CHAPTER 1

INTRODUCTION

1.1 WEB SERVICES

Web Services are offer services to users with eXtensible Markup Language (XML) technology, which facilitates messaging, service description through better representations, discovery and extended features. Moreover, a Web Service platform provides open standards for distributed computing including features for interface descriptions and document exchange through messages. Moreover, Web Services are provided independent of the underlying technology and application platforms. They can be extended for enhancing the enterprise Qualities of Service with respect to security, reliability and transactions. Finally, Web Services support for composite applications in terms of flexibility and features for business process flows, multi-channel access and rapid integration.

1.1.1 Security Issues in Web Services

The advances in Web Services technologies have extensive and remarkable effects on the Internet and enterprise community. Web Services based on XML, Simple Object Access Protocol (SOAP), and related open standards allow the bases data and applications to interact directly without the intervention of human through dynamic and ad hoc connections. Web Services technologies are implemented with a wide variety of architectures and they co-exists with other existing software technologies and design approaches. Therefore, they can be adopted in an evolutionary manner
without the need for major transformations in both legacy applications and databases. In short, The features that make Web Services usable, include fast access to huge amount of data, dynamic connections between applications, and autonomy in operations (lack of human intervention). In spite of all these advantages, the security challenges presented by Web Services are alarming and serious. The major security challenges include the maintenance of confidentiality and integrity of the data while transferring through Web Services protocols. Moreover, the functional integrity of the Web Services require the establishment of trust between services.

1.1.2 Challenges of Web Services

Though most of the challenges on Web Services have been addressed by the existing standards, there are still several new challenges that are to be addressed for providing secure Web Services discovery and reliability features. The Web Services Interoperability Organization (WS-I) acknowledges that there are many challenges that have yet to be addressed (Bertino et al 2010). Some of the important challenges are:

- Unpredicted form of script injection attacks
- Repudiation of transactions
- Secure issuance of credentials
- Exploitation of covert channels
- Spread of malwares, such as viruses and Trojan horses through SOAP messages
- Denial of Service attacks
- Incorrect Service implementations.
1.1.3 Counter Measures

Generally, Web Services are developed as built-in functions in their native software packages. However, these functions are not able to prevent enormous types of attacks in various forms. Therefore, many researchers have proposed solutions for detecting and preventing attacks targeting Web Services. The existing solutions are divided in two categories namely schema validation and schema hardening.

Before discussing the countermeasures for Web Service attacks, it is necessary to discuss the differences between Web Applications and Web Service applications.

1.2 WEB APPLICATION

An application that requires the user to access it over the internet is called a Web application. Generally, any software, which is accessed through a Web browser, can be called as a Web application (ref...). While using Web applications, users need install any software for developing and maintaining the application. Moreover, Web applications work in different platforms. It is very easy to use a Web applications since the only requirement is the availability of a Web browser. Due to all these reasons, Web applications have gained a fast and immense popularity. Popular Web applications in the internet include Web mail applications, online auctions, Wikis, etc. Generally, Web applications are organized into tiers, where each tier is responsible for a specific task. In the past, Web applications were made up of using single tier. On the other hand most Web applications are built now on three-tier architecture and also some complex applications use n-tier architecture. In the three-tier architecture, the three tiers are dedicated for presentation, application (or logic) and storage from top tier to bottom tier. Skaruz & Seredynski (2004, 2009, 2010) presented a novel approach based on
applying a modern Meta heuristic Gene Expression Programming for detecting web application attacks.

1.3 WEB SERVICE

A Web Service is a system of softwares that allows different machines to interact with each other through internet. Web Services use Extensible Markup Language, Simple Object Access Protocol, Web Services Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI) open standards to achieve this task. The XML language is used in Web Services to send messages to system with different platforms and programming languages where XML is used to tag data effectively.

SOAP is a protocol that is based on XML allows applications to communicate over Hypertext Transfer Protocol (HTTP) and it is used for accessing a Web Service. WSDL is used to describe and locate a Web Service. Web Services are mainly used to achieve reusability of application components. Web Applications Services such as weather reports, real estate trends and share market are widely popular among the public. Therefore, without developing them repeatedly, they are offered as Web Services, which can be easily used. Web services are capable of exchanging data between different applications running on different platforms.

1.3.1 Difference between a Web Application and a Web Service

A Web application is a software, which can be accessed only through a Web browser running on client’s machine on the other hand a Web Service is a system which allows different heterogeneous machines to interact with each other through the internet. Mostly, Web Services do not necessarily have a user interface. This is due to the fact that it is used as a component in a Web application which itself has a Graphical User Interface
(GUI). Moreover, Web Services are used in communication or data transfer between Web applications that are running on different platforms. Web Services are platform and language-independent due to the use of XML for data exchange. Web applications can make use of the features provided by other Web Services through calling their methods.

### 1.3.2 XML Web Services

XML Web Services use eXtensible Markup Language messages that follow the SOAP standard and have been popular with the traditional enterprises. In such systems, there is often a machine-readable description of the operations offered by the service written in the WSDL. XML Web Services are the essential building blocks in the move to distributed computing on the Internet. Open standards and the focus on communication and collaboration among devices and applications have created an environment, where XML Web Services are becoming the platform for application integration. Such as, the user sends instructions from mobile device to home appliances. Applications are constructed using multiple XML Web Services from various sources that work together regardless of where they reside or how they were developed.

### 1.3.3 Platform Elements of Web Services

The basic architecture of Web Services (Christian Geuer-Pollmann & Joris Claessens 2005) is shown in Figure 1.1 includes Web Services technologies capable of Exchanging messages, describing Web Services, publishing and discovering the Web Service descriptions.
Figure 1.1 Basic architecture of Web Services

This Web Services architecture defines an interaction between software agents as an exchange of messages between service requesters and service providers. Requesters are software agents that request the execution of a service. Providers are software agents that provide a service. Agents can be both service requesters and providers. Providers are responsible for publishing a description of the services they provide. Requesters must be able to find the descriptions of the services.

The basic Web Service architecture models has the interactions between three roles: the service provider, service discovery agency, and service requestor. The interactions involve publish, find and bind operations. These roles and operations act upon the Web Service artifacts namely the Web Service software module and its description. In a typical scenario, a service provider hosts a network accessible software module (an implementation of a Web Service). The service provider defines a service description for the Web Service and publishes it to a requestor or service discovery agency. The service requestor uses a find operation to retrieve the service description locally or from the discovery agency (i.e., a registry or
repository) and uses the service description to bind with the service provider and invoke or interact with the Web Service implementation. Service provider and service requestor roles are logical constructs and a service may exhibit characteristics of both.

Requesters and providers interact using one or more Message Exchange Patterns (MEPs) that define the sequence of one or more messages exchanged between them. A service description is hosted by a discovery service, to which a provider publishes the description, and from which the requester discovers the description. The description includes data type and structure information, identifies the MEP, and contains the address of the service provider.

1.4 WEB SERVICE AND COUNTERMEASURES

1.4.1 Introduction to Attacks

The vulnerable Web and Web Service pages are deformed by attackers. In the past many researchers proposed wide range of solutions to defend it (Axelsson 2000, Wang et al 2009, Khaleel Ahamed et al 2011). Since attackers are introducing different kinds of hits with advanced technologies, it is very difficult to maintain updated black list in the security system. The security programmer must understand the way of hacking to find the vulnerable points of Web pages. Ethical hacking expertises the researcher in producing the defending mechanism to protect the Web attacks efficiently. Hence, different attacks of Web Services are listed and solutions explained to mitigate it.

A Web Service is defined as a software system designed to support interoperable machine-to-machine interaction over a network. Moreover, Web Services provide a framework for system integration, independent of programming language and operating system. A Web Service is used to
implement SOA based system. It provides interoperability and loose coupling infrastructure for SOA based system. The Web Service is built on SOAP and HTTP messages to interact with the Web Services. An XML based SOAP messages enable Web Services platform independent. For example, a service written using .Net technology can be consumed from the java based clients and vice-versa.

Web Services have become dependable platform for enterprise models. Widespread adaptation of Web Services has resulted in a bunch of standards such as WS-Security, WS-Trust, WS-Addressing etc., to support business and security requirements for the same. The XML provides foundation for core entities on which Web Service infrastructure is built. These entities comprise SOAP, WSDL and UDDI. Large adaptation of Web Services for businesses is endorsed to excess of rich features and advantages it offers. The salient features of Web Services are listed below.

- Uses open, XML-based standards, which enable components written in different languages and for different platforms to communicate.
- It helps to significantly reduce the costs of enterprise application integration and B2B communications.
- They are more helpful for the implementation of loosely coupled distributed system. One business process component can easily be migrated without modifying other components.

The entities of Web Service have been developed using XML. It provides a common, independent data format across the enterprise, pervasive technology for defining for defining business documents and exchanging business information. Before discussing the attacks in Web Services, the components of Web Services must be discussed.
1.4.2 Components of Web Services

The Web Service infrastructure can best be explained with respect to three entities: SOAP, WSDL and UDDI (Cremonini 2003).

1.4.2.1 SOAP

It is an XML based terminology that enables programs on separate computers to interact across a network. The Web Service is invoked by sending a message called “SOAP request” and the response returned is called SOAP response. The basic skeleton of SOAP request is shown below.

```xml
<?xml version="1.0"?>
<soap:Envelope>
  <soap:Header>
  </soap:Header>
  <soap:Body xmlns:m="http://www.E-Banking.org/temp">
    <m:GetAmount>
      <m:UserAccount>ICCI23df3j78hj</m:UserAccount>
    </m:GetAmount>
  </soap:Body>
</soap:Envelope>
```

The Envelope is root level element which covers whole request. The request has mainly two components; Header and Body. The header is mostly used to carry information specific to Web Service standards. The body identifies exact operation being called along with parameters.

1.4.2.2 WSDL

The Web Services Description Language is another XML based component that allows developers to describe Web Services and their
capabilities in a standardized format. It specifies the address (endpoint) at which service is available. Also, it depicts names of operation parameters and equivalent data types. This information is then used to generate Web Service clients capable of producing SOAP request adhering to message structure dictated in WSDL and also it can process SOAP response.

1.4.2.3 UDDI

The Universal Description Discovery and Integration is a framework that defines XML-based registries in which businesses can publish information about services they offer. An organization publishes WSDL documents in UDDI which is the equivalent of an on-line service directory. Business partners can access these documents and search for a particular Web Service.

![Figure 1.2 Communication flow between Web Service components](image)
First, the Web Service provider registers their service into UDDI registry to make it available to the client. The client browses through UDDI registry to locate available services. After locating the desired service, client can now refer to WSDL of the service. Client may use this WSDL to generate Web Service proxy. Now, client sends SOAP message request and service returns SOAP message response as shown in Figure 1.2.

### 1.5 VULNERABLE WEB SERVICES

An increasing significance has brought the Web Services under the scanner of attackers; as a result of which Web Services have started experiencing instability. The Web Services are offered over HTTP similar to Web applications which makes them vulnerable to attacks to which most of the Web applications are vulnerable. Also, there exist attacks that are specific only to Web Services. Some of these attacks are attributed to immaturity of the standards; some of them are still in nascent stage which causes some error scenario to slip in incidentally. As SOAP messages are XML based, Web Services are prone to XML vulnerabilities specifically those which target XML parser. Common attacks among the Web Services are XML injection, SQL / XPath injection, XSS injection, malicious SOAP attachments, XML Denial of Service (XDoS), coercive parsing and buffer overflow attack. The conventional firewall can’t really address those vulnerabilities.

Consider a scenario where an online banking has established a set of banking services to allow users to query their account details. The users require updated information for every transaction. Certain tellers would also need this information quickly and frequently, so that they can respond to emerging changes in the account. As shown in Figure 1.3, a user can access the Account Information Service (AIS) and Transaction Service (TS) through a real-time banking service. A banker may provide online banking service to a user through the same application. In addition to the stock prices, The AIS is
designed as an internal service because it can only be accessed by a
authenticated user or a system administrator.

Since users verify their accounts, the endpoint address of a AIS
must be configured by an administrator in an XML settings file. In addition,
the XML settings file also contains authentication information of users. Thus,
at runtime, the AIS needs to make an XPath query to authenticate a user, read
the location of the adopted AIS from the XML settings file, and invoke AIS
dynamically. For example, AIS requires invocation of TS in order to provide
transaction service.

![Figure 1.3 XML-based attacks on Online Banking service providers](image)

Figure 1.3 XML-based attacks on Online Banking service providers

Now suppose there is an attacker who attempts to compromise the
banking service. The attacker first tries to attack AIS or TS using an
overloaded payload attack or XDoS attack; however, since these services
were designed for concurrent access with a large volume of data by many
users, they are typically very robust. Upon failure on such attempts, the
attacker turns his attention to the internal service AIS. As expected, internal
services are not published to the public and can be accessed only by
authorized personnel, thus the information for invocation of AIS is unknown
to the attacker. In order to capture such information, the attacker pretends to be a normal user and utilizes an XPath injection attack to the public service AIS. The XPath attack brings the related information regarding the internal service AIS such as its endpoint address, credentials to access the service and connection timeout and limits that should be imposed on the response. Based on the captured information, the attacker performs a hybrid of XDoS/overloaded payload attack to compromise AIS by combining elements of an XDoS attack with elements of an overloaded payload attack. Since the above attack consists of two major steps, namely XPath injection attack and a hybrid of XDoS/overloaded payload attack.

1.6 ATTACKS ON WEB SERVICES

Web Services are standardized and there hardly exists any proprietary protocol and hence security through anonymity does not go well with the Web Services. Web Services are made available in much the same way as Web applications through HTTP protocol which makes them vulnerable to same set of attacks to which Web applications are also vulnerable. In this research, three classes of attacks are carried broadly, those are injection attacks and authentication based attacks. At high level, injection attacks on Web Services can be classified as script based injection attacks and XML based injection attacks. The script injections are the various form of malicious scripts inserted into vulnerable Web Services. These malicious scripts are written in HTML, Java script, VB script, flash script etc,

Oversized message attack, XML based injection attacks, code injection attacks and brute force attack. The DoS attack can be launched either in a conventional way by flood server with more number of requests than it can handle or by sending malicious payload. Oversized payload, recursive XML payload, SOAP message with large number of encrypted elements are some examples of malicious payloads.
The WSDL is an open mechanism that can reveal sensitive service. Sometimes, selected Web operations are meant for internal use and not supposed to be made available to outsiders. Most of the WSDL are generated using automated tool which publishes information pertaining to all Web operations and hence discloses operations intended for internal use. WSDL may also reveal what tools generated the Web Service providing attackers with more information on the environment. WSDL documents are the first point of weakness because they may be used as a “guide book” for vulnerabilities. Using this information, a security breach can penetrate deep into the business logic.

WS-Routing is a standard for sending SOAP messages from an initial sender to the target receiver typically via a set of intermediaries. The routing information is assorted inside the SOAP header. The Routing Detours are a type of "Man in the Middle" attack where intermediaries can be hijacked to route sensitive messages to an outside location. Routing information inside SOAP header is modified at compromised intermediary.

The Web Service engine running on the server does the job of de-serialization by parsing SOAP request and converting extracted parameters to native types. Injection attacks alter parameters sent as a part of SOAP request that are subsequently processed by the business logic. If a message generated by a client is tweaked a bit then the validation on a server side can easily be bypassed without getting noticed. Also, some implementations of Web Service engine are not powerful enough to perform rigorous schema validation. These issues were observed with Web Service engine running for .Net Web Services.
This work describes attacks targeted at Web Services such as XML injection, SQL injection attacks, XPath Injection Attack, Buffer Overflow Attack, Message Replay Attack, Parameter Tampering Attack, Coercive Parsing Attack, Semantic URL attack, XSS injection, HTTP header manipulation, SOAP attachment based attacks and XML re-writing attack. These are most common attacks on the Web Services. XML injection attack comes handy for the attacker to launch DoS attack and manipulate operation parameters. The attacker uses hacking tool to break the vulnerable Web Services. The hacking tool imports the SOAP request and response with its WSDL documents. Here, this tool allows the user to change attributes of WSDL elements. After modifying the values, the tool converts that as permanent values and forwards the request to the service provider. Familiar hacking tools are burb, WS-Chess, Brute forcer.

1.6.1 Script Based Injection Attacks

Web developers are alarmed with attacks that cause a malicious script, typically written in JavaScript, to be added into the content of a trusted Web site. While a visitor views a page on the site, the added script is loaded and executed in the visitor’s browser with the trusted site’s privileges. The injected script can gain privileged information (browsing history, cookies and any confidential content from the site). The script can also utilize the visitor’s browser to generate denial of service attacks or other attacks on the Web site. If the Web site is very popular, the attack can be greatly amplified.

Script injection can be achieved in many ways. In cross site scripting, the attacker often exploits Web sites that insert user-provided text into pages without properly filtering the text. For example, users in on-line communities like MySpace, Blogger and Flicker may enter their own content and add comments to the content of others. This content is stored on the site and may be viewed by anyone.
Because different users’ profiles are hosted by the same site, if a malicious user were able to include a script in his content, any viewers of that content would run the script with the privileges of the site. This would allow the malicious user’s script to steal or modify the viewer’s content, including private information stored at the site or at the browser (e.g., as a cookie). Another way of injecting a script is by “reflection.” For example, when asked for a non-existent page, many sites try to produce a helpful page in response, with a “not found” message that includes the URL of the non-existent page that was requested. Therefore, if the site is not careful, an occurrence of the text `<script>...</script>` in the URL can be executed on the visitor’s browser when it renders the “not found” page. To exploit this, an attacker can try to entice victims to follow URLs with targets that include scripts, e.g.,

```
```

### 1.6.1.1 XSS injection

In XML, a CDATA section is used to get away a block of text that would not be parsed as mark up. Characters like "<" and "&" are considered illegal in XML when used as a content of some element. If an attacker wants to insert java script then it becomes essential to enclose script within “<Script>” tag. To keep away from XML parsing errors, script code can be defined as CDATA. Everything inside a CDATA section is ignored by the XML parser i.e. it passes as it is to the application logic. Hence, this is a powerful feature that enables an attacker to successfully inject the JavaScript. The code snippet in Figure 1.4 explains how the java script is embedded in the CDATA element and executed from the data collector of Web Service that simply stores data passed as an operation parameter.
It is obvious that ‘<’ and ‘>’ of script tag have been enclosed inside CDATA tag. Presence of these prohibited symbols goes unnoticed by XML parser and they are supplied as a part of a method argument. Our Web Service simply dumps it in a data store which is subsequently consumed by Web Application that merely does the job of data rendering. Web app reads this data source and displays uploaded content in the form of html tables. If stored data contains active content such as JavaScript then it easily paves way for an XSS, Cross-Site Request Forgery (CSRF) or other attacks.

1.6.1.2 SQL injection attack

Simple SQL statements are inserted into the database using the input fields of the SOAP message by the attacker. If the statements are not sanitized, the attacker may gain access to the databases. This attack uses SQL commands such as SELECT, CREATE, UPDATE, DELETE, DROP, 1=1, ALTER and INSERT. These commands are used to embed malicious input to client side interface.

```xml
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    </BookSearch:searchByISBN>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Figure 1.4 Javascript embedded inside CDATA**
<BookSearch:searchByIBSN/>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

This message is processed by the following C#.NET code, which inserts the content of the ISBN element into a SQL statement:

```csharp
Set myRecordset = myConnection.execute("SELECT * FROM my Books Table WHERE ISBN = '" & ISBN_Element_Text & "'")
```

In the case of the preceding SOAP message, this becomes

```
SELECT * FROM myBooksTable WHERE ISBN = '23HJ567'
```

Now consider what happens when the following SOAP message is received:

```xml
<SOAP-ENV:Header></SOAP-ENV:Header>
<SOAP-ENV:Body>
<BookSearch:ISBN>"23HJ567"; exec master..xp_cmdshell 'net user admin pass /ADD'; --
<BookSearch:ISBN>
</BookSearch:searchByISBN>
</SOAP-ENV:Body></SOAP-ENV:Envelope>
```

In this case, the SQL statement will read

```
SELECT * FROM BooksTable WHERE ISBN = '23HJ567'; exec master..xp_cmdshell 'net user Alice pass /ADD'; --
```
The code after the SELECT statement attempts to create a user called ‘alice’ with password of pass. An attacker could then try to use this new user account to violate the access to the target machine.

Attempting to manipulate SQL queries violate the following benefits to the attacker

- Data Leakage
- Exposure of stored data
- Exploitation of stored data
- Detouring authorisation controls
- Consumer-side SQL Injection

1.6.1.3 XPath injection attack

An XPath is used to locate and access the elements and attributes in an XML document. The XPath is used in a “XML - Supported” databases like SQL Server 2005. In SQL, XPath uses delimiters to segregate code and data. Because, there is no access control in XML or XPath, the attacker can control data through the XPath statement to access arbitrary parts of the XML file or return arbitrary data. The sample code snippet for XPath Injection attack is shown in Figure 1.5.
In the above code snippet, the attacker generates the code to access the user name and password. This can be achieved by XPath technology in XML. The XPath finds the element and access all sub elements under the targeted root or parent element. The root element is ‘Users’ tag and the sub element of root element is ‘User’ with the attributes ‘Login’ and ‘Password’. This search query statement is assigned to xpathQuery variable. Then the query is iterated through the XML document unconditionally by the use of XPathNodeIterator class. Then returned dataset of user credentials are gathered and gained by the attacker.

1.6.2 XML Based Injection Attacks

1.6.2.1 XML Injection

The XML Injection happens when user input is passed to XML fields. Since user fields can be accessed by Web user interface through forms, XML Injection overwrites the sensitive element. For example consider the customer database defined by the following database schema. This attack tries to modify the XML structure of a SOAP message by inserting XML tags or modifying existing tags inside operation parameters. At the server side, this content is regarded as part of the SOAP message and can lead to undesired effects. XML document is created in Web Service as depicted in Figure 1.6.
In this schema, all the fields can be edited by a user. Hence, this can be used by an attacker to destroy the valuable data.

1.6.2.2 Buffer overflow attack

The attacker inserts malicious content with well-formed message, which is beyond the allowable size of the buffer and causes DoS attack. It is called as buffer overflow attack. Following code shows how the buffer overflow attack occurs:

```xml
<soap:Envelope xmlns:soap="">
  <soap:Body>
    <fn:Function xmlns:fn=" " xmlns:ns=" ">
      <DataSet xsi:type="ns:Array" ns:arrayType="xsd:string[100000]">
        <item xsi:type="xsd:string">Data1</item>
        <item xsi:type="xsd:string">Data2</item>
        <item xsi:type="xsd:string">Data3</item>
      </DataSet>
    </fn:Function>
  </soap:Body>
</soap:Envelope>
```

Figure 1.6 XML attack injection in XML document
1.6.2.3 Parameter tampering attack

The WSDL document has parameters to receive inputs from the client. The parameters are visible in a WSDL structure to all users. Here, the attacker tries to send different data types of parameters several times. Then the Web Services will have possibility to crash as shown in Figures 1.7 a and 1.7 b.

![Parameter tampering attack](Image1)

![Response from the server](Image2)

1.6.2.4 Coercive parsing attack

The attacker sends a SOAP message with an unlimited amount of opening tags in the SOAP Body. It means the attacker sends a very deeply nested XML document in to the targeted Web Service. If the parser receives a peculiar format of SOAP messages, it reduces its processing capability and this may result Distributed Denial-of-Service attack. For example, consider the following code
The nesting provided by tag <x> will lead to an infinite loop which leads to a DDOS attack.

1.6.2.5 Semantic URL attack

In a semantic URL attack, a client manually adjusts the parameters of its request by maintaining the URL's syntax but altering its semantic meaning. This can be avoided by giving token and timestamp for expiration. The existing Web Service allows the user to reset their password by answering the security question correctly and allows the users to send the password to the e-mail address of their choosing. The receiving page has all the information it needs to send the password to the new e-mail. The hidden variable username contains the value uid001, which is the user identification of the e-mail account as shown in the URL http://urlsemanticdemo.com/resetpwd.aspx?userid=uid001&altemail=alternative%40emailexample.com. When this URL appears in the location bar of the browser, it is possible to identify the user details and the e-mail address through the URL parameters. The malicious user may decide to steal other people's (uid002) e-mail address by visiting the following URL: http://urlsemanticdemo.com/resetpwd.aspx?userid=uid002&altemail=alternative%40emailexample.com. If the resetpwd.php accepts these values, it is vulnerable to a semantic URL attack. The new password of the uid002 e-mail address will be generated and sent to alternative@emailexmaple.com which causes uid002's e-mail account to be stolen.
1.6.3 Authentication Targeted Attacks

1.6.3.1 Message replay attack

A message replay attack is one in which an attacker eavesdrops and obtains a copy of an encrypted message and then re-uses the message later in an attempt to reveal the secret messages or to provide a fake identity. For example, when a legitimate client transfers money from his account in bank to the receiver, the attacker steals the password and uses it multiple times by sending it to the Web Service in order to cause money loss. Figure 1.8 shows the flow of Replay attacker in Web Service platform.

1.6.3.2 Brute force attacks

A method of breaking a cipher text is by trying all possible combinations of keys. The feasibility of brute force attack depends on the key length of the cipher, and on the amount of computational power available to the attacker.
1.6.3.3 Dictionary attacks

A dictionary attack is a technique for defeating a cipher or authentication mechanism by trying to determine its decryption key by searching likely possibilities. A dictionary attack uses a targeted technique of successively trying all the words in an exhaustive list called a dictionary (from a pre-arranged list of values).

1.6.3.4 Cookie replay attacks

The attacker can read authentication information that is submitted for the application to gain access. The attacker can then replay the same information to the application causing cookie replay attacks.

1.6.3.5 Credential theft

The server has to maintain its own user data that contains name of user id, password and other sensitive information in the database. In client side, the user information has stored by browser history and cache for future use. If the terminal is accessed by someone other than the user who logged on, and the same page is hit, the saved login will be available is called as credential theft.

1.7 COUNTER MEASURES OF ATTACKS

All legitimate or illegitimate input should be thoroughly analyzed by the filters installed in Client/Server side. We have surveyed so many research papers and listed. The computer science research community has been actively involved in finding attacks targeted at Web Services and designing mitigation techniques. The attacks documented in the literature can fall into one of two categories. First category of mitigation mechanism is protecting the attacks through static filters. In this approach the nature of
attacks are studied and filtered directly through some regular expression validator or string comparing filter. Next category falls under dynamic approach. In this approach the attacks are studied and refined by dynamic filters. The general skeleton of attack filter is formed and analyzed by the filter. The static filter requires updated blacklist of attacks. The attack not appeared in the blacklist could be escaped in static filter. The dynamic approach would filter all possible attacks in the network. Since, it is designed to analyze the nature of input.

1.7.1  Static Approach

1.7.1.1  Updated blacklist

Inputs are compared with existing attacks format by maintaining the recent attacks list in server side.

1.7.1.2  Regular expression validator

All parameters should be validated before executed or stored in server side. The validator is utilized to filter the attacks.

1.7.2  Dynamic Approach

1.7.2.1  Taint based analysis

Filters have to be designed to increase security by preventing malicious users from executing commands on a host computer. Taint checks highlight specific security risks primarily associated with Web sites which are attacked using techniques such as SQL injection or buffer overflow attack approaches.
1.7.2.2 **HTTP request analysis**

By submitting unexpected values in HTTP requests and viewing the Web application’s responses, filter can identify places where tainted parameters are used.

1.8 **OBJECTIVES OF THE THESIS**

Even though a lot of detecting and preventing algorithms have been developed, just a few of them are sustainable to be used in real-time Web applications. New self-aware message analyzing and validating algorithms against attacks on Web Services are proposed in this research which can extensively be used as a detection and prevention system. The objectives of this research are:

- To study of attacks in various levels and to understand the existing countermeasures.
- To survey the various protection mechanisms of existing firewalls.
- To develop new architectural model for injection filter and authentication services to prevent injection and authentication based attacks.
- To develop efficient algorithms for real-time detection and prevention of attacks and to make a comparison with the existing algorithm.
- To develop a code setup to demonstrate the detection and prevention of attacks practically.
• To study the performance of various conventional firewalls and to submit a performance analysis report to show the robustness of proposed security filter and to create a graphical user interface.

1.9 ORGANIZATION OF THE THESIS

The thesis titled “Dynamic Security Algorithms for the Detection and Prevention of Attacks in Web Services” protects the Web Services that can be used in e-business application to maintain integrity of the sensitive data. This dissertation is divided into seven chapters, including the introduction.

Chapter 1 introduces the need for Web Service security to protect from Web attacks and authentication. In addition, it explains the new techniques proposed in this research work for enhancing the Web security.

Chapter 2 provides the related work in the area of Web services security.

Chapter 3 depicts the architecture of the security system proposed in this research work.

Chapter 4 explains the new techniques proposed to detect and prevent XSS attacks. A detailed step-by-step procedure of self aware schema validation implementation is given in this chapter.

Chapter 5 deals the XML injection based attacks and proposes new techniques for the detection and prevention of such attacks. It also
explain the secured architecture proposed in this work which uses SOAP message validating approach to protect the server.

Chapter 6 discusses the proposed nonce based authentication scheme and explains the development of this approach for low-end systems. A comparison is made between the existing algorithms and the proposed algorithms.

Chapter 7 gives the conclusions on this work and suggests some possible future works.

1.10 CONCLUSION

The basic architecture of Web Services and its components have been discussed. Then different kinds of attacks are listed and solutions carried in existing systems. In addition, we proposed http request analysis approach. It is a dynamic approach to protect system from attacks. The motivation for the present work has been brought out. The organization of the thesis has also been presented briefly.