IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

SECURITY TESTING FRAMEWORK FOR SOA MIDDLEWARE IN BANKING DOMAIN

Damini Jambhulkar, Tarun Yengantiwar,
Mtech Student, HOD CSE,
V M Institute of Engineering & Technology,
V M Institute of Engineering & Technology,
Nagpur, India

Abstract : In the banking domain, a high level of security must be considered and achieved to prevent a core-banking system from vulnerabilities and attackers. This is especially true when implementing Service Oriented Architecture Middleware (SOAM), which enables all banking e-services to be connected in a unified way and then allows banking e-services to transmit and share information using simple Object Access Protocol (SOAP). The main challenge in this research is that SOAP is designed without security in mind and there are no security testing tools that guarantee a secure SOAM solution in all its layers. Thus, this paper studies and analyses the importance of implementing secure banking SOAM design architecture and of having an automated security testing framework. Therefore, Secure SOAM (SSOAM) is proposed, which works in parallel with the banking production environment. SSOAM contains a group of integrated security plugins that are responsible for scanning, finding, analysing and fixing vulnerabilities and also forecasting new vulnerabilities and attacks in all banking SOAM layers.

***IndexTerms* - SOA Middleware, BPEL, Automation Security Testing Framework, Orchestrated Business Process, SOAP Protocol, Secure Banking Architecture**

INTRODUCTION

Service-Oriented Architecture (SOA) has been introduced for solving the problem of ensuring effective, reliable and secure interaction of complex distributed systems. SOA assumes that such systems are constructed from separate functional application modules (services) that have interfaces, defined by common rules (WSDL - description), and a dedicated invoke mechanism (SOAP messages). Software trends show that maintainability is paramount in new software design. User's requirements and expectations are continually changing and need to be met. SOA is designed specifically, to satisfy that need [9]. SOA has the intent to have several autonomous services loosely coupled together by a central controller that will coordinate the services for the users. The architecture naturally facilitates design goals such as modifiability, reusability, reliability and much more.

2. BENEFITS

The benefits of using SOA stem mostly from the loose coupling between the individual services. Each service is added to a registry separately, and therefore does not affect how other services operate. The separation of functional blocks promotes an easy system to understand, as developers and testers only need to study code for the individual service, a much smaller scale compared to the entire system. The meaning of processes also become clearer, as services abstracted away the internal details, and only the I/O interfaces need to be grasped. Since services are very easily understood, the system as a whole, very simple to debug and modify. This is an important contributing factor to the systems reliability. Since the services of the system are easily testable, this reduces the chances of the system crashing. Even in the event that a service crashes, SOA is fairly fault tolerant [6]. The system should continue to function properly despite a failed service, it should simply allow the user to query for alternative services that may accomplish the required task.

3. SOA STRUCTURE

The basic assumption of SOA is that there are many consumers that require services. In literature, consumers are also referred to as clients or customers. These terms are used interchangeably here. On the other side, there are many providers that provide services

on the network. These two groups have to be linked together in a dynamic and adaptive way. This is usually done by a service broker [7], [5].
Service providers register their services at the broker (registry), service consumers request a service from the service broker, which returns a known provider for the requested service. Consumer and provider agree on the semantics. The consumer then binds himself to the service provider and uses the service. The structure of SOA architecture is shown in Figure. 1.

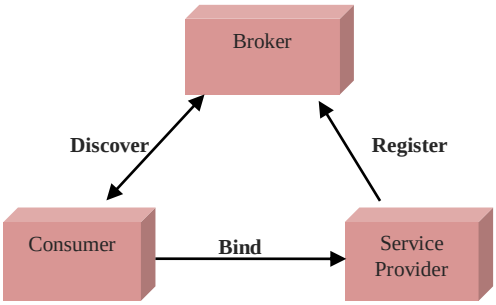

Fig 1: SOA Structure [1]

A. Services
The Service is a unit of software that will perform some functionality for the client. The service can provide meaningful functionality in itself in order to promote reuse. The service must be defined by its ability to communicate with a client, and it must advertise these details in a way that the client can understand. Other important feature of a SOA is low coupling between the client and the service. The IDEs, such as eclipse, have features that allow you to select a service you wish to use in your client, and then the IDE creates client side," Proxy or Stub classes", allowing the client to communicate with service. This practice allows for fast and easy SOA development, but may cause trouble down the line if the client needs to change the service it uses [6].

B. WSDL
Before a service can be discovered and consumed by the client, it must be published with a clearly defined contract outlining its capabilities, as well as input and output interfaces. This is done with the Web Services Description Language (WSDL). WSDL is an XML formatted machine-readable language, designed by Microsoft and IBM.[9] It is used for describing how the service can be called, what parameters the service expects, and what kind of data is re-turned. It serves a similar purpose as a method signature in a conventional programming language. [6]

A WSDL document has 4 main sections [10]:

Types: The data types used by the web service
Message: The messages used by the web service
Port Type: The operations performed by the web service.
Binding: The communication protocols used by the web service.
The Types section defines what data types will be used in the web service. It simply defines variable names and assigns a type such as string or int to them. Variables can be any complex data type that has been defined in XML. Otherwise types must be either a primitive data type, string or an array of primitives.
The Message section defines how the input and output messages are formatted. For each message that will be sent or received by the service, the WSDL document gives the message a name, and declares what parameters will be sent.
The Port Type section declares what operations can be per-formed by the service. Each entry in the Port Type section defines a name for an operation, the message that is needed for input, and the message that will be sent as output. Operations don't always have a request-response structure.
The Binding section defines the message protocol details for a web service. Each Binding entry defines how an operation from Port Type will be accessed using SOAP.

C. SOAP
In order to send and receive the messages defined in WSDL, a standard was needed to allow Internet transfer of these messages over HTTP. SOAP is one standard that can be used. Originally an acronym for Simple Object Access Protocol, It is no longer referred to as an acronym, and is now simply the soap protocol. [9] Soap messages are ordinary XML documents that are sent over HTTP. Each SOAP message contains 4 main sections: [9] Envelope: A small section identifying the document as a SOAP message Header: Small section that carries relevant header information. Body: Contains the call and response information. Fault: An element that defines errors and status information. The Envelope element is the root XML element of the message. It has a field to identify the XML as a SOAP message, and another field, containing a URI that details the SOAP encoding used in the message.

The Header section contains some optional application specific information such as authentication and payment. The header section is not mandatory, but it does have a field that can be set to force whatever system receives the message to parses it.

The SOAP Body element contains the actual message that is being sent. The client or service must format this message in a way

that is described in the WSDL document. The Fault element is an optional section that indicates error messages. SOAP has some default fault codes that can be sent to indicate whether the problem was on the client side with a malformed message, or on the server side, where the service couldn't respond for any number of reasons.

D. Error in ESB layer

Enterprise Service Bus (ESB) layer is at the core of typical enterprise SOA stack. This layer supports the transformation and routing capabilities required off of the enterprise re-usable services. Components in this layer provide a well-defined interface to **t**he various provider implementations such as existing underlying assets and partner or vendor-based services, by applying appropriate message and protocol transformations. Error handling by the mediation components mostly involves transforming the provider error structures into well-defined error structures defined in the context of business domain. These components also could handle applying some complex transformation and mapping rules on the errors returned from the backend functional components to provide more simplified error info to the service consumers within the enterprise.

E. Error Handling Technique in ESB Layer:

A lot of error handling considerations mentioned for this layer is also possible to be implemented in the component layer. But there are number of ESBs and frameworks in the market that does these things in a lot more configurable and flexible manner than what individual platform developers could implement in their functional component implementations. Separation of such error handling mediation concerns to ESB layer relieves the platform developers from having to satisfy a variety of error handling consideration and have them focus more on implementing the business functionality resulting in greater developer productivity.

4. RESERCH METHODOLOGY

Enterprise Service Bus (ESB) layer is at the core of typical enterprise SOA stack. This layer supports the transformation and routing capabilities required off of the enterprise re-usable services. Components in this layer provide a well-defined interface to **t**he various provider implementations such as existing underlying assets and partner or vendor-based services, by applying appropriate message and protocol transformations. Error handling by the mediation components mostly involves transforming the provider error structures into well-defined error structures defined in the context of business domain. These components also could handle applying some complex transformation and mapping rules on the errors returned from the backend functional components to provide more simplified error info to the service consumers within the enterprise.

5. DISTRIBUTED APPLICATION TOOLKITS

A. Net IDE

Handling Error for SOA application used these steps:

1) Creating a service application.

For creating a service application used following steps, they are:

a) To establish contract between client & the service, use WCF Service Contract & include Service Contract, Operation Contract & Data Contract (i.e. Interface). All these contracts can include by adding the reference System.ServiceModel
b) Add WCF Methods/ Operation Contract
c) Create Class Library and add references Service Model & Contract Library.
d) Add one more class which will be the implemented class which is inherited from interface.
e) Add one more class Account.
f) Create Service Provider & add references, Service Model, Contract Library & Implementation.
g) To publish end point, we need App.config which can be created using MS service configuration editor i.e. WCF service configuration editor.
h) Create client application.

App.config:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
<system.serviceModel>
<client>
<endpoint address="http://localhost:9095/BankService"
binding="wsHttpBinding"
bindingConfiguration="" contract="BankContractLib.IBank" name="BankEP" kind="" endpointConfiguration="">
</endpoint>
</client>
</system.serviceModel>
</configuration>
```

2) Checking the exception in service

a) The exception in SOA architecture is BasicHttpBinding.

b) To identify, AccID is valid or not, & see the exception on client side. This exception message is very big.

3) Handling the exception

a) The BasicHttpBinding exception is resolve using wsHttpBinding. BasicHttpBinding is a stateless protocol, it doesn't store any previous state. Therefore, change the binding to wsHttpBinding, which is meant for statefull binding.

b) To avoid big message, log it on server side.

B. NetBeans IDE/Java System Application Server bundle

Web services are Web-based enterprise applications that use open, XML-based standards and transport protocols to exchange data with calling clients. NetBeans IDE1, which is a powerful integrated development environment for developing applications on Java platform, supports web services technologies through the Java Platform Enterprise Edition. Sun Java System (SJS) Application Server2 is the Java EE implementation at Sun Microsystems. NetBeans IDE with SJS Application Server support JSR-109, which is a development paradigm that is suited for J2EE development, based on JAX-RPC. NetBeans IDE provides wizards to create web services and web service clients[4].

C. IBMWebSphere Software Developer Kit for Web Services

IBMWebSphere Software Developer Kit for Web Services (WSDK) is an integrated kit for creating, discovering, invoking, and testing Web services. WSDK can be used with the Eclipse IDE. Eclipse provides a graphical interactive development environment, which provides tools for building and testing Java applications. WSDK adds to the standard Eclipse package the tools relating to Web services, making it suitable for building Web services. Supporting the latest specifications for Web services including WS-Security, SOAP, WSDL, and UDDI, WSDK enables to build, test, and deploy Web services on industry-leading IBM WebSphere Application Server. Functionality of the WSDK has been incorporated into the IBMWebSphere Studio family of products [4].

6. CONCLUSIONS

Achieving a high level of security in the banking domain is a challenging topic. In this paper discussed the SOA conceptual architecture and promotes SOAM, which contains two major ideas that should be applied in the banking domain to increase the security level. The integrated Enterprise Service Bus (ESB), will allow users to combine their bank transactions into a single platform. Besides that, the ongoing updates for different transactions will be handled by the integration of ESB.

7. REFERENCES

[1] Ardi Suryatmojo,Emil R Kaburuan, Ahmad Nurul Fajar, Sepsellona Sutarty, Abba Suganda Girsang, "Financial Technology Integration Based on Service Oriented Architecture", Conference Paper · October 2018

[2] Mahmoud. Q , "The Road to Enterprise Application Integration (EAI) ", Service-Oriented Architecture (SOA) and Web Services published April 2005, Available: http://www.oracle.com/technetwork/articles/javase/soa-142870.html, [Accessed: 30 Apr 2014].

[3]. Ceccato, M., C.D. Nguyen, D. Appelt and L.C. Briand, 2016. SOFIA: An automated security oracle for black-box testing of SQL-injection vulnerabilities. Proceedings of the 31st IEEEACM International Conference on Automated Software Engineering, Sept. 03-07, ACM, Singapore, pp: 167-177. DOI: 10.1145/2970276.2970343

[4]. Early, A. and D. Free, 2002. SOA: A 'must have' for core banking (ID: SPA-17-9683).

[5]. Erl, T., 2009. SOA Design Patterns. 1st Edn., Prentice Hall, Upper Saddle River, ISBN-10: 0136135161, pp: 814.

[6]. Hariharan, C. and C. Babu, 2014. Security testing of orchestrated business processes in SOA. IEEE International Conference on Advanced Communications, Control and Computing Technologies, May 8-10, IEEE Xplore Press, Ramanathapuram, India, pp: 1426-30. DOI: 10.1109/ICACCCT.2014.7019337.

[7]. Inaganti, S. and S. Aravamudan. 2008. Testing a SOA application. BPTrends.

[8]. Kajtazovic, N., A. Höller, T. Rauter and C. Kreiner, 2017. A lightweight framework for testing safety-critical component-based systems on embedded targets.

[9]. Keen, M., R. Kaushik, K. Singh, B.A. Aghara and S. Simmons *et al*., 2017. Case study: SOA banking business pattern.

[10]. Last, D., 2015. Using Historical Software Vulnerability Data to Forecast Future Vulnerabilities. Proceedings of the Resilience Week, Aug. 18-20, IEEE Xplore Press, Philadelphia, PA, USA, pp: 1-7. DOI: 10.1109/RWEEK.2015.7287429

[[11] Baskerville, Richard. (2005), Extensible Architectures: The Strategic Value of Service-oriented Architecture in Banking, Department of Computer Information Systems, Robinson College of Business, Georgia State University.