CHAPTER 3

SYSTEM ARCHITECTURE

3.1 INTRODUCTION

Architecture of the Web Service security system proposed in this work is shown in Figure 3.1. This proposed architecture for Web Service Security was implemented with three major components namely script based injection filter service, XML based injection filter service and Authentication service. This system has been implemented using C# sharp in .net framework.

The filtering policies are implemented to detect and prevent the intruder’s attack. It is accessed before the service providers of business concerns. The request of client is passed through injection filter services and authentication service. It is not easily identified as legitimate request or illegitimate request using single attack detection system. For this reason, the service providers are in need to set a big set of attack detection system because the attacks can be of any type and in any form. Hence, the injection filter and authentication service is installed before accessing the service provider in business concern.

The filtering policies are script based injection filter, parameter tampering filter, coercive parsing filter, oversized message filter, message replay filter and semantic URL filter. It receives XML document through parser and validates each and every lines of statements. If it encounters any anomaly request, then it sends alert response to the attacker. If there is no anomaly in authentication request then the request is forwarded to real
authentication service. At this time, the user name, password and related
dynamic random nonces are verified using database in the authentication
service.

Figure 3.1 Architecture of the proposed system for the detection and
prevention of attacks on Web Services

Thereafter, the legitimate client gets verified and received access
for corresponding business services from the authentication service.
Otherwise the illegitimate attacker is identified and blocked. These filtering
policies are built with SAX parser which is faster than DOM parser. Thus, the
system was implemented with SAX parser to improve the filtering speed of
the filter. The proposed system has been built to filter the injection attacks in
Web Services. It is designed to filter attacks using a self-aware XML message
validating algorithm and a nonce based authentication scheme.
3.2 USER INTERFACE

This is built at client side, it contains interface to supply user credential. It includes random nonce generator to combine the user credential with random numbers.

3.2.1 Interface to Supply User Credentials

It gets user name and password from the user and it is concatenated with random nonces. Then it is forwarded into authentication service by the user interface.

3.3 SCRIPT BASED INJECTION FILTER

The proposed system consists of two major phases: Dynamic and Static filter.

3.3.1 Dynamic Filter

This filter checks the request for malicious contents by forwarding the request to dummy service provider and process request in it with dummy database. Then the data from the dummy service provider sends back to the requester. Here, the request and response are analyzed by significant components of injection filter. The ASCII converter retrieves the input in any form then, it verifies the encoding format of the request. If it is not in native format of the service provider, then the converter converts this to native format of the server. Later, it forwards request into dummy service provider.

Dummy service provider and database receives the input from the user and performs validation with dummy page. Then it responds back to user. Flow analyzer component is implemented with attack tester which holds the dummy service provider with dummy database. The attack tester contains
HTTP request/ response analyzer and load analyzer. HTTP request/ response analyzer and load analyzer module captures the response then analyzes all fields of the header. At that time, it matches the nature of requested data from HTTP request header and also the nature of expected data from the dummy database. If they do not match with the expected database then, it checks set of alert HTTP response codes like server redirection. Blacklist maintains the distrust list for attackers. It helps the filter for repeated attacks of the adversary.

3.3.2 Static Filter

It identifies the malicious script characters in the received input. Next, it validates the input type, length, format and range.

The server-side code has been designed to constrain input supplied through client-side input controls or input from other sources such as query strings or cookies for XML injection, SQL injection, XPath Injection and cookie replay detection.

3.4 XML BASED INJECTION FILTER

The architecture of the validation approach is composed of important components in XML based injection filter.

3.4.1 Parameter Tampering Filter

The attacker adjusts the parameters in a SOAP message in an attempt to redirect the input validation in order to access unauthorized information. This filter checks the XML schema definition of received message for data type, null values.
3.4.2 **Coercive Parsing Filter**

This filter verifies the namespaces and version mismatch received in the WSDL and SOAP files. This filter blocks the input that has a strange format. This policy used the values in SOAP fault code. They are version mismatch and must understand fault code.

3.4.3 **Oversized Message Filter**

The XML parsing of the service provider is directly affected by the size of the SOAP message. Hence the filter does the checking of three important parameters for the received SOAP message. First, it sets a request timeout to prevent infinite delay attacks. Then, it limits the amount of data that it retrieves. Last, it restricts the message from retrieving resources on the local host.

3.4.4 **Message Replay Filter**

A hacker can resend the SOAP message requests to access the Web Service using other’s login credentials. The filter assigns an identifier for each incoming message and stored into the database to identify the replayed message.

3.4.5 **Semantic URL Filter**

The semantic URL attack is the client manually retypes the parameters of its request by keeping the URL’s structure but altering its semantic meaning. This is protected by giving token and timestamp for expiration.
3.5 AUTHENTICATION SERVICE

The proposed system has been implemented using dynamic nonce for validating the user with username and password which are embedded with WS-Security. The Dynamic nonce has been implemented with the user’s mouse movement by satisfying the condition given in the proposed scheme.

3.5.1 Dynamic Nonce Generator

The nonces are generated from chaotic of computer parameters and from user’s activities. The components of this model are initiate function, core state function, refeed function, uninitiate function, robustness test function and generate function.

3.5.2 Authentication Service

The following components are involved in authentication process. Chaotic input is collected from the parameters of the system that is collected at the time of processing the instructions. This is collected from user’s mouse movement, time stamps and application identifiers. The nonces are generated from chaotic of computer parameters and from user’s activities. The DNG includes a source of chaotic input and includes a nonce source. Then it forwards the random number to hash generator. Afterwards, it is concatenated with user credentials, private keys and public key of the authentication.

3.5.3 Authentication Database

A set of claims used to prove the identity of a client. It contains an identifier for the client and a proof of the client’s identity, such as a hashed password. This also includes the information of user name and password.
which is concatenated with dynamic random nonce generator, to indicate that the issuer service identifies the claims in the credential.

3.6 WEB SERVICE PROVIDER

The authenticated user’s request gets access for the Web service provider from the authentication services. Now the authenticated user can get the data of Web Service based on their role.

3.6.1 Service Provider

The register service provider can get service from injection filter service and authentication services. These service providers register their data in protection services to prevent the access from malicious requesters.

3.6.2 Web Service Database

It is used to store and compare user realms. The data used in the Web Services are stored here.

3.7 SUMMARY

This architecture model is designed to prevent main classes of attacks with dynamic approach. The script injection violates almost all conventional filters because, of its lacking approach. It also provides suitable user interface. The authentication approach is also designed to maintain the secrecy of the user’s credentials by the use of dynamic nonce generator.