International Journal of Latest Trends in Engineering and Technology
Vol.(9)Issue(4), pp.192-202
DOI: http://dx.doi.org/10.21172/1.94.29
e-ISSN:2278-621X

SECURING WEB SERVICES FROM DOS ATTACK BY SOAP MESSAGE VALIDATION

Arti A. Naik[1] & Amit Patil[2]

Abstract: Web Services become more and more popular, not only inside closed intranets but also for inter-enterprise communications, few efforts have been made so far to secure a Web Service's availability. Existing security standards like e.g. WS-Security only address message integrity and confidentiality, and user authentication and authorization. This paper presents a system for protecting Web Services from Denial-of-Service (DoS) attacks. DoS attacks often rely on long messages that engage a server in resource-consuming computations. Therefore, a suitable means to prevent such kinds of attacks is the full validation of messages by an application level gateway before forwarding them to the server. This show how messages can automatically be derived from formal Web Service descriptions (written in the Web Service Description Language).
Keywords: Web Services, SOAP Security, DoS Attack, Cryptography, WSDL.

1. INTRODUCTION

As Web Services become more and more popular, not only inside closed intranets but also for inter-enterprise communications, security is becoming crucial for operating Web Services. While the basic Web Service specifications themselves do not address any security topics, a large number of additional specifications (WS-Security, WS-Security Policy, WS-Trust, WS-Secure Conversation for Web Services security exists. However, all these standards focus on the aspects of message integrity and confidentiality and user authentication and authorization. Few efforts have been made so far to secure the Web Service server itself and ensure a Web Service's availability. Of course, traditional perimeter protection systems like packet filters, application level gateways, and intrusion detection systems contribute to this, but we will show that these are unable to secure a Web Service server's availability in an adequate manner. In this paper we present an application level gateway system for protecting Web Services from Denial-of-Service (DoS) attacks. DoS attacks often rely on mis formed and or overly long messages that engage a server in resource-consuming computations. For Web Services a suitable means to prevent such kinds of attacks is the full grammatical validation of messages by an application level gateway before forwarding them to the server. Web Service messages are XML documents and these are usually defined by an XML Schema, written in the XML Schema definition language-a grammar language for XML. Our system generates an XML Schema from a Web Service description and validates all Web Service messages against this schema.

2. LITERATURE SURVEY

Reviewing literature, the article named "Protecting Web Services from DoS Attacks by SOAP Message Validation" [1] tries to provide security to a soap web service from DoS attack by validation of a soap message and provide check way for validating soap message. The article named "Assessing the security of web service frameworks against Denial of Service attacks" [2] tries to protect web service against denial of service attacks, but didn't talk about other types of attacks. In a study named "possible attacks on Web services" [3] general suggestions for dealing with each of these attacks have been reported. This survey has proposed encryption technology to stop hijacking the parameters. In addition, the limiting of WSDL has been suggested in order to confront its attacks. Another idea in relation to securing WSDL is an article named "web services security" [4] which tries to propose a comprehensive model to improve the security of all these three technologies, includes UDDI, SOAP and WSDL, using encryption to provide confidentiality and hiding important information that WSDL contains, but it does not explain details of implementation. One of the models recommended for improving web service security that is suggested in other studies, is the model known as IAPF [4] which proposed an abstract solution to protect UDDI, WSDL and SOAP. IAPF suggests to enhance the security of WSDL by limiting the access of users to UDDI. In this project we will offer a practical approach to protect WSDL by applying cryptography on critical parts of WSDL to minimize the limitation on it.

2.1 Web service

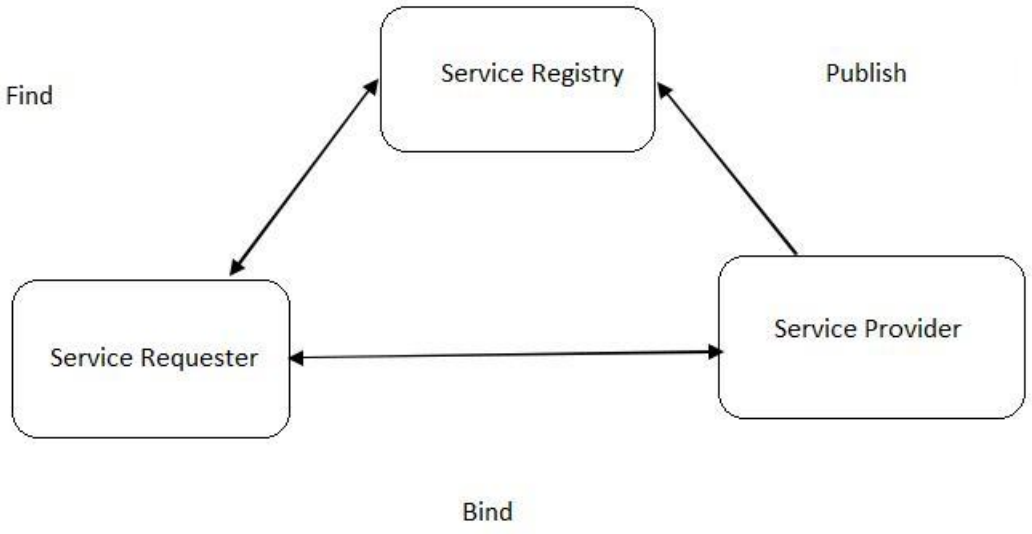
As per W3C the definition for a Web Service is:
A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine process able format (specifically WSDL). Other systems interact with the web service in

[1] Student, Dept. of Computer Goa College of Engineering, goa, India
[2] Assistant Professor, Dept. of Computer Goa College of Engineering, goa, India

a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web - related standards.

2.2. Business prospective of Web Service:
A web service is a business process or step within a business process that is made available over a network to internal and/or external business partners to achieve a business goal. The key is ease of integration, particularly between organizations, allowing business Systems to be built quickly by combining Web Services built internally with those of business partners.

2.3. Technical prospective of web service:
A web service is nothing more than a collection of one or more related operations that are accessible over a network and are described by a service description.



Any SOA contains three roles: A Service Requestor, a Service Provider, and a Service Registry.
1.Service Provider
is responsible for creating a service description.
2.Service Requestor
is responsible for binding a service description published to one or more service registries and is responsible for using service descriptions to bind to or invoke Web Service hosted by service providers

2.4. Service Registry
is responsible for advertising Web Service descriptions published to it by service providers and for allowing service requestors to search the collection of service descriptions contained within the service registry.

3. SECURITY IN WEB SERVICE
There is a need for security to Safeguard Web Service from Outsiders (Hackers/Malicious Users/Business Opponents)

3.1. They might tamper the content of SOAP message or attachment.
For E.g.: -Let's talk about the Weather Report Web Service Hosted and a web service client like Airways. In this scenario the web service is dealing with a very critical data of weather report for the operation of flights and any mistake might lead to a big disaster. In such scenario when the messages are of critical importance the Web Services Security comes into play.

3.2. A Non-Authentic/Malicious user or an attacker might send fake message.
For E.g.: - Let's talk about the Bank debit card web service and web service client like Amazon. In this scenario a hacker can generate the fake transaction (purchase details) and can generate fake SOAP messages and send them to bank Web Service to get the money from Some-one's A/c into his a/c.

Business Competitors/Opponents some time opt for negative business strategies. They might try to overload the web service through DOS attack. In such scenario the actual server running the web service incurs heavy losses due to unavailable downtime.
For E.g.: - If a banking web service is attacked with DOS attack and been brought down then during the period of unavailability the bank will make huge losses.
In Short, we need Confidentiality, Integrity, Authentication, Authorization and Non-repudiation.
1.Confidentiality:
guarantees that exchanged information is protected against eavesdroppers.
2.Integrity:
refers to the assurance that a message isn't modified accidentally or deliberately in transit.
3.Authentication: guarantees that access to e-business applications and data is restricted to those who can provide appropriate proof of identity.
4.Authorization
is a process that decides whether an entity with a given identity can access a particular resource.
5.Non-repudiation
guarantees that the message's sender can't deny having sent it.

3.3. WS Security Specification:

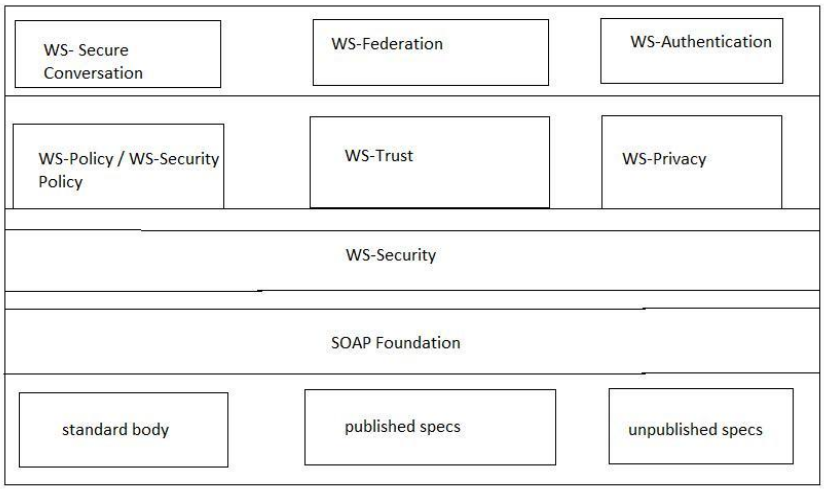


Fig1: WS Security Specification

1.WS-Security
defines how to include security tokens in SOAP messages and how to protect messages with digital signatures and encryption.
2.WS-Policy
provides a framework for describing Web Services meta-information. Based on the framework, domain-specific languages can be defined, such as WS-Security Policy.
3.WS-Trust
prescribes an interaction protocol to access Security Token Services.
4.WS-Secure
Conversation defines a security context with which parties can share a secret key to sign and encrypt parts of messages efficiently.
5.WS-Federation
provides a framework for federating multiple security domains.
6.WS-Privacy
provides a framework for describing the privacy policy of Web Services.
7.WS-Authorization
defines how to exchange authorization information among parties. The authorization is defined as a security token.

3.4. WS-Security in SOAP

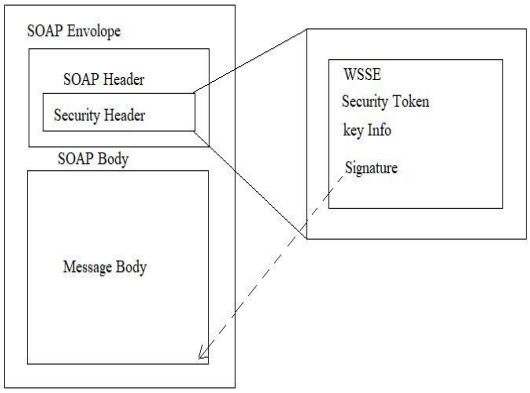


Fig2: WS Security in SOAP

- WSS information stored in SOAP security header.
- One or more security tokens carried in header to identify the transaction.
- XML Signature blocks provide integrity and link the identity to the transaction.
- Key information within the security token may be used.
- Privacy provided using XML encryption. F.

System Design for a Web Service

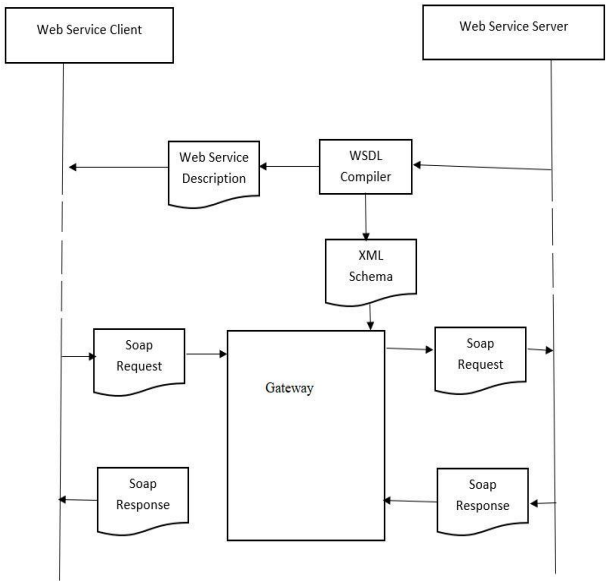


Fig 3: Integration of the Check Way Web Service Firewall.

The considerations above regarding SOAP message validation lead to our Web Service firewall, called Check Way. Figure 3 shows the Integration of a Web Service firewall between Web Service client and server. The Check Way WSDL Compiler gets the Web Service Server's Web Service description, generates the corresponding XML message Schema, "hardens" the description, and advertises the modified description to a Web Service client. The Check Way Gateway validates all SOAP messages against the Schema, forwards the message if it is valid, and rejects the message if it is not.

3.5. Web Service Interface Description

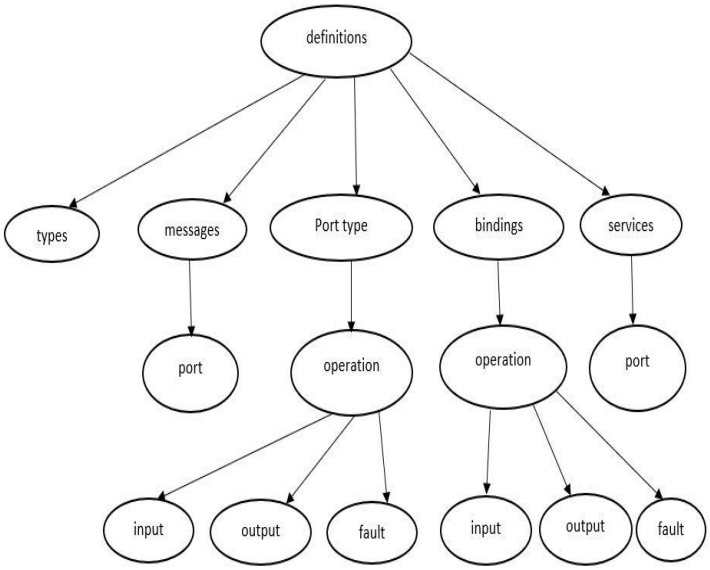


Fig 4: WSDL Structure

Figure 4 shows the WSDL document structure. It contains two sections: - an abstract interface, describing the Web Service's operation signatures. It includes the operation-organized in port Types-defining the input, output, and fault messages composed of parts, which refer to a datatype, defined in the types section. A concrete implementation. It includes a binding section-assigning the operations to a wire format and a transport protocol-and the ports-defining the service's network endpoint address. The WSDL specification [5] either allows a variety of concrete implementations or does not make any regulations at all. This applies for example to the grammar language used for data types (XML Schema, DTD, etc.), the encoding rules (literal, SOAP encoded.) or the transport binding (SOAP, HTTP POST, HTTP GET). This creates a problem in implementing compatible systems

4. SOFTWARE REQUIREMENTS

The tools used are as follows:
1.Eclipse-jee-indigo-SR2 IDE
2.Apache Tomcat Server Version 7.0
3.Apache Axis2 Version 1.7.6
4.Apache Rampart Version 1.7.1
5.Java Key Tool (JDK 7)

4.1 Eclipse-jee-indigo IDE

Eclipse is a universal tool platform - an open, extensible IDE for anything, but nothing in particular. The real value comes from tool plug-ins that "teach" Eclipse how to work with things - Java™ files, Web content, graphics, video - almost anything you can imagine. Eclipse allows you to independently develop tools that integrate with other people's tools so seamlessly, you won't know where one tool ends and another starts. The very notion of a tool, as we know it, disappears completely.

4.2 Apache Tomcat Server 7.0

Apache Tomcat is a servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the Java Server Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run. Implements the Servlet 2.5 and JSP 2.1 specifications. Support for Unified Expression Language 2.1. Designed to run on Java SE 5.0 and later.
Reduced garbage collection, improved performance and scalability. Native Windows and Unix wrappers for platform integration. Faster JSP parsing.

4.3 APACHE AXIS2 1.7.6ver

Apache Axis2 is a core engine for Web services. It is a complete re-design and re-write of the widely used Apache Axis SOAP stack. Implementations of Axis2 are available in Java and C. Axis2 not only provides the capability to add Web services interfaces to Web applications, but can also function as a standalone server application. Axis2 has support for REST by just removing the SOAP headers both on the client and on the server. Axis2 has support for Spring Framework.

4.4 APACHE RAMPART 1.7.1ver

Apache Rampart is an implementation of the WS-Security standard for the Axis2 Web services engine by the Apache Software Foundation. It supplies security features to web services by implementing the following specifications

1. WS-Security
2. WS-Security Policy
3. WS-Trust
4. WS-Secure Conversation
5. JAVA KEYTOOL

Java (JDK) has a key and certificate management utility called as key tool. It manages a KeyStore(database) of cryptographic keys, X.509 certificate chains, and trusted certificates. It allows users to administer their own public/private key pairs and associated certificates for use in self-authentication (where the user authenticates himself/herself to other users/services) or data integrity and authentication services, using digital signatures. It also allows users to cache the public keys (in the form of certificates) of their communicating peers.

Key tool also enables users to administer secret keys used in symmetric encryption/decryption (e.g. DES). Key tool stores the keys and certificates in a KeyStore.

5. IMPLEMENTATION AND RESULTS

5.1 Creation of Web Service:

Java SE Development Kit (JDK) 7 is used for java development. Eclipse indigo is used as the IDE. If not already installed, download Web Tools Platform (WTP) project files. It extends the Eclipse functionality for developing Web and Java EE applications. Apache Tomcat 7 based distribution is downloaded and under the Eclipse preferences, the path of the Server is configured to Tomcat directory. Now in the preferences window, there will be a web services option for configuring Axis2 preferences. Axis2 Standard Binary Distribution 1.7 is downloaded and extracted to a directory. Axis 2 preferences window under Eclipse is configured to point to this directory. Everything is left as default.

1.Create a new dynamic web project under Eclipse.

2.Under the configuration, Axis2WS Web Service is selected. Now click Finish.

3.A Secure web service, where communications are encrypted

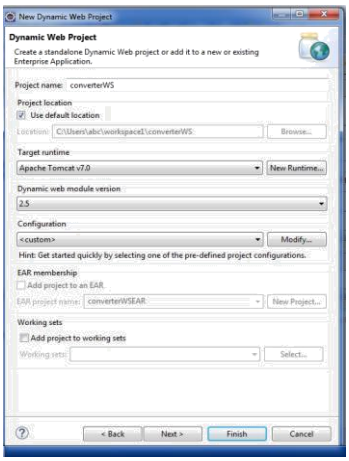


fig 1: Create web project

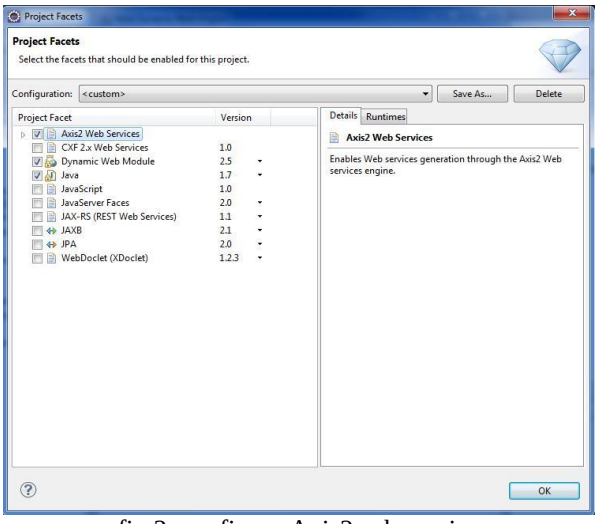


fig 2: configure Axis2web service

This will create the dynamic web project. The business logic used is the Converter.java which converts temperature from Celsius to Fahrenheit and vice versa.

converter.java

```
1  package wtp;
2
3  public class converter {
4      public float celsiusTOFarenhit (float celsius)
5
6      {
7          return (celsius *9/5) +32;
8      }
9      public float farenhitToCelsius (float farenhit)
10     {
11         return (farenhit -32)* 5/9;
12     }
13
14 }
15
```

Fig: 3 web service (temperature converter)

5.2.Add the Converter.java to the project

Preserve the package while adding. Build the project. Right click the Converter.java and select Web Service option to create the Web Service. It is possible to create the web service client simultaneously or it can be created later as well.

5.3.Configure the options as shown below.

Click Next. Select ―Generate a default services.xml file. Click next. On the next page, select start server. Now the web service has started. Click next.

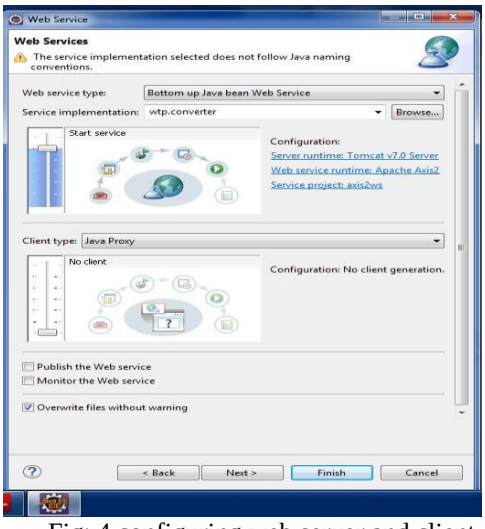
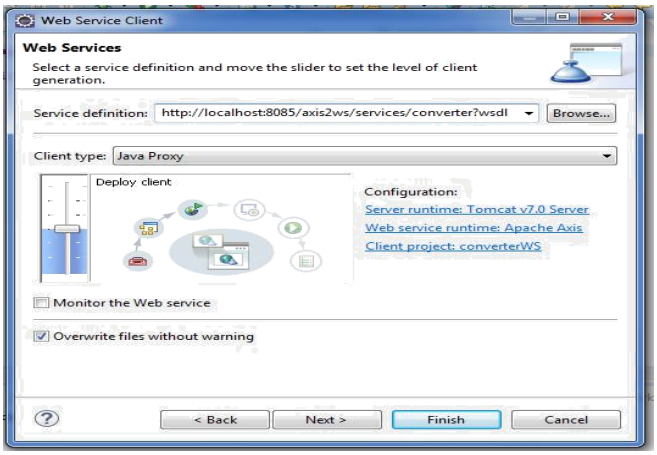


Fig: 4 configuring web server and client



Now the client side is being configured. The port name will be selected as SOAP12 version. Everything else will be left as default. Click Finish.
The code for testing the service called as ConverterClient.java is added in the Web Service Client project.

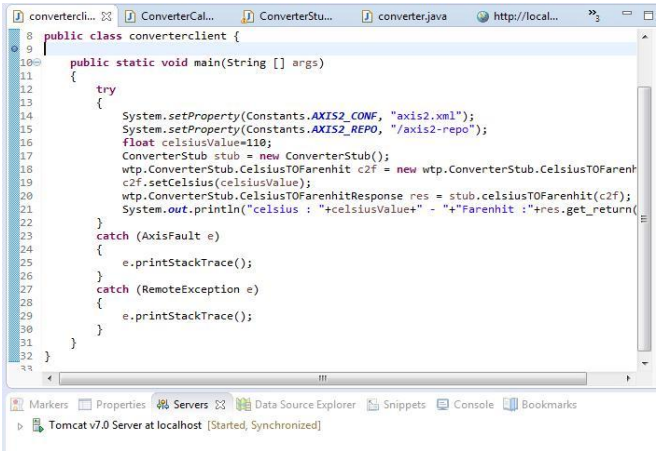


Fig: 5 codes for testing client.

The two lines of codes starting with System are used to configure the axis2.xml and the respective modules needed for the client to encrypt the soap message.
To invoke the service, the ConverterStub.java is created. To monitor the soap messages, send to the web service, the port for invoking is changed from 8080 to 8888 so that Tcpmon can intercept the message before sending it to the service.

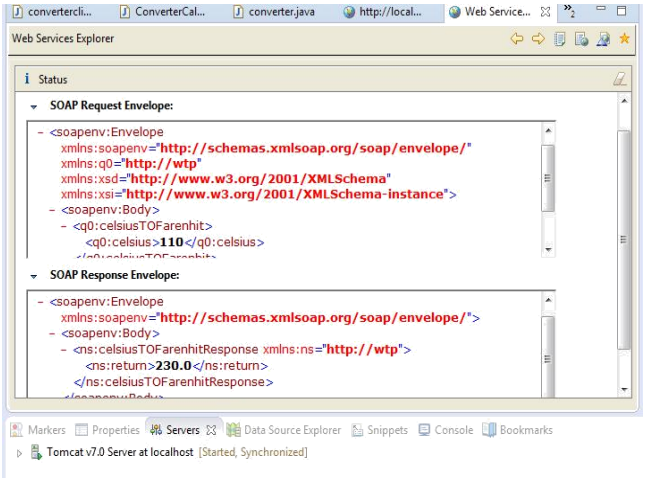


Fig: 6 soap request and response for a web service

(temperature converter from CelsiusTOfarenhit)
Adding the Encryption files
Setting up the environment
Rampart 1.7 binary distribution is added to the [AXIS2_HOME] \repository\modules. Its library files are added to [AXIS2_HOME] \lib. For encryption, latest Bouncy Castle jar is download and added to[AXIS2_HOME] \lib. Xalan and Jaxen jars are added as well to the library folder. If Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files are not present, download it and add local_policy.jar and US_export_policy.jar) to JAVA_HOME/jre/lib/security. The java. Security file in the %JAVA_HOME%\jre\lib\security should be modified to add the bouncy castle jar in ―security. Provider.

```
public class PWCBHandler implements CallbackHandler {
    public void handle(Callback[] callbacks) throws IOException,
            UnsupportedCallbackException {
        for (int i = 0; i < callbacks.length; i++) {
            WSPasswordCallback pwcb = (WSPasswordCallback)callbacks[i];

            String id = pwcb.getIdentifer();
            if("client".equals(id)) {
                pwcb.setPassword("clientkey");
            } else if("service".equals(id)) {
                pwcb.setPassword("servicekey");
            } } } }
```

Fig : 7 retrieving keys for service and client

For the client side, to use the encryption, a few lines of code will need to be added to the axis2.xmlfile.

```
<module ref="rampart" />

  <parameter name="OutflowSecurity">
    <action>
      <items>Encrypt</items>
      <encryptionUser>service</encryptionUser>
      <encryptionPropFile>client.properties</encryptionPropFile>
    </action>
  </parameter>

  <parameter name="InflowSecurity">
    <action>
      <items>Encrypt</items>
      <passwordCallbackClass>wtp.PWCBHandler</passwordCallbackClass>
      <decryptionPropFile>client.properties</decryptionPropFile>
    </action>
  </parameter>
```

Fig 8: encrypting service at client side

This will enable rampart for encryption and provide the location of the password handler and also the location of the properties file. For the service side, it is the same configuration. But this will be added into the services.xml file

```
    <module ref="rampart" />

<parameter name="InflowSecurity">
  <action>
    <items>Encrypt</items>
    <passwordCallbackClass>wtp.PWCBHandler</passwordCallbackClass>
    <decryptionPropFile>service.properties</decryptionPropFile>
  </action>
</parameter>

<parameter name="OutflowSecurity">
  <action>
    <items>Encrypt</items>
    <encryptionUser>client</encryptionUser>
    <encryptionPropFile>service.properties</encryptionPropFile>
  </action>
</parameter>
```

Fig 9: encrypting service at server side

A Secure web service, where communications are encrypted. Now care is to be taken that the class path for Java and Axis2 and that all the library files needed for the project are present in their respective directories.The client side is now tested and the message is send encrypted.

6. TEST RESULTS:

6.1 Request with Encryption Snippet

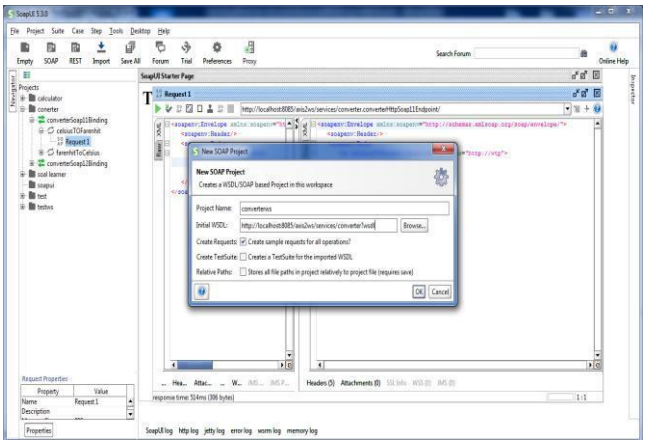
<soapenv:Body><xenc:EncryptedDataId="EncDataId-31216096"
Type="http://www.w3.org/2001/04/xmlenc#Content"><xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/><ds:KeyInfo
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"><wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"><wsse:Reference URI="#EncKeyId-181086"
/></wsse:SecurityTokenReference></ds:KeyInfo><xenc:CipherData><xenc:CipherValue>9HS6Tf78KX6iqdLL4mv
95SY7+XrM2gFCJ6jIHLmQVLeIk+dXFz99LQMuUfK2mzyGN59VQMjRdWTLgWqFZN+Bx+9mD1kP1dOkbzP
KWI9ixBVhA0mw7jDBFSoGGdWflo5oqQX7jcC6gSnWpAQBry8
/4DRxBUXeMKEOiM2fg+Hu8/esf5YxbE/GXsGWm4N4QysQJEhJlH3QVMkSx5/gFwKxY41s2xfFqBFr3hfDTugb
z1c5IbQ5jzWAQyAnJNvzlZObGIoDJoVwzNN9QMXATGablZvsXYCXvPrYfus667q725a04tQXeiO6dQGzPxhN
atR8iN7qCptq+rhrMdwARe7sNacbRRIIhjqdb68RyX+VwzCqsnqRZ3QdFPwdWuCKaGzx</xenc:CipherValue></x
enc:CipherData></xenc:EncryptedData></soapenv:Body>

6.2 Response with Encryption Snippet

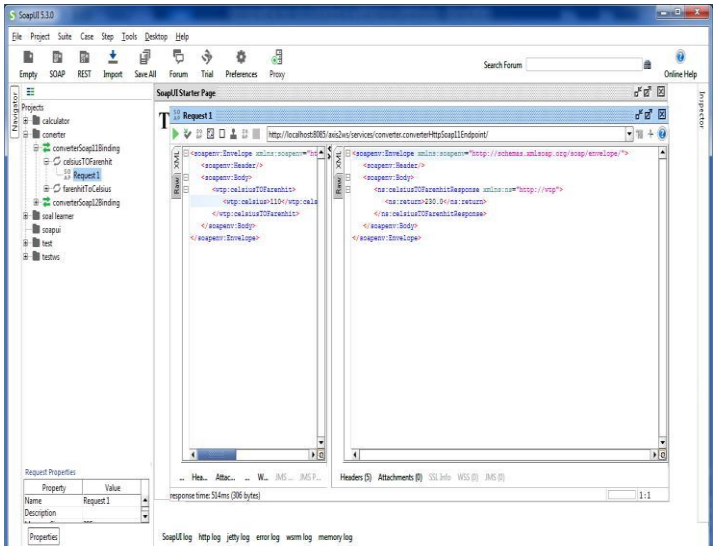
<soapenv:Body><xenc:EncryptedDataId="EncDataId-12462239"
Type="http://www.w3.org/2001/04/xmlenc#Content"><xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/><ds:KeyInfo
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"><wsse:SecurityTokenReferencexmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"><wsse:ReferenceURI="#EncKeyId-22555260"
/></wsse:SecurityTokenReference></ds:KeyInfo><xenc:CipherData><xenc:CipherValue>Qsa6W8q+472xikZZC9J
wHfTcK44tWNFFwhX1GUi2pXL8BB9gbxj3ARN9arzm3fS/hPmWkjg6KsHsCiK5mrCQPRjh/tFlqAnEHnvsbpez2j
7n5gitFzj1DBowJr3uX2FBe7MW84zv07rtcdgHMzu0k3JVu9xV9PNAHzAYCifEzvFm0kkKogHCb8L0zdBCBGw1
8CzCQL4JCwlgPqZvUXhL0GHAOlHq/SLldlj/p+tys1el/E6f+ode2svPJFU16DYw4KbPqiAJhXUAMdZa9AaZJ27q6
UfSyHEagDhmcWXEASglSR29d+vr4385TureCH2wUX/xrR5ohNGJDEw+Vg31JQd/yYHCvSkJz0y4asZuZMUTu
F7AnPn4YPEj1hZeVaJy</xenc:CipherValue></xenc:CipherData></xenc:EncryptedData></soapenv:Body>

6.3 testing web service on SOAPUI 5.3 tool

1. create a new soap project and provide WSDL path.



2. double click on soap project maximize it and click on soap request and click on run button.



7. CONCLUSION

The web services use XML as data exchange format for its communication with service provider, service registry and service requester. As the XML format is in a simple text, it is prone for attacks. It became vulnerability for web service where the attackers can use this retrieved information in order to make attacks on SOAP web services. The proposed system deals with securing a soap message from DoS attack by validating a soap message and providing security to WSDL file.

8. REFERENCE

[1] Nils Gruschka and Norbert Luttenberger “Protecting Web Services from DoS Attacks by SOAP Message Validation”, Department for Computer Science Christian-Albrechts-University of Kiel, Germany,2010

[2] R. A. Oliveira, N. Laranjeiro, and M. Vieira, "Assessing the security of web service frameworks against Denial of Service attacks," Journal of Systems and Software, vol. 109, pp. 18-31, 2015.

[3] E. Moradian and A. Håkansson, "Possible attacks on XML Web Services," IJCSNS International Journal of Computer Science and Network Security, vol. 6, pp. 154-170, 2006.

[4] H Krawczyk and M. Wielgus, "Security of Web services," in International conference on dependability of computer system 2006.

[5] Erik Christensen et al. Web Services Description Language (WSDL).W3C Note,2001.

[6] Keith Ballinger et al. Basic Profile Version 1.1. WS-I Organisation, 2004.

[7] SoapUIFunctionalTesting[WWWDocument],2012.

[8] Shahgholi, Narges, Mehran Mohsenzadeh, Mir Ali Seyyedi, and Saleh Hafez Qorani, “A New Security Framework against Web Services' XML attacks in SOA,” The 7th International Conference on Next Generation Web Services Practices, pp. 314-319, IEEE, 2011.