ATTACKS ON WEB SERVICES NEED TO SECURE XML ON WEB

Abhinav Nath Gupta and Dr. P. Santhi Thilagam
Department of Computer Science & Engineering, NITK Surathkal, India

ABSTRACT

Web Services are the newest mechanism of communication among applications. Web Services are independent of both hardware and software infrastructure, they are very flexible and scalable. Lack of security features provided by the web services creates a window of opportunity for attackers. Web Services are offered on Http with Simple Object Access Protocol (SOAP) as an underlying infrastructure. Both SOAP and Web Services relies heavily on XML, hence, Web Services are most vulnerable to attacks using XML as an attack parameter. Several attacks use XML and most of them lies in the category of XML injection.XML based attacks discussed in this study covered a variety of attacks for example Denial of Services and Data Theft, escalation of privileges etc. Among these attacks the injections attacks on the web services are more severe and being given special attention. This study is aimed at providing an insight of the various forms of XML injections such as XPath injection, Coercive Parsing, and oversize payload.

KEYWORDS

XML, Denail of Services, Injection, XPath, web services

1. INTRODUCTION

The term **Web Service** is often used to designate web resources that are accessed by the software applications rather than users. Web Services are used extensively in today scenario due to their independent nature and code reusability. Web Services are independent of both software and Hardware platforms and web services also standardized the business application and due to the code reusability and interoperability feature provided, the business environment is falling for Web Services. Due to extensive usage of Web Services in business scenario it gives enough importance to Web Services to raise the security concerns of Web Services [13, 8].

The Web Services acts as an agent of business logic to the user end or to the application programs. Once the agents are compromised possibilities are, the entire system is compromised as in context of the Web Applications Scenario as webApps depends on Web Services. Web Service Engine needs an XML parser to extract the required parameters from an incoming message. Exploiting this parser can successfully lead to Denial of Service attacks. Web Services are designed in SGML type languages preferably XML so there are some attacks like XML Injections, XSS and XPath Injections that targets the XML based Web Services to be highly prevalent and even the Web Services designed using SOAP and offered over http. These attacks often inject additional nodes or modify the existing nodes so as to change the operation parameters. Mitigations of these attacks are must for security of Web Service and underlying Business Model to work properly.

The typical requirements for a secure system are integrity, confidentiality and availability. Any action targeting violation of one of these properties is called an attack; the possibility for an attack is called vulnerability. This study list various attacks specific to the domain of the Web Services.

DOI : 10.5121/cseij.2013.3501

These attacks can be accomplished due to the un-sanitized input to the Web Service or unauthorized updating of the source code by injecting malicious code in Web Services. Among all the other reasons the major threat to web service is the injection of malicious XML code by un-sanitized input to the Web Service. In the coming sections various kind of attacks including the above named attacks are listed with their detection and corrective measures.

Section 2 describes the related work done on the same topic. Section 3 gives the details of most attacks to Web Services using XML as attacks parameters. Section 4 describes the most general and popular methodology's used to mitigate the attack vulnerabilities. Section 5 provides the motivations behind the study. Section 6 concludes the study by providing suggestions to defend web services as per current security standards followed by references considered for the study.

2. RELATED WORK

There has been considerable effort in exploring different kind of attacks on Web Services. Probing Attacks, Coercive Parsing, External Reference Attacks and SQL Injections attacks have been discussed by (Negm, 2004). (Vorobiev, 2006) has classified attacks as XML attacks, SOAP based attacks or Semantic WS attacks based on exploits used. (Jensen, 2007) has demonstrated SOAPAction spoofing, oversized payload and cryptography based attacks. The message validation as a technique to thwart DoS attack is shown by (Gruschka, 2006). It mentions the use of schema embedded in WSDL (Web Service Description Language) to validate messages. (McIntosh and Austel, 2005) have discussed XML Re-Writing attack which shall serve as a baseline for the attacks that we have described here. Use of context sensitive signature as a countermeasure of XML Re-Writing attack is outlined by (Gajek, 2009). XPath injections is an attack on Web Services with severe after effects, as demonstrated by (Antunes, 2009). Input Sanitization, static code analysis and penetration testing are popular methods for XPath attack vulnerabilities in Web Services [8]. Learning Based approaches are presented as an effective countermeasure against XPath injection attacks on Web Services [6].

3. ATTACKS ON THE WEB SERVICES

This Section of this study defines the various prominent attacks specific to the Web Services with their few suggested mitigation methods. Several attacks are presented in a tree form in Figure 1.

3.1 Injection attacks: Injection attacks generally deals with the attacks domain where malicious data is intentionally included to the input provided to the Web Service to interrupt the normal functioning of Web Services [8]. Some injection attacks are:

3.1.1 **SQL Injection:** SQL injection is an attack that generally occurs with databases. In this input query provided to the database is modified in such a way to alter the output of the query as per the attacker requirement [11]. Suppose, a web service is querying the database for the authentication of the user, the user provides the input credential to the form and from the form the web service takes the credential for validation. In normal cases the validation is done and access is granted to user but when the input is altered by attacker then unauthorized access can be gained by the user.

Example:

Good: *'select * from accounts where userid="$userid"' 'AND password="$pass"';*
Evil: *select * from accounts where userid="**1" or "1=1--**"AND password="";*

In the above example as we saw that the normal input validation occurs at **Good** but when modified in **Evil**, the string **1 or 1=1--** converts SQL query to bring out all the username and password and grant access to the attacker with any set of credentials stored in the user database.

***i.* Detection of SQL Injection Vulnerability**: The SQL injection vulnerability can be detected by provided any random input and analyzing the result occurred.

If on providing as input a simple apostrophe(‘) to the form if the result comes out to be an SQL error or database error or server error then there is a good chance of a SQL injection vulnerability to the service.

***ii.* Corrective Measures:** The various corrective measures provided so far for the SQL Injection attacks is to sanitize the input before passing it to the web service or web application[11,8]. Some standards are designed to do that like use of stored procedures and use of parameterized Statement but the essence of all the work here to sanitize the input query.

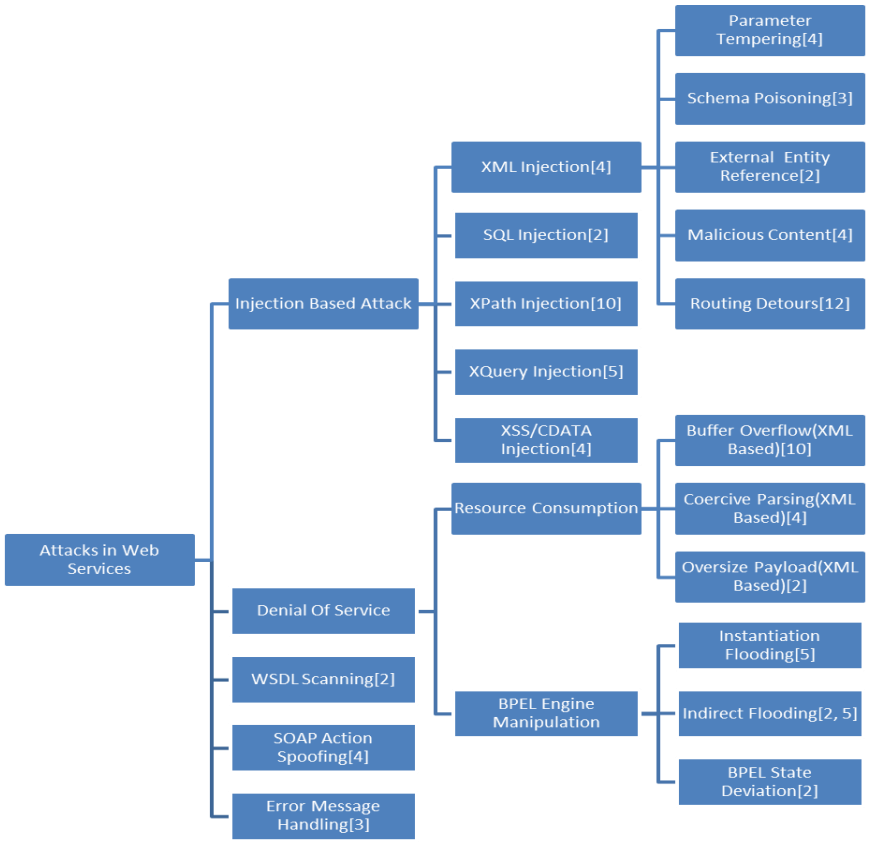

Figure 1. The Taxonomy on the Attacks on Web Services

***a.* Stored Procedures:** Stored procedures are a sequence of instructions in PL/SQL language. Is a programming language implemented by some DBMS, that lets you store sequences of queries frequently applied to your model, and share the processing load with the application layer.

***b.* Parameterized Statement:** Prepared statements are queries written with placeholders instead of actual values. You write the query and it is compiled just once by the DBMS, and then you just pass values to place into the placeholders. The advantage of using prepared statements is that you enhance the performance considerably, and protect your applications from SQL Injection.

3.1.2 **XML Injection:** An XML Injection attack tries to modify the XML structure of a SOAP message (or any other XML document) by inserting content e.g. operation parameters containing XML tags [2, 3]. Such attacks are possible if the special characters "<" and ">" are not escaped appropriately. This can lead to xml injection attacks. The detection and corrective measures regarding the XML injection attacks is strict schema validation of the SOAP messages including the data type validation of the data included in the message itself. Various types of XML injections are shown in fig 1.

3.1.3 **XSS/CDATA Injection:** An XSS (Cross Site Scripting) Injection attack is also prominent in case of the Web Services [8, 15]. A CDATA section in XML is used to escape the text and can be used to construct malicious JavaScript code which further leads to XSS attack over the web service.

The code given below presents an example of the XSS injection attack over the Web Services.

```
<BLOG_ENTRY>
 <EMAIL>john@due.com</EMAIL>
 <TEXT>
  <![CDATA[<S]]>CRIP<![CDATA[T>]]>
   alert(document.cookie);
  <![CDATA[</S]]>CRIP<![CDATA[T>]]>
 </TEXT>
</BLOG_ENTRY>
```

The manipulation in the <TEXT> node of the XML file using the CDATA Section leads to the JavaScript code given below.

```
<SCRIPT>
 alert(document.cookie);
</SCRIPT>
```

The detection of XSS injection vulnerability totally depends on the configuration of the web service to display the generic error messages to the user, like, URL not found. If the server specific errors are displayed then it leads to the misuse of the information in that error messages and leads to XSS injection attacks [8]. Another popular preventive measure against the XSS injection attacks is the input validation and filtration.

3.1.4 **XPath Injection:** XPath is language designed specifically to query nodes in an xml document. XPath is a query language for xml document like sql is for relational databases. Difference between sql and xpath is that xpath is implementation independent. Similar to Relational databases, xml document are also susceptible to xpath injection. Similar to sql injection xpath injection occurs when a web service is invoked or xml database is queried using unsanitized user input. We'll use this XML snippet for the examples.

```
<? XML version="1.0" encoding="utf-8"?>
<Users>
 <User>
<FirstName>Arnold</FirstName>
<LastName>Baker</LastName>
<UserID>ABaker</UserName>
<Password>SoSecret</Password>
<Type>Admin</Type>
```

```
</User>
</Users>
```

Suppose we have a user authentication system on a web page that used a data file of this sort to login users. Once a username and password have been supplied the software might use XPath to look up the user:

```
C#:
String FindUserXPath;
FindUserXPath = "//User[UserID/text()='" + Request("UserId") + "' And
Password/text()='" + Request("Password") + "']";
```

This query will work normally, but it is also susceptible to xpath injection by using the query string given below:

```
UserID: blah' or 1=1 or 'a'='a
Password: blah
FindUserXPath becomes //User[UserID/text()='blah' or 1=1 or
'a'='a' And Password/text()='blah']
```

Logically this is equivalent to:

```
//User[(UserID/text()='blah' or 1=1) or
('a'='a' And Password/text()='blah')]
```

In this case, only the first part of the XPath needs to be true. The password part becomes irrelevant, and the UserID part will match ALL employees because of the "1=1" part.

This types of attacks can be defended by sanitizing the input provided by the user and properly escaping the quotes and slashes present in the input parameters.

Another form of XPath injection is blind XPath which employs hit and trial method guess the root node of the XML document and then carry out attack while in above described XPath attack the attacker is aware of the structure of the XML document.

3.1.5 **XQuery Injection*:*** XQuery is the language specially designed to query the XML document in the same way as the SQL query does with the Relational Database. XQuery is the abstract form of XPath described above and the XQuery injection is same as the XPath Injection the Only difference is that we don't use the name of the name of the XML document in XPath but we do in XQuery. XQuery is inherited from the XPath so the syntax of the query and notations is similar in both XPath and XQuery.

3.2 Denial of Service (DoS): Denial of service attack is the attack which targets the server behind the web service and aimed at the request processing capability of the processor [2, 8]. In this attack the attacker manipulates the variables associated with the request made to Web Service for example, size of request and frequency in such a way that the processing of request becomes difficult for the web server's processor and which then leads to ultimate rejection of request, i.e. Denial of Service to user. Several types of DoS attacks are there as in Fig 1, which includes XML and XML with http i.e. SOAP.

3.2.1 **Oversize Payload/XML Bombs:** Oversize payload attack aims at exhausting the resources of the web server. Against web services, this is attack is easy to mount due to high computation

cost of xml processing. The reason behind this is that most Web Service frameworks make use of tree based XML processing model like *DOM (Document Object model)* .Using this model XML Document like SOAP request is completely read, parsed and converted into in-memory object representations which in case of large SOAP request takes more computation power and memory than the original SOAP request.

A good example of oversize payload is just keep on replicating a single tag again and again say 10000 times to increase the size of the message and this will do the trick. For example,

<Address>
<name>ABC</name>
<name>ABC</name>
<name>ABC</name>
….to 10000
</Address>

This request when put under SOAP envelope and send for execution will lead to the Denial of Service attack. Don't get on the number of repetitions it is just take for sake of convenience to explain the concept.

An obvious Countermeasure against the Oversize payload attack on the Web Services is to put a check on the size of the SOAP request.

3.2.2 **Coercive Parsing:** First step in processing any web service is parsing the SOAP message to get the parameters out of the request and transforming the request to make it accessible to the other application programs and this kind of processing leads a another kind of possibility for the Denial of Service attack known as Coercive Parsing [2]. In this attack a sequence of opening tags belonging to the different namespaces is created and this reduces the response efficiency to the web server to the considerable amount and further leads to the Denial of Service.

Such kinds of attack are very difficult to counter since XML does not provide any restriction for the declaration of the namespaces used in the XML Document and also no limit to the number of the External references used in the XML document, and if the referenced documents are also deeply nested then it becomes a lot more difficult to counter. But, still such kind of attacks can be mitigated to some extent by using Schema validation by providing the restriction to the number of elements regarding particular namespaces in the XML Schema Definition (XSD) file [3].

3.3 External Entity Reference Attack: In External Entity Reference attacks the attacker tries to bypass the security by attaching a reference according to schema which is downloaded during validation of the XML document but before processing of that document [4, 10]. Such an attack make use of XML tags like <!ENTITY> and <!DOCTYPE> to insert the external references to the XML Document. For Example, <!ENTITY rootelement SYSTEM "URI"> where URI is Uniform Resource Identifier and used to make reference external to the XML Document. The only way to detect such kind of vulnerability is to simulate attacks and testing. To prevent such kind of attacks generally recommended procedures are use of XML Schema Definition (XSD) in place of DTD and also disallow any user modification in the prolog section of the XML Document.

3.4 Malicious Content/Application: An Attacker sends a malicious code and attachment such as image, exe files and other application specific document with the SOAP message which contains malicious code, virus like Trojan horse and other which are transmitted with the XML doc as the firewall which are designed to protect system against malicious code are blind to XML

files [3]. The SOAP with attachment specifications allows transmitting packages using MIME. The SOAP message can refer to attachment through the use of URI. The detection can be done only by testing and analyzing and the only prevention against such attacks is to examine the packet for the attachments and data types and other input parameters.

3.5 Replay Attacks: In this attack the attacker repeatedly send the previous SOAP request message to the web service in order to crash the system or restart. The target behind such type of attacks is to overload the web service or to gain access to system by posing as a legitimate user. Prevention against such attacks is the use of timestamps and random session IDs.

3.6 Privilege Escalation Attack: Privilege Escalation is a special type of XML injection attacks targeted for web services used for authentication of user and using XML files as user Database Access Control List [5]. In this attack the attacker injects the fake user credentials as part of original request to add a fake unauthenticated user to the XML user database. The only way to prevent such kind of attack is to sanitize the input by proper filtration of tags and scripts from the input and also validating the SOAP request against the XML Schema [11].

3.7 Parameter Tampering Attack: Parameter Tampering is a kind of XML injection attack in which the attacker modifies the parameters in a SOAP message in attempt to bypass the input validation in order to access the unauthorized information [1, 7]. An attacker sends the SOAP message where the field values are different what the server expects. In SOAP the parameter can take the form of XML Elements. The server has XML Schema defined with a set of rules and restrictions regarding the input from the client. In parameter tampering attack the attacker violates this schema or provide oversized payload also known as XML Bombs**.** The Result is severe like Denial of Service is most common, and also information Disclosure, etc. The only way to test such kind of attacks is simulation of attacks by automation tools. The preventive measures are strict validation of SOAP message for input parameters and other attachments.

3.8 Schema Poisoning Attack: XML Schema models an XML Documents grammar and document structure.XML parser uses this grammar to properly interpret web services messages. Since grammar describes the necessary preprocessing instructions, they are vulnerable to tampering if not stored securely [13]. It is a variation of the XML injection and the possible attacks it can cause is the Denial of Service. The Only prevention to such kind of attack is to protect the XML Schema against unauthorized modification. For implementation, an application that uses a known schema, using a local copy or a known good repository instead of schema reference is a good preventive measure.

3.9 Buffer Overflows Attack: Generally the communication between the sender and receiver of the web service is of asynchronous nature, hence an attacker may send a malicious request containing specially crafted parameters to overload the input buffer. Buffer overflows generally lead to crashes. The Buffer overflow vulnerability can also be tested by simulating the attacks and there are no complete solutions to the problem but some of measures taken to cater this problem are filter the request messages for size bounds avoid use of risky API's during the development phase of web service.

3.10 BPEL State Deviation Attack: BPEL processes are called by external communication partners, and BPEL engine provides web service endpoints accepting service request. Each BPEL process have more than one process instances running concurrently, hence the endpoints are accepting request messages at all times. Thus an attacker may send a malicious request which is correct syntactically, but not related to any existing process instance. These types of message has

to be discarded but only after verifying all the instances of all running processes but this cause a huge computation overload and eventually cause denial of service..

3.11 Instantiation Flooding Attack: As a new message arrives a new instance of the BPEL process is generated. Such instance immediately starts executing the instructions given in the process description. Once all the instructions are executed only then the instance is terminated. An attacker can repeatedly send request messages to the same BPEL process. For each message, the BPEL engine will create a process instance and start its execution, running until its instructions are executed completely and it gets terminated. This will flood the BPEL engine with new requests and eventually lead to denial of service.

3.12 Indirect Flooding Attack: This attack works in same way as the instantiation flooding attack but it targets the backend architecture of the web service. This attack uses the BPEL engine as an asset to mount attack on backend architecture. The attacker will flood the backend system with web service request messages and for each message a process instance is created. This will put a heavy load on BPEL engine and also on the backend systems. Thus, if the target system is not powerful, it will lose availability, finally resulting in a Denial-of-Service.

3.13 Routing Detour Attack: The Web Services-Routing specification help directing the xml messages by attaching the information about the intermediate nodes. These intermediate nodes can be compromised, allowing attackers to insert malicious code inside the message. The attackers can then remove the malicious code before forwarding the document to its destination.

3.14 SOAPAction Spoofing: Operation invoked by the web service can be identified in two ways, one by scanning the first child of the soap body, second by an http header attached to the message names SOAPAction. Although this value only represents a hint to the actual operation, the SOAPAction field value is often used as the only qualifier for the requested operation. SOAPAction http header can be tampered to cause man in the middle attack and used to invoke other operation than the original intended one.

3.15 WSDL Scanning: WSDL stands for web service description language and wsdl document related to a particular web service contains all the details about that web service like endpoints, operations and parameters details, and binding details. WSDL documents are easily available on the web and an attacker can study the document to learn in depth about the target web service. This information gained from the wsdl can then be used to mount sophisticated attacks on the web service. This attack is called the wsdl scanning. To defend against wsdl scanning, wsdl documents has to be kept secure and in a separate network from the web service network. Another way to keep them secure is by providing external client a separate wsdl which contains details about external operations only.

3.16 Error Message Handling: Error messages are a great source of information from an attacker's point of view and in case of web application it generally happens due to the bad programming practices the server specific or application specific error messages are displayed to the user. These messages have valuable information with them such as the version number of the server and by getting information about the server number the attacker can exploit the known vulnerability associated with that particular server or application.

4. COUNTER ATTACK TECHNIQUES

The Counter attacks mechanism first begins with detection of Vulnerability. Generally used approaches to counter attacks are shown in figure 2. Methods Used to detect Vulnerability are

1. Static Code Analysis: "white-box" approach that consists of the analysis of the web application source code [8].

2. Penetration Testing: "black-box" approach that consists of the analyses of the web application execution in search for vulnerabilities [8]. In this approach, the tester uses fuzzing techniques over the web HTTP requests. Methods generally applied to prevent an attack to occur, as shown in Figure 2.

3. Input Filtration: Input taken from users must be filtered to remove escape characters like **'**,**"**, /,--, <script></script> and then only it is sent to web services for further processing.

4. Centralized Schema Validation: Centralizing the validation process of request and response message within the network perimeter with largely reduces the possibility of outside attack to a considerable amount.

5. Access Control Mechanisms: Use of XACML's to provide access control to prevent unauthorized modification in XML schema.

6. External References Blocking: Blocking the external referenced element and using only validated and sanitized urls can successfully avert the external references attack.

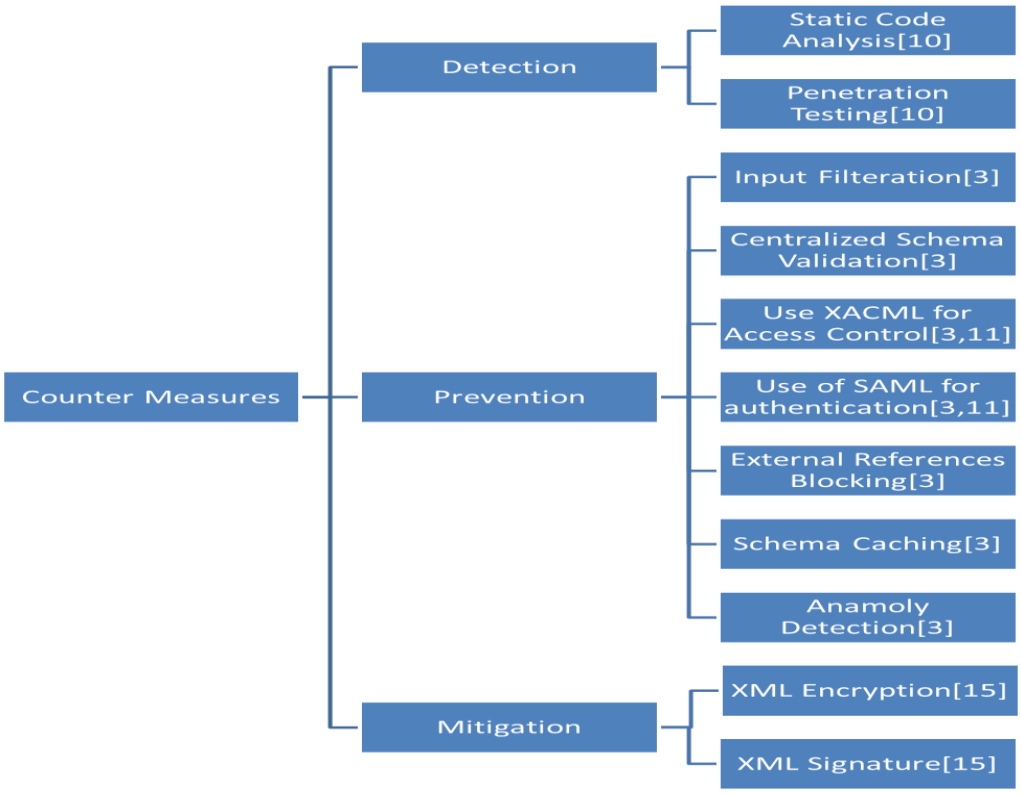

Figure 2. Taxonomy on The Counter Measures to attacks on Web Services

7. Schema Caching: A secure repository of XML schemas that is periodically updated is one of the best ways to fend the usage of compromised, unapproved XML schemas.

8. Use of SAML for Authentication: SOAP messages can cross multiple organizational bound aries, each requiring its own authentication process. It is better to have authentication system for each document. This can be done using SAML. SAML allows for a single sign-on approach by embedding authentication status, as well as authorization permission, within a particular message.

9. Anomaly Detection: Anomaly Detection is generally applied method to detect the abnormal behavior of the web service to detect whether it is an attack or not. It is based on analyzing parameters associated with SOAP request like size, frequency of same incoming message.
Few Mitigation techniques that can often use in prevention mechanisms can also reduce the after effects of an attack. These are:

1. W3C XML Encryption: To encrypt the content of the soap request message.
2. W3C XML Signature: To digitally sign the xml document to ensure its authenticity and integrity.

5. DISCUSSION

The survey on the web services described in this paper aimed at presenting potential threats and vulnerabilities to the web services and also gives an idea about there after effects on the architecture behind.

What is the major Threat? Out of 18 major attacks on web services described in this paper, 12 are just mere variations or different flavor of XML Injections which includes XPath/XQuery Injection, Coercive Parsing, CDATA Injections, Schema Poisoning, Parameter tempering, etc. Due to this immense effect of XML Injection based effects on web services, a large amount of consideration is been laid on prevention mechanisms of XML injection vulnerability in web services scenario.

Why it is important to mitigate? Web services are standard for applications to exchange information over the internet and also these are the building blocks of most prevalent business architecture today i.e., Service Oriented Architecture (SOA). Web Service relies heavily on the XML, due to the use of SOAP and WSDL and this is the reason that makes them so vulnerable to the XML based attacks. These attacks are important to prevent as if doesn't, they can cause a huge harm to the Entire Services architecture of the Business Enterprise.

What we have done? Several organizations like OWASP,OASIS and NIST comes forward to counter the threats posed by various attacks and vulnerabilities to the web services architecture and hence proposed several standards like WS* family to counter the attacks on web service, of them all the most prominent is the WS-SECURITY standard. WS-Security along with soap specification provides a set of protocols to ensure the authenticity, confidentiality and integrity of the web services requests. Standards like XML Encryption are designed to secure communication of web service while ensuring the confidentiality. XML Digital Signature is used to ensure the authenticity and integrity of the web service request message during communication among client and service provider.

Finally, current approaches to Web service security focus on security of individual services and secure messaging among the services. As Web services become more intelligent, aspects such as secure business process and workflows and trust negotiation should also be included in the security model.

6. CONCLUSIONS

While an SOA implementation based on XML and Web services can offer various benefits, it can also introduce new avenues of attacks for hackers to exploit critical business applications and processes. Web Services are susceptible to many attacks because the information about the web services are publically available as the wsdl document is freely available. A lot of attacks which are discussed in this study exploits vulnerabilities related to xml processing and since web service highly depends on xml those attacks can easily be mounted on web services. Most of the attacks discussed are in the owasp top ten list. These attacks on the web services can be defended by reducing the amount of information disclosure and applying secure coding practices.

REFERENCES

[1] Laranjeiro, N., Vieira, M., Madeira, H.," A Learning-Based Approach to Secure Web Services from SQL/XPath Injection Attacks" IEEE 16th Pacific Rim International Symposium on Dependable Computing (PRDC), 2010, pp 191 – 198.

[2] Meiko Jensen, Nils Gruschka and Ralph Herkenhoner, "A survey of attack on Web Services", Computer Science- Research and Developmemt Volume 24, Number 4, pp 185-197, Nov. 2009.

[3] Wali Negm, "Anatomy of Web Services Attack: A guide to threat and preventive Countermeasures", Forum Systems Inc 2004.

[4] Vipul Patel, Radhesh Mohandas, Alwyn R. Pais ,"Attack on Web Services and Mitigation Schemas" information Security labs NITK Surathkal Security and Cryptography (SECRYPT) , 2010, pp 1-6.

[5] Yin-Soon Loh, Wei-Chuen Yau, Chien-Thang Wong, Wai-Chuen Ho," Design and Implementation of XML Firewall" , Faculty of Engineering Multimedia University Cyberjaya, Malaysia, 2006, pp 1147 - 1150.

[6] Nuno Antunes, Nuno Laranjeiro, Marco Vieira, Henrique Madeira ," Effective Detection of SQL/XPath Injection Vulnerabilities in Web Services" IEEE International Conference on Services Computing 2009, pp 260 -267.

[7] Dave Wichers Input validation cheatsheet (1st edition) [On-line] available: www.owasp.org [Oct 7, 2011]

[8] Matteo Melluci (1999, June 1). OWASP Testing Guide (1st edition). [On-line]. 3(1). Available: https://www.owasp.org/index.php/OWASP_Testing_Project [Dec 9, 2008].

[9] Vic (J.R.) Winkler and Bill Meine, Securing the Cloud Cloud Computer Security: Techniques and Tactics Amsterdam,Boston,Elsevier Inc 2011, pp 123-35.

[10] Bertino E, Martino L, Paci F, Squicciarini A ,"Security for Web Services and Service Oriented Architectures", Springer, 2010,XII, pp. 218.

[11] Dave Wichers, Jim Manico SQL prevention cheatsheet (1st edition) [On-line] available: www.owasp.org [Dec 21, 2011]

[12] Michael Coates, Dave Wichers Transport Layer protection cheatsheet (1st edition) [On-line] available: www.owasp.org [Oct 16, 2011]

[13] Dave Wichers, Jim Manico Web Service Security cheatsheet (1st edition) [On-line] available: www.owasp.org [Dec 22, 2011]

[14] Don Patterson, "XML firewall Structure and Best practices for Configuration and auditing", Sans Institute InfoSec 2007.

[15] Jeff Williams, Jim Manico XSS prevention cheatsheet (1st edition) [On-line] available: www.owasp.org [Feb 23, 2012]

[16] Gruschka N., "Vulnerable Cloud: SOAP Message Security Validation Revisited", IEEE International Conference on Web Services 2009, pp 625-631.