ENHANCEMENT OF FRAMEWORK FOR WEBSERVICES SECURITY IN SOA

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Name – Ruchi Mishra
(Roll No. - 801331024)

Under the supervision of:
Dr. Seema Bawa
Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004
July 2015

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Enhancement of Security framework for Webservices in SOA*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Seema Bawa* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Ruchi

(Ruchi Mishra)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Seema Bawa
13/July/15

(Dr. Seema Bawa)
Professor
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

(Dr. Deepak Garg)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. S. Bhatia)
Dean (academic Affairs)
Thapar University
Patiala

i

Acknowledgement

The successful completion of any task would be incomplete without acknowledging the people who made it possible and whose constant guidance and encouragement secured the success.

With the profound sense of gratitude and heartiest regard, I express my sincere feelings of indebtedness to my guide **Dr. Seema Bawa**, Professor, Computer Science and Engineering Department, Thapar University for her positive attitude, excellent guidance, constant encouragement, keen interest, invaluable cooperation, a generous attitude. She has been a source of inspiration for me.

I am indebted to **Dr. Deepak Garg**, Head of Department, Computer Science and Engineering Department, Thapar University for providing me opportunity to explore industrial projects and motivation and inspiration for the completion of this thesis. I will be failing in my duty if I do not express my gratitude to **Dr. S. S. Bhatia**, Senior Professor and Dean of Academics Affairs at the University, for providing me with all the necessary facilities for the research.

I am very grateful to **Oracle Retail,** to hire me as intern and work for Retail Technology Group (RTG) for a year and providing me opportunity to explore Oracle Retail Integration products and standards. I am thankful to RTG team for giving me live time products to work on, their thought provoking views, veracity and whole hearted cooperation helped me in completing internship program.

I take this opportunity to express gratitude to all of the Department faculty members for their help and support. I also place on record, my sense of gratitude to one and all, who directly or indirectly, have lent their hand in this venture.

Ruchi

Ruchi Mishra

(801331024)

Abstract

Enterprise systems have been evolving from a centralized monolithic application to distributed applications where Service Oriented Architecture is playing a keyrole, and limits dependency on shared resources. Service Oriented Architecture (SOA) frames system based on services, where service can be specific functionality based discrete units of software. Web services technology indeed solution to SOA infrastructure and provides solution for comprehensively representing, discovering and invoking a functional services in a distributed environment like SOA and wide varieties of systems like grid systems. Web Services also forms base of cloud computing. The Web services being self contained modular applications can be located, invoked or consumed independent of platform fulfills requirement of SOA infrastructure based on XML technology, which forms core of webservices like SOAP protocol.

In SOA infrastructure, business functions as exposed to services to achieve interoperability, transparency and extensibility, there is need for security of webservices over common distributed applications. Webservices security is effective mainly in terms of authenticity, confidentiality, non-repudiation and integrity. This analysis discusses possible webservices security standards and review of efforts aimed at web services security in a concise way. Along with that, a security framework is proposed in a manner to cover security of deployed services, communicating parties and communication links to defend webservices threats.

Table of Contents

List of Figures

Chapter1 Introduction

Service Oriented Architecture is archetype that designs infrastructure architecture based on units of specific functionality named services. SOA enables client and server interact via services, and provides a way to integrate services from different providers independent of underlying technology and platform and modularize complex systems. SOA promotes abstraction, loose coupled services, easily discover ability of services and reusability [17]. SOA architecture can be implemented using webservices. SOA can be said a strategy for developing business-oriented software systems, combined with interoperable and reusable building blocks named services. A service can be defined as a modular, independent, measurable, functional unit that can be managed, versioned and discovered as packaged reusable software code that provides specific business oriented functionality either to other services and external enterprise applications [2]. Services can be said to have few logical components: Contract (a description of what functionalities and constraints), Interface (application which can invoke service), Implementation (Configuration and deployment data of infrastructure) [32].

1.1 Introduction to Web Services

A web service can be said as software system identified by a Uniform Resource Information, whose interfaces and bindings are defined and described using XML. Web services refer a distributed environment where providers can expose their business functions in form of services to be consumed by subscribers in an integrated environment over internet protocols [12]. A webservice is like a web API that allows integration of applications using XMLs over web infrastructure and protocols. Webservices architecture for B2B transactions need applications to dynamically bind and find distributed components. All these business transactions can possibly be quite sensitive in nature or high dollar transactions that requires comprehensive secure infrastructure [4]. Any secure infrastructure employed must address issues like if for most parts of transactions can occur over secure internet and trust could be made between participants that might have no previous history.

1.2 Web Services technologies

Web Services can be implemented in any language and can be accessed in any platform as it is based on standards such as SOAP, UDDI, WSDL, WSIF and REST, communicating via XML messages [12]. XML is a self describing way of representing data, independent of protocol, application, vocabulary, programming language and operating system and considered a business language for portable data transfer. Describing rules of XML instance is XML schema. Simple Object Access Protocol (SOAP) [38] is an extensible mechanism and provides a way to transport common message format (in XML) to be exchanged between service provider and service consumer, using various transport technologies like HTTP, SMTP, etc. REST protocol uses HTTP transport and supports other data formats like XML, JSON etc. Representational State Transfer (REST) is lightweight and flexible protocol but not much used. Universal Description, Discovery and Integration (UDDI) is a service registry, registering WSDL where both service provider and service consumer access the registered services [7]. Web Service Invocation Framework (WSIF) provides a architecture for designing and defining interfaces as URI and enabling dynamic or stubless invocation of webservices. Web service description language (WSDL) is common service specification format that service providers use for service detailing like service type, input output operations, operation parameters and service access points etc.[10].

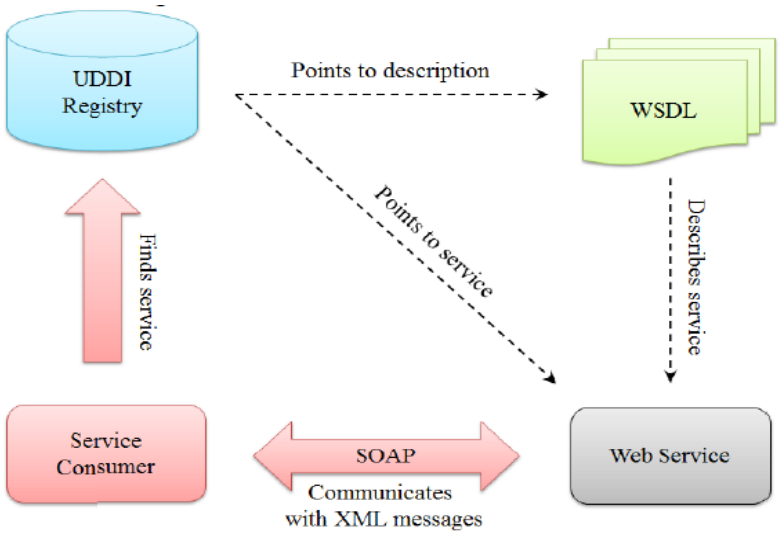


Figure 1: webservices architecture [15]

1.2.1 Simple Object Access Protocol (SOAP)

SOAP protocol aims at application integration and allows publisher and consumer to communicate data based on common data transfer protocol. The basic structure of SOAP message has SOAP envelope which acts as a container to place XML message. SOAP envelope is further divided as SOAP body and SOAP header, where header contains details of publisher and consumer related to authorization, authentication and security and body has message payload having business data. SOAP body can be based on different styles like RPC-style or Document style, where document-style is preferred because of better performance [35].

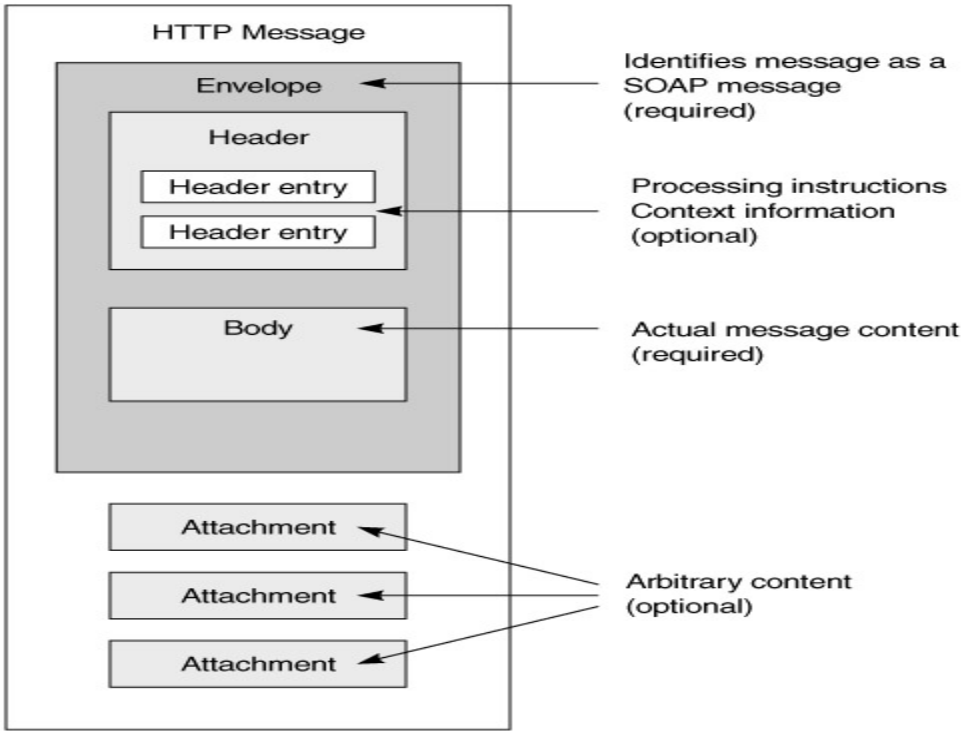


Figure 2: A SOAP message

1.2.2 Web service description language (WSDL)

WSDL is a language to describe set of operations and structure of SOAP message and rules to access and wok on it. It defines what input is expected and will be the output after performing operations. WSDL describes how service interacts with other service, where service is residing, what service meant to do, and how to invoke service. When WSDL is published for a service, a contract is created that how other services will be

interacting to utilize the service. WSDL composes what section for specifying input and output messages, how section for specifying rules for packaging message in SOAP envelope, and transferring same, and where section specifies endpoint of service and implementation rules [19].

WSDL contains service, interface, access protocol, implementation details, and contact endpoint information related to an operation. WSDL document describes webservice using elements: portType (the operations to perform and messages involved), and message (data elements involved in operation and messages description), service (name and location of service, and access context), types (data types used by operation), binding (describes communication protocol used by operation for communication between webservice provider and consumer). WSDL can be mapped and exposed to java method if abstract description is provided in WSDL document. The default style attribute is considered document style, if not declared whether rpc-style or document style.

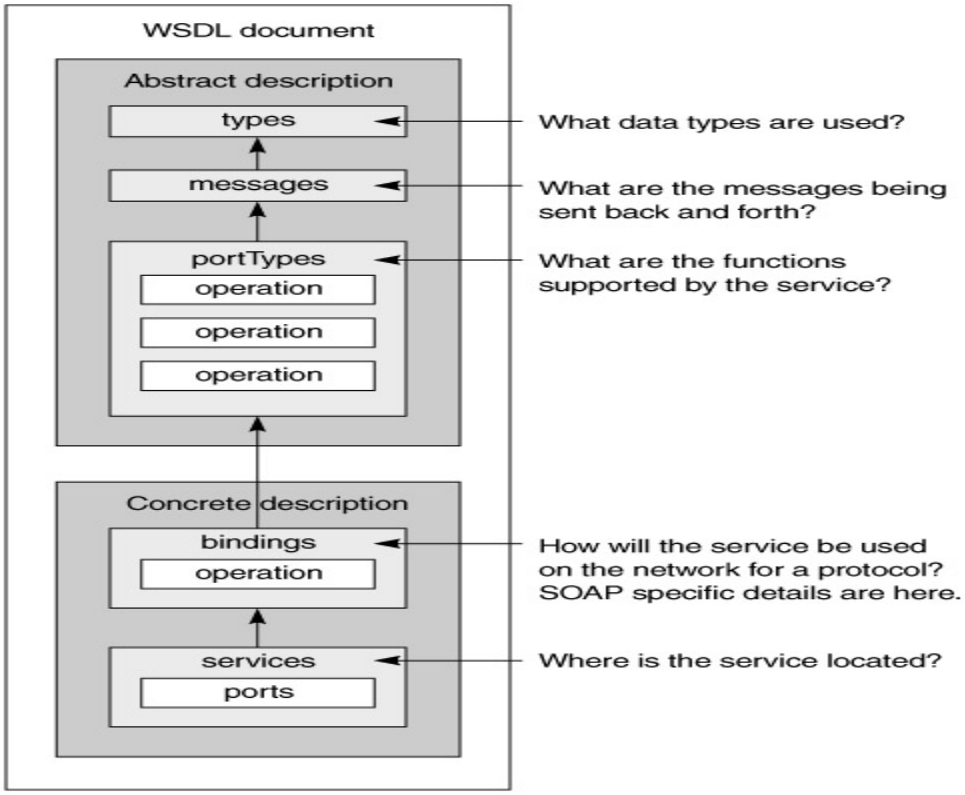


Figure 3: WSDL Structure

1.2.3 Universal Description, Discovery and Integration (UDDI)

UDDI acts as webservice registry and meant for discovering registered businesses and web services. UDDI registers webservice metadata, and pointer to describe WSDL of service. It supports inquiring APIs' for publishing and querying a webservice. UDDI offers a platform-independent framework for discovering web services, describing webservices, and integrating webservices.

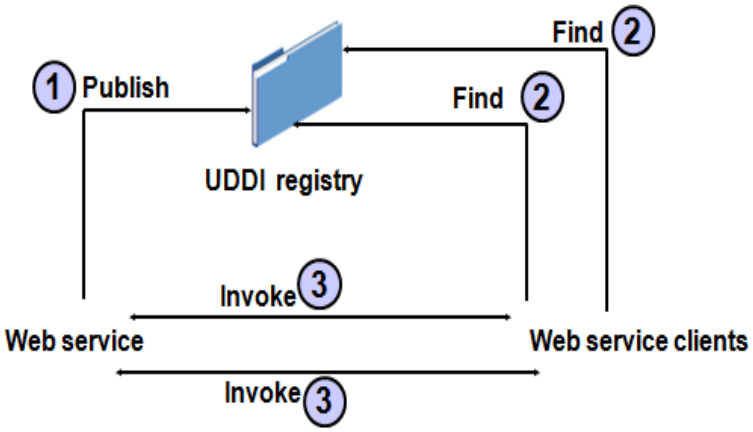


Figure 4: UDDI in webservice

1.3 Web Services Security

SOA offers various plus points to implement business service like flexibility, scalability, reduced cost and operation, reusability, interoperability and platform independence [1]. But along with plus points, there are few minus points like high data transfer using XML, speed and security. Webservice security is one of the major concerns while using SOA platform for business implementation. Web services expose valuable and critical XML encoded business data and information, thus webservices application to application based infrastructure need as much security. Web services standards can be trapped to various unsecured attacks like WSDL scanning and tampering, XML threats, SOAP threats etc [4]. Web services also lie as foundation of cloud, inter organizational and cross domain implementation of business logic. Web service security includes aspects as per its nature. W3C and OASIS has also standardized stack for web service stack [1].

There are number of security standards defined to overcome security breaches, and security threats. Web Services security differs from protocol to transport used, thus its needed to protect communication link, communicated message, securing provider, consumers, and intermediates involved. To achieve required level of confidence, Business Systems demand for access control of services, non-repudiation guarantees message integrity, confidentiality along with transport level security. WS-security provides technology stack to deal security concerns of protecting SOAP based web services [3].

1.3.1 Security Concepts

There are few words which are common to understand security concepts. Cryptography is science of keeping message secrecy, and guarantee trust by binding the identity across message that can be interpreted and seen, and keeps message confidential., Authorization is allowing what that identity can access and do. Integrity is receiving message as it is intended to receive without any changes and is what published.

1.3.1.1 Cryptography

Cryptography [12] aims to develop and design and applying algorithms to secure message, protocols, systems, and applications. Encryption is further area of cryptography, is basis for security standards like XML encryption and Hash function is an cryptographic algorithm to digest a digital signature to form XML signature. A plaintext is readable and unscrambled message, and thus an unencrypted data. Applying encryption on plaintext scrambles or disguise a plaintext using a digest algorithm makes it ciphertext, an encrypted data. Further decryption reveres the encryption and decrypts ciphertext into readable form, plaintext. Encryption goals to achieve Confidentiality so that encrypted data cannot be deciphered by unintended receiver [10].

Figure 5: Encryption and decryption process

a. Key

Encryption and decryption utilizes a special numeric value named key as a parameter in algorithm on which all task is based. If wrong key used, it will not give correct output, will get garbage out. Cryptography fundamental principle is to keep algorithms standard, public, distributed, and carefully scrutinized, to ensure cryptographers' can shake out algorithms in security flaws. Variable that makes algorithm output secret and unique is key. Key can be a random number or chosen complex variable mathematically driven. To prevent guessing keys, and large of possible key combinations a key space, and large key length is needed [12].

b. Shared Key Cryptography

Communicating parties shares same key to encrypt and decrypt data. It is fast and can be operated on any size data. Issue is in managing and to retain secret key across network [12].

c. Public key Cryptography

Communicating parties' shares public and private key pair, where different keys are used to encrypt and decrypt the data. Public key is shared and distributed but private key is never shared. Depending upon situations, its decided how key pairs will used, if public key is used to encrypt the message then private key will be used to decrypt and on other side if message is encrypted using private key to add sender identity into same, then message can be read using public key [12].

1.3.1.2 XML Signature

XML signature acts a foundational technology in WS security standards, and built on top of digital signature. Digital Signature provides a means to ensure message has not changed i.e. integrity and not to refute message change i.e. non-repudiation. Digital signature [18] encoded into XML message is XML signature. XML signature has reference URIs for contents digitally signed, and address resources located by URL. Message can have multiple XML signature and can be applied on whole or part of document whether XML or non-XML content.

XML signature comprises a set of signed references or pointers, actual signature, optional key information to verify signature, and optional object tag. XML signature types can be Enveloping Signature where reference is to signed element, Enveloped Signature where parent element is referenced and detached signature where reference is to any internal or external element [10]. XML Signature uses asymmetric key encryption, using public private key pairs where publisher signs message using its private key and any consumer having access to public key can verify whether sent by authorized user and integrity of message. XML Signature process can be: a publisher first hash the message and form digest message, then encrypt the digest message using private key that generated value is called digital signature, which is embedded into XML message and sent on link. At consumer side, message validation message is done where received message is hashed at one end and signature is decrypted with public key, if the hashed messaged and decrypted signature matches, message is subscribed [12].

1.3.1.3 XML Encryption

XML Encryption and XML signature lays off on same technologies but addresses different issues. XML Encryption maintains secrecy of message by allowing hiding whole message or part of message referring either internal document or external document. Complementary to XML Signature, XML Encryption can exist multiple times, can have same key or different keys in a document. XML Encryption Structure mainly documents [13] EncryptedData which is core data element that wraps encrypted data into XML, or references to external encrypted data, reference tag to point to external non-xml data, keyinfo to include key information, EncryptedType-abstract information of encrypteddata and key, EncryptedMethod- to specify information of encryption algorithm used, CipherData, EncryptionProperties - to define optional other information related to XML encryption, AgreementMethod – key agreement protocol used for encryption key generation. XML Encryption process involves related but different process encryption at sender side and decryption at receiver side.

Encryption process can have these steps:

- Choose Encryption Algorithm: Sender and Receiver need to negotiate over a supported algorithm like AES, or 3DES [10], which will be represented under

EncryptionMethod element in XML Encryption structure.

- Generate Encryption Key and Representation: Using either shared key encryption or public key encryption, key pair is generated. Key details are represented in KeyInfo element in ways like including key as pointer to key, or encrypting key using public key or pointing to key agreement protocol.
- Serialize Message Data: XML data is converted to octets as encryption algorithm is applied on stream of bytes.
- Encrypt data: Generated key, algorithm, details of encrypted data type and data structure are finally implied on XML data to encrypt data.

Decryption process can be summarized as:

- Get Algorithm, keyInfo and parameters and locate key: All required details like encryption algorithm, related parameters like encrypted method, key info elements are collected.
- Decrypt Data and Process XML and non-XML data: Cipherdata can be either encrypted using inline CipherValue or CipherReference, and accordingly decryption process is applied. For ciphervalue base-64 is decoded into bytes, and for cipherreference URI reference is dereferenced.

Common XML Encryption types are Symmetric or secret Key Algorithm and Asymmetric Key Algorithm. In secret key encryption, same shared key is used at both publisher and consumer side, distributed at both ends. A secure channel is needed for distribution of keys [11] [19].

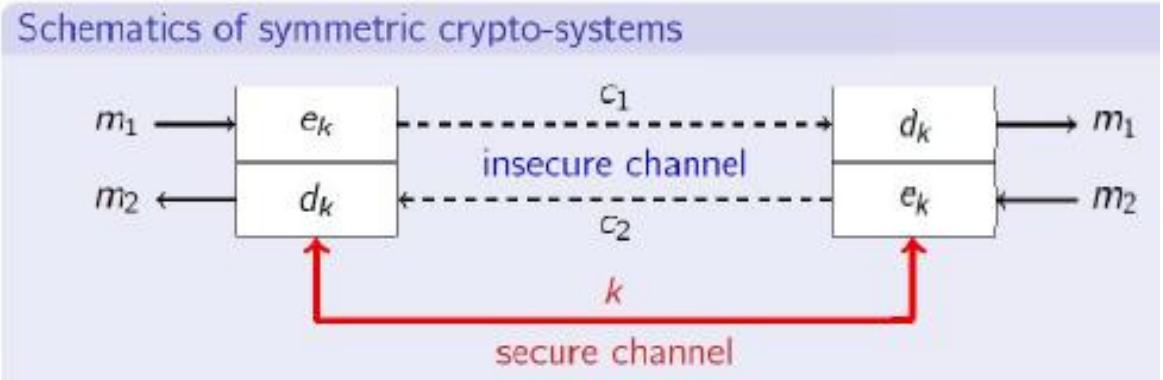


Figure 6: Symmetric Key Encryption

Asymmetric Encryption employs two keys, named private key and public key where both are related mathematically. Public key is accessible to all users, where private key is only for intended user. As shown in figure, public key p is used by server to encrypt message where as secret (private key) s used to decrypt [11] [19].

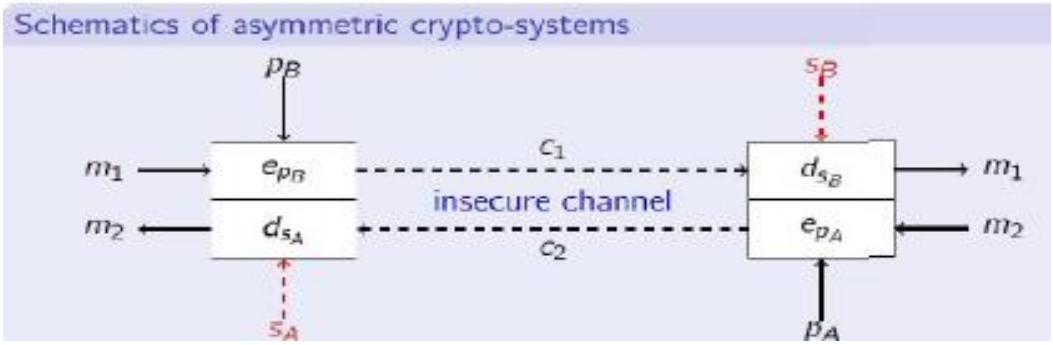


Figure 7: Asymmetric Key Encryption

1.4 Thesis Outline

This thesis has been divided into 6 chapters.

- Chapter 1 includes the introduction to Service Oriented Architecture and Webservices. It also covers basic webservices security techniques.
- Chapter 2 describes the different Web services security standards and frameworks available by organizations working for webservices. Also chapter explains research globally and presented view of different authors.
- Chapter 3 presents the problem statement, objectives and how proposed solution addresses problem.
- Chapter 4 provides overview of security framework proposed '**Certificate based Message Protection and Username token over SSL'.** Proposed security framework deployment diagram is illustrated along with a glance of Oracle Retail Service Bus.
- Chapter 5 describes proposed security framework deployment and creation of integrated SOA environment where framework will be employed. Finally, results of the framework have been discussed.
- Chapter 6 gives the conclusion and the future scope of the work done in the thesis.

Chapter2 Review of State of Art in Webservices Security

2.1 Web Services Security Standards

Webservices based on SOAP protocol lays ground for SOA infrastructure, thus with increased adoption of SOAP in developing distributed applications, Webservices security is one of main consideration. The World Wide Web consortium (W3C) defines Web Services as following [3]: a Web Service is a software system identified by a Uniform Resource Identifier (URI), whose public interfaces and bindings are defined and described using Extensible Markup Language (XML). These systems may then interact with the Web Service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols. In particular, a service provider uses Web Service Description Language (WSDL) to describe the functionality which a service offers and publish the description in Universal Description Discovery and Integration (UDDI). On the other hand, a service requester discovers the service in UDDI and consumes it by sending Simple Object Access Protocol (SOAP) messages. SOAP based security consideration is always a buzz word as many critical software applications e-commerce, integration applications, cloud computing are based on service environments. To abide by that, SOAP security foundation has many standardized frameworks, protocols stack and polices.

W3C, Web Services Interoperability Organization (WS-I), OASIS, JCP organizations have developed guidelines, specifications and tools for webservices security. [1] W3C Specifications [2] are related to XML Encryption, XKMS, and XML Digital Signature. OASIS Specifications [5] are related to development, adoption and convergence of webservices security standards like SOAP message security, SAML, and XACML. JSR Java technical security is specified by JCP. WS-I [3] specifies protocols for interoperability of messages like guidance of using X.509 security tokens using basic security profile, SAML token profile, threats challenges, and countermeasures for web service security.

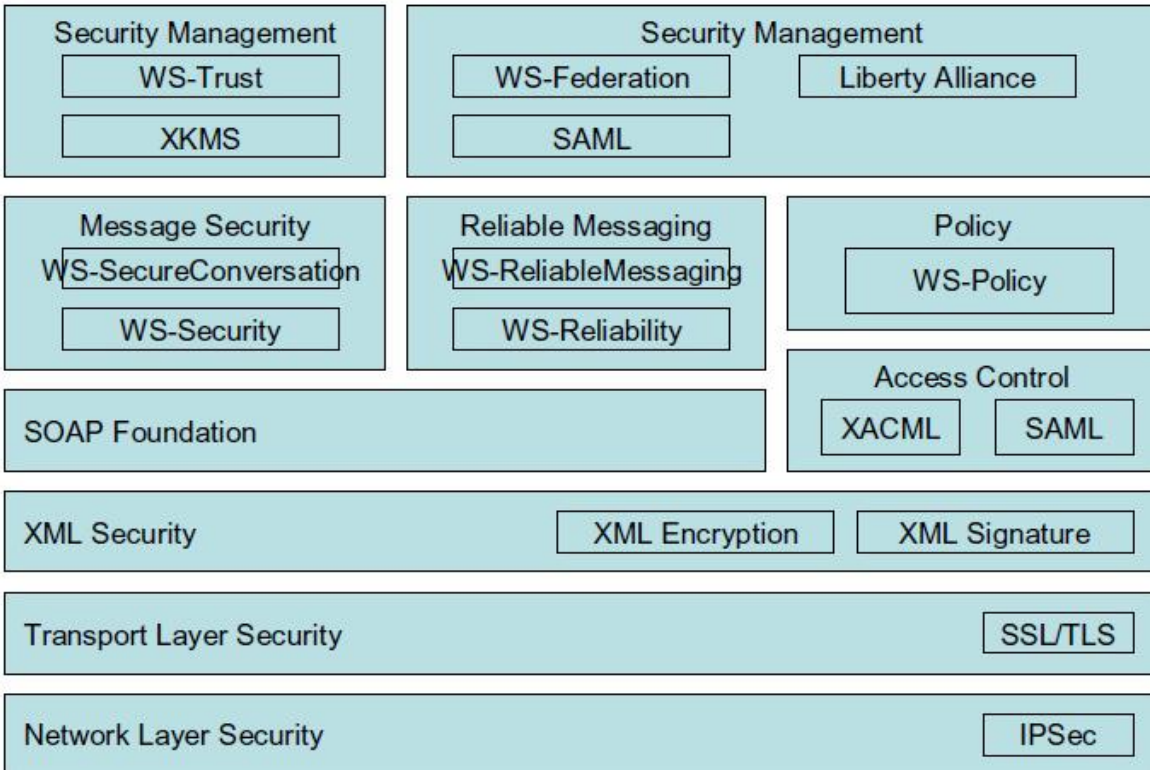


Figure 8: Web service security Stack [1]

2.1.1 Transport Layer Security

It is a point to point transport security mechanism where communication link is secured between service provider and consumer secured by Https using Secure Socket Layer Protocol, thus providing confidentiality, authenticity and integrity. In transport layer security, provider and consumer run on SSL enabled protected session, where service provider and service consumer authenticate each other and communication takes over encrypted network [8]. For a transmission to begin, a cryptographic key and encryption algorithm is chosen and all communication takes place over secured session. Digital Certificates are cryptographically signed that act as digital driver license to authenticate and assure self identity over internet SSL protocol implements public key cryptography based on key pairs, where value encrypted using server's private key called digital certificate, and same can be decrypted at client end using server's public key, authenticating the sender [27]. Running on SSL, client and server authenticates each other using certificates and before transmission or receiving data, cryptographic keys and

encryption algorithm are decided between communicating parties. Transport level security mechanism secures transport channel but not data, if intermediaries or proxies are involved environment is vulnerable to security threats.

2.1.2 XML Security

SOAP based webservices communication is based entirely on XML which lays ground for platform independence and can be language for protocols like JMS, Http, SMTP. XML Encryption is transcripting data into an encoded form that cannot be read be recipient not intended to receive. Secured encrypted data can be decrypted with knowledge of key, either encrypted fully or partially. It strives to achieve data confidentiality. XML Signature [8] ensures data integrity and authenticity. Digital Signature as sender's identity is bound with SOAP message to insure message integrity that it's not altered while in transmit. WS Security has provided specifications on how to structure SOAP Message with Digital Signature and encryption.

Digital signing and encryption are based on private public keys. XML Encryption can be achieved using either symmetric key encryption (secret key) or asymmetric key encryption (public key). Symmetric key encryption, publisher encrypts message using a key and consumer decrypts encrypted message using same key. Asymmetric Key encryption, message encryption and decryption is done using public private key pairs. Digital Signature as sender's identity is embedded as XML tags bound in SOAP message. In digital signature process, first SOAP message is hashed called digest, then outcome is encrypted with private key of sender and added to original SOAP messaged and sent over link. At receiver end, encountered message is hashed and digital signature is decrypted with public key. If both the values match, message is validated to be received from authorized user and ensured for integrity [12] [18].

2.1.3 Access Control

Access Control Policy Specification ensures who has what access to webservices. It is needed to restrict access to webservice even to authorized users to ensure authenticity, confidentiality and integrity of messages exchanges in distributed environment [27]. Access Control specifies security roles and privileges assigned to users and their groups.

OASIS defined XACML (XML Access Control Markup Language) standard as a policy language for access control decision and defined standards for defining new data types, combining logic or new functions [5]. Access control action responses called policy frames a query whether access is allowed based on output named as permit, intermediate, deny, and not applicable [40]. XACML revolves around subject, action and object as language oriented. XACML express access control policies element wise and action primitive as create, read, write delete, where elements can be single or group of elements specified as XPATH expression referred in XML document [32].

SAML (Security Assertions Markup Language) is OASIS specification for user authentication, attribute and based on six components: assertions (validate and authenticate), protocols (structure and contents for XML schema), bindings (mapping with communication protocols), profiles(constraints and extensions) , metadata (configuration information of entities). Through SAML, entities create assertions for authorization, access control, identity subjected to either enterprise application, or partner application or other third parties. Created assertions are presented as XML documents sent either by asserting party to relying party or relying part pulls from asserting party [32][39].

2.1.4 Policy

WS-Policy is a framework to express webservices properties. It is an extensible grammar based building block for defining general characteristics, requirements and capabilities of webservices entities in form of policies. WS-Policy provides mechanisms for defining and associating policies for webservices resources like WSDL artifacts, UDDI elements to include encryption rules, security tokens and privacy rules. WS-policy specifies policy expression that contains policy information called policy expression as an XML structure [41].

2.1.5 Reliable Messaging

For an uninterrupted communication, it is needed to have a reliable network that guarantees successful message delivery between webservices provider and consumer. WS reliable messaging is an OASIS specification that defines a framework and message

protocol to track, identify, and manage reliability of message delivery for application hosts communicating over reliable messaging source and destination.[csp doc] WS-RM specifies interoperable protocol where a message from provider to consumer is invoked reliably either by a delivery assurance(in order, exactly once, at most once or at least once) or with a error [42].

2.1.6 Message Level Security

Transport Layer Security does not ensure end to end security, i.e. message is prone to security threats. To overcome this WS-Security specifies security principals on SOAP message. Application level or message level secures by encrypting the transmitted message, thus providing message confidentiality and integrity [27]. Message Confidentiality can be achieved by encrypting message at service provider end and decrypting SOAP message on service consumer end.

Message Integrity to make sure while transition message is unaltered, and can be achieved by digitally signing the message. WS-Security provides security token profiles : Kerberos ticket, REL (rights markup) document, X.509 certificate, Security Assertion Markup Language (SAML) assertion [40].

2.1.7 Security Management

For secure messaging, basic mechanism are defined by security management standards. WS-trust [46] specifies protocols for managing, issuing and cancelling security tokens i.e username tokens and SAML tokens, based on Security token service (STS). STS [43] is a software responsible for security token issue and exchange to enable issue and distribution of credentials token on shared webservices trust domains. WS-Federation [45] specifies management of security token exchange and trust relationships on webservices domain. XKMS a W3C standard is a trust service that provides an interface between PKI and XML application to register, retrieve and validate public keys in order to simplify enterprise deployment and transferring complex client tasks to trust service [44].

2.2 Web Services Security Frameworks

2.2.1 Entrust Webservices framework

Entrust provides Web services Trust Framework [32] to describe general component of webservices architecture to distinguish between Trust Services, Trust Context, and trust messaging. To secure a communication transaction between service provider and service consumer is what trust messaging. Supported mechanisms and processes to achieve trust messaging is trust context and trust services. As shown in figure, the provider and consumer interact via exchange of signed and encrypted trust messages, accompanied by further trust information to establish trust context entitlements of participants and identity.

Additionally, either the consumer or the provider or both the communicating parties need services of Trust Service, and they needs to search in the broker registry, and bind like any other webservice. Entrust Web Services Trust addresses fundamental security issues along with core requirements to deal with authentication, authorization, support for nonrepudiation, confidentiality, and integrity, through products and technology that extensible and standards-based, that will benefit customers' to easily integrate applications into a trusted, open and flexible infrastructure.

Trust framework also accompanies webservices security standards. As a common practice by default its considered that message will be XML, contained in SOAP envelope, and communicating over HTTP at transport layer. Fundamental challenge is to ensure base XML message is trusted, where trusted means the origin sender or publisher is authenticated, XML message confidentially is assured, message integrity is determined, and finally communication participants cannot deny sending or receiving messages at later stage i.e nonrepudiation.

Trust messaging standards are XML encryption, signature in SOAP with an addition concept of canonical XML. A basic test of message integrity is to verify message digest, but even small space issue and tag delimiter may mean that message is tempered. In such scenario, even if signature is verified but digests comparison failure results in signature not trusted. To address and deal such issues, Canonical XML [32] is proposed.

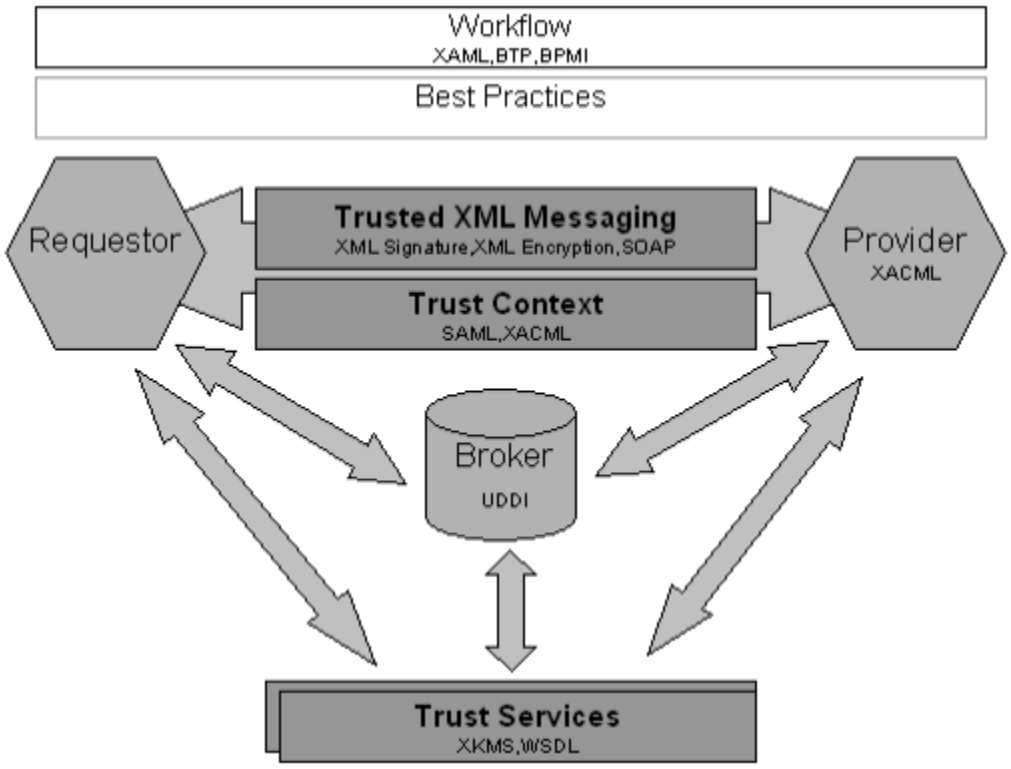


Figure 9: Web services Trust Framework [32]

Canonical form of XML message can be a physical normalized representation to establish baseline standard for signature processing. The message digest is computed during digital signature generation utilizing canonical form of message. The message is communicated to relying party to validate signature by reading message and thereafter computing digest from canonical form of received message. If output from both ends match, i.e. digests equivalence computed by relying and signing parties which means canonical forms equivalence when computed ensures message integrity that information content is what intended and is not altered in transit after signing.

Trust Service can be said a webservice needed to invoke enable trust certificate for transactions. Trust Services are there to deliver the security functions, like encryption, signing, time-stamping, and the accompanying administrative functions, e.g. revocation, key registration, validation, that comprises necessary parameters needed to guarantee the

trust to other parties for secure. This helps to overcome complexities of PKI, and right Trust service can be invoked at appropriate time.

Trust Service is based on XKMS, one of the basic WS standards. A XML based messaging standard to process key related tasks like registration, verification, and key management outsourced to trust webservices. XKMS provides customized XML interfaces integrated with webservices to access PKI functionality residing in remote servers. Trust Context Standards provides additional information to achieve acceptable level of confidence and verification of creditworthiness as a support for trusted business transaction. Trust context is available in form of entitlements static policy provided by trusted 3rd party or negotiated between participants. Trust Context standards are SAML and XACML [32].

2.2.2 Security framework against XML attacks

Mohsenzadeh et al., [19] discussed on securing web service against XML attacks and a proposed a framework using PKI and XML signature along with few other standards. To secure WSDL and SOAP against oversize payload attack or recursive payload attack, proposed solution adds a new XML tag called <XMLSize> in XML file, which is further encrypted by PKI, and if added tag value is received incorrect at receiver end shows message is tampered in transits. The authors also suggest to restrict access of UDDI registry, so that only authenticated users can access same.

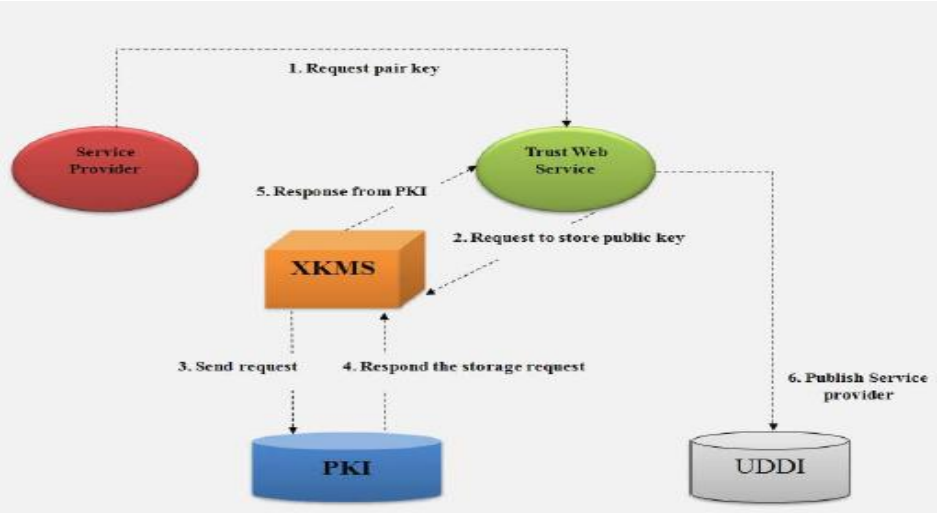


Figure 10: Security Framework against XML attacks [19]

2.2.3 Security framework against WSDL attacks

The core of Webservice lies in XML based SOAP standard, which ensures webservices characteristics – interoperability, transparency, scalability and extensibility, concerning which we have lots of security standards available for SOAP [9]. Bt WSDL describes web service in machine readable form in a XML grammar structure. Shahgholi et al., [4] discussed after web service security against WSDL attacks like parameter tempering, WSDL scanning and proposed a framework for WSDL security. The proposed security framework structure here explores XML Encryption and XKMS standards and has encrypted WSDL and extended it by adding a tag <wssecured>, only authenticated used having authority with Trust Webservice can decrypt WSDL using key combination.

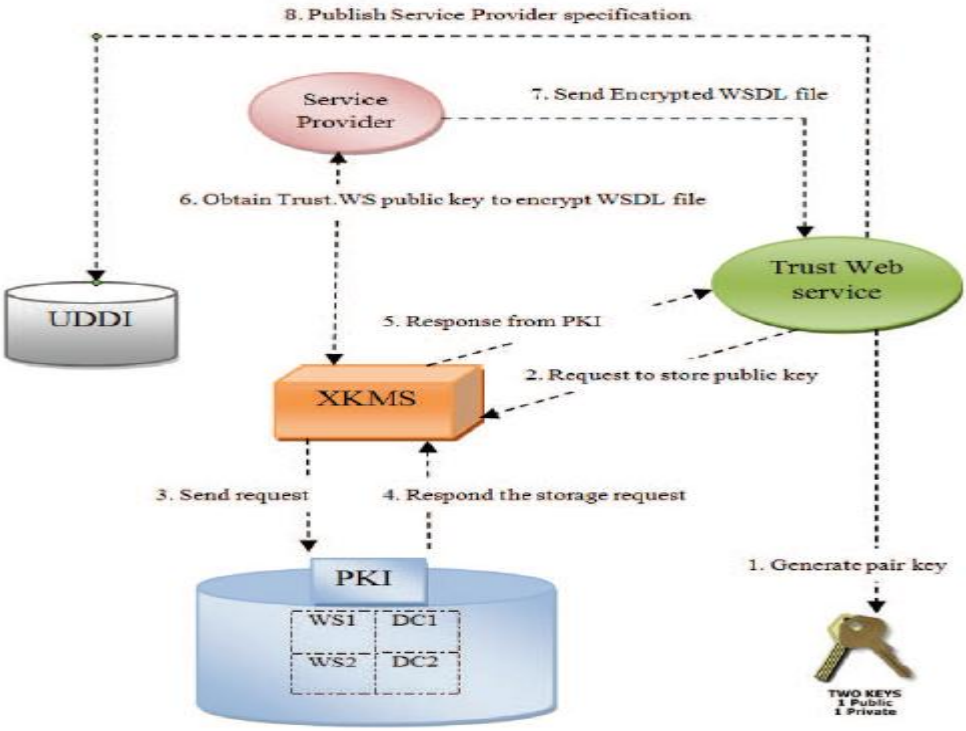


Figure 11: Security Framework against WSDL attacks [4]

2.2.4 Restful Services Stack

Gorski et al., [36] had shown importance and difference between SOAP and RESTful services, and proposes a framework for RESTful services by revisiting security stack available for SOAP based webservices. SOAP based webservices are prone to more external entity attack like XML bomb where URI is attacked to parse malicious XML

content or denial of service attack making parser to go infinite loop or XML injection where malicious content can be injected in XML data or XPath or attack of forged XML signature. While REST based webservices are safe from such attacks as it is not based on XML. RESTful services can be based on JSON, where W3C and JSON Object Signing and Encryption (JOSE) [47] organizations have given JSON security specifications. JSON Web encryption uses cryptographic ciphers to encrypt JSON message using either compact serialization or JSON serialization. Serialization has encrypted metadata specifications in header with only difference that JSON serialization can address multiple recipients while compact serialization to one only. JSON Web signature is applying digital signatures in json message in form of <header> .<payload> .<signature>, where header contains metadata for generating signature, which is validated at recipient end. If message need to be signed by multiple entities, JSON serialization can be combined with JWS. Besides, JSON web signature and JSON Web encryption, JSON Web algorithm and JSON Web key proposed by JOSE. JSON web algorithm specifies cryptographic identifiers and algorithms to be used for web encryption and signature in JSON message.

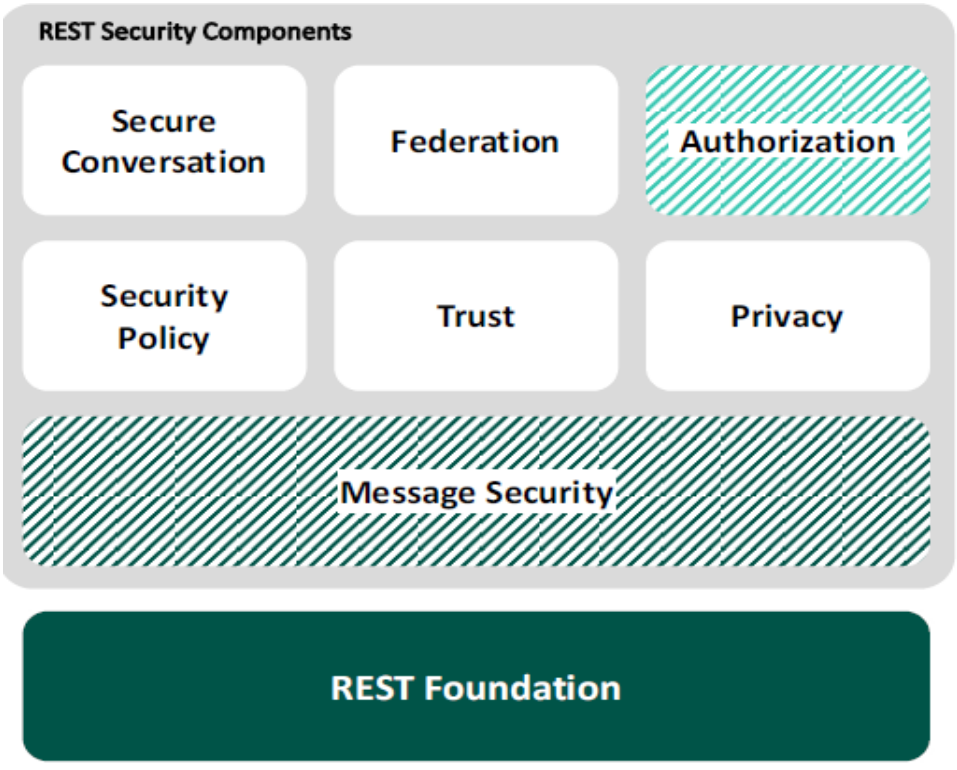


Figure 12: Revisited Security Stack for Restful [36]

2.2.5 Model driven Security Architecture

Bleier et al., [50] proposed a work where a model-driven architecture patterns are applied on webservices security and practically implemented on e-government project. Considering security requirements and internal standards on eGovernmnet business project a model driven architecture utilizes Platform independent Model (PIM) – a model for describing business transactions on which Domain specific language represents business requirements in graphical form and an automated XML tool named Platform specific model to generate XACML polices based configuration file. The proposed architecture wraps polices outside business logic named Policy Decision point, where incoming and outgoing messages to access webservices are passed to Policy enforcement point which decides access control policies to allow or block the requester. Security architecture is implemented on pilot application modeled on BPEL/BPMN standards and risk analysis is done.

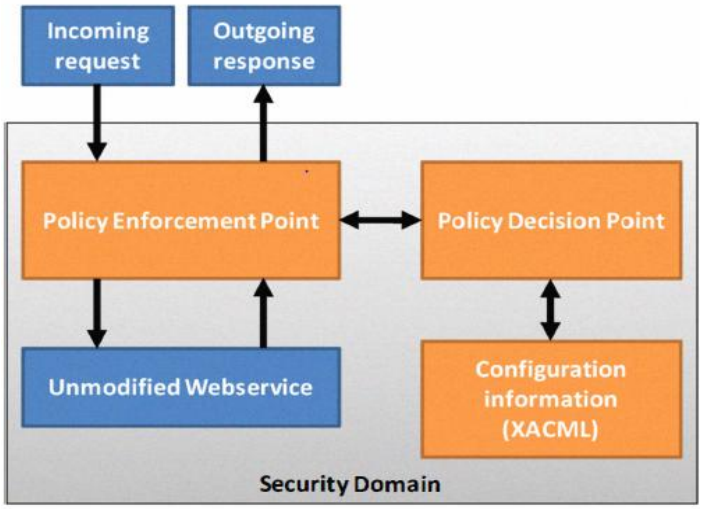


Figure 13: Model driven security architecture [50]

2.2.6 Framework against DoS attacks

Pinzon et al., [34] deals with Denial of service (DoS) attacks in web service environment and presents a multi-agent self-adaptive architecture to deal of DoS attacks. DoS [35] attacks are security threats where legitimate users are prevented to access web services or related information at server. A scenario where attacker floods network with lots of invalid request to target service hosted on a web server until service provider unable to

handle requests and services. DOS attacks can be combined with XML rewriting attacks and can lead messages to be redirected to server different locations or replaying messages, therefore same malicious message will be processed by server, making server to perform redundant work. To deal with it, authors' represents SMAS architecture as alternative to centralized service hosting and proposes a prototype or distributed hierarchical multi-agent for service hosting as a 2-phase classification mechanism.

2.3. Webservices Security Specifications

2.3.1 Specifications against XML rewriting attacks

XML rewriting attacks [15] on SOAP messages is modifying message and adding malicious content in message without invalidating XML signature and bypassing security checks. XML rewriting class of attacks maliciously manipulates, transmits and intercepts SOAP messages. Rahamna et al., [11] proposed approach to add details of SOAP message in form an element named SOAP account in message header that includes number of header elements, number of child elements in message envelope, details of signed object etc with concatenated details of successive signed SOAP node. In [13], authors proposed an inline approach of SOAP account is not enough to handle XML rewriting attacks.

Enhancing SOAP account, an approach named Rewriting healer [14] adds further parameters like successive elements of signed node and depth of information and identities of parents elements to SOAP header. Barhoom, et al., [16] proposed approach to use signed elements positions as a parameter in message header. Referring Document Object Model tree nodes as elements positions, a post order traversal output is used as elements position is added in soap header. At receiver side, if location of signed element is found changed results that SOAP message is modified in transit by attacker.

Smriti et al., [22] derived solution for XML rewriting attack threat after discussing potential scenarios for same. The proposed approach considers root to signed element absolute path using XPATH expressions and uses fastxpath variable of XPath for referencing signature instead of id attributes in SOAP messages. It is also discussed that using fastXpath compared to other parameters of Xpath expressions provide better

performance.

2.3.2 Specification against Fault tolerance

Fang et al., [33] proposed a framework for Fault tolerant in Web service architecture, to protect web service domain environment from soap message related security threats. The framework proposes components to be deployed in environment named Fault Detector for monitoring webservices, Replication Management for replicating elements like user membership management and user group's constitution, then Fault Notifier for notifying any reported issues and finally component for capturing and logging logs for accessing issues and recovery purpose.

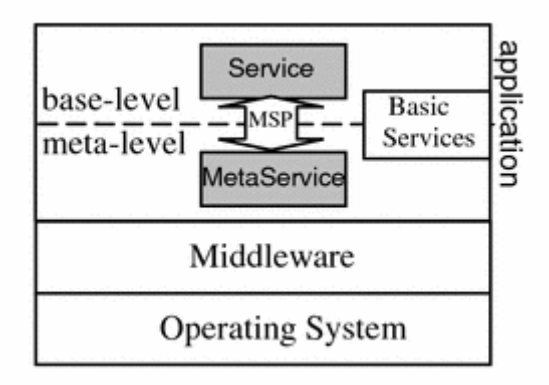


Figure 14: Architecture against Fault tolerance

2.3.3 Specifications for Message Integrity

Somorovsky et al., [20] suggested verification steps for provider to validate incoming clients' SOAP request effectively. The work is done reviewing vulnerabilities of webservice threats on Amazon Elastic Compute Cloud and accordingly security validation mechanism to achieve an effective, robust SOAP message is proposed. Byun et al., [21] the authors proposed a self-adaptive approach to ensure the integrity of SOAP messages.

Blanco et al., [6], the proposed approach first builds SOAP message structure as ontology, and then attaches it in SOAP message's header. Here, if any changes in the pattern of attack are occurred, the proposed method learn them and save in a reliable storage. This can be used later in order to detect new attacks.

Ontology [6] is a way to provide abstract and conceptualize view of a model containing elements, objects, and their relationship. At receiver end, any modifications in signed elements relationship are detected for attack. Along with detection, logging is done for SOAP messages. In security failures, logs can be accessed and root cause can be found for recovery.

Nasridinov et al., [15] proposed an approach against XML Signature wrapping attacks to wrap a SOAP message header with ontological generated SOAP message elements. In proposed approach, ontology applied on SOAP message structure expresses how message elements relate to each other, and accordingly build thesauri and element hierarchies.

Chapter3 Problem Statement

3.1 Gap Analysis

Transport level security is one of the most widely used and easy to configure which encrypts communication. This security configuration is based on transport level security and encrypts communication link and is one of the most widely used and easy to configure. As transport level security involving SSL, offers required performance based on level of security offered, and supported by almost all programming languages. But transport level security is unable to handle security demands in distributed service oriented applications.

Message level security encrypts communicating parties' details and payload transmitted. The main entities of webservices architecture are the adoption of transport level and message level security policies. In this study, a security framework for SOA environment is proposed, whose main objective is securing webservices in order to protect communicated SOAP message and communication link between service provider and service consumer [12] [27]

After analyzing, Transport Level Security and Message Level Security, and various standards present, any of the frameworks does not deal with both communication channel and communication parties. It's been focus on one of the security layer and to comply with security standards of one layer.

In a business specific, webservices based SOA environment like for sites involved in e-commerce or in any other business transaction which are built up on SOA architecture using SOAP based webservices as communication protocol, its very important to take place transaction in secure manner, where authentication of identity of both service consumer and service provider is considered, then is important to assure message integrity (that transaction remains unaltered and not tempered during transit by having an authority digitally sign that message) , message privacy and confidentiality (transaction is kept secret as webservices request or response data could contain confidential, sensitive

business data or personally identifiable information) and last but not least communication link (identities of all communicating parties should be known and trusted by each other involved in transaction).

3.2 Need for security framework for an integrated webservices based SOA environment

The proposed security framework combines plus points of the security approaches of transport layer, message layer and access control principles and security standards available in these respective security layers and proposes a security framework to enhancement webservices named as **Certificate based Message Protection and Username Token over SSL**, i.e securing communication link, communication parties and communicated message.

Approach ensures end to end security including proxies (like message queue, or router) or any intermediaries involved. Security Policies describe requirements of web service and capabilities like how message can be secured and conforms whether or not message is sent reliably and meet security requirements as expected. Intention of this thesis work is to propose a framework for SOA environment to establish a secure communication channel and to get out of common message transportation threats.

An Integrated environment can be a heterogeneous SOA environment which involves an Application publisher that act as service provider based on webservices architecture, intermediaries that can be proxy service, router or queue that acts as intermediate communication layer between service provider and service consumer, and lastly an Application Subscriber that acts as a service consumer that can built as a java client application or a webservices based application. In such integrated environment, it is very important to have integrated security framework that deals with all communicating parties and communication channel and applies security policies at respective ends to meet end to end security. Security standards are available at different layers that need to be implemented in a real time environment and assure a secured environment to establish communication and exchange of business specific critical data.

3.3 Objectives of Proposed framework

The proposed framework named as “Certificate based Message Protection and Username Token over SSL” to provide enhancement in web security in an integrated SOA environment, specifies that the following security measures that must be applied by web service provider, and client to the SOAP messages for invoking operations:

a. Authentication through Username Token

Requires a client to present an identity that can be compared with user accounts in the domain's authentication provider.

b. Secure Connection through Digital Certificate

To assure identity of server that provider is what intended and to whom negotiated to communicate any sensitive or non-sensitive information. To reflect digital driver license (information related to basic site administrator or owner along with which site belong and its associated company for an internet address), an application presents its associated certificate in form of digital certificate over SSL.

c. Message integrity through Digital Signatures

Establishes the identity of the client that is requesting to invoke an operation and guarantees that no intermediary has altered the request. Also guarantees that the return values of the operation are returned to the client without being altered by an intermediary.

d. Message confidentiality through XML encryption

Encrypts the request and the return value in the response and guarantees that no intermediary has viewed the request or the response.

3.4. Methodologies

To obtain the specified objectives of proposed framework following methodologies have been applied. The proposed security framework specifies below high level abstraction of steps to be configured to achieve an integrated secured webservices environment.

a. All communication parties' enables listen port on SSL/HTTPS for communication to take place over secure connection.
b. A Web services client generates a SOAP header and adds the header to the SOAP message envelope. The header includes digital signatures, and username token. The client encrypts and signs its SOAP message and sends it to the proxy service.
c. When the proxy service processes the secured envelope, it decrypts the message. The proxy service then verifies that the message conforms to its security requirements. For example, the proxy service confirms that the required message parts were signed and encrypted and that the required tokens are present.
d. Then the proxy service routes the message to the business service. The business service again signs and encrypts the SOAP message. Then business service invokes the edge app service.
e. The edge app service decrypts the messages, authenticates the user and invokes the web service operation. After the operation is complete, SOAP response is signed and encrypted and returned to the Business service.
f. The response is then sent back through the business service and the proxy service to the client.

Chapter4 Proposed work "Enhancement of Framework for Webservices Security in SOA"

Proposed work "Enhancement of Framework for Webservices Security in SOA" proposes a framework named 'Certificate based Message Protection and Username Token over SSL'. The proposed framework combines message level and transport level security policies and proposes an integrated security framework to secure message and communication link in order to secure webservices from SOAP threats and WSDL attacks like message tampering, or message scanning to oversize payloads.

To deal with same, Oracle Web Services Manager (WSM) [23] addresses SOA based web services security and management that declares and defines web services policies to attach to SOA infrastructure, enforce security policies to infrastructure using configurable agents, and monitoring and tracking run time security management. OWSM supports number of security standards like WS- security, WS-Policy, WS-ReliableMessaging, WS-Addressing, Encryption Algorithm, JKS and Signature Algorithm.

4.1 Oracle Retail Service Backbone (RSB)

Proposed work explores OWSM policies and security principles to apply on Oracle Retail product -Oracle Retail Service Backbone (RSB). RSB offers service oriented communication as an enterprise infrastructure. RSB is built on top of Oracle Service Bus (OSB) which offers SOA service infrastructure [28]. OSB is Oracle SOA suite product, to manage, mediate, connect legacy systems and heterogeneous services, and offers SOA advantages like service pooling, centralized service provisioning, securing services on oracle applications.

RSB is enterprise integration product having prebuilt integration flows and services for request response style of integration and communication of oracle retail products RSB exposes Oracle Retail Applications as service providers, and edge app services as decorator services that gives virtualized operational view of services and lastly, service consumers are proxy WSDL that invokes SOAP services deployed in RSB.

As a solution of business processing, RSB integration API points to retail applications web services or external enterprise applications [25]. Security policies need to be applied on all components involved in communication, i.e. Service Providers, Decorator Services, and Edge application/Service Consumer. Further, OSB and RSB runs on Oracle Weblogic Server, which provides ease of deployment and high performance execution, availability and scalability environment. OSB acts a communication layer and sandwiched between weblogic and RSB [27].

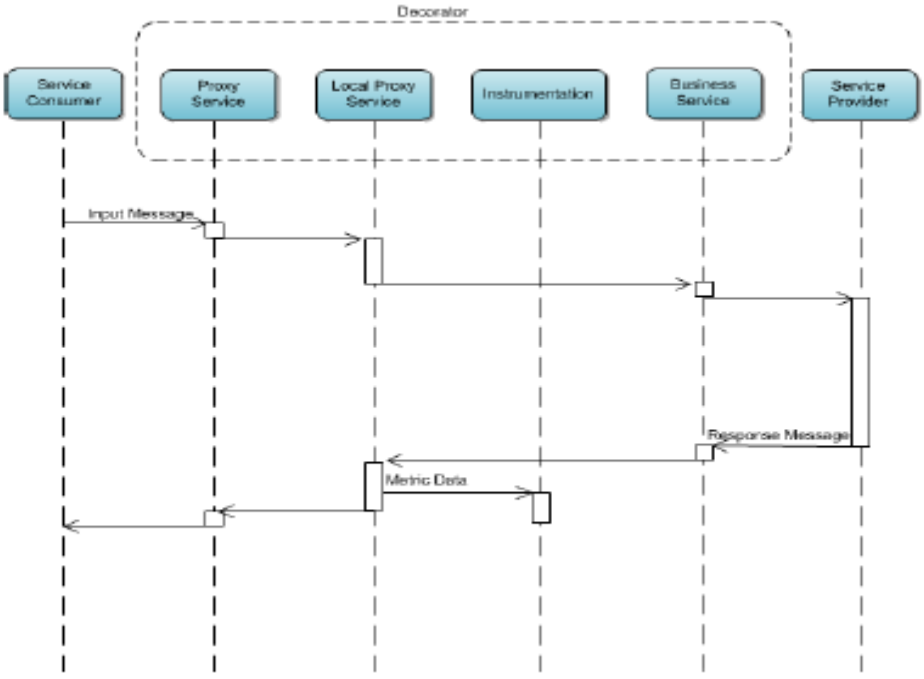


Figure 15: Sequence Diagram for service invocation

4.1.1 RSB and OSB concepts

To implement services on RSB, its needed to explore core concepts of RSB and OSB and basic terms behind its creation [25].

- Proxy Services: An OSB word for intermediary webservice as a layer between application service and service consumer, deployed on weblogic server. Service consumer connects proxy service as an interface of application service.
- Business Service: An OSB word for enterprise information services coupled with actual service provider on which communication done. Service consumers invoke

proxy service which further invokes business services which route finally to service providers. It helps for instrumentation, routing, mediating and transmitting webservice calls.

- Decorators: RSB term for packaged proxy and business service and instrumentation code.
- Payloads: Term for XML message communicated between service provider and service consumer, conforming to schema standards.
- Router/Injector Service: A facility to involve external webservices as service consumers
- Edge application: Actual service provider hosting webservices on application server is termed edge application.

4.1.2 RSB Support Tools

For development, configuration and deployment and testing purpose, support tools needed by RSB like Retail SOA enabler (RSE), Artifact generator (AG), Javaee Service Interface Tester (JSIT), and PLSQL stubby interface Tester (PSIT).

A. Artifact Generator (AG)

A tool to generate and package business objects, which are logical representation of business entity and physically represented in form of XML schemas (XSD) is named artifact generator. Functional artifacts i.e business entities like message payload, message schema varies from technologies.

For JAVA enterprise applications, AG generates jaxb java beans, jaxb [31] beans is java standard for XML binding and converts java objects to xml instances and vice versa. To marshall and unmarshall SOAP messages internally JAXB technology is used in webservices infrastructure. For PLSQL applications, AG generates oracle object artifacts that represent XML structure in form of SQL to store in database [29].

B. Retail SOA Enabler (RSE)

A tool to develop webservices for JAVAEE applications and PLSQL enterprise applications confirming to webservices standards in a consistent way is named RSE tool. RSE generates SOAP based or REST based web service consumer client, service

provider end point and interface templates for PLSQL APIs and JAVAEE APIs. Works with conjunction with AG tool, to produce run time or design time artifacts [37].

RSE features:

- Generation of SOAP based web services in a standard and consistent manner, which are HTTP based. Generated services are complied with JAX-WS specification having WS-Addressing enabled. Generated services are document literal and can be customized to plug web services security and.
- Generation of architectural based RESTful services, complying with JAX-RS specification.
- Generate PLSQL based web service provider, and JAVAEE based webservice consumer and providers.
- Supports Top down service development, using Service Definitation Library having all service operations, input and output against XML Schema.
- For generated web services, WSDL also generated for service level documentation. Deployed webservices as application ear in any JAVAEE application server like glassfish server or weblogic server can be published and registered in UDDI and communicates with UDDI registry with infrastructure management service.

C. SIT tools

RSB being an integration product cannot be tested as standalone, and need stubby to communicate as edge applications. SIT tools [28] mimics' role of edge application.

- Java Service Interface Tester (JSIT): Stubby tool to mimic J2EE application based service provider for validation and testing. RSE generates JAVAEE based service provider jar that can be assembled with JSIT, to expose and access webservices of provider. To JSIT end points, service subscribers can consume services.

- PLSQL Service Interface Tester (PSIT): Stubby tool to mimic PLSQL application based service provider for validation and testing. RSE generated PLSQL service provider ear can be deployed and accessed through PSIT database.

4.2 Security Deployment architecture and illustration

Security policies applied on service providers drive the security architecture, service consumers and decorator services need to adhere according to security policies implied on service provider. In proposed work, service consumer is deployed in basic weblogic domain, where OWSM policies are not employed, need to secure with weblogic policy.

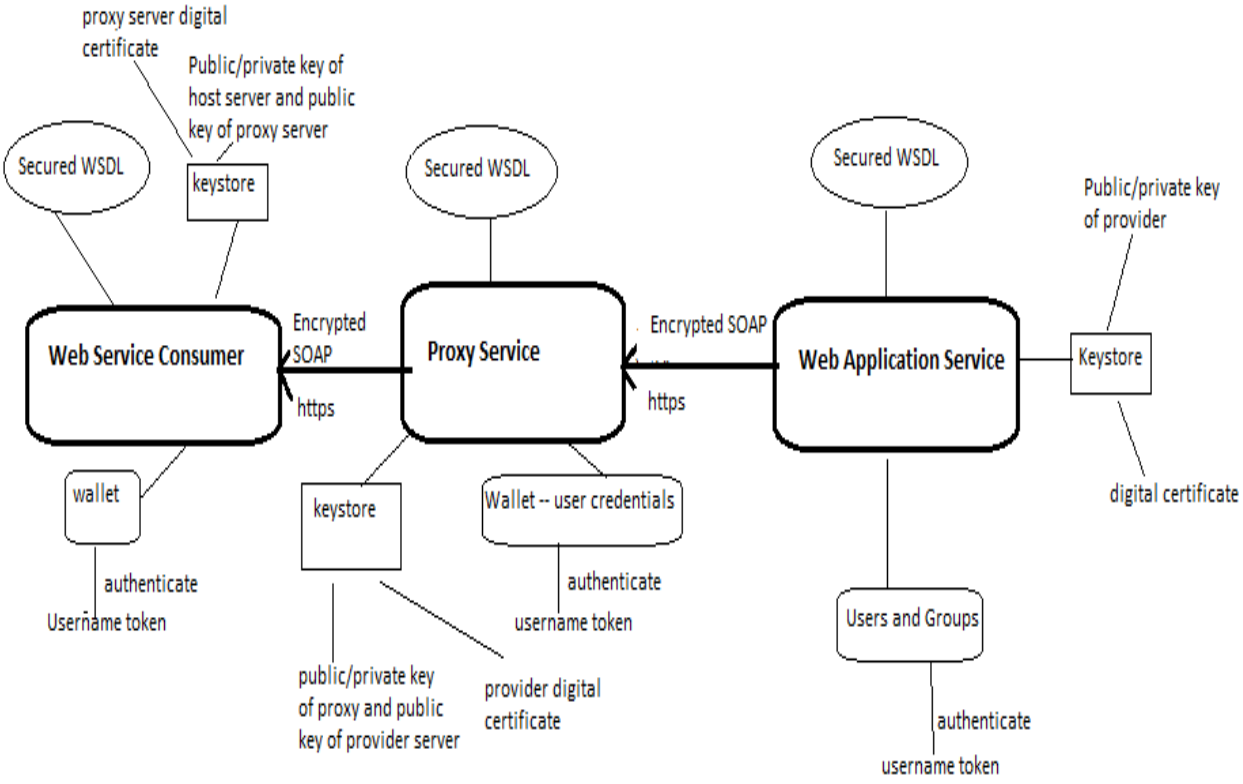


Figure 16: Deployment Architecture

RSB Decorator Services are deployed in OWSM OSB enabled weblogic domain, thus secured with OWSM security policies. For this integrated distributed environment, need to configure applications with weblogic and OWSM inter operable policies, such that handshaking between integrated applications will happen [26].

Chapter5 Implementation and Testing

5.1 Deployment and Implementation of Security framework

As its explored RSB is an SOA based webservices enterprise infrastructure, where architecturally oracle retail applications can be exposed as service providers, and OSB decorators services can virtualize operational visibility wrapping edge app service [48]. Further, Oracle Retail applications or any third party application exposed by Proxy WSDL exposed by decorator services can be visualized as service consumers invoked by SOAP, giving a complete solution for webservice architecture. Using these applications, web service security framework can be applied. RSB key components are platform independent and loosely coupled across physical topologies, technology and platform [24].

In order to apply proposed security framework and ensure end to end security as framework applies, need to secure all communication parties i.e. all intermediary layers and communication channel. So, main layers in our environment setup i.e Service Consumer, Service Provider and Decorator Services, each must be configured securely in order to have secure environment when web services operation is performed and handshaking between communicating parties begin through webservices calls.

Principally, in an integrated environment, security policies decisions based on service provider security policies and security provider driven, to which service consumers and decorators need to comply and adhere. As different communication parties can be deployed in different servers, so it's important to verify compatibility among servers [26].

There can be two ways to apply and process: Active Intermediary and Pass through.

- Active Intermediary is where SOAP message header is processed by proxy service and message level security and access control policy can be enforced on communicated messages. For example, a client signs header and encrypts its SOAP message and communicate it to a proxy server. The proxy service on receiving message decrypts soap message and verifies soap header for digital signature, then on successful verification and validation routes the message to service provider. To

response back, the proxy service on this end signs soap header and encrypts the message. On receiving message, the client decrypts the soap message and verifies authentication of proxy service's digital signature.

- Pass-Through process, proxy server does not validate message received from client and passes it to service provider untouched. Message routing takes place based on values in header. The Service provider on receiving message processes security header and validates signature of client and accordingly acts on the request. In this scenario, business service can use the WS-Policy framework which can help to describe operations which can be secured with message-level security. Again on receiving message response proxy service passes message untouched to client, simply routing response happens on proxy service side [24] [27].

Proposed security architecture, security policies applies on below components:

A. Edge Application Services

Edge app ear file is deployed in plain web logic domain server and does not involve any JRF or middleware components. OWSM security policies cannot be applied here, so weblogic server policies for web services are applied. Edge app services are mimicked by JSIT stubby tool. So JSIT ear packaged with edge app services is deployed in weblogic server.

B. Decorator Services

Decorator services wraps and packages business and proxy webservices. Deploying these services need middleware components like OSB and thus deployed in OSB Sbconsole server. To secure decorator services, OWSM and web logic compatible policies are needed and used for handshake between decorator service and egde app services. The security policy files for proxy and business services depends on security policy applied on edge app services.

C. Client Application

Client Application can be any java client or any retail application. . In same manner like decorator service and edge service, service consumer client of proxy service in turn of service provider needs client security policy matched compatible with edge app security poli cies for OSB proxy service.

5.1.1 Creating web services based SOA environment

RSB provides the toolset and framework for centralized product lifecycle management where centralized location named as rsb-home handles all configuration and management tasks on rsb components employing specific tools and support tools for configurations in all phases and management of RSB product lifecycle. The RSB toolset and framework is referred to RSB Builder tool. Before applying security, its important to verify working of unsecured environment [24].

Webservices security configuration can be performed once services are deployed and domain is in up and running status. Before applying security policies in RSB which acts as integration layer, application services i.e. edge services need to secure. Security policy configuration need to be executed with accuracy as even small mistake while employing security policies can corrupt domain and make system non functional.

On getting security related issues, troubleshooting domain and deployment can become harder, so its recommended to analyze and verify every step of configuration and minimize chances of error. Proposed work has used automated steps of RSB builder tools and manual steps to configure security framework and achieve level of desired security [25].

Diagram on next page represents high level data flow in a typical integrated system. Service Client can be on owsm or any weblogic server, where OSB system is deployed in clustered environment to handle performance and data aggregator issues. Based on incoming request and data, cluster server decided to which manage server request to route, thtis also helps in load balancing. Service provider can be configured to use weblogic.

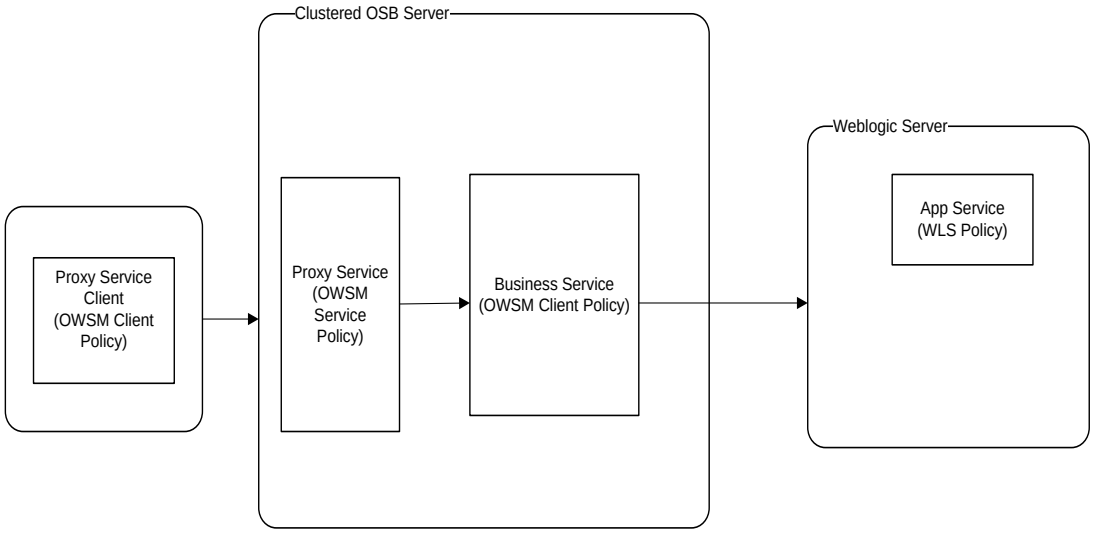


Figure 17: High level representation of flow

5.1.2 Illustration and deployments

5.1.2.1 Pre Installation step

Verified integration environment working in unsecured way where all communication parties are deployed in unsecured, utilizing webservice generated by RSE. Then need to make all deployment server to be SSL enabled and setting up domain to run over Https and setting up default variables for all servers.

5.1.2.2 Setting up App service

App services should be deployed in a base weblogic server, it should not have FMW components installed. The app services are secured using WLS policies. Below steps followed for configuring app services with message protection:

a. Generate service provider web services for app like customerservice using RSE and wrap generated webservices to JSIT to mimic service provider and act as edge service in weblogic server. While deploying services, make sure to select “Custom Roles and Policies” in the Security Model page. This is required to be able to add security policy to the web service.

b. Service Provider needs to generate Username token for authentication. Username

Token will be stored in security realm users and groups of application server. At other subsequent steps and at client side same username token will be used.

c. After deploying the ear file, Explore weblogic server, go to webservice > Configuration > ws-policy page of the web service for webservices to secure.

d. Following policies attached to the app service like customerservice.

    i. Policy: Wssp1.2-2007-Https-UsernameToken-Plain.xml

    ii. policy:Wssp1.2-2007-Wss1.1-UsernameToken-Plain-EncryptedKey-Basic128.xml

    iii. policy:Wssp1.2-2007-EncryptBody.xml

    iv. policy:Wssp1.2-2007-SignBody.xml

e. In security realm of the domain, user for username token authentication, and user attached o the web service.

f. PKI certificates generated in config path of weblogic server where app service is deployed.

g. Domain environment is set, by running setDomainEnv.sh file, this will export environment variables and set profiles like java home, oracle home

h. Generates certificate files and a keystore with public/private key. The generated file names will be as like following:

    <hostname>-certificate.der,<hostname>-certificate.pem, <hostname>-keystore.jks, <hostname>-public-private-key.der, and <hostname>-public-private-key.pem

    For remembrance purpose, name is kept server/host/machine name.

i. As shown in above file listing, .der and .pem files contain server certificate. The .jks file is a java keystore which contains a public-private key pair for the server.

j. Following are the details of the generated keystore:

i. Alias name in keystore: <hostname>-public-private-key-alias

ii. Keystore password: <hostname>-public-private-key

iii. Password for the alias <hostname>-public-private-key-alias: <hostname>-public-private-key

k. App services are configured to utilize generated keystores and certificates from PKI store as per app server domain details. It configures the weblogic server to use the new keystore for encryption and signing of request and response xmls of web services.

l. Generated Key pairs and username token will be applied on all or few parts of webservices to secure them. Keypairs and Username Token can be applied at application ejb level.WS policies will be applied at service level for proposed work, OWSM and weblogic server policies are used.

   This completes securing service provider webservices where keystore stores public private key pair, WSDL is protected with weblogic policies to employ digital signature. Hosted service provider servers will be listening on SSL enabled port accessed with username token

m. Configured certificates and keystores can be verified in weblogic console server After running the above script, go to weblogic console and navigate to domain_name > Web Service Security > default_wss > default_x509_cp page

n. Admin server and managed servers are restarted to take the effect. App services WSDL are verified to include the attached policies.

Access secured WSDL and verify all the polices are applied on WSDLs'.

5.1.2.3 Setting up RSB/OSB Domain for Security Framework

RSB decorator services use OWSM policies for message protection. To use OWSM policies, it's needed to install OWSM components into the RSB/OSB domain. OSB

domain need RCU schemas, which is used by OWSM to manage the security policies. Repository Creation Utility (RCU) is CLI based graphical tool to create and manage middleware db schemas and is part of Oracle Fusion middleware system. RCU provides mechanism to create db custom schemas and their table spaces, and also flexibility to manage schemas like change tablespace allocations and rename schemas or distribute table space among single or multiple components. RSB domain is created pointing to RCU MDS schemas and in a clustered environment, where manage servers are clustered to act cluster as proxy server. To access OSB weblogic sbconsole is available, and to access RSB weblogic console is available [49].

RSB-home process for security framework

a. RSB Home is configured for decorator paks over RSB/OSB domain created using RSB builder tool. Edge App service WSDL are downloaded and saved in RSB home. Properties file to map web service with service provider policy is created, properties file also contains policy keyname for edge app.

b. At RSB/OSB Domain location need to generate certificates and keystores' for RSB using scripts. This will generate a self-signed certificate with common name hostname and key strength 1024. Few default parameters are there, they can be changed as per security configuration: Disabling CryptoJ JCE Provider self-integrity check for better startup performance. To enable this check, specify Dweblogic. security. Allow CryptoJDefault JCE Verification =true, Changing the default Random Number Generator in RSA CryptoJ from ECDRBG to FIPS186PRNG. To disable this change, specify Dweblogic.security.allowCryptoJDefaultPRNG=true. Following files are generated as part of keystores and certificates: <hostname-certificate.der>, <hostname-certificate.pem>, <hostname-keystore.jks>, <hostname-public-private-key.der>, <hostname-public-private-key.pem>

c. Next is to import private key <hostname>-public-private-key.der and certificate <hostname>-certificate.der into a new keystore generated of type jks <hostname>-keystore.jks under username token <hostname>-public-private-key-alias.

d. Copied edge app service certificate located in app service's weblogic server to domain path of RSB/OSB domain.

e. This step imports edge application remote server public key certificate into RSB keystore will be used for message encryption/decryption process. This will need public key at prompt to verify, and add security policy on selected webservice followed by CN name that is kept hostname for easy use and remembrance purpose. Below is output of step.

Owner: CN=blrqa01.idc.oracle.com, OU=FOR TESTING ONLY, O=MyOrganization, L=MyTown, ST=MyState, C=US
Issuer: CN=CertGenCAB, OU=FOR TESTING ONLY, O=MyOrganization, L=MyTown, ST=MyState, C=US
Serial number: -39a51e9a7aa8b0158d48d8952eb73424
Valid from: Fri Mar 16 10:02:31 CST 2012 until: Sun Jan 23 10:02:31 CST 2028
Certificate fingerprints:
MD5: 39:53:84:48:19:13:15:A5:AE:60:B6:BB:C1:E1:01:8F
SHA1:
CE:D4:05:89:A7:02:B3:AC:83:E8:92:2F:9B:14:E7:33:A9:37:35:7A
Signature algorithm name: MD5withRSA
Version: 1
Trust this certificate? [no]: yes
Certificate was added to keystore

f. Next is to setup username token credentials for digital signature**.** It creates users in OSB server and also updates domain's wallet file with credentials required for message signing using digital signature imposed in SOAP message.

g. The wallet file of rsb-home is updated with the generated username token credentials that are required for configuring proposed security policy in the server. In this step, we store all credentials like username token, public key, private key, webservice to secure with proposed security policy.

The purpose of providing parameters or name of webservice to secure is to encrypt specific part of whole webservice, one of the advantage of WS-framework. These stored credentials from OSB wallet file are read and verified at time of accessing webservice and handle request from service consumer.

h. The various parameters configured in above step are:

- Local Server Keystore Password

  This field contains the password for accessing the keystore. The username is fixed to keystore-csf-key and provided the keystore password of OSB server as value here.

- <hostname>-public-private-key-alias

  This field contains the username and password for the public/private key of the OSB server in the keystore. The username is fixed to <hostname>-public-private-key, password here is private key in keystore.

- <appName>-<edgeapphostname>-remote-host-public-key-alias

  This field contains the alias name of the edge app service's remote host server. This field is created as one per application in the wallet file. So if there are services of multiple app-names secured with message protection, then this field will be prompted once per app-name of those services. The username is fixed to <appName>-<edgeapphostname>-remote-host-public-key-alias and need to provide password as public key of edge app as value.

- <appName>-user-alias

  This field contains the user name and password which are required for usernametoken authentication by the edge app service. Again this field is prompted once per app name whose services are message protected. After user enters the username/password, code creates a username/password combination for all services of that app and stores in the wallet file.

i. RSB builder is executed to compile and creates users in OSB server and also updates domain's wallet file with credentials required for message signing and encryption.

j. Admin Server and all managed servers are restarted of the OSB domain. It is required to restart servers at this point because weblogic maintains cache of wallet file

contents and new contents need to be reloaded.

k. RSB decorator jars are deployed and managed servers are again restarted. It is required for managed servers to load instrumentation jar file.

Proxy Server handshakes with service provider and to access secured WSDL deployed on application server and map policies at proxy server which will be done using username token and keypairs, will be stored in proxy domain wallet PKI will be used in this step that will generate and store certificates for each webservice at proxy server. Proxy server will store secured WSDL. It is needed to ensure and employ interoperable policies between proxy and application server.

5.1.2.4 Setting up Service Consumer

Service consumer can vary from java client to external application or any web service. Now, to establish communication between service consumer to proxy that will further communicate with service provider, proxy server digital certificate will be exported and then imported to store in service consumer domain. Service Consumer application can be deployed and run in weblogic server without any fusion components.

a. To consume services secured with proposed security framework application need to access keystore from config folder of deployed weblogic server. The client application need to provide usernametoken credentials for authenticity to proxy service. Username credentials can be stored in weblogic wallet
b. Import OSB server digital certificate into keystore of Client service consumer from domain home config folder of RSB/OSB server to service consumer application web logic domain config folder.
c. The public private key and the certificate for the service consumer client of weblogic server is generated. Service consumer generates its public private keypairs for consumer application trust, so keystore stores keypairs and public key of proxy server for encryption/decryption purpose and digitally signing the message to ensure its identity. PKI will be used for intermediary purpose between different servers.

d. Username Token will be stored in wallet of service consumer server Service consumer will hit the proxy server using username token and get the secured services with policies attached, i.e encrypted and digitally signed message. To decrypt and subscribe webservices, keystores will access employing asymmetric message process.

5.2 Testing of Security Framework

The proposed security framework is applied on Oracle RTG products version 14.1.1 where Java Service Interface tester acts as service provider or edge application deployed on weblogic 12c, Retail Service Bus deployed on weblogic 10.3.6 on OSB template, acting as service consumer OSB sbconsole test client and external open source software SOAPUI. Applied proposed security framework can be tested from OSB sbconsole or from service client webservice WSDL can be hit from SOAPUI.

5.2.1 Verification of security policies in WSDLs'

Before starting testing, it's important to verify that security policies are applied to WSDLs' hosted at edge application that acts as service provider. Verification steps include:

- Launched edge application weblogic server and gone to deployments to access edge application named javaee-service-interface-tester-14.1.0.
- Gone to overview tab and clicked on ejb level for customer bean and verified username token applied as roles and policies in Security tab -> Policies.
- To verify security policies at webservice level, gone to customerservice webservice and clicked on configuration tab and further WS-policy tab.
- Now, to verify keystore configure at domain level. Clicked on domain name, further on default_wss -> default_x509_cp is credential provider name, and here clicked on WS-Configuration subtab where table shows credential provider properties.
- Credential provider properties shows name, value (password as encrypted) for parameters CredentialityKeyAlias, CredentialityKeyStore, IntegrityKeyAlias, IntegrityKeyStore.

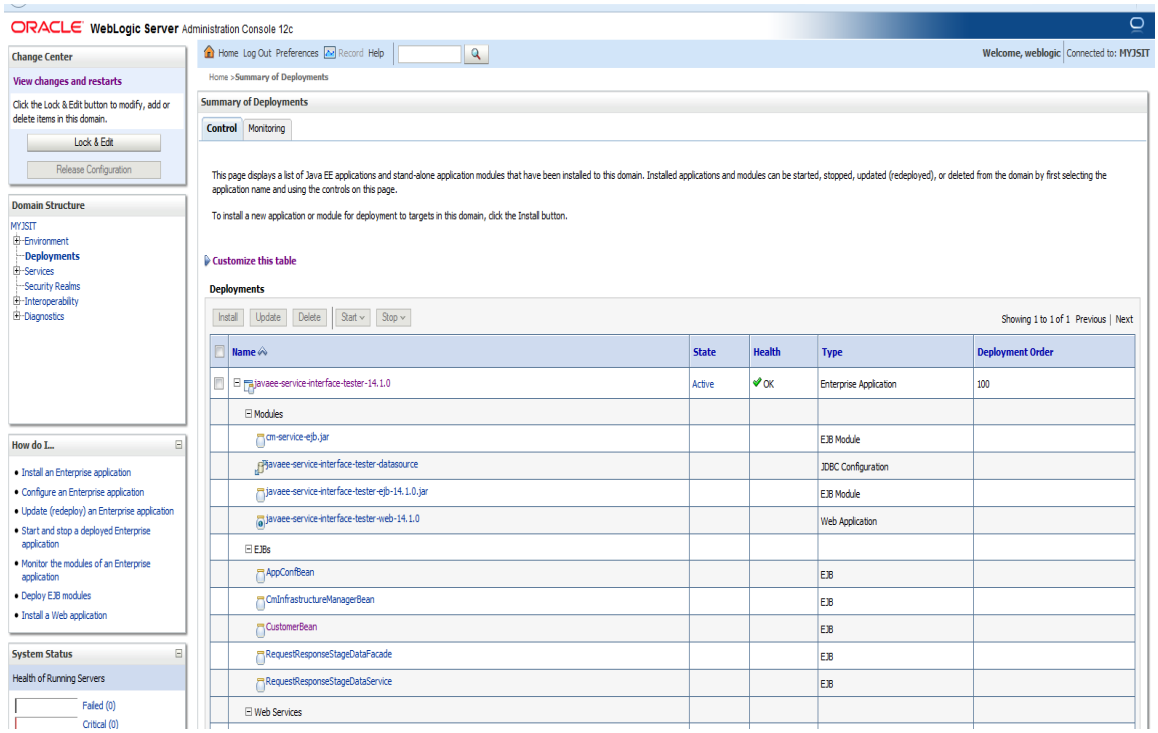


Figure 18: Edge application service overview

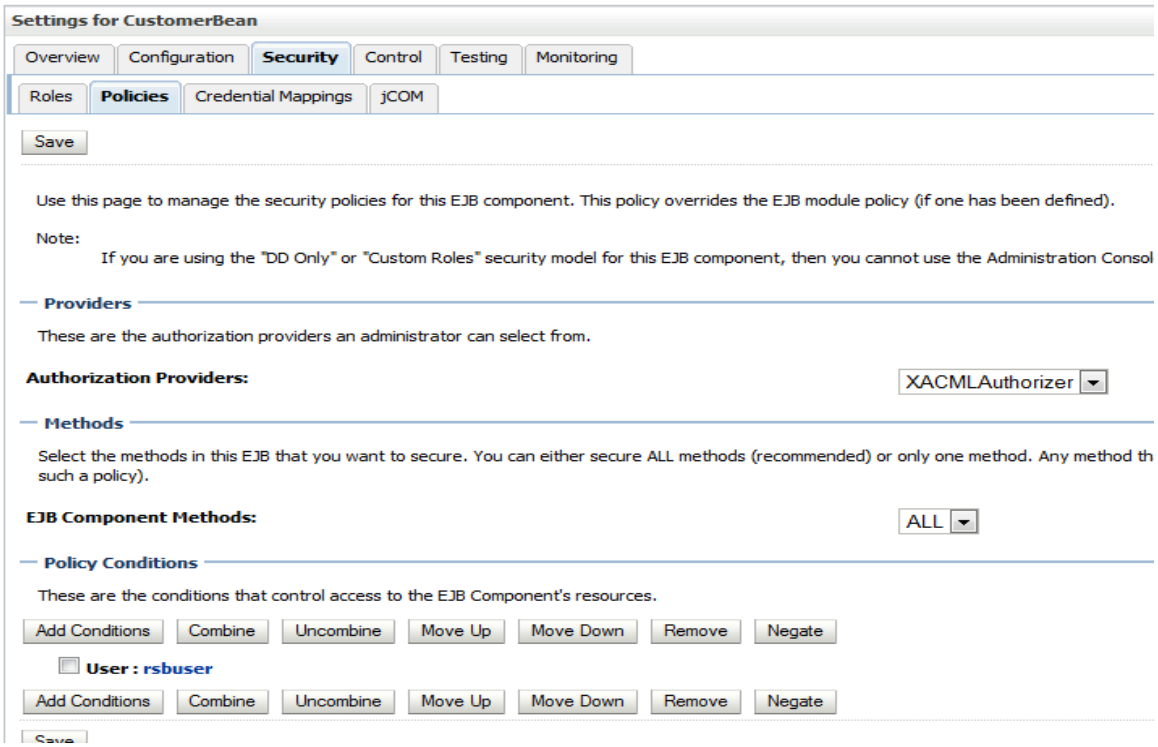


Figure 19: Username token at EJB

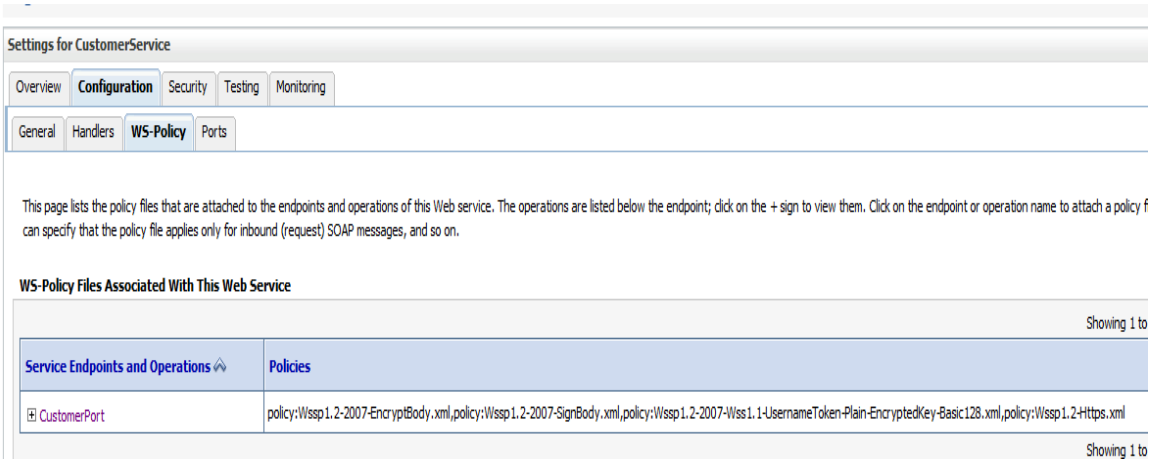


Figure 20: Security policy at webservice

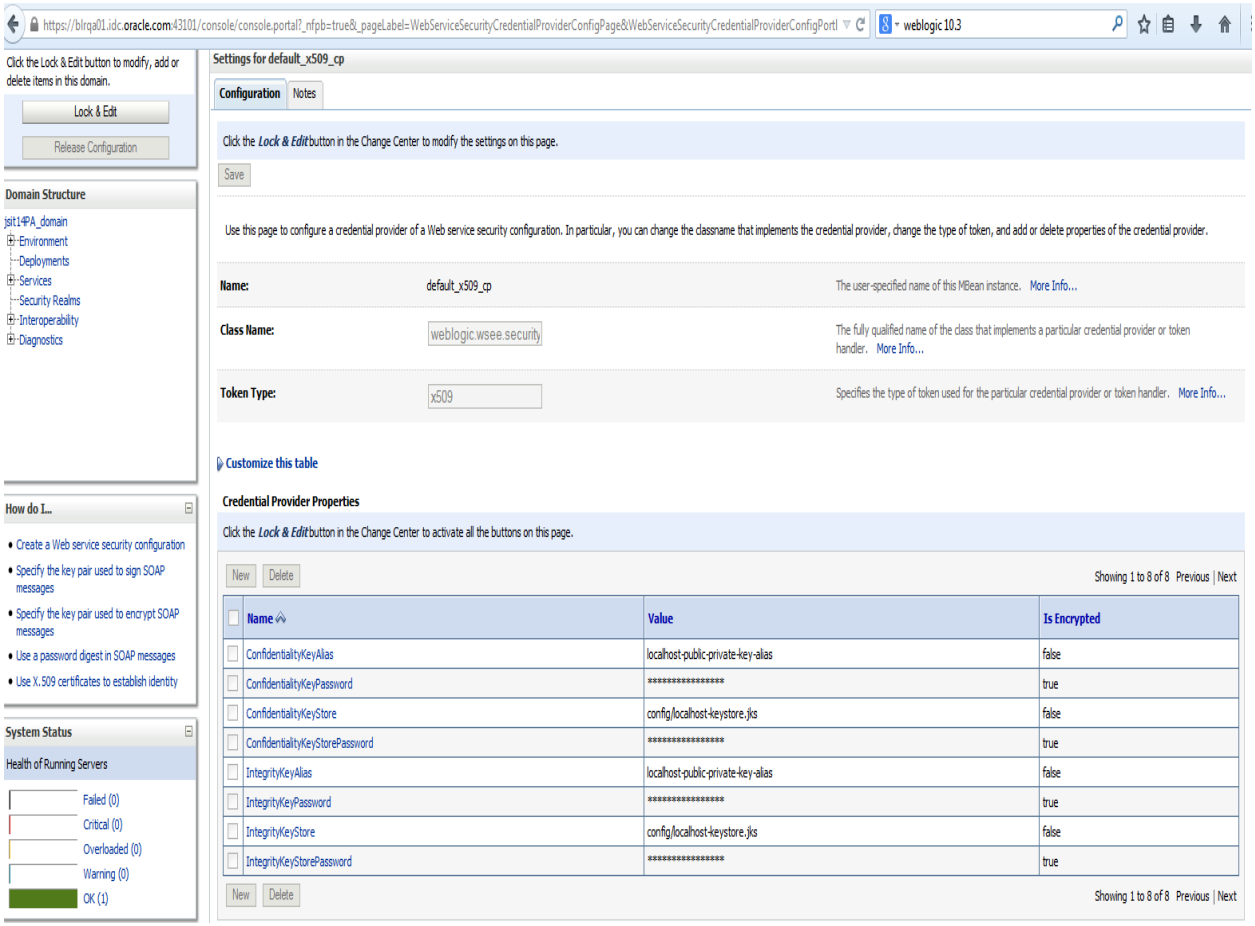


Figure 21: Keystore configured at Domain

Figure 22: Secured WSDL with Username token authentication Policy

Figure 23: Secured WSDL file configured for services

Verified WSDL accessibility at webservice level on testing tab and clicking test point.

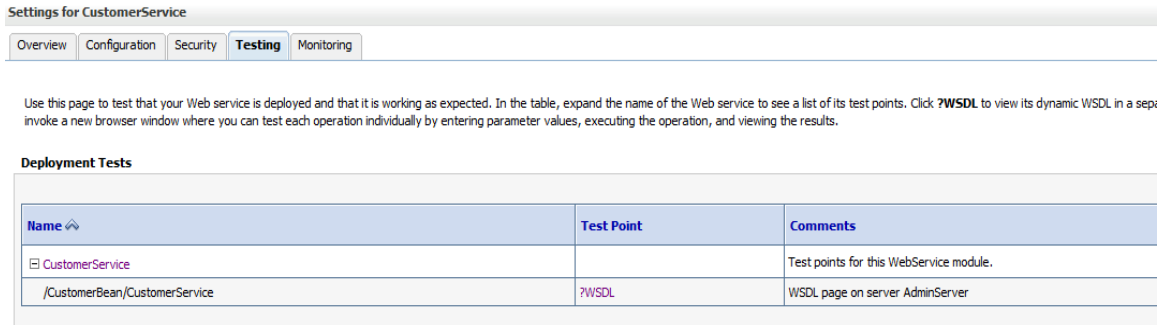


Figure 24: WSDL Testing

```
- <ns0:Policy>
  - <ns2:UsernameToken ns2:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
    - <ns0:Policy>
        <ns2:WssUsernameToken10/>
      </ns0:Policy>
    </ns2:UsernameToken>
  </ns0:Policy>
</ns2:SignedEncryptedSupportingTokens>
- <ns2:Wss11>
  - <ns0:Policy>
      <ns2:MustSupportRefKeyIdentifier/>
      <ns2:MustSupportRefIssuerSerial/>
      <ns2:MustSupportRefThumbprint/>
      <ns2:MustSupportRefEncryptedKey/>
      <ns2:RequireSignatureConfirmation/>
    </ns0:Policy>
  </ns2:Wss11>
</ns0:Policy>
- <ns0:Policy wssutil:Id="Wssp1.2-2007-EncryptBody.xml">
  - <ns2:EncryptedParts>
      <ns2:Body/>
    </ns2:EncryptedParts>
</ns0:Policy>
- <ns0:Policy wssutil:Id="Wssp1.2-2007-SignBody.xml">
  - <ns2:SignedParts>
      <ns2:Body/>
    </ns2:SignedParts>
</ns0:Policy>
- <ns0:Policy wssutil:Id="Wssp1.2-2007-Https-UsernameToken-Plain.xml">
  - <ns2:TransportBinding>
    - <ns0:Policy>
      - <ns2:TransportToken>
        - <ns0:Policy>
            <ns2:HttpsToken/>
          </ns0:Policy>
        </ns2:TransportToken>
      - <ns2:AlgorithmSuite>
        - <ns0:Policy>
            <ns2:Basic256/>
```

Figure 25: Secured WSDL with https policy and encrypt body and encrypt sign policy

5.2.2 Testing from OSB SBconsole

OSB Sbconsole can test proxy services and business services of deployed decorators. Logged to sbconsole and test the services. On the business service test page, test the security between business service and app service. Proxy and Business services deployed are shown in sbconsole. As we need to test integrated system proxy service will be executed, that will route request to business service and business service will check parameters configured at RSB. On successful authentication and keystore check, business service will send request to edge app service, where further configured policies will check the request received. We will hit execute button for testing the proxy services, in the bottom section of test page, there is "security" section. In this section, it will need to provide override values. A bug icon is there to launch testing console. Following are the security values provided for promted parameter, which are set as part of security framework:

- Keystore.recipient.alias

This is the alias name of public key of the remote app service. When we import public key of app service in the keystore of OSB server, we use alias name in the format: rms-blrqa01-remote-host-public-key-alias The same alias name should be provided in this field.

- Keystore.enc.csf.key

This is the alias name for username token of remote edge service public private key pair.

- Csf-key

This is the alias name for usernametoken authentication. The value should be in the format rms-ActivityLockService-user-alias.

Here, we give alias name not username or keystore or public private key, as alias are configured with scripts executed. These values get fetched from decorator projects, business service, security tab.

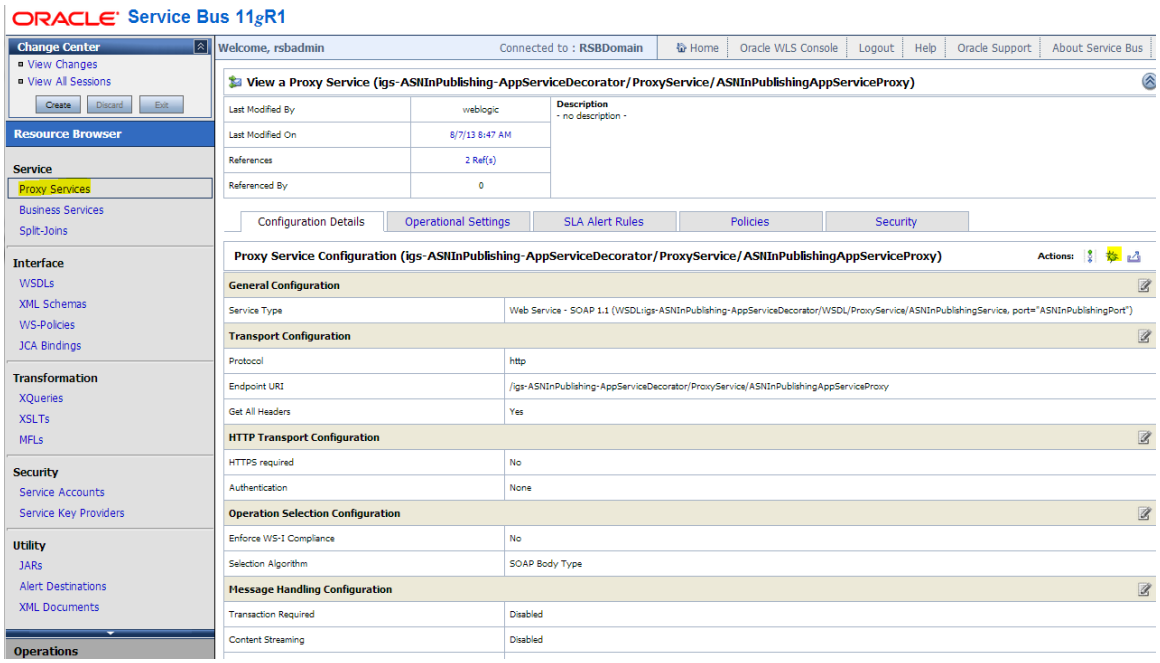


Figure 26: Launching Proxy service

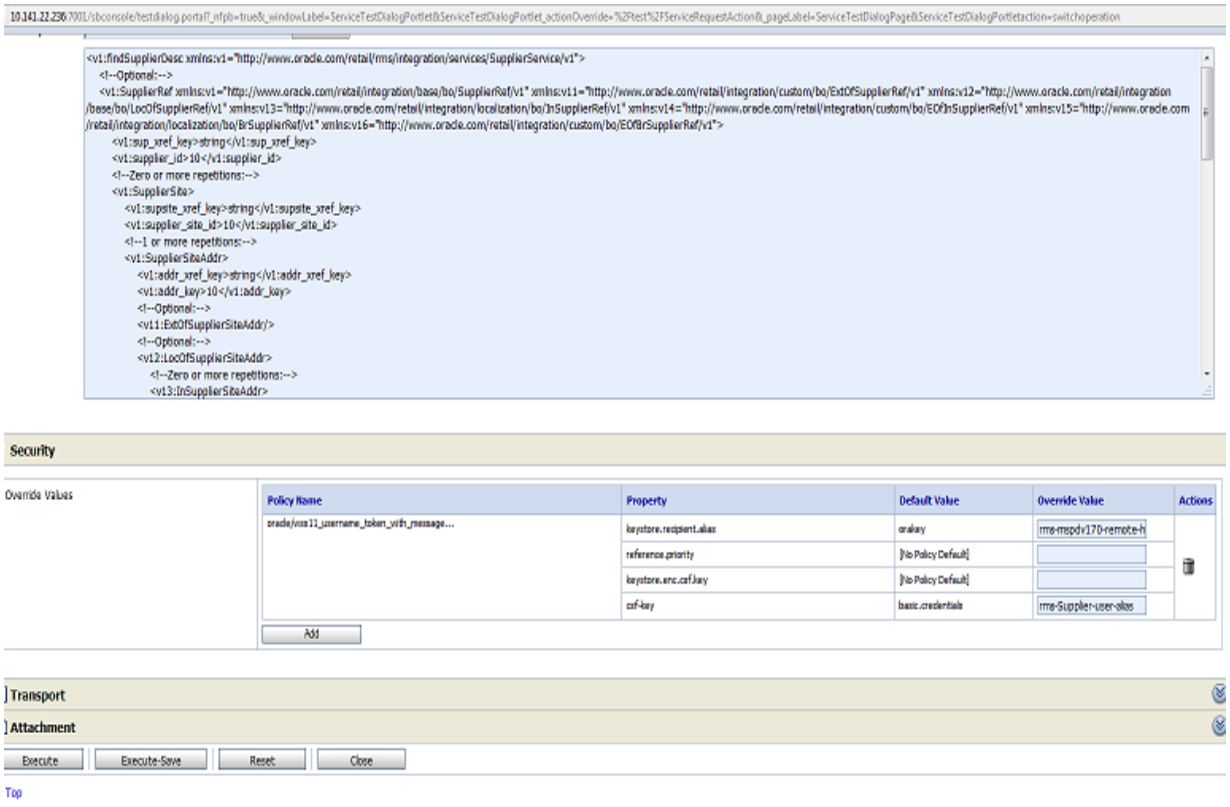


Figure 27: Testing Secured webservice from OSB Sbconsole

5.2.2. Testing from SOAPUI

SOAPUI [30] is open source webservices functional, load, performance and security testing tool. To test applied security framework, at first WSDL is imported as SOAP project either by browsing or giving address of wsdl. Once imported, all the operations available for service will be shown. To test and of secured wsdl operation, Request properties header will be provided parameters for username token is provided for authenticated, and keystore password will be provided. WS-addressing and WS-realiable messaging parameters will be set as false. Once given required inputs, service request will be hit on SSL. Once request processed, received response will be encrypted message, and digital certificate is imported in SOAP header and encrypted communication is taken place based on policies embedded. Figure shows results of policy implemented where communication link is SSL secured and communicated SOAP message is encrypted request and response is received.

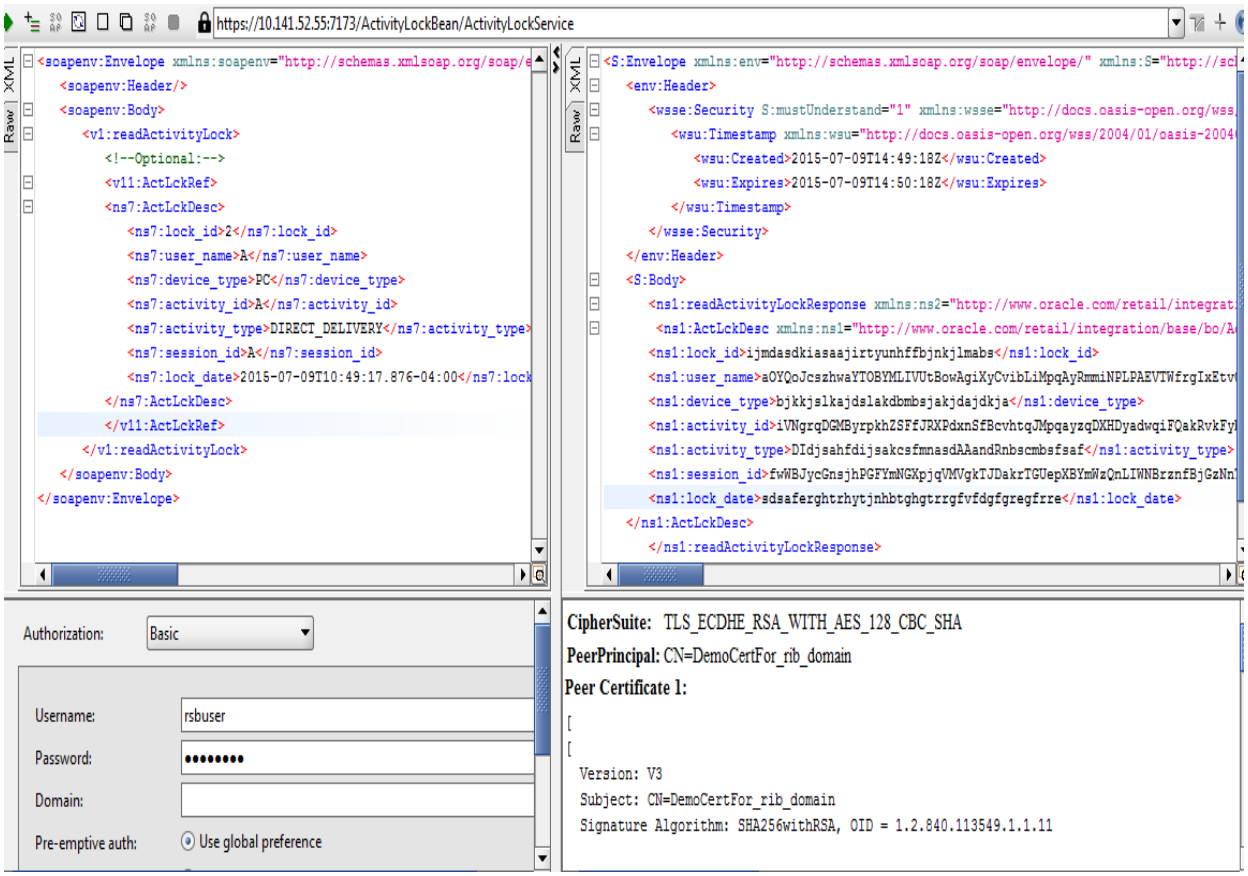


Figure 28: Testing Secured webservice from SOAPUI

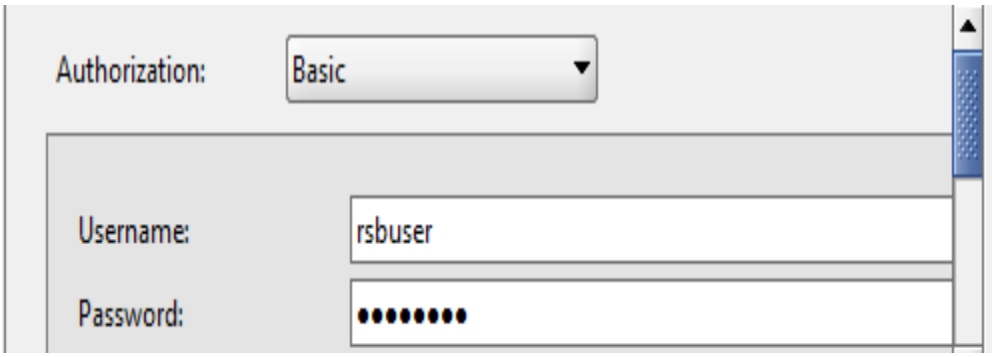
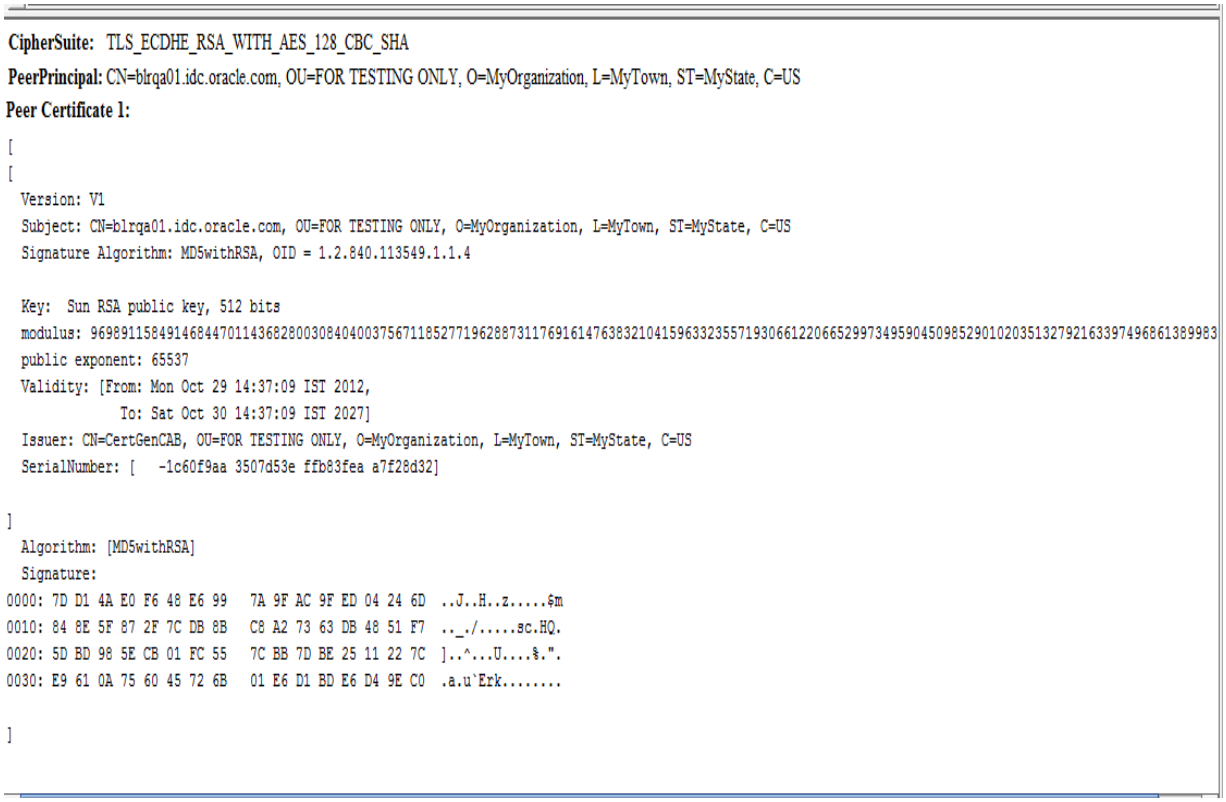


Figure 29: Prompted to Username Token for Digital Signature



CipherSuite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

PeerPrincipal: CN=blrqa01.idc.oracle.com, OU=FOR TESTING ONLY, O=MyOrganization, L=MyTown, ST=MyState, C=US

Peer Certificate 1:

```
[
[
  Version: V1
  Subject: CN=blrqa01.idc.oracle.com, OU=FOR TESTING ONLY, O=MyOrganization, L=MyTown, ST=MyState, C=US
  Signature Algorithm: MD5withRSA, OID = 1.2.840.113549.1.1.4

  Key:  Sun RSA public key, 512 bits
  modulus: 9698911584914684470114368280030840400375671185277196288731176916147638321041596332355719306612206652997349590450985290102035132792163397496861389983
  public exponent: 65537
  Validity: [From: Mon Oct 29 14:37:09 IST 2012,
              To: Sat Oct 30 14:37:09 IST 2027]
  Issuer: CN=CertGenCAB, OU=FOR TESTING ONLY, O=MyOrganization, L=MyTown, ST=MyState, C=US
  SerialNumber: [    -1c60f9aa 3507d53e ffb83fea a7f28d32]

]
  Algorithm: [MD5withRSA]
  Signature:
0000: 7D D1 4A E0 F6 48 E6 99   7A 9F AC 9F ED 04 24 6D  ..J..H..z.....$m
0010: 84 8E 5F 87 2F 7C DB 8B   C8 A2 73 63 DB 48 51 F7  .._./.....sc.HQ.
0020: 5D BD 98 5E CB 01 FC 55   7C BB 7D BE 25 11 22 7C  ]..^...U....%."
0030: E9 61 0A 75 60 45 72 6B   01 E6 D1 BD E6 D4 9E C0  .a.u`Erk........

]
```

Figure 30: Imported Digital Certificate

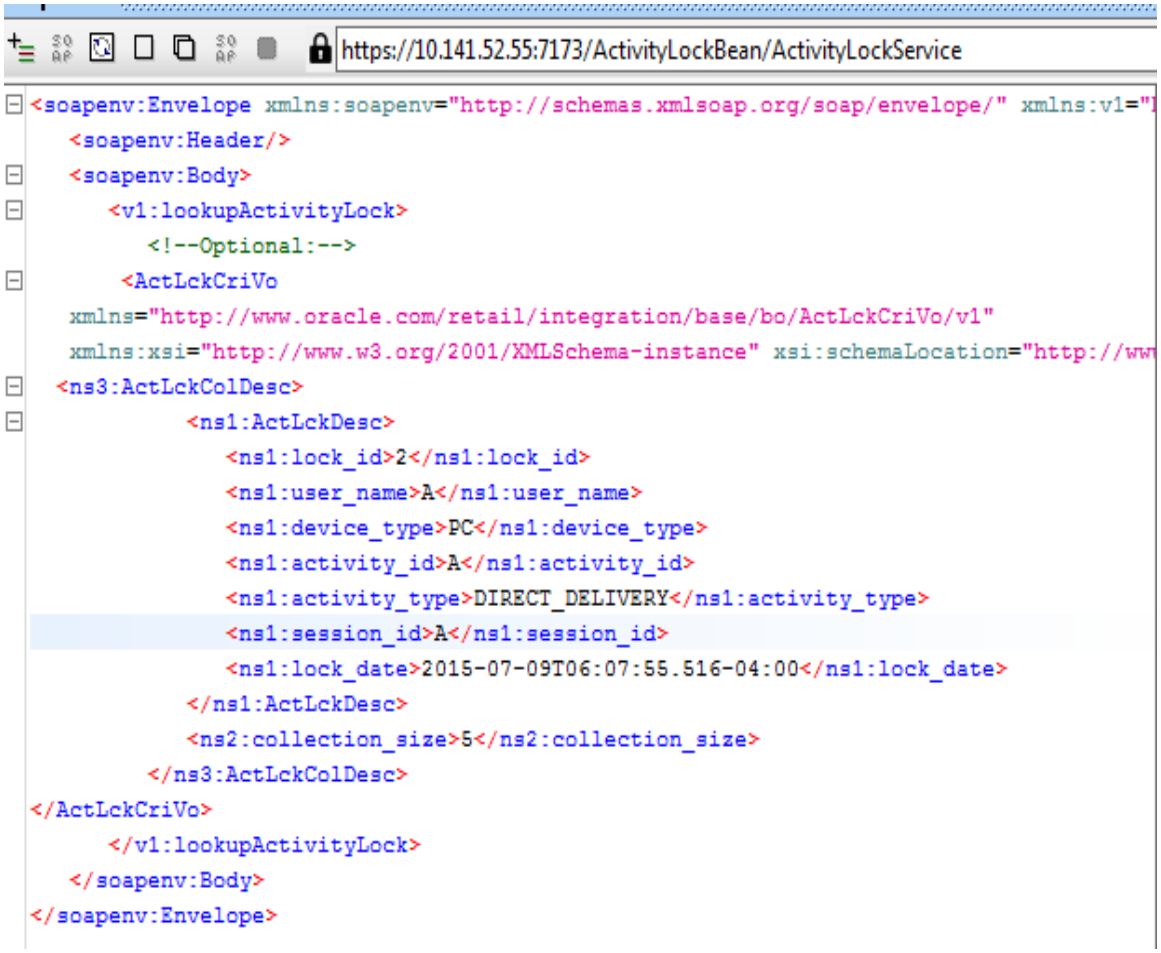
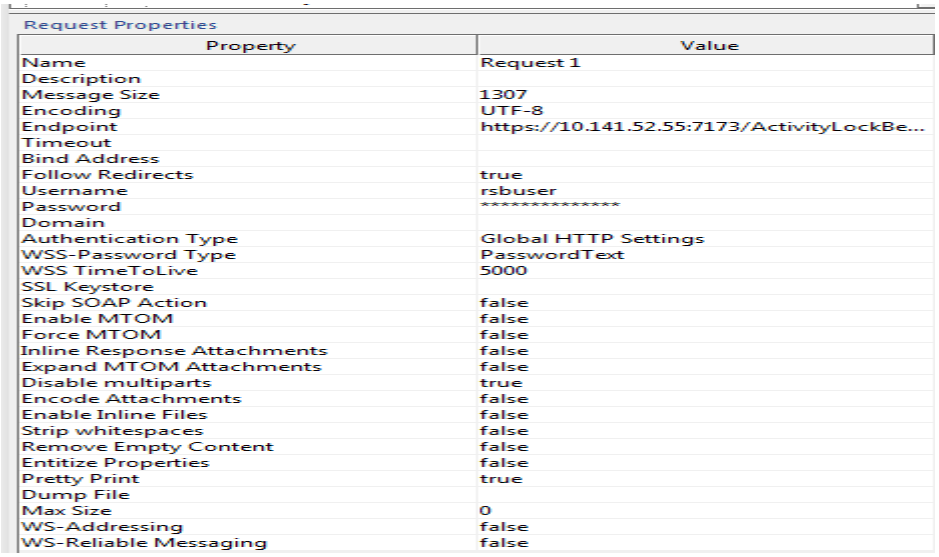


Figure 31: SOAP request communicated on SSL



Request Properties

Property	Value
Name	Request 1
Description	
Message Size	1307
Encoding	UTF-8
Endpoint	https://10.141.52.55:7173/ActivityLockBe...
Timeout	
Bind Address	
Follow Redirects	true
Username	rsbuser
Password	**********
Domain	
Authentication Type	Global HTTP Settings
WSS-Password Type	PasswordText
WSS TimeToLive	5000
SSL Keystore	
Skip SOAP Action	false
Enable MTOM	false
Force MTOM	false
Inline Response Attachments	false
Expand MTOM Attachments	false
Disable multiparts	true
Encode Attachments	false
Enable Inline Files	false
Strip whitespaces	false
Remove Empty Content	false
Entitize Properties	false
Pretty Print	true
Dump File	
Max Size	0
WS-Addressing	false
WS-Reliable Messaging	false

Figure 32: SOAP request properties

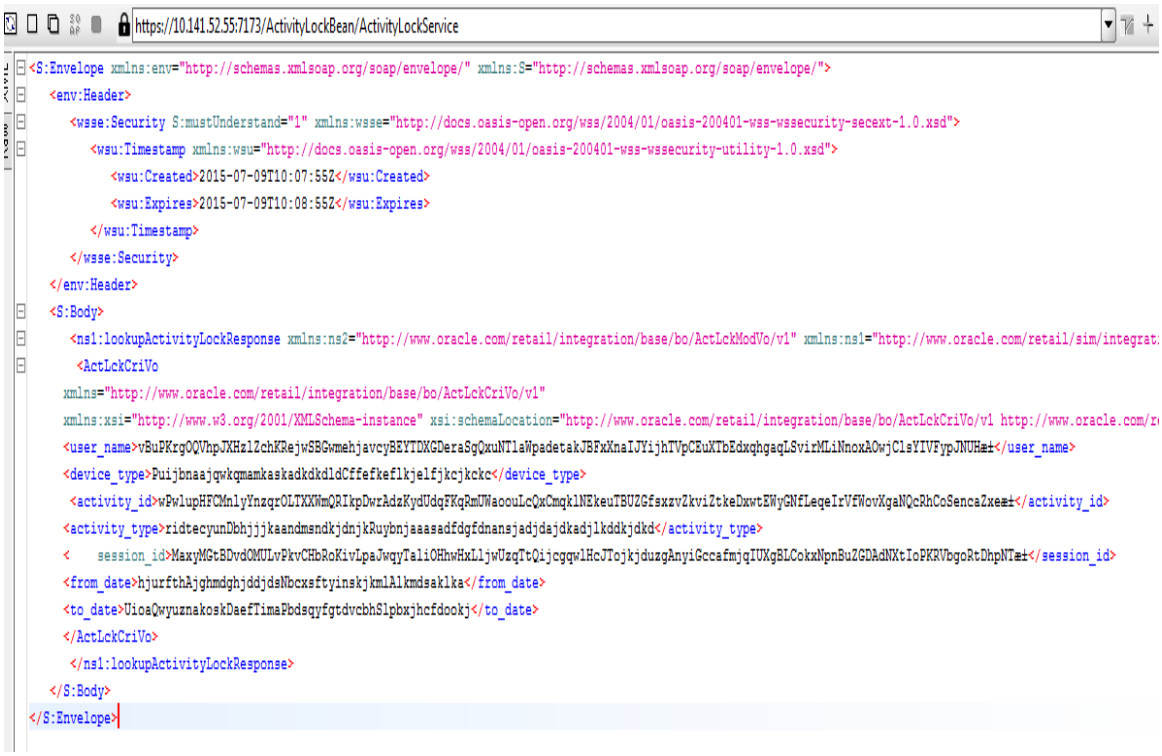


Figure 33: An Encrypted message response communicated on SSL

Results shows that employing proposed work, SOA infrastructure based on webservices architecture based on SOAP protocol can be used to secure communication link and exchanged services between service provider and service consumer for an end to end protection.

Chapter6 Conclusion and Future Work

It can be concluded that the proposed work for Enhancement of Security framework for Web services in SOA named '**Certificate based Message Protection and Username Token over SSL'** specifies a solution to have an integrated security framework for webservices based SOA environment, where SOAP message threats are handled to order to have a secure communication channel and secured communicated transaction of business messages.

The proposed security framework has been presented in section 4 and section 5, where section 4 provides high level design of deployment architecture and high level illustration of security framework configuration, and section 5 illustrates deployment of security framework.. Along with that, respective sections also provides high level analysis of how proposed security framework fits on Oracle Retail Service Backbone (a real time webservices architectural tool based SOA infrastructure) and explores Oracle Web services manager security policies to achieve web services security aspects i.e. authorization and authentication, message confidentiality, message integrity and secured transport channel.

Summary of Contributions

Although there are many WS standards and policies available for web service security, but there is need to frame these policies and standards for a integrated webservices based SOA environment. Analyzing and exposing those standards and policies, framework suggest for Certificate based Message protection and Username token over SSL.

A. Proposed solution provides an integrated security framework for webservices business infrastructure to overcome XML vulnerabilities related to SOAP and WSDL threats and threats related to unsecured communication channel.

B. Proposed framework is illustrated on Oracle Retail Service Backbone (RSB) an enterprise infrastructure for integrated webservices oriented communication.

C. Implementation and testing shows successful configuration of security framework and achieved required secured environment.

Shortcomings and Future Scope

Encryption decryption mechanism takes place at SOAP layer not in transport layer, and processing is done at higher layer of internet stack, that results in degradation of performance. Proposed security framework can be complicate to configure in an integrated domains. This policy is less supported by programming languages, so need proper understanding and defined process, as a small mistake can corrupt the domain environment. Proposed security framework for SOAP based webservices implemented on distributed applications and integrated interoperable applications within and across organizational boundaries can suffer from non-functional requirements like performance, cost requirements, load balancing for multiple service consumer clients, and fault tolerance for proxy service, business service and service provider, along with optimization parameters of Quality of Service (availability, throughput, and response time) etc.

Keeping these in mind, future work is to devise processes, mechanism or frameworks to improve non-functional requirements for SOAP message like

- Enhancement of processing time and improve performance to reduce latency,
- Employing replicas and controller to deal with service breakdown and fault tolerance,
- Finding parameters where QoS can be explicitly managed.

Popularity of SOAP based web architecture populates number of security stack but REST based webservices offers better performance. For business system where a main criterion is achieving satisfactory results of non-functional requirements parameters, challenge can be to devise such security framework for REST interaction style of webservices instead of traditional SOAP based webservices.

References

[1] Web Services Security Initiatives and Organizations http://docs.oracle.com/cd/E17802_01/webservices/webservices/docs/2.0/tutorial/doc/Security-WebSvcs6.html.

[2] World Wide Web consortium, "Web Services Architecture Requirements", (2008), http://www.w3.org/TR/wsa-reqs/.

[3] Web Services Interoperability Organization (WS-I), (2010) http://xml.coverpages.org/ws-i.html.

[4] Narges Shahgholi, Mehran Mohsenzadeh , Mir Ali Seyyedi , Saleh Hafez Qorani A New SOA Security Framework Defending Web services Against WSDL Attacks", IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing (2011).

[5] OASIS and W3C Cosponsor Forum on Security Standards for Web Services, https://www.oasis-open.org/news/pr/oasis-and-w3c-cosponsor-forum-on-security-standards-for-web-services.

[6] Blanco, J. Lasheras, E. Fernandez, R. Valencia-Garcia, A. Toval, "Basis for an Integrated Security Ontology According to a Systematic Review of Existing Proposals," In International Journal on the Development and Application of Standards for Computers, Software Quality, Data Communications, Interfaces and Measurement, Vol.33, pp. 372-388, 2011.

[7] Martin Naedele, ABB Corporate Research Standards for XML and Web Services Security http://www.tik.ee.ethz.ch/~naedele/Computer03.pdf.

[8] Understanding Web Services Security Concepts, http://docs.oracle.com/cd/E17904_01/web.1111/b32511/intro_security.htm#WSSEC1112.

[9] The cryptography guide: triple des, http://www.cryptographyworld.com/des.htm.

[10] W. Stallings, Cryptography and Network Security Principles and Practices, Fourth Edition, Prentice Hall, 2005.

[11] M. A. Rahaman, M. Rits and A. Schaad, “An Inline Approach for Secure SOAP Requests and Early Validation”, Proceeding of the Open Web Application Security Project Europe Conference (OWASP), (2006) May; Leuven, Belgium.

[12] Jothy Rosenberg and David Remy , “Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption”, 2004.

[13] S. Gajek, L. Liao and J. Schwenk, “Breaking and Fixing the Inline Approach”, Proceeding of the ACM Workshop on Secure Web Services (SWS), (2007) November; Fairfax, VA, USA.

[14] A. Kadir, “RewritingHealer: An Approach for Securing Web Service Communication”, Master Thesis in KTH Royal Institute of Technology, Sweden, (2008).

[15] Aziz Nasridinov, Jeong-Yong Byun and Young-Ho Park, “A Study on Detection Techniques of XML Rewriting Attacks in Web Services” International Journal of Control and Automation Vol.7, No.1 (2014).

[16] T. S. Barhoom and R. S. K. Rasheed, “Position of Signed Element for SOAP Message Integrity”, International Journal of Computer Information Systems, vol. 2, no. 1, (2011).

[17] Nicolas Serrano, Josune Hernantes, and Gorka Gallardo, “Service-Oriented Architecture and Legacy Systems”, IEEE SOFTWARE (2014).

[18] Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Bode Carson, Ian Evans, Dale Green, Kim Haase, Eric Jendrock J2EE™ 1.4 Tutorial Sun Java System Application Server Platform Edition 8.2.

[19] Mehran Mohsenzadeh, Narges Shahgholi, Mir Ali Seyyedi , Saleh Hafez Qorani “A new security framework against Web services’ XML attacks in SOA”, IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing (2011).

[20] J. Somorovsky, M. Heiderich, M. Jensen, J. Schwenk, N. Gruschka and L. Iacono, “All Your Clouds are Belong to us - Security Analysis of Cloud Management Interfaces”, Proceedings of the 3rd ACM workshop on Cloud computing security workshop (CCSW), (2011) October; Chicago, IL, USA.

[21] J. Y. Byun and A. Nasridinov and, “A Self-Adaptive Approach to Secure Integrity of SOAP Messages”, Journal of KIISE (Korea Institute of Information Scientists and Engineering): Computing Practices and Letters, vol. 18, no. 9, (2012).

[22] K. S. Smriti and B. Azzadine, “A Formal Solution to Rewriting Attacks on SOAP Messages”, Proceeding of the ACM Workshop on Secure Web Services (SWS), (2008) October; Alexandria, VA, USA.

[23] Oracle Fusion Middleware Developer's Guide for Oracle Service Bus http://docs.oracle.com/cd/E17904_01/doc.1111/e15866/toc.htm.

[24] Oracle Fusion Middleware Security and Administrator's Guide for Web Services http://docs.oracle.com/cd/E17904_01/web.1111/b32511/toc.htm.

[25] Oracle® Retail Service Backbone Implementation Guide 14.1, December 2014,

[26] http://docs.oracle.com/cd/E12461_01/141/rsb_implementation_guide/rsb-141-impg.pdf.

[27] Oracle® Retail Service Backbone Developers Guide 14.1, December 2014, http://docs.oracle.com/cd/E12461_01/141/rsb_developer_guide/rsb-141-dg.pdf.

[28] Oracle Retail Service Backbone Security Guide 14.1, December 2014,

[29] http://docs.oracle.com/cd/E12461_01/141/rsb_security_guide/rsb-141-sg.pdf.

[30] Oracle® Retail Service Backbone (RSB) Release Notes Release 14.1, December 2014 http://docs.oracle.com/cd/E12461_01/141/rsb_release_notes/rsb-141-rn.pdf.

[31] Oracle® Retail Functional Artifacts Guide Release 14.1 December 2014 http://docs.oracle.com/cd/E12461_01/141/funtional_artifacts_guide/or-141-fasg.pdf.

[32] SOAPUI, http://www.soapui.org/about-soapui/what-is-soapui-.html.

[33] Project JAXB, https://jaxb.java.net/.

[34] Entrust Securing Digital Identities and Information, “Web Services Trust and XML Security Standards”, version 1.0, (2001), https://www.entrust.com/wp-content/uploads/2013/05/TWS_apr9.pdf.

[35] L. Fang, D. Liang, F. Lin and C. C. Lin, “Fault-Tolerant Web Services”, Journal of System Architecture, vol. 53, no. 1, (2007).

[36] I. Pinzon, J. Bajo, J. F. De Paz and J. M. Corchado, “S-MAS: An Adaptive Hierarchical Distributed MultiAgent Architecture for Blocking Malicious SOAP

Messages within Web Services Environments", Expert Systems with Applications, vol. 38, no. 5, (2011).

[37] Prof Sunil R Dhore,+IHariswaroop Gangwar, Pradyumn Mishra, Rahul Sharma, Rajeev Singh "Systematic Approach for Composing WebService using XML", IEEE ICCCNT Coimbatore, India (2012).

[38] Peter Leo Gorski, Luigi Lo Iacono, Hoai Viet Nguyen, Daniel Behnam Torkian, "Service Security Revisited", IEEE International Conference on Services Computing (2014).

[39] Oracle® Retail Integration Bus Service-Oriented Architecture Enabler Tool Guide Release14.1,http://docs.oracle.com/cd/E12461_01/141/retail_soa_enabler_tool_guide/or-141-rse.pdf.

[40] Joe M. Tekli, Ernesto Damiani, Gabriele Gianini , Richard Chbeir, "SOAP Processing Performance, IEEE transactions on services computing, vol. 5, no. 3, july-september 2012 and Enhancement"

[41] Yoon Jae Kim "Access Control Service Oriented Architecture Security", http://www.cse.wustl.edu/~jain/cse571-09/ftp/soa.pdf.

[42] Anderson, Simon Godik, OASIS XACML, version 1.1., august 2013, https://www.oasis-open.org/.../xacml/.../cs-xacml-specification-1.1.pdf.

[43] Web Services Policy Framework (WSPolicy), Version 1.2, March 2006 http://specs.xmlsoap.org/ws/2004/09/policy/ws-policy.pdf.

[44] Web Services Reliable Messaging Protocol (WS-ReliableMessaging), February 2005, http://specs.xmlsoap.org/ws/2005/02/rm/ws-reliablemessaging.pdf.

[45] Oracle Identity Management, http://www.oracle.com/technetwork/middleware/id-mgmt/overview/index.html.

[46] Phillip Hallam-Baker, Shivaram "XML Key Management Specification (XKMS 2.0)", june 2005, http://www.w3.org/TR/xkms2/.

[47] OASIS Web Services Federation Language (WS-Federation) Version 1.2, May 2009 http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.html.

[48] Empowered Identity Management WS-Trust and WS-Federation, http://www.empowerid.com/learningcenter/standards/ws-trust-fed.

[49] Javascript Object Signing and Encryption (JOSE), http://jose.readthedocs.org/en/latest.

[50] Fusion Middleware Security and Administrator's Guide for Oracle Fusion Middleware Online Documentation library, 11g Release 1(11.1.1.4) Web Services , http://docs.oracle.com/cd/E17904_01/web.1111/b32511/toc.htm.

[51] Oracle Retail Service Backbone Installation Guide Release 14.1, December 2014, http://docs.oracle.com/cd/E12461_01/141/rsb_installation_guide/rsb-141-ig.pdf.

[52] Thomas Bleier, Arndt Bonitz, Christian Wagner, "Efficient Security Implementations in eGovernment Workflows: a Practical Application", World Telecommunications Congress 2010, Vienna.

List of Publications

- Communicated
    - Ruchi Mishra and Seema Bawa "**Analysis of Security Framework for Web Services in SOA**"