Security for SOAP based Communication among Web Services

R. KISHORE KUMAR
PG Scholar
Dept. of Computer Science
SSN College of Engineering
Anna University, Chennai, India
Kishore1115@cse.ssn.edu.in

R. KANCHANA
Assistant Professor
Dept. of Computer Science
SSN College of Engineering
Anna University, Chennai, India
rkanch@ssn.edu.in

CHITRA BABU
Professor
Dept. of Computer Science
SSN College of Engineering
Anna University, Chennai, India
chitra@ssn.edu.in

ABSTRACT
Several companies and individuals are increasingly dependent on the electronic means to provide and consume services. A single service may not fulfil the requirement of a service consumer and hence, composition of web services which together satisfy the requirements is essential. In service provisioning, security attacks such as message alteration attack may happen at the service level or at the composition level. It is required to ensure secure data flow without information leaking through covert channels. Most of the existing works on web services security provide solutions for ensuring only confidentiality, client authorization, and integrity of information. Hence, it is proposed to design pluggable APIs that protect SOAP messages, from service based threats while accessing a service and during service composition. These APIs are generic in nature and can be installed on the server where the services are deployed. A generic API that protects the SOAP messages from message alteration attacks is proposed in this paper.

Keywords
MAA API, SOAP, Web Service Attacks, Web Services, XML Encryption.

1. INTRODUCTION

Service Oriented Architecture (SOA) [1] involves a set of principles and methodologies for designing and developing software in the form of interoperable web services. A web service is a standard way of integrating web-based applications using the XML, SOAP, WSDL, and UDDI open standards over an Internet protocol. A web service is self-contained and self-described. These services are well-defined business functionalities that are built as software components (discrete pieces of code and/or data structures) which can be reused for different purposes. The web services communicate using open protocols like SOAP (Simple Object Access Protocol). The SOAP is a protocol specification for exchanging structured information among web services. It relies on eXtensible Markup Language (XML) for its message format, and on Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.

Web services make it possible to achieve interoperability, by interconnecting services offered by multiple business partners based on the business process. This interconnection of web services to realize a certain business process is called web service composition. Existing web services are combined together on the basis of the rules of composition to meet a demand which cannot be realized by a single service alone.

Security is an important issue while exchanging the business data among web services. The security challenges [2] presented by the web services approach are formidable and unavoidable. Since web services enable grater accessibility of data, dynamic connections and relatively less human interventions, ensuring confidentiality and integrity of data that is transmitted via web services protocols gains more importance. Moreover, availability of services in the phase of denial of service attacks must be ensured. Perimeter-based network security technologies (e.g., firewalls) are inadequate to protect SOA based applications for the following reasons:

- SOA based applications are dynamic and can never be fully constrained to the physical boundaries of a single network.
- SOAP is transmitted over Hypertext Transfer Protocol (HTTP), which is allowed to flow without restriction through firewalls.

Transport Layer Security (TLS), which is used to authenticate and encrypt Web-based messages, is inadequate for protecting SOAP messages because it is designed to operate between two endpoints. TLS cannot accommodate the inherent ability of web services to forward messages to multiple other web services simultaneously. The web service processing model requires the ability to secure SOAP messages and XML documents as they are forwarded along potentially long complex chains of consumer, provider, and intermediary services. The nature of Web services processing makes those services subject to unique attacks, as well as variations of familiar attacks targeting Web servers.

This paper, focuses on securing SOAP messages as they are forwarded along services. The rest of the paper is organized as follows, Section 2 discusses the existing work, Section 3 summarizes the drawbacks of the existing system and the motivation behind the proposed approach. Section 4 discusses the SOAP message communication scenario. Section 5 describes the proposed approach and Section 6 concludes the paper.

2. LITERATURE SURVEY

Web services rely on the Internet for communication. Since SOAP was not designed with security, SOAP messages can be viewed or modified by attackers as the messages traverse the Internet [2]. Security decisions must always be made with an understanding of the threats facing the system to be secured. While there are a wealth of security standards and technologies available for securing web services, they may not be adequate since, they need to be configured and customized for each web server or organization. Hence, it is important to understand the threats that face the web services. According to WS-I [2], the top threats facing web services are:

Message alteration: An attacker inserts, removes or modifies information within a message to deceive the receiver.

Loss of confidentiality: Information within a message is disclosed to an unauthorized individual.

Falsified messages: Fictitious messages that an attacker intends the receiver to believe are sent from a valid sender.

Man in the middle: A third party sits between the sender and provider and forwards messages such that the two participants are unaware, allowing the attacker to view and modify all messages.

Principal spoofing: An attacker constructs and sends a message with credentials such that it appears to be from a different, authorized principal.

Forged claims: An attacker constructs a message with false credentials that appear valid to the receiver.

Replay of message: An attacker resends a previously sent message.

Replay of message parts: An attacker includes portions of one or more previously sent messages in a new message.

Denial of service: An attacker causes the system to expend resources disproportionately such that the valid requests cannot be met.

Few authors have proposed security solutions for service-oriented applications.

Abdelkader et al. [3] proposed a comprehensive Quality of Security Service (QoSS) model for addressing security within a Service-Oriented Architecture (SOA). It utilizes symmetric keys, public keys and hash functions techniques, in order to provide different levels of QoS agreements to satisfy the requirements of both the services providers and requesters. This model addresses only core networking security requirements such as mutual authentication, session keys, anonymity, and perfect forward secrecy. Hence, it is suitable to handle only network level attacks such as Replay, Man-in-the-Middle, and Denial-of-Services.

Hua Yue et al. [4] discussed the security issues of Web based services on heterogeneous platforms. The authors introduced two different security solutions; WSE 3.0 for Microsoft .Net web services and Rampart module in Apache Axis 2 platform. A third-party certification body was created to realize the security of Web services in heterogeneous platforms. However, this approach is susceptible for replay attacks and uses computationally intensive asymmetric cryptographic algorithms.

3. MOTIVATION

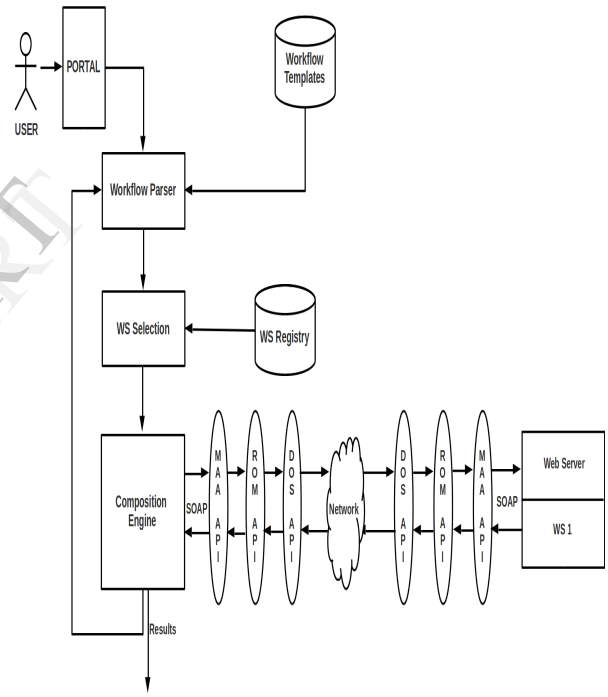
The rapid development of modern information and communication technologies has an impact on Government applications. E-Governance has the capability to ensure greater engagement with citizens, higher productivity in terms of reduced costs, more efficient administrative procedures, and delivery of higher quality services. Security is an important non-functional requirement that is essential for E-Governance applications. Some of the network security technologies like HTTPS and TLS were used for protecting the sensitive information. However, presently, these applications are increasingly built using web services. The network security technologies are inadequate for protecting them as they are designed to operate between two endpoints. The sensitive information travels in the SOAP body while communicating among the web services. These SOAP

messages are prone to some of the unique attacks like message alteration attack, man in the middle attack, and denial of service attack. For example, in a Government Land Registration process, when a buyer buys a land, any third party or the hacker can alter the sensitive information such as buyer details before generating the title deed. Hence, a new approach is proposed to secure web services that involve SOAP message communication from such attacks.

4. SOAP MESSAGE COMMUNICATION

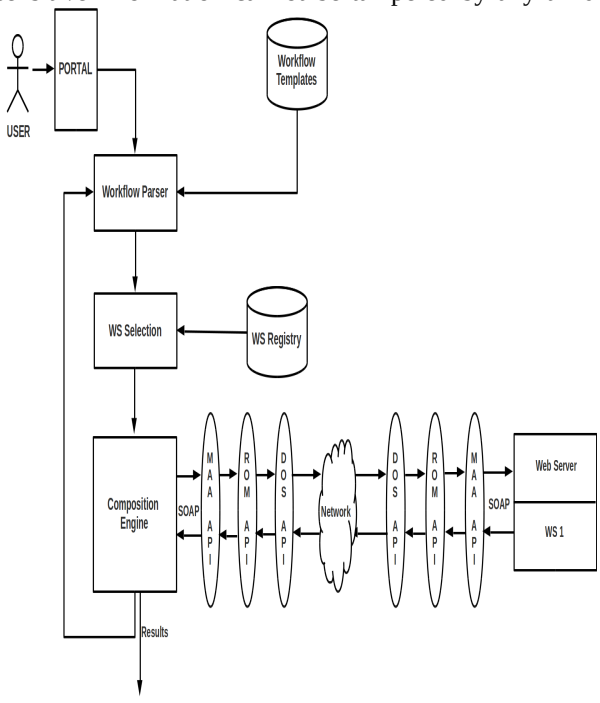
The proposed approach uses template based dynamic composition method [5], since templates are suitable for business applications. The workflow parser parses the workflow template, chosen based on the business scenario. The template specifies the control flow among the activities using workflow patterns. For each activity in the workflow template, the corresponding service is selected dynamically, from the web service Registry. The composite service communicates with the domain web service using SOAP protocol. This SOAP message contains the envelope part, header part, and the body part. The body of the SOAP message contains sensitive information pertaining to the user. This sensitive information is passed in the public network to access the service. In order to protect these SOAP messages, the proposed APIs such as MAA API, ROM API, and DOS API will provide appropriate solutions to ensure secure communication over the internet. The scenario involving SOAP message communication among services and the role of APIs is shown in Figure. 1.

Fig 1: SOAP Message Communication



5. MESSAGE ALTERATION ATTACK API

The Message Alteration Attack API (MAA API) has been proposed for providing security against message alteration attack. The SOAP request message sent from the middleware service is captured and the content of the SOAP message body is encrypted and sent via network to the web service. The API installed on the machine hosting the web service, captures the incoming SOAP message and decrypts it before redirecting it to the web service. Similarly, the response message from the service is captured and encrypted by the API and sent to the middleware service. By this way, the sensitive information cannot be tampered by any third party or hacker. The procedure adapted by MAA API is illustrated using the following algorithm.



Algorithm *MAA API*
Input: SOAP Message

Output: Secured SOAP Message

1. **if** SOAP message = outbound **then**
   intercept SOAP message
   call *Encrypt()*
   save changes in the message
2. **if** SOAP message = inbound **then**
   intercept SOAP message
   call *Decrypt()*

save changes in the message

Algorithm *Encrypt()*

Input: SOAP Message

Output: Encrypted SOAP Message

1. parse SOAP message to identify the element to be encrypted

2. apply DES encryption

Algorithm *Decrypt()*

Input : Encrypted SOAP Message

Output : SOAP Message

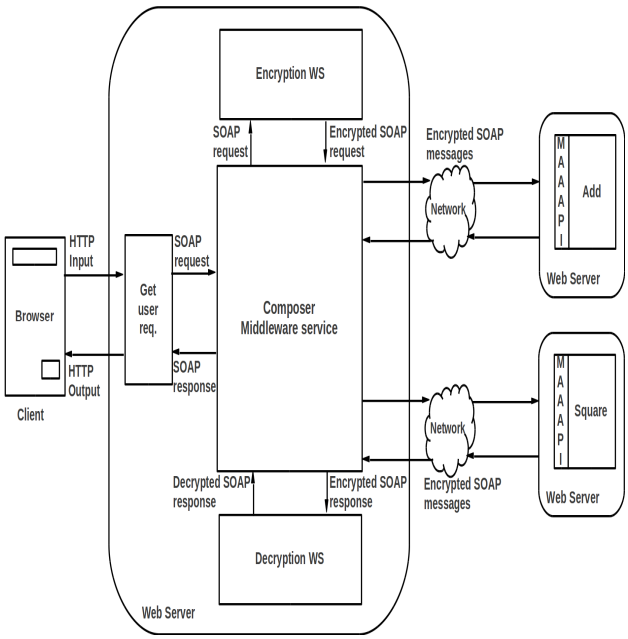
1. parse SOAP body to identify <EncryptedData> element

2. apply DES decryption

5.1 Proposed Approach

The MAA API can handle only one incoming and outgoing SOAP message since it cannot intercept messages in more than one communication path or channel. Hence, MAA API cannot be plugged at the middleware service side, since there are two channels; one in which the composition middleware is invoked by another middleware service and the second one in which the composer invokes several services. Thus,

Fig 2: Propoed Approach



there are more than one communication channels, through which the composer places SOAP request and response messages. Hence, web services are used for encrypting and decrypting outbound and inbound SOAP messages at the middleware service side, for communicating with the domain services. Two simple domain services such as *add* and *square* are considered to find the square of sum of two numbers. The usage of the proposed API in composition scenario is depicted in Figure 2.

In *add* web service, an MAA API is installed to intercept and to decrypt the incoming SOAP message which is then used to invoke the *add* web service. The outgoing SOAP message is intercepted my MAA API again which encrypts it before sending to the middleware service. The server log of middleware service depicting SOAP message communication in the middleware service side is shown in Figure 3. The server log of Glassfish server where add web service is deployed, as shown in Figure 4. depicts the usage of MAA API in encrypting and decrypting the SOAP messages. Similarly, the MAA API is installed on the machine where the various domain services are deployed, to provide protection against the message alteration attack.

While encrypting the SOAP message, if the data values alone are encrypted, its description or the enclosing element name may provide a clue to the hacker about the contents. This enables the hacker to focus on trapping the more sensitive data. Therefore, the whole body of the SOAP message is encrypted using the DES algorithm. The Java program *GetUserReq* that obtains user supplied functional requirements and invokes the composer middleware service resides in the same web server as the composer. The SOAP messages exchanged between *GetUserReq* and *Composer* services do not flow in network and consequently, there is no possibility of message alteration threat. Hence, there is no need for any security API in the middleware service side.

6. CONCLUSION

Middleware services and Web Services are created for demonstrating the secure service communication. The functionality of the middleware service is to obtain the input from the user and generate the SOAP request message and send it to the service side. The proposed MAA API has been plugged into all the machines where the domain web services are deployed. It captures the SOAP request for encrypting the contents. The encrypted message is then sent to the recipient service. The received message is captured by the MAA API installed in the service side for decryption of contents before directing the request to the service. The future work involves creating generic APIs for other

service based security attacks such as Replay of Message Attack and Denial of Service Attack.

7. REFERENCES

[1] Erl, T., Service-Oriented Architecture Concept, Technology, and Design, Pearson Education, 2006.

[2] Singhal, A., Winograd, T., Scarfone, K., Guide to Secure Web Services, Tech report of National Institute of Standards and Technology, Special Publication 800−95, U.S. Department of Commerce, MD, 2007.

[3] Abdelkader, H., David, S., and Miriam, A.M., Security Protocols in Service-Oriented Architecture, IEEE 6th World Congress on Services, pp. 185−186, 2010.

[4] Yue, H., Tao, X., Web Services Security Problem in Service-oriented Architecture, 2012 International Conference on Applied Physics and Industrial Engineering, Physics Procedia, vol. 24, pp. 1635 − 1641, 2012.

[5] Rajaram, K., Babu, C., Template based SOA framework for dynamic and adaptive composition of Web Services, International Conference on Networking and Information Technology, pp. 49-53, 2010.

[6] Han, J., Kowalczyk, R., Khan, K.M., Security-Oriented Service Composition and Evolution, XIII Asia pacific software engineering conference (APSEC'06), pp. 71-78, 2006.

[7] Netbeans tutorial for web service interoperability technologies. http://netbeans.org/kb/docs/websvc/wsit.html.

[8] Menzel, M., Meinel, C., SecureSOA − Modelling Security Requirements for Service-oriented Architectures, IEEE International Conference on Services Computing, pp. 146 − 153, 2010.

[9] Monson, R., The Ultimate Guide J2EE Web Services, Pearson Education, 2011.

[10] XML Encryption Syntax and Processing, http://www.w3.org/TR/xmlenc-core/

R. Kishore Kumar received B.E degree in Computer Science and Engineering from Rajalakshmi Engineering College, Anna University, Chennai, India in 2011. Currently he is pursuing M.E degree in Computer Science and Engineering in SSN College of Engineering.

Kanchana Rajaram is an Assistant Professor at the Department of Computer Science, SSN College of Engineering, Chennai. She holds a M.E. in Computer Science from National Institute of Technology, Tiruchirapalli and currently pursuing her research in Service Oriented Architecture. She has more than 20 years of teaching experience.

Chitra Babu is a Professor and Head of the Department of Computer Science, SSN College of Engineering, Chennai. She obtained PhD in Computer Science from IIT, Madras, Chennai and M.S. in CIS from Ohio State University, USA. She is currently guiding 6 PhD research scholars.