Security Analysis and Testing in Service Oriented Architecture

Noor A. Altaani,,Ameera S. Jaradat

Abstract— Now adays, Service Oriented Architecture (SOA) becomes the latest trend for software architectures to combine distributed services in an IT environment. SOA architecture supports an interoperable, cost efficient and reusable approach to develop **IT** systems for large scaled boundaries. SOA relies on web services technology that are developed independently and with increase in connectivity among these services, the security risk rise exponentially. Many security problems related with SOA applications have serious consequences unless it is managed in early phases. This research focus on the architecture of service oriented and the security problem, which it faces, therefore soapUI tool will use to test security of the services in this environment and so overcome the security problem.

Keywords— Service Oriented, SOAP, WSDL, UDDI, Service Brokering, soapUI.

1 INTRODUCTION

Currently, new business requirements need to be responding rapidly by IT organizations, the software application has run on multiple computing architectures, which allow distributed processing, and programming languages designed to run on any platform and reduce the implementation, which result in better and faster combination of applications. Therefore, SOA is considered as the new technology step to provide IT organizations meet their challenges.

Service Oriented Architecture is an architectural paradigm where it is essential purpose is to loose coupling by reducing dependency between software agents. The motivations of SOA entirely provide us to link data and its processing together. SOA defines the concept of communicating between service provider and service consumer. The mechanism of providing the service is done over the Internet or Intranet, where service provider provides a service that the service consumer consumes. The term service is defined as "course-grained, discoverable software entity that exists as single instance and interacts with applications and other services" [7], where it is an implementation of a well defined business functionality, and it is self contained i.e. may be composed of other services [7]. As examples on services are check customer credit and provide weather data.

- *Noor A. Altaani is currently pursuing masters degree program in Computer Sciences in Yarmouk University, Jordan. E-mail: taany_noor@yahoo.com*
- *Ameera S. Jaradat is currently Assistant Professor in Computer Sciences in Yarmouk University, Jordan. E-mail: Ameera@yu.edu.jo*

The popularity of SOA comes from the way that SOA is followed in separation of the service interface from its implementation. The consumers can only deal with the interfaces and are not interested with how these services will run their requests. On the other hand, these services are modular in design where each change in them does not need to be affected in the consumer's design. In addition, SOA provides interoperability between heterogeneous applications and technologies [8].

SOA security is a difficult task because each business has its own security infrastructure. Security in SOA is defined as "The sum of all techniques, methods, procedures and activities employed to maintain an ideal state specified through a set of rules of what is not in a heterogeneous, decentralized, and inter-connected computing system" [6]. Therefore, this research attends to focus on this problem and use supporting technology that proposed for security configuration, which is soapUI testing tool.

SoapUI is a "desktop application for inspecting, invoking, developing and functional/ load/ compliance testing of web services over HTTP" [12]. This tool specified to use by developers, tester whom use or consume web services. On the other hand, functional and load testing can be both interactively in soapUI and within an automated build or integration process using the soapUI command-line tools [13]. In next sections design and performance, analysis of web application and web services based online registration system, as case study will carry out while the security requirement for this system will identify. The web services based online registration system will test for response with and without different types of security profile that will identify in the next sections.

2 LITERATURE REVIEW

Over the last decade, the popularity of SOA has increased with the new trend in system evolution. A service-oriented architecture (SOA) can be viewed as an architectural approach for building systems, where this environment consists of service users and service providers. Furthermore, service is the main component that characterizes SOA [11].
A service is "course-grained, discoverable software entity that exists as single instance and interacts with applications and other services" [7]. A major result of these new trends is the movement from simple closed system in which specification are kept proprietary to prevent third hardware or software from being used, and then moving to the distributed open system which indicate to the system that allows third parties to make products that allow plug into or interact with it [8].

One of the first service oriented architectures was the use of DCOM or Object Request Brokers (ORBs) based on the CORBA specification [3]. SOA supports an information environment that is built upon loosely coupled, which refers to the mechanism of allowing the service or application using these service to be neutral to the underlying technical details of partner services, such that they can use it to achieve their fullest functionality. According to the architectural view what this points to is being represented by black box, this methodology use to indicate that we know what must go in and come out without concerned with how it does the translations [4]. Furthermore, reusability, agility and data interoperability is another advantage in SOA. Each service component in a SOA is stand-alone unit, where the service software is independent of the requester systems. In addition, agility use to point that a SOA potentially enables the enterprise to respond quickly to changes in the business environment by changing services [8].

SOA architecture consist of two key roles, service requestor (client) and service provider, which communicate via service requests, this service requests are considered as a message and are represented according to the Simple Object Access Protocol (SOAP). The question that arise in this situation is how does the service requester determine which service providers, should be selected for their service offering? Therefore, the service requester could keep the right to choose an application service provider according to those, which can be discovered from a registry service, such as UDDI. SOA standard such as SAML and WS-Trust, introduce another role, which addresses these issues, called a service broker as illustrate in figure 1. A service broker contains an index of service provider that is available. The service broker characterize by the ability to add value to the registry of application service providers through providing additional information about their services [5].

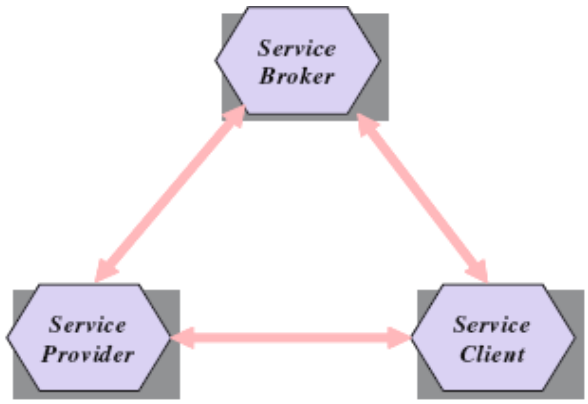

Fig. 1. Service brokering [5]

UDDI used as specialized instance of service broker, where the services providers publish the definitions of the services they offer using WSDL, after that the service requestors find information about the services available [5]. SOAs have been increasingly used in more and more domains with particular interests to ubiquitous computing in all its forms (ad hoc, nomadic and pervasive) [7]. The reasons that stand behind this popularity have been due to popular use of web services, and the heterogeneity of today's systems [4]. Web services is defined as "a software system designed to support interoperable machine to machine interaction over a network", as defined by the World Wide Web Consortium (W3C) [9].

Software architecture forms the bridge between business goals and the software system. Therefore, choosing and designing an architecture that satisfies the functional as well as the quality attribute requirements is vital to the success of the system [11]. Software architecture often includes and represents both functional and non-functional requirements of a system. Functional requirements represent the desired features of the system, while non-functional requirements also called quality attributes, which identify various attributes such as security, performance and usability. The security of SOA based systems can be viewed from three different corners: the security of the organization, the security of service and the security of service interaction [2]. Han et al has two views for SOA based security: individual service and services composition, and the security attributes for this viewpoint are authentication, authorization of services, maintability and service level reliability [2].

In service oriented environment, which consist of several services that are combined into an application, the process of, validate and monitor Quality of Service (QoS) is challenging phase. The following challenges are recognizable and any developer of service-oriented applications [1] is facing most of the time:

1- Application developers must be confident that services and composition of them will reach the phase of end user quality requirements.
2- Application developers must understand both the cost and risk of satisfying quality requirements in the system, where system qualities often must be traded off or built in.
3- Information about QoS to monitor and enforce service level agreements (SLAs) is another need for application developers.

In addition to the traditional security objective that illustrate in figure 2, there are several areas specification to SOA security objectives these are [11] [4]:

1- Authentication: This points to trust that the indicated sender is the one responsible for the information.
2- Confidentiality: This guarantees that access to service or to information just accessible by the authorized subjects.
3- Integrity: This guarantees that information is not corrupted.
4- Availability: This guarantees that the service is available in a timely manner.
5- Authorization.
6- Auditing.

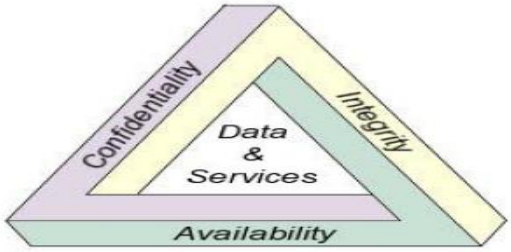

Fig. 2. Security objectives [4]

The reasons that make its difficult to guarantee security proprieties that are the service invoked by the application may invoke other services with their own set of distributed nodes, therefore any of which must be trustworthy. Another factor that arise security difficulties in a SOA environments is service reuse. In this case, a service has the ability to be reused by multiple consumers in different domain, where they have their own security requirements. Therefore, from what was shown previously, we can obviously say that is hard to design a service that guarantees multiple securities need [1].

Message transmission is one of the vital services that needs for system integration in SOA environment. This transmission is usually done via the SOAP protocol. On the hand that messages may carry vital business information, there integrity and confidentiality needs to be preserved, and therefore the mechanism in which SOAP message exchange in a meaningful and secured manner remains a challenging part of systems integration. SOAP is XML messaging used to transmit encoded information over several protocols (e.g. HTTP, SMTP). SOAP gives an easy approach to design protocols for communication between applications in an intranet over the internet. So security information must be contained within the SOAP message and/or SOAP message attachment [3].

There are many standards related to web services, as we illustrated above in the message transition between consumer, provider and repository. The fundamental ones are five, which are XML, HTTP, WSDL, SOAP and UDDI. The security objectives of service oriented achieved through several XML based security standards, as illustrate in figure 3, where it consist of XML- digital signature, XML-encryption and WS-security etc [9] [10].

XML signature as defined by the W3C, "provide integrity, message authentication, and/or signer authentication services for data of any type" [9]. The essential factor for XML signature using is to provide a mechanism, which makes it possible to sign only specific parts of a document. As XML, documents are made authored by different persons or systems, the need for ensuring the origin and the integrity of blocks seem legitimate [10].

XML encryption provides an ability to make it possible to encrypt whole or parts of an XML documents. This standard seems to be necessary in the case that intermediate web services may not be authorized to access some of the contents either send by the requester or by the service provider. XML encryption can be applied to totality of the document, to XML elements, to the content of an XML element or to the data part of an XML element [10].

The essential purpose of WS-Security is to provide the capability to implement an end-to-end security mechanism, which would be independent of the transport protocol. This purpose is satisfied by providing security methods to SOAP using standard extensions into the header [10]. Three main mechanisms are defined by the standard: "ability to send security tokens as part of a message, message integrity, and message confidentiality" [9], message integrity is achieved by XML signature while message confidentiality is achieved by using XML encryption; both can be used in conjunction with security tokens, where security token implements a set of claims. A claim is a standard about an entity (e.g. name, identity, key, etc) [9]. Furthermore, SAML is another security standard, which stands for Security Assertion Markup Language, and it is indicate to "an XML-based framework for communication user authentication, entitlement, and attribute information" [9]. SAML consists of two parties: the assertion party in which information is provided regarding a subject (such as a user, its authentication status and its attributes), and the relying party which chose or not to trust the assertion and makes decision according to the information read in the assertion [10].

These security standards are used to satisfy security objective in service-oriented environment. To ensure that

the security reaches to the system soapUI tool will be used in the next section to make test to the system, which is online registration, while the security requirement of this system will identify.

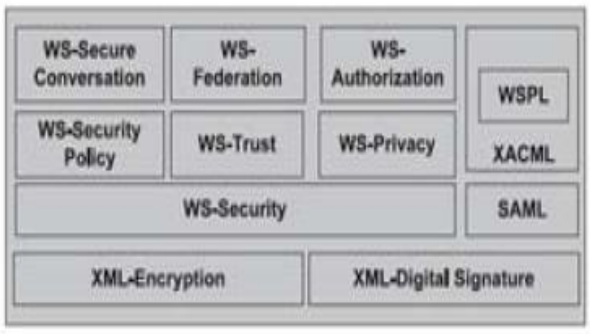

Fig. 3. Web Services Security Standards [6]

3 CASE STUDY – STUDENT REGISTRATION SYSTEM (SRS)

According to the process of registration, the student requests a course schedule that includes list of the courses that specified to the corresponding semester. In that states, student variety information some of it about the course, another about professor, department and prerequisites which attend to identify if student can register specific course or not.

The system that addresses the previous processes is an online registration system. Therefore, the system users will be professors who identified the courses that they will teach. On the other hand, students are another user of the system, so they attend to select courses. Furthermore, the registrar task in the system is to complete the registration process. Finally, billing system is an external system that bills student each semester [14].

The use case diagram for the system can be illustrated in figure 4, where the student activity is register for course. on the other hand, the professor select course where in this state provides the capability to select, review, modify, and delete a list of courses to teach for a specified semester. Furthermore, the professor can request class roster, where it provides the capability to request a printed list of all students assigned to a specified course offering. According to registrar task is to generate course schedule, maintain professor information by create, review, modify, and delete professor information. The same process, the student information can maintain. Finally, the registrar maintain curriculum by create, review, modify, and delete a list of course offerings for a given semester [14].

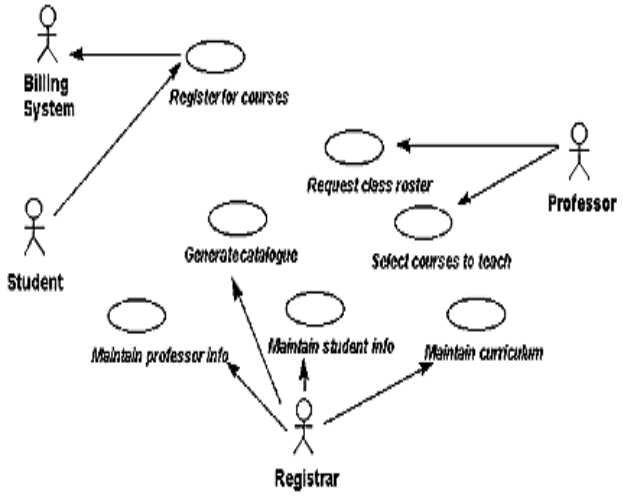

Fig. 4. Use case diagram for online registration system [14]

When the students attend to add courses, they must take in consideration at first the maximum course load not to be exceeded. Secondly, check that prerequisites are satisfied for the requested course and add the student to the course offering if the course offering is open.

Furthermore, out of our research concentrate on security. We will talk about several security requirements that must be addressed to achieve more security in the registration system which are [15]:

1. Access Requirements

For each student there must be secure and private access to his or her self-data. Therefore, both ITS and the registrar can have access to every part of the system. In addition, that is all these accesses need identification by ID and password.

2. Integrity Requirements

The integrity of the data achieve by assured reducing or limiting access to the database, appropriate synchronization and back up functionalities.

3. Privacy Requirements

These requirements attend to the protection of the database as the one that the university provides. Furthermore, this level of protection can grow due to the personal data made available on the system, and the larger share of people that will be having access to it through the online registration. The users' privacy achieved through the limited access that the login process is going to give to the database. In addition, the system does not have accessible direct access to the database itself. For user who needs to access the database will have to access it from a source independent from the registration system [15].

4. Immunity Requirements

This requirement attends to develop a security system that will reduce to the minimum the possibility of corruption from systems and/or humans [15].

4 RESEARCH EXPERIMENTS AND ANALYSIS

In this research, to address the structure of the SOA as design phase, the web service and the protocols that are used will be identified. We create authenticate web service that is used in the registration system which is log in service. Log in service require two input, user name and password. We choose this service to implement the authentication for entering the system, which forms security requirement in the registration system. Both students and professors can use this service. The implementation of this web service builds in C# programming language.

Student login is a security requirement in the online registration system where:

1) Student logs in to student registration login system using user name and password.
2) Student login system checks if the student exists and then, validates the student.
3) If the student does not exist, student login system results invalid user

In the same way, the professor can login the system. The following sequence diagram in figure 5 illustrates this process.

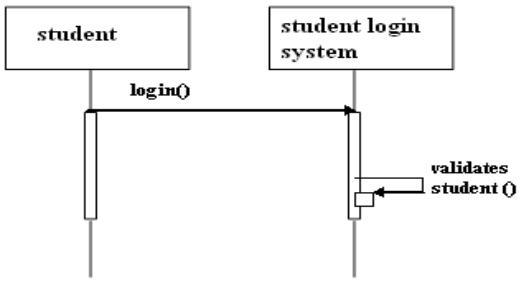

Fig. 5. Student login

After creating this web service and run it, we can easily identify the protocols that it uses. According to the WSDL, the XML document illustrates the methods, method parameters, namespace and handling URL for a web service that is necessary to the soapUI tool, as we will mention it later in the research.

Fig. 6. Authenticate web service

When web service code run, this brings up a web page that describes the method of authenticates web service as in figure 6 along with some other information. Furthermore, as selecting the link for the authenticate method from this page will bring up another page in figure 7, that allows you to test the operations of the service.

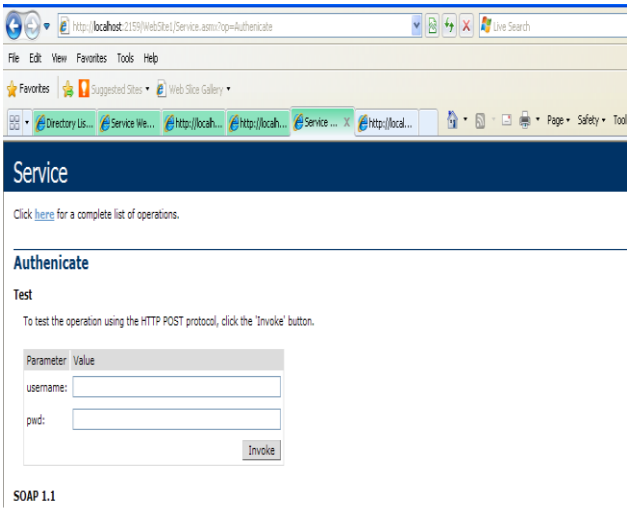

Fig. 7. Authenticate web service test

On the other hand, when we try to enter the URL of the web service, which is http://localhost:2159/WebSite1/Service.asmx?WSDL. The WSDL of the web service will appear. WSDL has four main sections that are element types, messages, portType and binding i.e. the communication protocol used by the web service. According to the portType element, it is

considered the most important element due to the description of the web service as well as all its operations include in this part of the WSDL. Figure 8 illustrates the components of the WSDL.

```
</s:element>
- <s:element name="Authenicate">
  - <s:complexType>
    - <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="username" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="pwd" type="s:string" />
      </s:sequence>
    </s:complexType>
  </s:element>
```

8-a. Element type

```
- <wsdl:message name="AuthenicateSoapIn">
    <wsdl:part name="parameters" element="tns:Authenicate" />
  </wsdl:message>
- <wsdl:message name="AuthenicateSoapOut">
    <wsdl:part name="parameters" element="tns:AuthenicateResponse" />
  </wsdl:message>
```

8-b. Messages

```
- <wsdl:operation name="Authenicate">
    <soap:operation soapAction="http://tempuri.org/Authenicate" style="document" />
  - <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
  - <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
- <wsdl:binding name="ServiceSoap12" type="tns:ServiceSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
  - <wsdl:operation name="HelloWorld">
      <soap12:operation soapAction="http://tempuri.org/HelloWorld" style="document" />
    - <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
    - <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  - <wsdl:operation name="Authenicate">
      <soap12:operation soapAction="http://tempuri.org/Authenicate" style="document" />
    - <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
    - <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
```

8-c. Binding

```
- <wsdl:port name="ServiceSoap" binding="tns:ServiceSoap">
    <soap:address location="http://localhost:2159/WebSite1/Service.asmx" />
  </wsdl:port>
- <wsdl:port name="ServiceSoap12" binding="tns:ServiceSoap12">
    <soap12:address location="http://localhost:2159/WebSite1/Service.asmx" />
  </wsdl:port>
```

8-d. PortType

Fig. 8. WSDL components

Recently, after the creation of the web service. We must have an approach to test the validation of the web service, so assure the authenticate user that will use it. For this reason, SoapUI testing tool will be used.

SoapUI is a "desktop application for inspecting, invoking, developing and functional/ load/ compliance testing of web services over HTTP" [12]. Furthermore, functional and load testing can be both interactively in soapUI and within an automated build or integration process using the soapUI command line tools [13]. SoapUI has many command for testing the web service, in this research, we illustrate two of them.

By looking to the web service request and response. First, WSDL is imported to the soapUI tool that needs essentially the WSDL, because it includes all of the information need to interact with web services. SoapUI should import the authenticate method. After that, by choosing authenticate request the following result in figure 9 will appear. Furthermore, login web service is a kind of web service, where the requester sends a request to service and waits. After that, the service processes the request and sends a response.

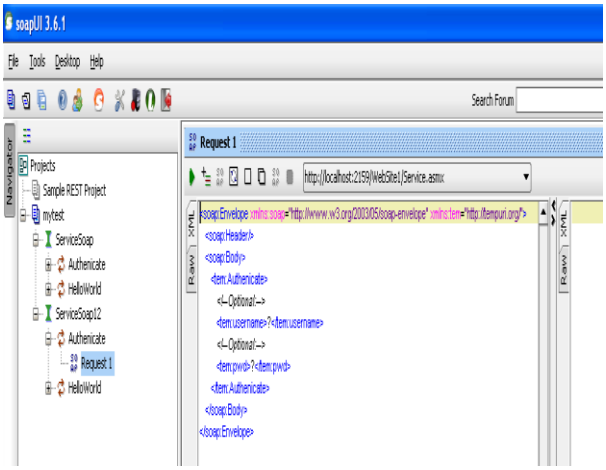

Fig. 9. Login request

For a request, we can do positive test through sending valid values i.e. sending valid user name and password:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:tem="http://tempuri.org/">
  <soap:Header/>
  <soap:Body>
    <tem:Authenicate>
      <!--Optional:-->
      <tem:username>noor</tem:username>
      <!--Optional:-->
      <tem:pwd>2011</tem:pwd>
    </tem:Authenicate>
  </soap:Body>
</soap:Envelope>
```

The result that we achieve when we send the request from soapUI:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instan
  <soap:Body>
    <AuthenicateResponse xmlns="http://tempuri.org/">
      <AuthenicateResult>true</AuthenicateResult>
    </AuthenicateResponse>
  </soap:Body>
</soap:Envelope>
```

As illustrate above, we can see that the communications during request and response run essentially by soap that is XML based protocol used for communicating with a web service, sends information to the request over HTTP. Furthermore, we can see the request soap elements such as soap envelope, header and body.

In other hand, if we attend to make a negative test through sending wrong data in the request, soap fault should occur:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:tem="http://tempuri.org/">
  <soap:Header/>
  <soap:Body>
    <tem:Authenicate>
      <!--Optional:-->
      <tem:username>noor</tem:username>
      <!--Optional:-->
      <tem:pwd>2020</tem:pwd>
    </tem:Authenicate>
  </soap:Body>
</soap:Envelope>
```

The result that we achieve from soapUI is:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instar
  <soap:Body>
    <AuthenicateResponse xmlns="http://tempuri.org/">
      <AuthenicateResult>false</AuthenicateResult>
    </AuthenicateResponse>
  </soap:Body>
</soap:Envelope>
```

SoapUI has a feature of test suite option that is needed to create series of tests, replay them and add the assertions. This is done by selecting the command adds test case on the request that appears in the navigator, figure 10 illustrates that.

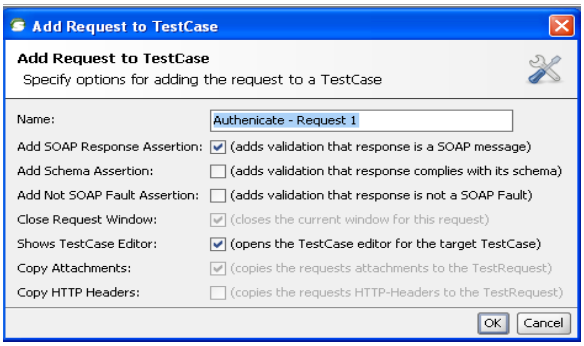

Fig. 10. Add request to test case box

Once the acceptance of this request, the window in figure 11 will appear. From the test suite, we can create a number of test cases. According to the assertion that we see, it is a test to run against the web service response to ensure we get the result we require, we can add many type of assertions. To choose an assertion that ensures the result is not soap fault we can select not a soap fault option as illustrates in figure 12.

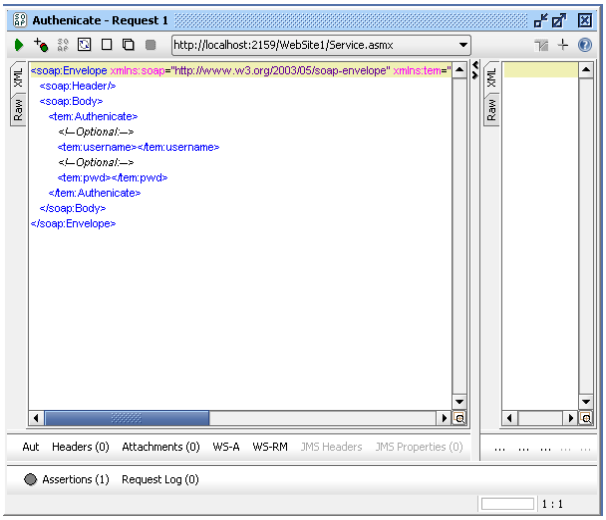

Fig. 11. Test suite that has one assertion to response

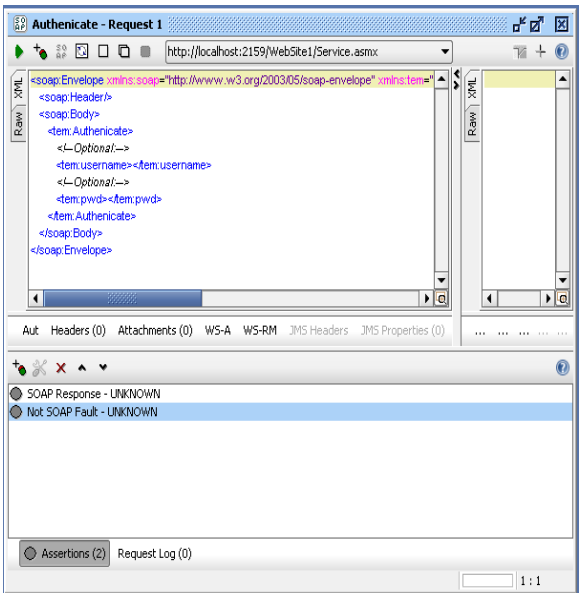

Fig. 12. Add not soap fault assertion

When we have set up our tests and assertions, we want to run them. We return to test case in the project navigator, open it and select the run to run complete test suite. So that we assure all things go well, we should see the following result in figure 13.

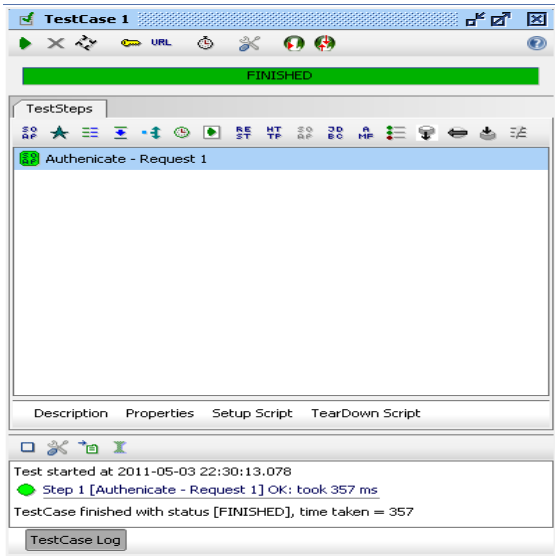

Fig. 13. Test case running result

5 CONCLUSION

As we can see these days, that SOA becomes the latest trend for software architectures to combine distributed services in an IT environment. SOA relies on web services technology that are developed independently and with increase in connectivity among these services, the security risk rise exponentially. For this reason, this research concentrates upon this problem and takes registration system as case study, so we identified the security requirements in it. Therefore, we take one of these requirements that are authentication requirement and build it as web service. After that, testing using soapUI tool to the validation of this web service by assure the authenticate user for this service in the response result that we get. Another feature in this tool was deployed through the research, that prove the powerfully of this tool on testing web service. Through using this tool all SOA protocols that are mentioned early in the research illustrate in easy way, and so we can identified the component of each protocol.

REFERENCES

[1] S. Balasubramania, G. Lewis, E. Moris, S. Simanta and D. Smith, "Challenges for assuring quality in service oriented environment", proceedings of 2009 ICSE workshop on principles of engineering Service Oriented System, 2009.

[2] S. Kou, M. Babar and A. Sangroya, "Modeling security for service oriented applications", ECSA '10 proceedings of the Fourth European Conference on Software Architecture, ACM, 2010.

[3] M. Rahaman, A. Schaad and M. Rits, "Towards secure SOAP message exchange in a SOA", SWS '06 proceedings of the 3rd ACM workshop on secure web services, ACM, 2006.

[4] D. Sanders, J. Hamilton and R. MacDonald, "Supporting a service-oriented architecture", SpringSim '08 proceedings of the 2008 spring simulation multiconference, ACM, 2008.

[5] M. Papazoglou, W. Heuvel, "Service oriented architectures: approaches, technologies and research issues", the VLDB JOURNAL, 2007.

[6] M. Saleem, J. Jaafar, M. Hassan, "Model Driven Security Frameworks for Addressing Security Problems of Service Oriented Architecture", International Symposium in Information Technology, Kuala Lampur, Malaysia, 2010.

[7] D. Controneo, A.Graziano and S. Russo, "Security Requirements in Service Oriented Architectures for Ubiquitous Computing", MPAC 04 Proceedings of the 2nd workshop on Middleware for pervasive and ad hoc computing, ACM, 2004.

[8] J. Andary, A. Sage, " The role of service oriented architectures in systems engineering", Information, Knowledge, Systems Management, IOS Press, 2010.

[9] R. Sassoon, "Security in SOA-Based Healthcare Systems", Norwegian University of Science and Technology Department of Telematics, 2009.

[10] R. Bidou, "Web Services Security".

[11] L. O'Brien, P. Merson and L. Bass, "quality Attributes for Service-Oriented Architectures", System Development in SOA Environments, SDSOA '07: ICSE workshops, 2007.

[12] E. Geirnaert, "Getting started with OWASP WebGoat 4.0 and SOAPUI. Hacking web services, an introduction", http://ebookbrowse.com/getting-started-with-owasp-webgoat-4-0-and-soapui-pdf-d50961913, 2011.

[13] www.soapui.org.

[14] http://sce.uhcl.edu/helm/cs4931a/CaseStudy.htm.

[15] A.Cardone, R. Lafountain, J. Mun and F. Rabbat, "Requirements Specification for Online Registration System".