message is likely to be forwarded to other WCF services are routed through intermediate systems.
- WCF clients are accessing the WCF service over the Internet and messages are routed through intermediate systems.

Using message security offers the following advantages-

- It provides end-to-end security. Because message security directly encrypts and signs the message, having intermediaries does not break the security.
- It allows partial or selective message encryption and signing, thus improving overall application performance.
- Message security is transport-independent and used with any transport protocol.
- It supports a wide set of credentials and claims, including the issue token that enables federated security.

Using message security has following disadvantages:

- This option may reduce performance compared to transport security because each individual message is encrypted and signed.
- It does not support interoperability with older ASMX clients, as it requires both the client and service to support WS-Security specifications.

The traditional approach to authentication takes the credentials presented by the requestor send it to an authentication service for validation. The authentication service validates the credentials and issues a security token which is then used for the rest of the interaction. The token has a lifetime which is set by the authentication service. The credentials supplied by the requestor is called the digital identity of the requestor.

VI.Web Services with Message Security

Web services use the messages method based on XML to create and access services. XML is the security foundation of web services. In SOA, two requirements in the security area:

a) Identify management service. The mainly includes:

- Identity recognition verification
- Identity authorization
- Identity federation management and single logon.

b) Message level security: The confidential and privacy of the sensitive information must be protected during the exchange of sensitive information between the partners and probably by the safe way.

c) Security strategy dynamic adjustment: Security strategy updates and execute dynamically according to the changes of context to realize web service security access intelligently.

The different requirement with specification for secures the message.

Table 2 Secure specifications and standards addressing

Dimension	Requirement	Specification
Messaging	Confidentiality	WS-Security
		SSL/TLS
	Authentication	WS-Security Tokens
		SSL/TLS
		X.509 Certificate

d) Confidentiality: It is used to keep the information secret so that only intended recipient are read. Data confidentiality is accomplished by using security services i.e. encryption. With the help of encryption method, the data are accessed only to the authorized parties. In this case use openSSL tool measure the integrity or confidentiality.

e) Authentication: It is used to establish or valid the identity throughout the system. Authentication is accomplished by using validation method. With the use of validation, access the secured system with username and/or password.

Web service security is a message level standard that is based on securing SOAP messages through XML digital signature, confidentiality through XML encryption, and credential propagation through security tokens. The web services security specification defines the core facilities for protecting

the integrity and confidentiality of a message and provided mechanisms for associating security-related claims with the message.

Figure 10:Different Message Security Services

In web services, authentication is done by the security token service which authenticates the requestor and issues a token. The requestor then contact the web service with the token which is verified by the web service against the Security Token Service and policies on what the requestor can access is obtained.

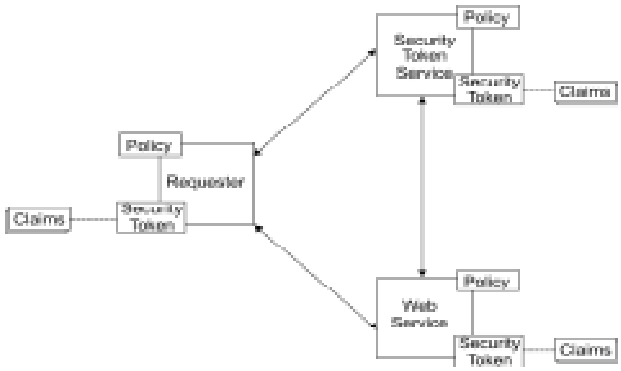

Figure 11:Security token models

This model works perfectly when the requestor and the web service are in the same security domain.

WS-Security uses security tokens and xml encryption standards to maintain the confidentiality, integrity to and authenticate the user. The security token plays a pivotal role in the specifications, makes no demand on what type of token is used. The specifications for the token to be in the form of X.509, Kerberos ticket, mobile device security tokens provided by the SIM cards, username and so on. There are two types of security tokens

- Unsigned security token (e.g. username)
- Signed Security token (X.509 Certificate or Kerberos ticket).

In WS-security other specification builds on top of the WS-security some other specification to provide complete security wrapper for web service messages.

Figure 12:WS-specification

- **WS-Policy:** WS-Policy describes the general framework to exchange policy information between web services. This may relate to sort of encryption standards are supported.
- **WS-Trust:** Describes the process that enables security token interoperability allowing one SOAP actor to request a security token issuing authority to replace one security token issued by another authority with a new one.
- **WS-Privacy:** This specification defines how privacy policies are exchanged between organizations.
- **WS-Secure Conversation:** Defines the security contexts are established, amended and the security tokens are computed and then passed around.
- **WS-Federation:** This specification describes federated trust models are established in collaborating web services using a combination of WS-security, WS-trust, WS-policy and WS-Secure Conversion.
- **WS-Authorization:** This specification provides a mechanism to proved common authorization across organization boundaries. Provide an alternate method of using role based authorization control.

VII.Conclusion

The study of REST web services and REST relay web services as distributed applications including REST clients and addresses the problems faced by transport layer security in REST web services and presents a solution to the problem as message level security.

REST web services only addresses transport security and message security functionality is absent. However, transport security protects the confidential data from point-to-point only. Any intermediary is present between client and service, to protect data either data must be retransmitted on another HTTPS channel or use message security which protects sensitive data from end-to-end.

With message security in REST web services providing an end-to-end security, distributed applications can freely transmit confidential or sensitive data over internet and intranet environment with no fear of man-in-the-middle attack. The message level security provides true end-to-end security in with or without intermediary environment.

References

[1] "Representational State Transfer Document". February 20, 2011.

[2] "SOAP Document". http://en.wikipedia.org/wiki/SAOP, March, 2011.

[3] Alex Rodriquez. "Restful Web Services: The Basics". October 10, 2010.

[4] Amit Asarawala. "Giving SOAP a REST". March 4, 2011.

[5] Austin, D., et al. (eds.). "Web Services Architecture Requirements". W3C Working Draft 14, November 2002.

[6] B. Mohamad, Samer Ei-sawda, I. Hajjeh. "Web Services Security Technology Overview". Computer Engineering and Application, Page no 38-41, 2004.

[7] C. Ferris and D. Langworthy, "Web Services Reliable Messaging Protocol (WS-ReliableMessaging)", March 13, 2003.

[8] C. Irek: IBM Global Services, Realizing a Service Oriented Architecture with .NET, 2005

[9] C. Kaler, B. Lampson, K. Lawrence, A. Nadalin. "Security in a Web Services World: A Proposed Architecture and Roadmap". IBM Corporation and Microsoft Corporation, 09 October 2004.

[10] C. Pautasso, O. Zimermann and F. Leymann. "Restful web services vs "big" web services: making right architectural decision". In www, ACM, page no 805-814, 2008.

[11] D. Eastlake and J. Reagle. "XML Encryption Syntax and Processing". W3C Recommendation 10, December 2003.

[12] D. Wessels, K. Claffy. "Internet Cache Protocol(ICP), version 2". 2006.

[13] Dr. Roy Thomas Fielding. "Architectural Styles and the Design of Network-Based Software Architecture". Department of Computer and Information Science, University of California, Irvine 2000.

[14] Dropbox model. "REST Application Programming Interface". 2012.

[15] Eric Newcomer. "Understanding Web Services:XML, WSDL, SOAP, and UDDI". Addison-Wesley Professional, May 23, 2002.

[16] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. "WSDL 1.1 Document". March 7, 2011.

[17] http://www.w3schools.com, Sep 2008.

[18] J. Hodges, C. Jackson, and A. Barth. "HTTP Strict Transport Security (HSTS)". RFC 6797, IETF, Nov. 2012.