SECURITY TESTING FRAMEWORK FOR SOA MIDDLEWARE IN BANKING DOMAIN

Ms. Bhargavi Wakhre, Prof. Hirendra Hajare
MTech Student, Assistant Professor
Department of CSE, Ballarpur Institute of Technology (BIT), Ballarpur.

ABSTRACT

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 analyzes 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, analyzing and fixing vulnerabilities and also forecasting new vulnerabilities and attacks in all banking SOAM layers.

Keywords: SOA Middleware, BPEL, Automation Security Testing Framework, Orchestrated Business Process, SOAP Protocol, Secure Banking Architecture.

INTRODUCTION

Service-Oriented Architecture Middleware (SOAM) is considered a paradigm shift in designing banking infrastructure that allows core banking to integrate heterogeneous and legacy systems or e-services to innovate a bank’s service delivery and guarantee efficient business process management and orchestration. In the banking domain, the real competition is to stay innovative, competitive and cost-effective and to fulfil customers’ needs, in order to achieve greater profitability. SOAM is an open, extensible and compassable architecture that provides services interoperability and reusability. Furthermore, SOAM allows enterprises to apply the agile methodology and become flexible during their application development cycle (Hariharan and Babu, 2014).

Banking solutions are rarely implemented in a standalone way and must have the ability to securely, inter-operate and integrate different data sources and their systems such as mobile banking, internet banking, electronic check clearing (ECC), ATMs and others. Each of these e-services will have been built by various vendors using different programming languages such as .NET and JAVA, so managing and controlling these eservices is a serious challenge for any bank, especially when the bank wants to achieve service quality, high performance, availability, customer’s satisfaction and a high level of security. Thus, having an integration layer is a must in the banking domain: This is SOAM. The SOAM integration layer allows banks to deploy new services quickly and with lower cost (Early and Free, 2002; Keen *et al*., 2017). The study in this study is based on a real example of an SOAM banking project. A similar approach may be taken in another integration projects to an extent depending on the e-services and the requirements of the banking system.

The rest of the paper is organized as follows. Section 2 discusses the literature review and their considerations. An overview of service-oriented architecture is presented in section 3. Section 4 discusses the motivation of the current study. The proposed SSOAM architecture design and its related work mechanism is presented in section 5 and section 6 discusses SSOAM security analysis and evaluations then finally section 7 contains the conclusion and suggestions for future work.

LITERATURE REVIEW

With regard to SOAM security there are many suggested plugins, patterns, framework and automated tools for achieving a high level of security. The most popular was proposed by Erl (2009). This considers data confidentiality and data origin authentication, which is used for message security level; it also considers direct and brokered authentication that uses an access control concept to allow specific users to access services with special permissions. However, the author did not study security issues related to the communication channel or data encryption pattern and did not consider the idea of building a secure banking architecture by using SOAM (Erl, 2009).

There are groups of commercial tools that execute security testing for SOAM, although they actually have several weaknesses and issues:

- Executing traditional security testing on SOAM technology may force the project to run over budget
- Traditional security testing does not work in parallel with the banking production environment
- Traditional security testing tools do not cover all SOAM layers

Service Oriented Architecture Overview

SOAM is a group of related services and applications that communicate with each other to achieve certain advantages for enterprises. It is a client-server architecture which contains two major concepts services and consumers. SOAM has additional features to the traditional client-server approach such as a loose coupling feature between service components and separately standing interfaces (Natis, 2003). Additionally, SOAM architecture supports banks to innovate and maintain their business processes and strengthen their IT infrastructure (Early and Free, 2002; Bhuvaneswari and Jujatha, 2011).

SOA Architecture Layers

In general, SOAM architecture contains the following major layers. (The number refer to items in Fig. 1).

Application and System Layer [1]

This layer contains all the connected bank's eservices such as the internet, mobile banking, ECC, ATMs and others. These published systems serve the bank's clients through the internet (HTTPS secure URL) (Oracle, 2017).

Service Orchestration Layer [2]

This layer is located in the middle of the SOAM architecture and is considered the source of functionalities containing all the common and unified bank's business rules and operations. For example, a request for transaction is a common function between internet banking, mobile banking and others, so it will be defined and upgraded and will then be used by all other e-services (Bhargavan *et al*., 2004; Lowis and Accorsi, 2009; Ye and Yang, 2013).

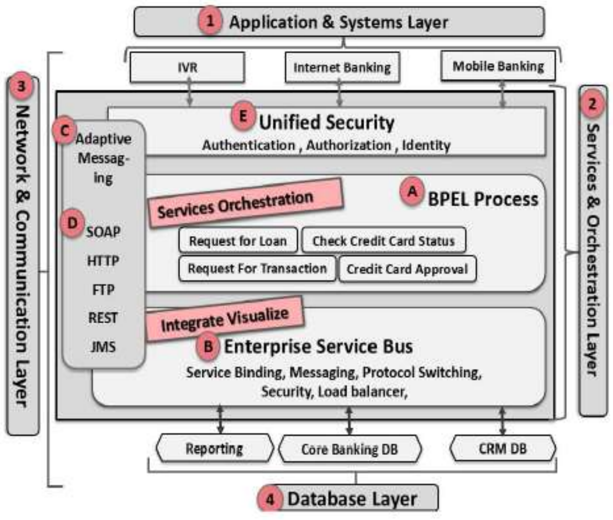

Fig. 1: SOA conceptual architecture

Network and Communication Layer [3]

This layer works in parallel with all other layers that are transferring messages and binding services through SOAP or any other protocols.

Database Layer [4]

This layer is considered as the source of truth. It is main purpose is to obtain all required information for the bank's e-services such as user transactions, client details, client account, credit card status, history and others.

SOA Architecture Components

It is important to highlight the major components of SOAM conceptual design. These are as follows (The letters refer to items in Fig. 1).

Business Process Execution Language (BPEL) [A]
BPEL is an XML based mark-up language that allows services within SOAM to interconnect, inter-operate and share both common business rules and data between core banking and e-services. It enables easy and flexible configuration for services into business processes and has recently become a major component in SOAM. BPEL is described as a WSDL (Oracle, 2017).

Enterprise Service Bus (ESB) [B]
ESB is the core of SOAM architecture. It is considered as a single point of entry for all connected eservices and systems. Each system needs to communicate with ESB to influence all other e-services and each system is required to create special interface to interact and connect directly with ESB instead of creating several interfaces to communicate with many other systems (Oracle, 2017).

Adaptive Messaging [C]
Adaptive messaging provides a communication pattern between orchestrated services and also between all clients and available e-services, such as requestor response, both synchronous and asynchronous. It helps in handling messages and manipulate them between different services through different protocols: SOAP, HTTP, REST and others (Oracle, 2017).

Simple Object Access Protocol (SOAP) [D]
SOAP is an exchange messaging framework within different web services and systems. SOAP is the heart of web services schema, while web services are loosely coupled software components and standard methods for connecting web-based applications such as internet banking to the core-banking system by using XML, SOAP, WSDL and Universal Description, Discovery and Integration (UDDI) over Internet protocol backbone.
XML: For tagging the data
SOAP: A message format for communication between systems involved in a web-service for transferring the data.
WSDL: A mechanism for describing the services available
UDDI: For listing what services are available within the architecture

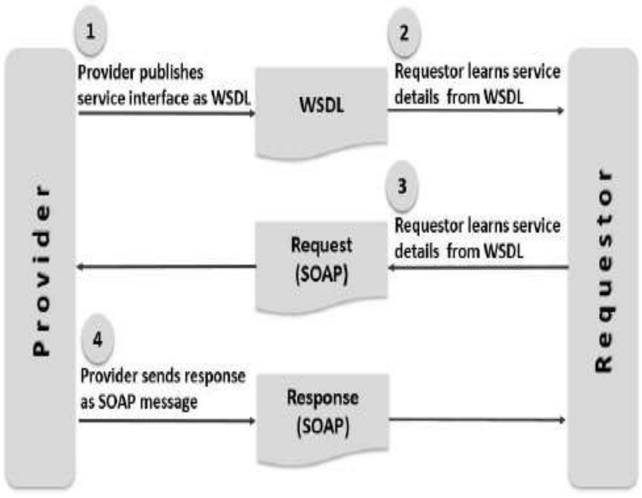
1
Provider publishes service interface as WSDL
WSDL
2
Requestor learns service details from WSDL
3
Requestor learns service details from WSDL
Request (SOAP)
4
Provider sends response as SOAP message
Response (SOAP)
Provider
Requestor

Fig2: SOAP Protocol

SOAP is the master leader in communications. Its main purpose is to send and receive XML information to allow a successful communication between service requestor, service registry and provider. As shown in Fig. 2, SOAP is used to publish the descriptor into a service registry to:

- Send a service request from a requestor to the registry
- Send information from the registry to the requestor;
- Allow the requestor to bind to the service provider and run the web service (Mainka *et al*., 2012; Al-Jaroodi and Al-Dhaheri, 2011; Bhargavan *et al*., 2007)

It is important to mention that SOAP is essential for all web services and communication between banking eservices (Hariharan and Babu, 2014; Lux et al., 2005; Keen et al., 2017).

SOA Middleware Unified Security [E]
SOAM unified security is responsible for establishing trust or handshaking relationship between two entities, such as core banking and mobile banking. In practice, in order to access secured e-services it must first provide information that express its origin of issuing or asserting.

This process is referred to as making a claim, it may happen by providing credentials which are stored in a security token to allow claims to be asserted by sender authentication, to establish the identity service consumer by confirming this claim to be a true claim (Al-Jaroodi and Al-Dhaheri, 2011; Bhargavan *et al*., 2007):

- Authenticating means determining what the requester is allowed to do and authorization typically occurs after authentication.
- Authorizing the consumer will require verifying the identity claim against predefined setup access rules.
- Confidentiality is focused on protecting the privacy of information and making sure this information is available only to the authorized services when a message is being transmitted. A confidentiality mechanism is responsible for protecting the content throughout the message path.
- Integrity is responsible for protecting the message from any alteration by unauthorized parties. The integrity mechanism ensures that the message remains un-attacked since its creation and transmission by the original sender (Bhargavan *et al*.2007; Lowis and Accorsi, 2009; Keen *et al*., 2017)

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.

REFERENCES

[1]. 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

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

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

[4]. 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.

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

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

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

[8]. 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