API based Security solutions for Communication among Web Services

A. Kanchana Rajaram, B. Chitra Babu, and C. Kishore Kumar R, *Member, IEEE*

Abstract—**The popularity of web services has largely influenced the way in which enterprise business is conducted. Since web services enable easy accessibility of data, dynamic connections, and relatively less human interventions, ensuring confidentiality and integrity of data that is transmitted via web services protocols becomes more significant. If a single service does not fulfill the service consumer requirements, it is necessary to compose several web services, which together satisfy the user requirements. Security attacks occur on SOAP messages that are communicated among web services while accessing a service or during service composition. Most of the existing works on web services security have provided solutions only for ensuring client authentication, confidentiality, and integrity of information in network layer and not in application layer. WS-Security and XML based web service security also provides message layer security in network layer and not in application layer. Hence, a novel approach that prevents the message alteration attack on SOAP messages and a security solution that detects and overcomes XML injection attack have been proposed in this paper. Our approach uses pluggable APIs in the service provider side and security services in the middleware side. The attacks were simulated and non-vulnerability of the proposed solutions to these attacks have been verified.**

Index Terms—**Web Services, Security, Composition, API, Message Alteration Attack, XML Injection Attack.**

I. 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 way of integrating web-based applications using standards such as SOAP, WSDL, and UDDI. The services are created for a specific business functionality and can be reused under different contexts. The web services communicate with open standard protocol called SOAP (Simple Object Access Protocol) [2].

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. Composition is useful in combining the services to meet a demand which cannot be satisfied by a single service alone.

Security is an important issue while exchanging the business data among web services. The security challenges [3] presented by the web services approach are formidable. Perimeter-based network security technologies (e.g., firewalls) are inadequate to protect web services used in SOA based applications for the following reasons:

- SOAs 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 most 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 security 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 targeting Web servers [3]. Existing web service security tools [4] such as *wsKnight* and *wsAudit* do not handle web service attacks like message alteration and they use only .Net web service frameworks for security assessment. Jensen et al. [5] in their survey of web service attacks, have classified the attacks into non-specific and specific attacks. Non-specific web service attacks such as buffer overflows and SQL injection occur at the back end of an application. Specific web service attacks such as SOAPAction spoofing and XML injection exploit vulnerabilities on SOAP and XML.

In this paper, a novel approach is proposed for securing SOAP message communication among web services. This security solution addresses two specific web service attacks such as message alteration attack and XML injection attack.

The message alteration attack is an attack that inserts,

A. Kanchana Rajaram is with the Computer Science & Engineering Department, SSN College of Engineering, Chennai, (phone: +91 9840623864; e-mail: rkanch@ssn.edu.in).

B. Chitra Babu, is with the Computer Science & Engineering Department, SSN College of Engineering, Chennai, (e-mail: chitra@ssn.edu.in).

C. Kishore Kumar R is with the Information Technology Department, Rajalakshmi Engineering College, Chennai, (e-mail: kishorekumar.r@rajalakshmi.edu.in).

removes or modifies information in a SOAP message body that carries the information. The XML injection attack is an attack that modifies or alters the encrypted information in a SOAP message.

The security solutions have been provided with security APIs plugged into the server of domain web services and security services deployed in the middleware server. The APIs and security services are built generic so that it can be plugged or deployed in any domain or middleware web server for preventing and detecting the attacks.

The rest of the paper is organized as follows, Section 2 discusses the existing literature, Section 3 illustrates the proposed security solution and Section 4 concludes the paper.

II. Existing Work

Web services are susceptible to security threats since; SOAP was not designed with security in mind. SOAP messages can be viewed or modified by attackers as the messages traverse the Internet. 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 are not adequate for a particular organization. For that reason, it is important to understand the threats that face Web services so that organizations can determine which threats their Web services must be secured against [3].

Abdelkader et al. [6] proposed a comprehensive Quality of Security Service (QoSS) model for addressing security within a Service-Oriented Architecture (SOA). This model addresses only core networking security requirements such as mutual authentication, session keys, anonymity, and perfect forward secrecy.

Hua Yue et al. [7] introduced the security issues of Web based services on heterogeneous platforms. The author introduced two security solutions in Microsoft Net and Apache Axis platform. This approach uses WS-Security to provide security solution with asymmetric cryptographic algorithms. The WS-Security has to be configured separately for each communication channel. Moreover, this approach is susceptible to Web Service attacks like XML injection attack, since it addresses only Message integrity, Message confidentiality and Message authenticity.

Mainka et al. [8] developed the first automated penetration testing tool for Web Services called WS-Attacker. This WS-Attacker provides solutions for WS-Addressing spoofing and SOAPAction spoofing attack while invoking a single web service. It does not test the attacks that affect SOAP messages exchanged among web services. Moreover, it does not work for JAX-WS technology, which is common for creating web services.

The security tool TulaFale proposed by Bhargavan K et al. [9] is based on Web Service Enhancement kit (WSE) with .Net Web Services. This tools works only for .Net Web Services and not for other frameworks like, JAX-WS, Axis 2 etc... This tool focuses on authentication properties of SOAP Protocols and not the Web Service attacks.

III. Proposed Security Solution

The shortcomings of the existing works on web services security motivated us to propose solutions for attacks on SOAP messages exchanged among web services. Two attacks have been considered in this paper, namely Message Alteration Attack (MAA) and XML Injection Attack (XIA). In order to demonstrate the proposed solutions, a composition scenario that involves a middleware service and a set of domain web services is considered. The middleware service is a composer which integrates all domain web services. The middleware service communicates with the domain web services using SOAP protocol. This SOAP message contains 3 parts namely *envelope*, *header*, and the *body*. The *body* of the SOAP message might contain sensitive information pertaining to the user and it is passed in the public network to access the domain service. The proposed approach consists of a set of security services in the middleware side and a set of pluggable APIs in the domain service side, as depicted in Fig. 1.

A user logs in to a portal through a web browser developed by a middleware service provider. The input information submitted by the user in the browser passes to the interface program through HTTP or HTTPS. This interface program in turn invokes the middleware service for composition using the SOAP request message and the body of that particular SOAP message contains the information provided by the client. The SOAP message is passed to the *SecurityProvider* middleware service for providing security features which then invokes the domain service.

In order to illustrate the proposed security solutions, a foreign exchange application is considered where two services must be composed for currency exchange. First service, *ConvertCurrency* converts money of one country currency to another and the second service, *CalcCommission*. It computes the commission for the foreign exchange dealer.

A. Security Solution for MAA

The proposed solution consists of an MAAAPI installed in the web server that hosts the domain web service and a set of security services such as encryption and decryption in the middleware web server. The SOAP request message received by the *SecurityProvider* service is encrypted using the encryption web service. The whole body of the request message is encrypted to prevent the hacker knowing the exact position of the sensitive information. A communication channel is acquired and the encrypted SOAP message is communicated to the *ConvertCurrency* service via that channel to invoke the service.

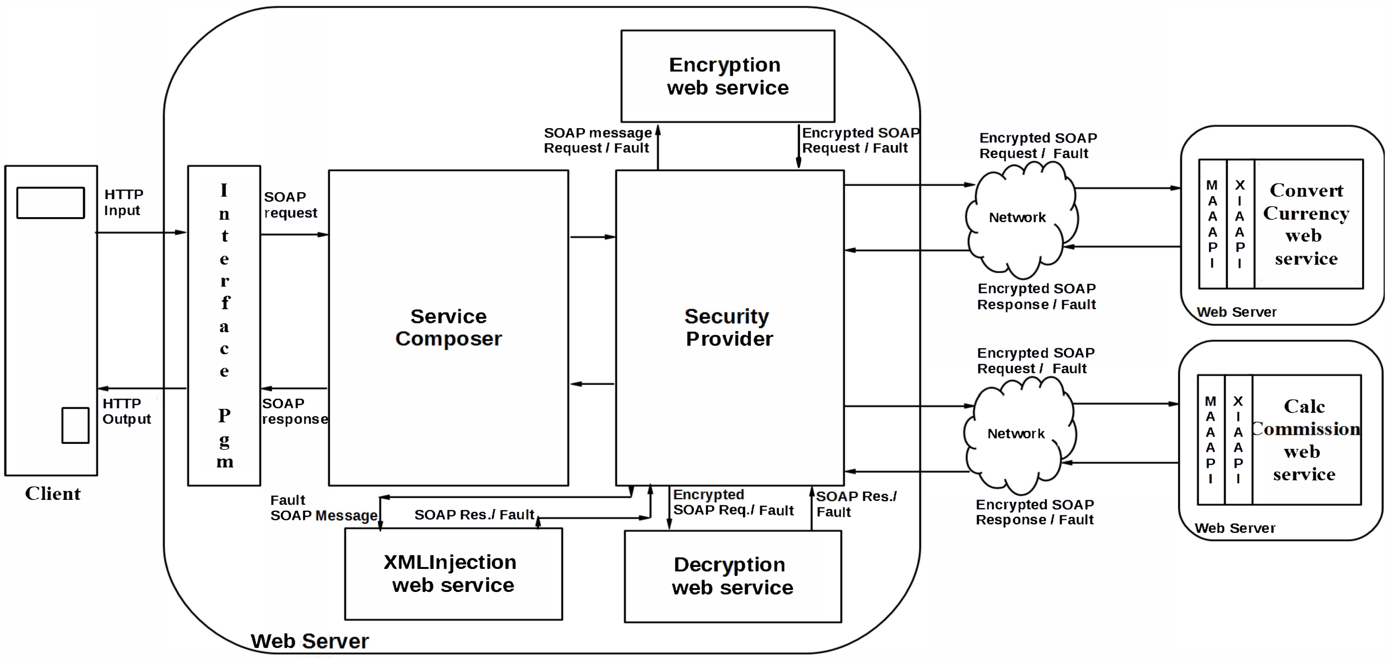
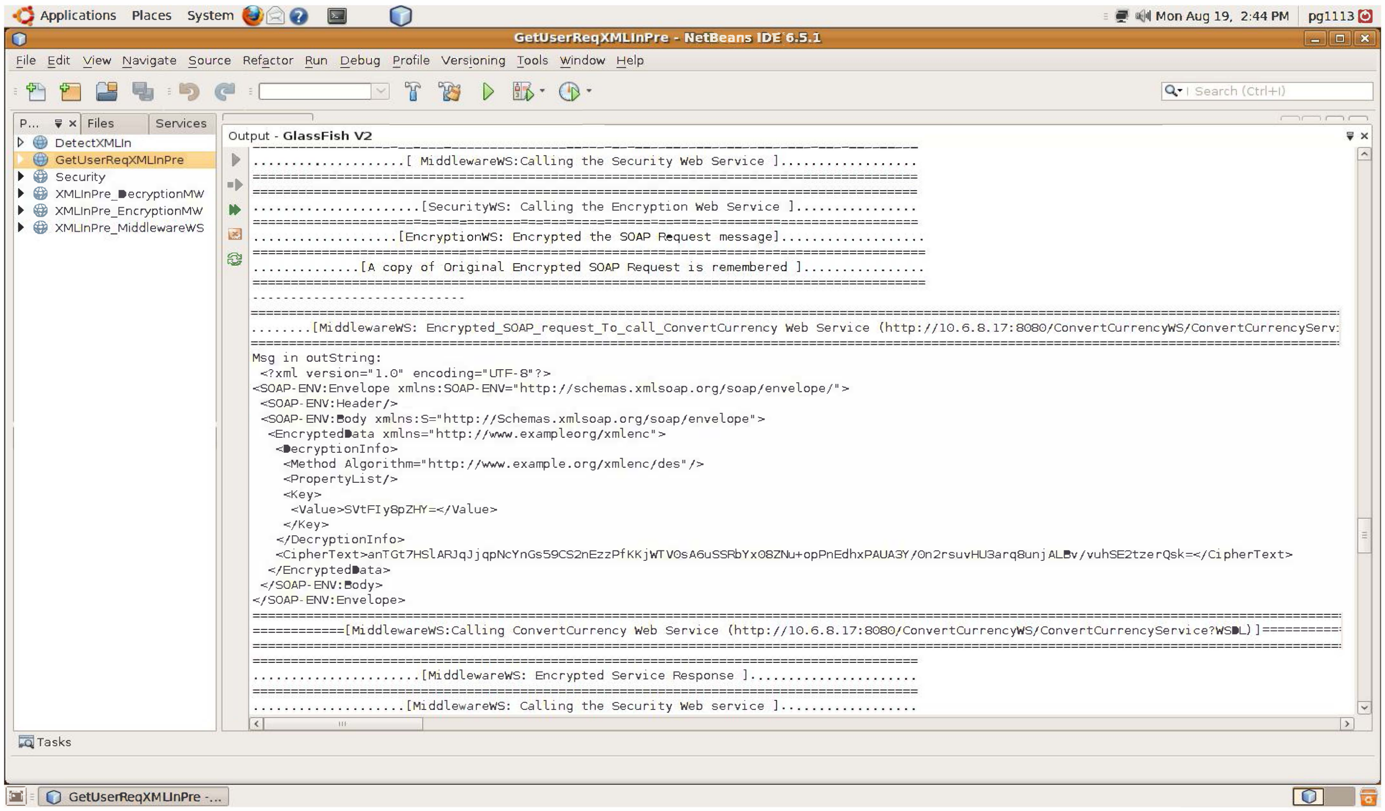


Fig 1. Proposed approach with security services and APIs.

The MAAAPI installed in the web server where *ConvertCurrency* service is deployed, intercepts the encrypted SOAP request message and decrypts it which may then be passed in the same channel to invoke the service. Similarly, after the execution of *ConvertCurrency* service, the response SOAP message passed in the same channel is intercepted by MAAAPI to encrypt and pass on. The security provider in the middleware side decrypts the encrypted SOAP response message using decryption web service and sends it to the composer service which then passes on to the end user. Since the request and response SOAP messages are communicated in the encrypted form, a hacker cannot modify the message contents and thus, message alteration attack is prevented. A snapshot showing the log of the web server in the middleware side is depicted in Fig. 2.



```
------------------------------------------------------------------------------------
....................[ MiddlewareWS:Calling the Security Web Service ]..................
====================================================================================
====================================================================================
......................[SecurityWS: Calling the Encryption Web Service ]................
====================================================================================
...................[EncryptionWS: Encrypted the SOAP Request message]...................
=====================================================================================
.............[A copy of Original Encrypted SOAP Request is remembered ]...............
=====================================================================================
-------------------------
==================================================================================================================================================
........[MiddlewareWS: Encrypted_SOAP_request_To_call_ConvertCurrency Web Service (http://10.6.8.17:8080/ConvertCurrencyWS/ConvertCurrencyServ:
==================================================================================================================================================
Msg in outString:
 <?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Header/>
 <SOAP-ENV:Body xmlns:S="http://Schemas.xmlsoap.org/soap/envelope">
  <EncryptedData xmlns="http://www.exampleorg/xmlenc">
   <DecryptionInfo>
    <Method Algorithm="http://www.example.org/xmlenc/des"/>
    <PropertyList/>
    <Key>
     <Value>SVtFIy8pZHY=</Value>
    </Key>
   </DecryptionInfo>
   <CipherText>anTGt7HSlARJqJjqpNcYnGs59CS2nEzzPfKKjWTV0sA6uSSRbYx08ZNu+opPnEdhxPAUA3Y/0n2rsuvHU3arq8unjALBv/vuhSE2tzerQsk=</CipherText>
  </EncryptedData>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
==================================================================================================================================================
===========[MiddlewareWS:Calling ConvertCurrency Web Service (http://10.6.8.17:8080/ConvertCurrencyWS/ConvertCurrencyService?WSDL)]==========
==================================================================================================================================================
====================================================================================
......................[MiddlewareWS: Encrypted Service Response ]......................
====================================================================================
....................[MiddlewareWS: Calling the Security Web service ]..................
```

Fig 2. Snapshot showing the log of middleware web server.

B. Security solution for XIA

The proposed solution consists of a XIAAPI installed in the web server of domain web server and a XML Injection service deployed in the middleware web server.

After the SOAP request message is encrypted, the XML Injection service remembers a copy. When the encrypted SOAP request message reaches the *ConvertCurrency* service, the MAAAPI intercepts and decrypts the SOAP message and passes in the same communication channel. Next, XIAAPI intercepts the decrypted SOAP message and validates against the SOAP_Request_DTD. If the message is not altered by the hacker, the XIAAPI allows the request to invoke the service.

Otherwise, it sends a *fault* SOAP message to the middleware service to indicate that somebody has hacked or altered the encrypted SOAP request message. This outgoing message is intercepted and encrypted by MAAAPI.

In the middleware side, the *SecurityProvider* service decrypts the received encrypted message and checks whether it is a fault message. In case of a fault message, XML Injection service communicates the remembered copy of encrypted SOAP request message to the domain service side. Otherwise, the received message is the response of service execution and its validity is checked against SOAP_Response_DTD. If the response is not altered in the middle of the communication, the response is passed on to the end user. If the response is hacked, the *SecurityProvider* service generates a fault message using XML Injection service, encrypts it using the encryption service and then passes it to the service side. In the domain service side, the fault message is first intercepted by the MAAAPI to decrypt it and then the decrypted message is intercepted by XIAAPI to send back the remembered copy of the response message. The procedure adapted by XIAAPI is explained using the following algorithm.

```
Algorithm XIAAPI
Input : SOAP Message
Output : Detect XML injected SOAP Message
1. if SOAP message = inbound
     then Intercept the SOAP message
       if fault message
       then send the copy of SOAP message to MW Service
           else if (!DomValidateDTD(SOAP Message))
                then send fault message to MW Service
                else send SOAP message to the Domain
                    Service
2. if SOAP message = outbound
     then Intercept the SOAP message
         remember a copy of SOAP message
         send SOAP message to MW Service.
```

Thus, when the encrypted SOAP request (response) message is altered using XIA, the XML injection service and XIAAPI together detects the attack and sends a copy of the request without regenerating it in the middleware side (without re-executing the domain service).

C. Experimentation

In order to test the feasibility of the proposed security solutions, domain services in the foreign exchange application and the middleware services for composition and security were deployed on a GlassFish Server V2 on 2.4 GHz machines with Intel Core2 Duo processor and 4 GB RAM.

These machines are connected in the college Intranet. The services were developed using JAX-WS framework with J2EE technology in NetBeans IDE 6.5. In order to control and handle specific communication channels, the SOAP messages were created using SAAJ 1.3 technology. The SOAP messages are intercepted using SOAP Message Handlers 1.2. The SOAP messages are encrypted and decrypted using XML Encryption and Decryption 1.1.

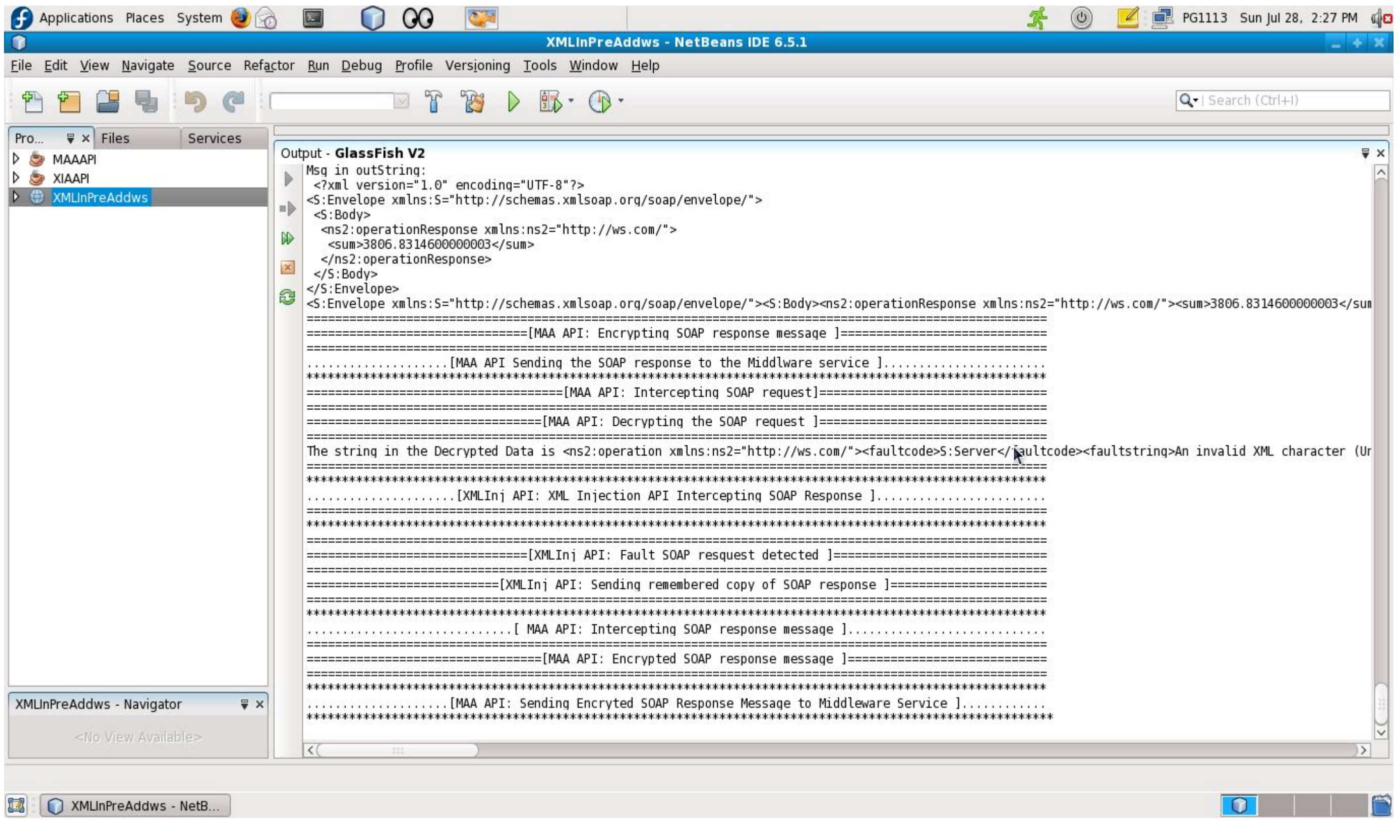
The Messages communicated between web services cannot be altered by a hacker since the entire body of the message is encrypted and the hacker may not know the position of sensitive information. The vulnerability of the proposed security solutions against the XML Injection attack was verified by simulating the attack. The attack was simulated by parsing the SOAP message using DOM parser and altering the cipher text randomly. It was verified that the proposed security solution is able to detect the attack and overcome by retransmitting the original message.

The functioning of XIAAPI is illustrated in the snapshot of Fig. 3 showing the log of web server in the domain service side. In Fig. 3, the XIAAPI retransmits the original response message as it was altered by the XML Injection attack.

IV. Conclusion

The preventive security solution that encrypts all the outbound messages has been implemented that prevents the message alteration attack. A detective security solution that detects and solves the XML Injection attack has also been implemented and tested. The security solutions allow deploying the security web services at the middleware side and installing pluggable APIs at the domain service side. The Security APIs are generic and pluggable in any web server where the domain web services are deployed. The proposed approach is generic, since, any number of APIs and security services can be added in the domain service side and middleware side respectively, for other web service threats.

The future work involves designing a descriptor file at the Middleware side to adopt to the features according to the security requirements. More generic APIs can be designed for other service based security attacks like Replay of Message Attack, Denial of Service Attack, etc...



```
Msg in outString:
 <?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns2:operationResponse xmlns:ns2="http://ws.com/">
   <sum>3806.8314600000003</sum>
  </ns2:operationResponse>
 </S:Body>
</S:Envelope>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"><S:Body><ns2:operationResponse xmlns:ns2="http://ws.com/"><sum>3806.8314600000003</sum
=========================================================================================================
===============================[MAA API: Encrypting SOAP response message ]==============================
=========================================================================================================
....................[MAA API Sending the SOAP response to the Middlware service ]........................
*********************************************************************************************************
====================================[MAA API: Intercepting SOAP request]=================================
=========================================================================================================
=================================[MAA API: Decrypting the SOAP request ]=================================
=========================================================================================================
The string in the Decrypted Data is <ns2:operation xmlns:ns2="http://ws.com/"><faultcode>S:Server</faultcode><faultstring>An invalid XML character (Ur
=========================================================================================================
*********************************************************************************************************
.....................[XMLInj API: XML Injection API Intercepting SOAP Response ].........................
=========================================================================================================
*********************************************************************************************************
=========================================================================================================
===============================[XMLInj API: Fault SOAP resquest detected ]===============================
=========================================================================================================
============================[XMLInj API: Sending remembered copy of SOAP response ]======================
=========================================================================================================
*********************************************************************************************************
..............................[ MAA API: Intercepting SOAP response message ]............................
=========================================================================================================
=================================[MAA API: Encrypted SOAP response message ]=============================
=========================================================================================================
*********************************************************************************************************
....................[MAA API: Sending Encryted SOAP Response Message to Middleware Service ].............
**********************************************************************************************************
```

Fig 3. Domain service web server log showing the functioning of XIAAPI.

REFERENCES

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

[2] Simple Object Access Protocol (SOAP). Version 1.1, W3C Recommendation. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, 2000.

[3] Singhal A., Winograd T., and 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.

[4] Shah S. *Hacking Web Services*. Career and Professional Group A part of cengage Learning, 2007.

[5] Jensen M., Gruschka N., and Herkenhoner R. *A Survey of Attacks on Web Services*. Journal Computer Science - Research and Development (CSRD), Vol. 24, pages 185-197, 2009.

[6] Abdelkader H., David S., and Miriam A. M. *Security Protocols In Service-Oriented Architecture*. In IEEE 6th World Congress on Services, pages 185 – 186, 2010.

[7] Yue H and Tao X. *Web Services Security Problem in Service-oriented Architecture*. In International Conference on Applied Physics and Industrial Engineering, Vol. 24, pages 1635 – 1641, 2012.

[8] Mainka C., Somorovsky J., Schwenk J. *Penetration Testing Tool for Web Services Security*. In IEEE Eighth World Congress on Services (SERVICES 12), pages 163 – 170, 2012.

[9] Bhargavan K., Fournet C., Gordon A. D., and Pucella R.. *TulaFale: A Security Tool for Web Services*. In International Symposium on Formal Methods for Components and Objects (FMCO'03), LNCS Vol. 3188, pages 197-222, 2004.

[10] Netbeans IDE. Version 6.5, Oracle Corporation. https://netbeans.org/kb/index.html, 2008.

[11] Oracle Corporation. JAX-WS - Message Handler. http://docs.oracle.com/javaee/5/tutorial/doc/bnbhr.html, 2008.

[12] XML Encryption Syntax and Processing.. W3C Recommendation. http://www.w3.org/TR/xmlenc-core/, 2002.

A. 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.

B. 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.

C. Kishore Kumar R is an Assistant Professor at the Department of Information Technology, Rajalakshmi Engineering College, Chennai. He holds a M.E. in Computer Science from SSN College of Engineering, Chennai and B.E in Computer Science from Rajalakshmi Engineering College.