Dynamic and Ubiquitous Security Architecture for Global SOA

Deven Shah, Shrikant Goswami, Nikhil Khekade, Rachna Pardeshi, Dhaval Doshi and Dhiren Patel

***Abstract*-- This work presents a detailed analysis of the security requirements for Global SOA. The Global SOA is about the entire Web being a reusable, shareable, public SOA The main problem in seamless ubiquitous integration of distributed network of web services into one Global Service oriented Architecture is that of security. Our strategy is to work on SOAP message interceptor or Handler for providing message level security in SOA. And when we move to Global SOA then we are proposing architecture for ubiquitous integration of security using handlers without any pre-configuration required at service requester side.**

***Keywords*-- Global SOA, Handlers, SOA, Security, Web Services.**

I. Introduction

THE world today is seeing the power of community intelligence and thoughts through the medium of blogs, wiki and other online communities. Wouldn't it be great if like humans, applications could also work together intelligently to give the users what they require more efficiently and effectively? Most service-oriented architectures (SOA) are still conceptually trapped inside an organization's firewall or VPN. Global SOA envisions the Web as the global stage upon which to act out grand visions of constructing vast supply chains of data and global application-to-application communication. The security is a major barrier for migrating SOA from enterprise network to web. Various Security mechanisms for SOA require static binding between communicating web services. Hence we propose ubiquitous security architecture for Global SOA.

Rest of the paper is organized as follows: Section 2 discusses SOA for EAI and Global SOA for Web. In next section we discuss the security strategy of SOAP message interception using handlers, for the SOA. Then we propose architecture for ubiquitous integration of security using handlers without any prep-configuration required at service requester side [6].

Prof.Deven Shah is with the Department of Information Technology Engineering SPIT, Mumbai University, Mumbai, INDIA (e-mail: devenshahin@yahoo.com).

Shrikant Goswami is a student of Information Technology Engineering SPIT, Mumbai University, Mumbai, INDIA.
(e-mail: shrikant.goswami@gmail.com).

Nikhil Khekade is with Cognizant Technologies, Mumbai, INDIA (e-mail: nikhil.khekade@gmail.com).

Rachna Pardeshi is with MindTree Consulting, Bangalore, INDIA (e-mail: rac1910@gmail.com).

Dhaval Doshi is with ThoughtWorks, Pune, INDIA (e-mail: dhaval.doshi@gmail.com).

Dr. Dhiren Patel is with the Department of Computer Engineering NIT, Surat, INDIA

II. SOA for EAI and Global SOA for Web

A. SOA for EAI

SOA is the exposure of software resources in the form of services, which can be accessed over a network. When SOA is used for EAI where diverse applications in an enterprise communicate and collaborate to achieve a business objective, binding between the web services is pre-configured and the interaction is static. The UDDI used is private and is accessible to organization, its business partners only. Fig. 1. shows basic SOA architecture. Fig. 2. shows implementation of SOA architecture for EAI where static binding between web services is mandatory.

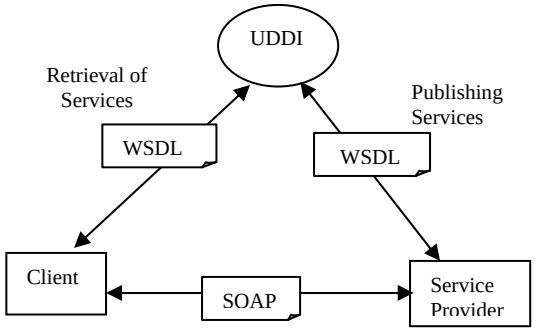

Fig. 1. SOA Architecture

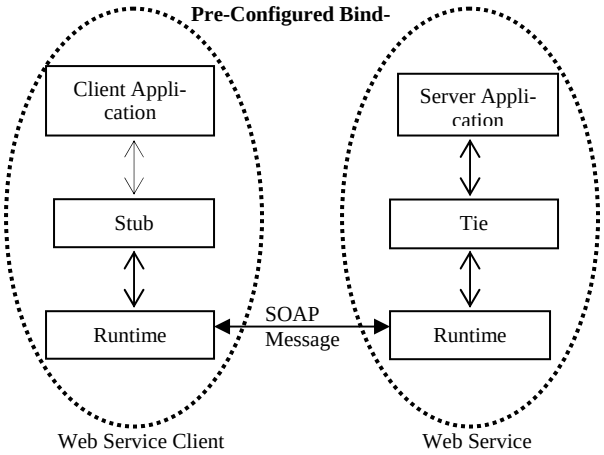

Fig. 2. Static web service interaction

B. GLOBAL SOA for WEB

As businesses become global there is a need for these applications to become available globally. Thus the pre-configured binding between web services becomes obsolete. Suppose an application for an online shopping chain accesses a service broker that specializes in shipping. The broker locates services from public UDDI registry that meet certain criteria such as fast delivery time and invokes them at run time. Thus the binding between broker and web services is dynamic. Fig. 3. shows the dynamic binding between client and web services in Global SOA.

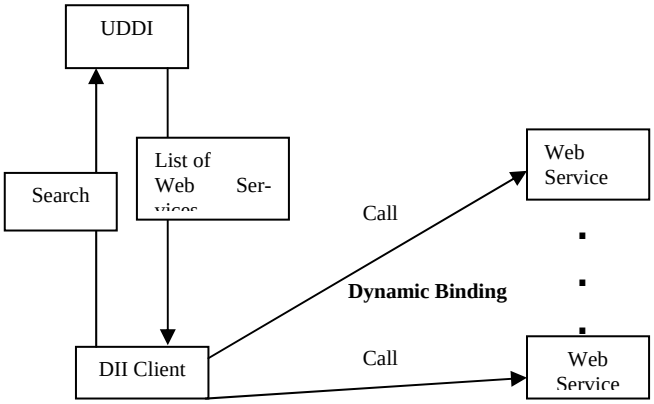

Fig. 3. Dynamic Invocation of multiple web services

III. SECURITY FOR SOA

A. Security for SOA

Security is one of the major concerns of SOA-based implementations, especially when it spans outside the boundaries of enterprise. We identify message level security as the best approach for securing web services [7]. One strategy to implement message level security is to embed the security processing logic in the application code. But this leads to complexity in testing of both business functionality and security requirements. When the business application grows larger in scale or becomes highly distributed, the maintenance effort and support to manage a change in the security processing logic is enormous. Also as the tools for creating web services directly from application code are available, this strategy puts additional burden on developer to study SOAP Message structure.

A better strategy would be to use Message Handlers [1]. Also known as SOAP interceptors they provide a way of modifying the SOAP Request/Response. A simple example of using handlers is to encrypt and decrypt secure data in the body of a SOAP message [3]. A client application uses a handler to encrypt the data before it sends the SOAP message request to the Web service. The Web service receives the request and uses a handler to decrypt the data before it sends the data to the back-end component that implements the Web service. This provides the advantage of making security independent of business functionality.

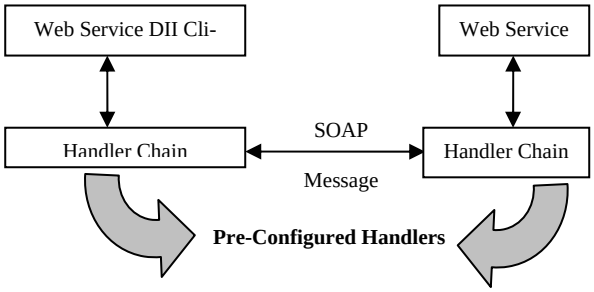

Fig. 4. Dynamic Invocation of web service with pre-configured handlers

The concept of message handler is now gaining a stand in context of web service security and is a best choice in SOA for EAI where handlers are pre-configured. But using message security handlers with global SOA raises many concerns.

B. Global SOA security concerns with handlers

1) Dynamic Configuration of handlers:

In Global SOA, Client dynamically invokes web services based on the result returned from the UDDI. Hence security cannot be pre-established. This necessitates dynamic configuration of security handlers. Also these handlers form a chain as per the order specific to web services. This chain should be configured at run time for each web service being called.

2) Identification of handlers:

Now issues remains, how client identifies handlers, which are needed to be incorporated at its side. These include, number of handlers, type of handlers, Sequence of handlers for creating handler chain etc.

3) Security handler information exchange:

As the configuration of handlers should be dynamic, handler information should be conveyed to the client in some manner. This becomes difficult, as it requires one more level of interaction, which conflicts, with request-response model of SOAP. Currently UDDI and WSDL in the only way that service information can be conveyed to the client.

4) Types of Universal Handlers:

Based on security requirements we have identified and designed handlers for Authentication, Authorization, Confidentiality, Message integrity, Non-repudiation, Denial of Service, XML Injection, and XML Rewriting.

IV. PROPOSED ARCHITECTURE

In global SOA, Client dynamically searches for the web services from UDDI and then sends SOAP request to that web services. Server-side Web Services has incorporated various handlers as per its security requirement.

1) Unique identification of handlers:

Our solution gives unique identification for handlers i.e. we have created Handlers based on various security requirements for e.g. handler for authentication, encryption, handler for identifying content-based DoS attack etc. All handlers are part of security package and can be bundled with application server software or be part of J2EE / .NET Framework, or can be freely downloadable from the Internet. Client will identify server-side handlers based on this unique identification, and then create a handler chain from the same package to modify the SOAP message.

2) Security handler information exchange at runtime:

We identify UDDI and WSDL as the only means to convey the web service handler info to the client, as they are available prior to the interaction with web service. But as UDDI is a global database used for discovery of web services, inserting handler info in UDDI leads to additional burden. On the other hand, WSDL is located on the Application server where the web service is present and is associated with each web service, hence is a better place to insert handler information.

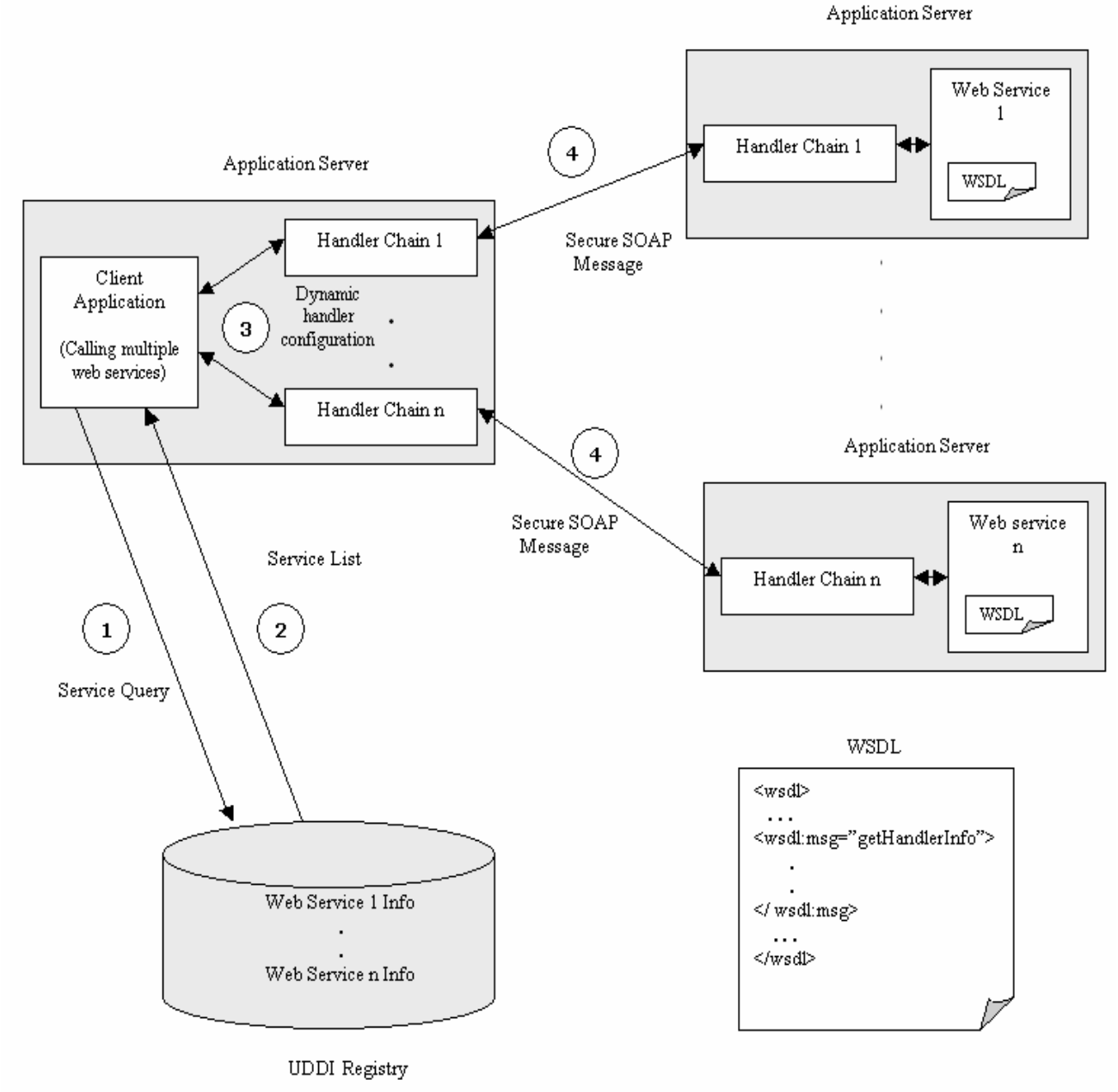

Fig. 5. Dynamic and Ubiquitous Security Architecture

This implementation is based on JAVA platform; it is probable that handlers can be developed using .NET framework. In this case, we are giving identification to handlers, irrespective of platform, solely based on final SOAP envelope generated by handlers. Hence a Microsoft .NET web service with handler can invoke a JAVA web service.

V. Implementation

As per proposed architecture we have the following setup.

1. Two application servers each have one web service with different configuration of handlers.

2. A UDDI server on which the two web services have been published under the same business category.

3. A DII (Dynamic Invocation Interface) client web service running on an application server, which will query the UDDI first and as per the results dynamically configure and call other web services.

Fig. 6 shows how a server-side web service is created and published in UDDI directory with security requirements.

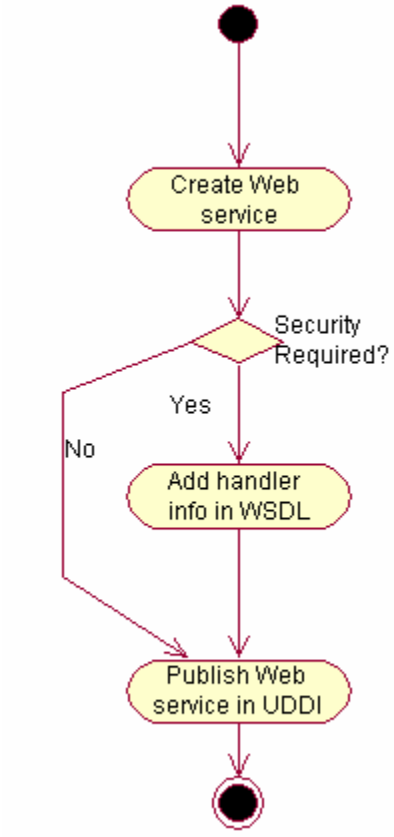

Fig. 6. Server-side

As current version of WSDL does not define a standard tag for message handler chain, we have created a standard message called 'getHandlerInfo' and inserted handlers information as the 'part' of the message in WSDL.

Fig. 7 shows configuration of two handlers Authentication and Digital Signature handler in WSDL.

```
<wsdl:message name="getHandlerInfo">
  <wsdl:part name="security.authentication.handler.ClientAuthenticationHandler" />
  <wsdl:part name="security.signature.handler.ClientSignatureHandler" />
</wsdl:message>
```

Fig. 7. Handler Info in WSDL

Fig. 8 shows how a client-side web service makes a call to the server side web service.

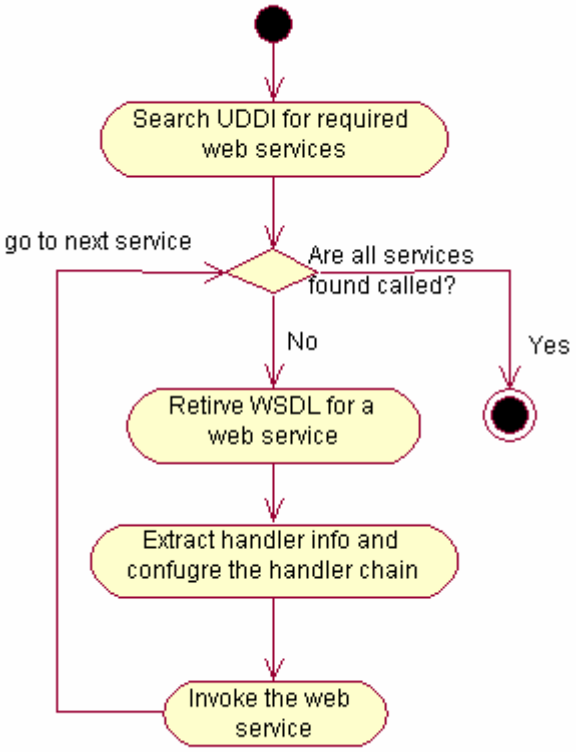

Fig. 8. Client-side

An intelligent module in DII client extracts handler information from WSDL and configures a handler chain at runtime using WSDL.

A. Result

As we configured two handlers dynamically at the client side (Authentication and Digital Signature), we captured the SOAP envelope at the following stages:

1. Between the client application and the first i.e. Authentication Handler.
2. Between Authentication Handler and Digital Signature Handler
3. After Digital Signature Handler execution

Each handler processes the SOAP envelope modifies or adds some security information in it and forwards it to the next handler. Final SOAP envelope generated is an output of handler chain execution which is forwarded to the target web Service.

VI. CONCLUSION

In this paper we have presented architecture and implementation detail for security in Global SOA. In the architecture, we use SOAP interceptor as a mechanism to implement various security standards for SOA. When SOA migrates over web as global SOA, then it is necessary to implement security mechanism between client and server web services dynamically and ubiquitously. We have taken step-by-step approach to solve the problem. First we created different interceptors based on various SOA security requirements, and gave unique identification to them. We discussed how service requester dynamically configures interceptors at its side based on security requirement at service provider side, and implemented the solution for the same.

VII. REFERENCES

[1] Reactive Inc: Security with in XML infrastructure, Available: http://www.reacivity.com, 2006.

[2] Hal Lockhart: "Web Services security update", BEA Systems Architecture Available: http://www.omg.org/news/meetings/workshops/SOA_MDA_WS_Workshop_CD/04-3_Lockhart.pdf

[3] SOA world Magazine: A Strategy for securing web services, 2007 SYS-CON Media Inc. Available: http://webservices.sys-con.com/read/39696_p.htm

[4] Apache Axis SOAP Toolkit: http://xml.apache.org/axis

[5] Marina Fisher, Ray Lai, Sonu Sharma, Laurence Moroney: "Java EE and .Net Interoperability", vol. 1, Prentice Hall, April 21, 2006.

[6] Domenico Cotroneo, Almerindo Graziano and Stefano Russo, "Security requirements in service oriented architectures for ubiquitous computing" ACM International Conference Proceeding Series; Vol. 77, Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing.

[7] Web Services Security: SOAP Message Security 1.0 (WS-Security 2004) Available: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsssoap-message-security-1.0.pdf

VIII. BIOGRAPHIES

Deven Shah is professor in IT Dept; S.P.College of engineering, Mumbai. He is currently pursuing PhD from NIT, Surat.
(E-mail: devenshahin@yahoo.com)

Shrikant Goswami is student of BE IT, S.P.College of engineering, Mumbai.
(E-mail: shrikant.goswami@gmail.com)

Nikhil Khekade is with Cognizant Technologies, Mumbai. INDIA.
(E-mail: nikhil.khekade@gmail.com)

Rachna Pardeshi is with MindTree Consulting, Bangalore, INDIA.
(E-mail: rac1910@gmail.com)

Dhaval Doshi is with ThoughtWorks, Pune, INDIA.
(E-mail: dhaval.doshi@gmail.com)

Dr. Dhiren Patel is with SVNIT Surat, India
Email:dhiren29p@gmail.com