National College of Ireland

Multi-layer security framework for cloud application

MSc Research Project
Masters in Cloud Computing

Akash Sadanand Hande
Student ID: X17156220

School of Computing
National College of Ireland

Supervisor: Dr. Sachin Sharma

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Akash Sadanand Hande		
Student ID:	X17156220		
Programme:	Cloud Computing	**Year:**	2018
Module:	MSc Research Project		
Supervisor:	Dr. Sachin Sharma		
Submission Due Date:	20/12/2018		
Project Title:	Multi-layer security framework for cloud applications		
Word Count:	7314	**Page Count**	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: .....

Date: 20/12/2018

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Multi-layer security framework for cloud application

Akash Hande
X17156220

Abstract

Rapid movement of financial and business applications towards cloud platforms not only increases mobility and flexibility but also introduces different vulnerability issues. Considering increased transactions and security attacks such as SQL Injection and Cross-site scripting single security breach could harm the reputation as well as the quality of service of a cloud application. Detecting and preventive solution are grouped into two broad categories i.e. third-party tools such as Intrusion Detection System or programming language dependent solution such as AspectJ. This paper proposes Multi-layer Security Framework which enables solution provider to design and develop self-defensive cloud application with respect to SQL Injection as well as cross-site scripting attacks. To defend against SQL Injection and cross-site scripting attacks, MLSF proposes security patterns and malicious script analyzer which is based on similar characters in SQL queries, scripts and user input. The experimental result clearly shows that MLSF not only recognizes attacks but also defend all of them.

Keywords: SQL Injection, cross-site scripting, the Multi-layer security framework

1 Introduction

Cloud environment provides a great platform for business applications however, the mobility of cloud computing attracts cybercriminals to make use of system resources and vulnerability to make more attacks. As millions of users are sharing computing resources these attacks are very harmful and effective. Cloud applications inherit the vulnerabilities of web applications and become victims of SQL Injection and cross-site scripting attacks [1]. Majority of attacks opposed to data are due to SQL Injection which allows an attacker to take complete control of the database [2][3] as well as cross-site scripting which allows an attacker to implement or execute harmful script in user's browser [4]. Open Web Application Security Project i.e. OWSAP is an international open community of cybersecurity professionals who are motivating software professionals to design, develop, maintain and acquire software application which can be trusted [5]. OWSAP Top 10 is an awareness document released by OWSAP with the help of security experts from the international background for web-based application development which highlights secured coding practices [6]. This document confirms that SQL Injection and cross-site scripting are the most challenging and increasing issues in web application development which could harm the quality of cloud application.

SQL Injection and cross-site scripting attacks are consistently on the peak of OWSAP Top 10 list for critical application security risks and most common vulnerability issue in SaaS application [6][7]. SQL Injection is basically a code injection attack in which malicious SQL script is formed by entering some special characters into the input fields from the presentation layer of SaaS application used to create dynamic SQL queries [6][7]. Several special characters such as <, >, -, /, =, ‘, “, #, SPACE along with standard keyword from SQL language such as UNION, DROP, it is possible to formulate harmful scripts for a database of web-based application [6][8]. Similarly, cross-site scripting is a code injection attack in which attacker implements and executes the malicious script on the user's browser

[4]. Special characters such as <, >, -, =, ‘, “ are considered harmful characters in respect to cross-site scripting attacks [4][8]. Figure 1.1 shows the general architecture of cloud-based application which includes presentation tier, business tier and data tier. The presentation layer is composed of UI component e.g. HTML and UI Process component e.g. JavaScript. Business tier is responsible for implementing business workflow, components and entities which is logical and domain-specific representation of the application. The service layer is an additional layer in MVC application which mediates communication between view and controller. Data tier is logical database management unit that defines all database objects and how the application is accessing them which composed of data access components, data helper and service agents e.g. connection pooling, JDBC, helper classes etc.

The attacker uses special characters and keywords to penetrate through these layers especially from the presentation layer and takes control over the database [6]. SQL Injection flaws are very usual in legacy code and very easy to discover while examining the code and error pages. These vulnerabilities can be found in SQL, NoSQL, LDAP queries, expression languages, SMTP headers and ORM queries which results in corruption, data loss, disclosure to unauthorized parties, denial of access and sometimes lead to complete database takeover [6]. Cross-site scripting attacks can be easily found with automated tools which can be used for DOM XSS, remote code execution on user’s browser, serve for stored XSS, stealing session information/credentials etc [6].

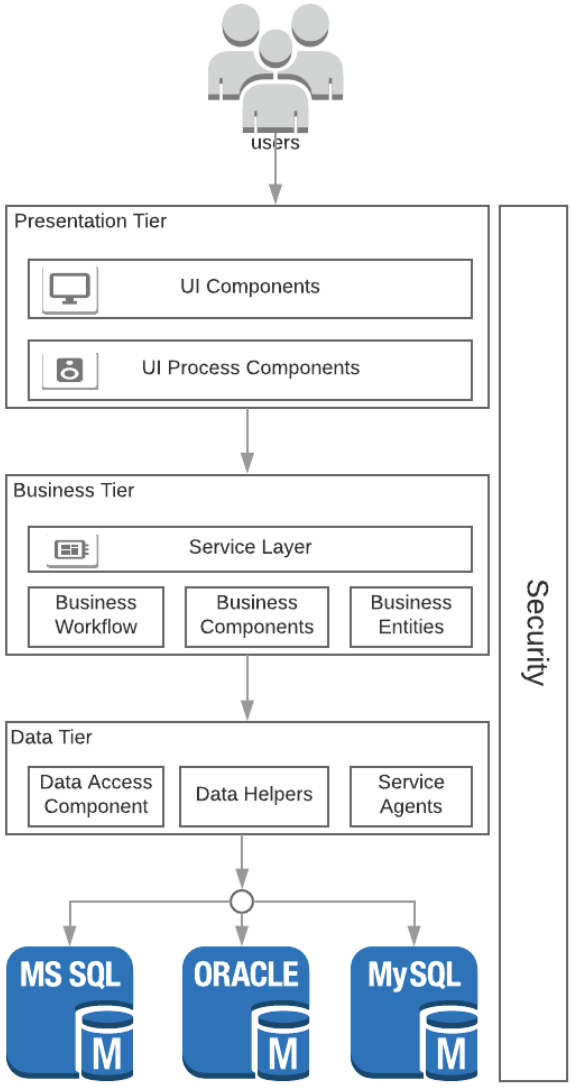

Figure 1.1 Structure of cloud application

Majority of proposed countermeasures of SQL Injection and cross-site scripting expects costly analysis tool or massive modification in the source code [7][9][10]. This is the main reason that software application needs to dependant on specific programming language and framework (e.g. PHP, J2EE). Most of the existing solutions are third-party solutions which analyze SQL and HTML traffic which increases spending on the security of cloud-based application [11][7]. Although these are effective solutions, all these approaches have not evaluated on cloud computing environment to protect cloud applications. In recent years, many researchers proposed various approaches to counteract SQL Injection as well as cross-site scripting which is dependent on the specific programming language or environment [4][8].

This paper proposes a framework for self-defensive cloud application which defends against SQL Injection attacks based on input filtering and trait analysis of SQL Injection attacks. Multi-layer Security Framework (MLSF) provides integrated, defensive and multilayer programming framework for the cloud application development. MLSF introduces security patterns and malicious script analyzer on different layers of cloud application for detection of harmful special characters and keywords which could take over a database of the application. Instead of depending on external monitoring tool or specific programming language cloud application can be developed or migrated with MLSF which increases self-defensiveness of cloud application and quality of service in terms of security. This paper is structured in four major sections. Section 2 discusses related work and existing solutions. Section 3 describes the research methodology. Implementation is described in section 4. Next, Section 5 discusses the evaluation of the proposed framework. Finally, Section 6 draws a conclusion and future work.

2 Related Work

There are enormous investigations, studies and researches have done in previous years for detecting and counteracting against SQL Injection and cross-site scripting attacks associated with web-based applications. Many solutions have been proposed by researchers, unfortunately both the attacks still major cybersecurity threat in web applications. These applications still facing SQL Injection and cross-site scripting attacks and tools, preventing techniques are designed to counteract them are divided into two major categories:

2.1 External Solution

SQLIDaaS is SQL Injection introduction detection framework which is composed of six modules to detect SQL Injection attacks proposed by Yasin et al [12]. This framework is integrated with AWS and delivered a set of Amazon Machine Images (AMI) by which SaaS providers can subscribe for framework on-demand as well as measured and scaled as per use. Backbone ideology for this framework is monitoring of HTTP traffic (e.g. HTTP session and web session), SQL traffic (e.g. SQL query) as well as the correlation between introduction detection system (IDS) and detection of SQL injection attacks.

Patil et al. proposed a multilevel system for security of cloud application and to counteract against SQL injection along with DDoS and Brute force attacks with similar ideology as SQLIDaaS [13]. Network-based intrusion detection (NIDS) agent installed on each cloud server of the network to form a system. This NIDS agent monitors incoming and outgoing traffic flowing through bridges which are responsible for traffic control among physical servers and VMs. For counteracting against SQL Injection in their solution they proposed four phases framework which compares SQL query entered by multiple clients with predefined fishy commands which are stored at server side. If they match system blocks user of the application to avoid the attack.

A very different and much interesting solution was proposed by Rajeh and Abed in the year 2017 which is three-tier SQL Injection detection and mitigation strategy [14]. This follows dynamic, static and runtime prevention as well as a detection mechanism. As a result, this approach introduced a three-tier approach which provides security on different layers such as client tier, access tier and data server tier. This proposal recommends using proxy server within service provider of cloud data along with the central firewall. This again used to differentiate queries with pre-stored metrics to protect the firewall to cross over. The result of their test was held on three different web applications achieved 100 percent detection and it prevents all types of SQL Injection attacks along with stored procedure attacks.

Trending and more advanced deep learning approach have been utilized by Nguyen and Dutkiewicz to counteract against cyber-attacks in mobile cloud computing [15]. The backbone principle of their approach is utilizing deep learning method with sample dataset which trains the preestablished neural network in offline mode. Biologically inspired computing algorithm neural network is used to learn from observational data over the network to detect cyber attacks in the cloud environment. The three modules system proposes data collection, detection of attacks and request processing module which lies between cloud resources and mobile users to monitor doubtful request and update security policies in the dataset. The observable advantage of this proposal is a neural network algorithm can be replaced by other machine learning algorithm as per need as well as this solution covers the majority of cyber-attacks with 97.11 percent high accuracy.

Hidden Markov Model was proposed by Li et al. which is based on the observation of a log analyzing combined with statistical characterizes and feature matching [16]. In the first place, this approach builds browsing behavior models of user and attacker. Furthermore, HMM is utilized to restore user browsing procedure from user log and SQL Injection attacks will be detected by analyzing user behavior without sensitive request made by users.

The proposed approach by Wang and Hou targets SQL Injection attacks in a cloud environment and suggests SQL detection algorithm which combines poisonous script analysis and input filtering [17]. This approach provides protection to web applications while embedding with a cloud environment. This proposal based on an analysis of lexical regulations for SQL statements to obtain SQL keywords and then it analyses the syntax rules of SQL statements to form rule tree. Based on SQL syntax, rules to detect attacks ternary tree get traversed. For investigational verification, web service has been designed, developed and deployed on Ali and Amazon cloud with MySQL database and Apache2 server which not only shows successful detection of common attacks but also the replacement of code attacks with a high detection accuracy of SQL Injection.

Proposed model by Sonewar and Thosar is a web server hosted application which is based on the network to monitors the behavior of attacking situation [18]. Their research is introducing three-tier web application with static as well as dynamic behavior. While considering static and dynamic analysis, parameter filtering, parameterized query and instruction detection system proposed by Xiao et al which is a unique approach to counteract with SQL Injection attack [19]. The backbone ideology behind this is monitoring attack behavior, response and state of a web-based application under the different attacking condition and formulate pattern of the network for the next vulnerable situation.

Majority of the proposed solution in this category are based on filters which clean the malicious input and those are not capable of preventing emerging attacks. In respect to this concept web security proxy for cross-site scripting has been recommended and implemented by Vijayalakshmi and Lemma [20] which is much similar as system proposed by Xiao *et al* [19]. However, instead of depending on observation their solution is following detection and protection mechanism for secure web access. Also, a considerable difference between both solutions is they are addressing one problem at a time i.e. either SQL Injection or Cross-site scripting. This proposed system is web security proxy for cross-site scripting attacks which is composed of WSP, WS policy management, Policy filters and

malevolent report which resides between world wide web and client machine to monitor request and response.

Below table 2.1 enlists different approaches and proposal addressed by researchers.

Proposed Tool	Author	Description	Issue Addressed
SQLIIDaaS	Yassin *et al.* [2017]	Intrusion Detection framework	SQL Injection
NISD	Patil *et al.*[2017]	Network-based Intrusion Detection System	1. DDoS 2. Brute force 3. SQL Injection
SQLi detection schema	Wahid Rajesh, Alshreef Abed [2017]	Three-tier SQL Injection detection and mitigation scheme	SQL Injection
Deep Learning approach	Nguyen *et al.* [2018]	Observational network data to detect cyber-attacks in cloud systems	1. Cross-Site Scripting 2. SQL Injection
Utilization of HMM	Li *et al* [2017]	log Observation with Hidden Markov Model	SQL Injection
web service for SQL detection	Kuisheng Wang, Van Hou [2016]	Dynamically generate a ternary rule tree to detect an attack	SQL Injection
Web-based tool	Sonewar and Thosar [2016]	Network dependent monitoring tool	1. Cross-Site Scripting 2. SQL Injection
Web Security Proxy	Xiao et al. [2017]	Observation-based tool	SQL Injection
Web Security Proxy	Vijayalakshmi and Leema [2017]	Detection and protection mechanism for a secure web access	Cross-Site Scripting

Table 2.1 External solution

2.2 Programming Language Dependent Solution

In software programming, Aspect-oriented programming is paradigm which increases modularity while separating tasks generally spread through different part of the system. This concept was utilized in an Aspect-oriented system for defeating against SQL Injection and cross-site scripting proposed by Kajo-Mece et al [8]. This intrusion detection ideology is implemented by AspectJ and embedded into a target system which scans input strings and compares with the pre-registered standard pattern. The main advantage of this approach is a separation of security code from business logic which can be maintained or extended to deal with new attacks. The evaluation of this proposed system's performance is considered by introducing the overhead of web application.

Aspect-oriented programming is again exploited in 'AProSec' by Hermosillo et al. which provides the capability of changing security policies at runtime with the help of JBoss AOP framework [10]. This proposed framework is composed of three different parts, first part is responsible for validation process, the second part is XML configuration file and can be controlled by the administrator which analyses input for SQL Query, finally AspectJ or JBoss AOP is used. This proposal utilizes advantages of AspectJ and JBoss AOP frameworks to defending against SQL Injection.

D'silva et al. have utilized hashing to defend against SQL Injection and Session Hijacking. Their proposal follows the ideology of authentication query which authenticates a valid user of the application [21]. While registering user his credentials i.e. user id and password are stored in hash

digest in the form of Hash of authentication. For each login hash of credentials will be dynamically generated and compared with pre-stored hash from hash digest when users want to interact with the database. This is a very efficient and lightweight way of preventing attacks however it gives the responsibility of securing hash digest as well.

A similar solution like the previous one has been introduced by Dubey and Gupta which is a combination of two security services which authenticates data and maintains confidentiality with integrity [22]. Even though their proposal is following a similar approach like D'silva et al. it overcomes disadvantages of security by changing approach for query comparison. Proposals which are based on hashing framework has a dependency on libraries which are used for generating hash values.

Signature-based SQL Injection framework which follows the backbone of fingerprinting method and pattern matching for differentiating vulnerable script from genuine SQL query has been proposed by Appiah et al [23]. This framework monitors SQL query at data tier and compares the same with a pre-defined dataset of signatures which can cause an attack. As this comparison is done at data tier this is very secured approach but introduces the overhead of pre-storing dataset of signatures.

The proposal by Ntagwabira and Kang follows the similar ideology of work like multi-layer security framework [24]. Their research aims the development of a method for detecting and preventing SQL Injection attacks with the help of Query tokenizer which implemented by Query parser method. The major drawback of this proposal is a dependency of java programming language. Chen Ping proposes a very nice approach to defending against SQL Injection attack which is based on Instruction Set Randomization. This approach gives defensiveness at database layer and dependant on MySQL database.

Recently, Taha and Karabatak have proposed, secured PHP functions which utilize built-in functions which detects and prevents XSS attack at two different layers [25]. Their two layered proposed system, the first layer uses a regular expression for authenticating data entered by a user of the application and second layer uses another regular expression for validating and protecting every single entry which may contain malicious script entered by a user of the application. Their future work heading towards developing a tool or browser extension which detects as well as prevents running harmful script in web form.

Ismail *et al.* proposed a java based twofold client-side system which not only protects users from cross-site scripting attacks but also warns web server about the attack [26]. This system automatically detects XSS vulnerability after modifying either client request or server response. The backbone ideology of this proposal is to validate HTTP request and response based on special characters and malicious scripts in detection proxy server. If detection proxy server finds fraud response or requests it gives HTML alert in web browser i.e. for client-side also it blocks database to execute script i.e. server side. This system has been implemented in Java/Linux/greSQL environment and for evaluation they used existing realistic examples which gives evidence of detection of the malicious script in response and request.

Amnesia is a tool proposed by Halfond and Orso which detects, and blocks SQL Injection attacks based on static analysis and runtime monitoring [7]. Underline technique for this tool is building SQL query models based on scanning application code which is consists of identify hotspots and build SQL-query models. After that each dynamically generated executing the query from the program is monitored in the instrument application and runtime monitoring steps if it found dangerous as per SQL query model it will not execute at the database level. A major limitation of this tool is SQL-query models takes space in the application also it assumes SQL queries from the application code only.

An advanced framework based on compiler platform and machine learning have been proposed by Kamtuo and Soomlek which uses a decision tree algorithm for predicting SQL Injection attacks

[27]. The machine learning part is the core of their approach which conducts 1100 datasets of vulnerabilities to train their model. A considerable drawback of this approach is it needs new datasets to train their model

Table 2.2 highlights programming language dependent solutions addressed by researchers.

Proposed Solution	Author	Programming dependency	Issue Addressed
Aspect-based defense system	Kajo-Mece *et al.* [2018]	AspectJ	1. SQL Injection 2. Cross-Site Scripting
AProSec	Hermosillo *et al.* [2007]	AspectJ, XML, Jboss	1. SQL Injection 2. Cross-Site Scripting
SHA1 based method	D'silva *et al.* [2017]	Hashing	1. SQL Injection 2. Session Hijacking
SQL Filtering	Rhythm Dubey, Himanshu Gupta [2016]	Hashing with Query comparison	SQL Injection
Signature-based framework	Appiah *et al.* [2017]	Not given	SQL Injection
Query tokenization	Lambert and Song Lin [2010]	Java	SQL Injection
Instruction set Randomization	Ping *et al.* [2016]	MySQL	SQL Injection
Algorithm for developers	Taha and Karabatak [2018]	PHP	Cross-Site Scripting
Automatic detection / collection system	Ismail et al. [2004]	Java	Cross-Site Scripting
Amnesia	Halfond and Orso [2006]	Java	SQL Injection
The framework of machine learning	Kamtuo and Soomlek [2016]	Not given	SQL Injection

Table 2.2 Programming language dependent solution

3 Research Methodology

This section gives a detailed description of the research procedure followed and an evaluation methodology to prove the defensive strength of the framework. The primary objective of this framework is to formulate cloud application for self-defensiveness so that dependency of external entities or programming languages can be removed. After closely examining and utilizing the benefits and drawbacks of existing systems mentioned in Section II a significant security framework will be provided to the cloud computing community for the development of secured cloud applications.

3.1 Objective

As mentioned previously, the motivation for this research project is to contribute framework for cloud computing community which helps cloud application developers to develop or migrate secured application for the cloud. This framework can remove the dependency of using external tools which are already using for monitoring and preventing from SQL injection and cross-site scripting attacks. Apart from that, this framework will not stick to any specific programming languages which gives freedom to cloud application developer for choosing programming languages. This framework will remove the dependency of using the external tool and formulate cloud application self-dependent in terms of security. This framework is not only useful for application developers but also it will play a very important role for solution architects for designing secured cloud applications. The Multi-layer security framework will be operative for all cloud-based applications to increase their Quality of Service to opposing against SQL Injection and cross-site scripting attacks.

3.2 Research Procedure

This framework has been evolved from the motivation of securing cloud application from application developers' minimal efforts. This framework should be able to use by multiple actors in the software development lifecycle such as application developers, solution architect etc. This framework should be easily pluggable and can be used while migrating an application or refactoring the application. With all these advantages MSLF has been carrying simple structure and applicable in every cloud application without restriction of any specific programming language.

3.3 Structure of the Framework

Multi-layer security framework is primarily divided into three separate layers, each with the specific functionality. OWSAPs guide for secured software development recommends three pillars for securing user data as follows [6]:

- Reject corrupt data
- Accept valid data
- Clean the data which can harm the system

The backbone of multi-layered security framework is based on these three pillars also it covers some of the significant approaches from the recent research held in the same direction. Here are the three layers of MSLF framework:

3.3.1 UI Component Layer Filtering

This layer of the framework is designed to defend against special characters which cause SQL Injection and cross-site scripting attacks. In this stage, the framework rejects the corrupt data and corrupt data at this layer can be defined as the special characters which may cause SQL injection and cross-site scripting attacks. Previous researchers and OWSAP have recommended special characters such as <, >, -, /, =, ‘, “, #, SPACE which causes SQL Injection and cross-site scripting attacks [4][6][8]. MLSF introduces security pattern at UI Component layer of cloud application for filtering of corrupted data which blocks these special characters to flow towards data tier of the application. Considerable superiority of this security pattern is that they can be reconfigured as per business requirements. These security patterns will be attached to input fields such as text box, text area etc. Here is the example of security patterns:

Allowed characters in the text box

Alphabet: a-z / A-Z	Numbers: 0-9
Special Characters (spaces should be allowed): ~@$^&*()_+[]{}|\\,.?:	

Security pattern for above requirement

[a-zA-Z0-9~@#$^*()_+=]*

3.3.2 UI Process component Layer Filtering

The second layer of the framework has been specially designed for detection and defending against malicious keywords which is another major cause of SQL Injection attack. Malicious keywords have been recommended by some of previous researchers and OWSAP such as SELECT, UPDATE, DELETE, UNION etc [6][8]. MLSF proposes malicious script analyzer at UI process component layer which scans user input, detect and rejects of malicious user input. By using core

principle of object-oriented programming i.e. inheritance single function could scan all input field of application for malicious keywords and can easily upgrade to future malicious keywords. All these malicious keywords included in the UI process component layer are given below:

SELECT	UPDATE	WHERE	OR
DELETE	UNION	HAVING	NOT
ORDER BY	GROUP BY	AND	SLEEP

Table 3.1 Malicious Keyword

3.3.3 Service Layer Filtering

As per Stephen Walter (2009), the service layer is the interface between middleware and graphical user interface which can be used effectively for validation purpose. First two layers of the multi-layer security framework are operating on client side i.e. end user's browser which can be bypassed with penetration tools such as SQLMap, Postman etc., and the attacker can target the application. To protecting cloud application against such attacks service layer filtering plays a vital role in the multi-layer security framework. MLSF utilizes a service layer for validation purpose and combines security pattern with malicious script analyzer at the service layer. All data entered by user or attacker must bypass through this security gate which is present at the service layer. Here is the code snippet for service layer filtering:

```
private static final String SQLIA_SECURITY_PATTERN = "[a-zA-Z0-9~@#$^*()_+=]*";
private static final List<String> MALICIOUS_SCRIPT = Arrays.asList("WHERE", "OR", "HAVING",
"NOT", "ORDER BY", "GROUP BY", "AND", "SLEEP");

public boolean validate(final String validationString) {
        Pattern pattern = Pattern.compile(SQLIA_SECURITY_PATTERN);
        Matcher matcher = pattern.matcher(validationString);
        return matcher.matches() && MALICIOUS_SCRIPT.contains(validationString);
}
```

3.3.4 Architectural structure

Figure 3.1 shows the architectural structure of multilayer security framework which is embedded in a cloud application. There are three different layers of this framework which are embedded with existing layers of cloud application namely UI components, UI process components and service layer. When the attacker tries to penetrate from presentation tier which is split into two layers i.e. UI component (e.g. HTML, CSS) and UI process component (e.g. coffee script, JavaScript) MLSF detects special characters and malicious keywords from those layers respectively. Service layer filtering is responsible for validating for special characters as well as malicious keywords which protects data tier of a cloud application.

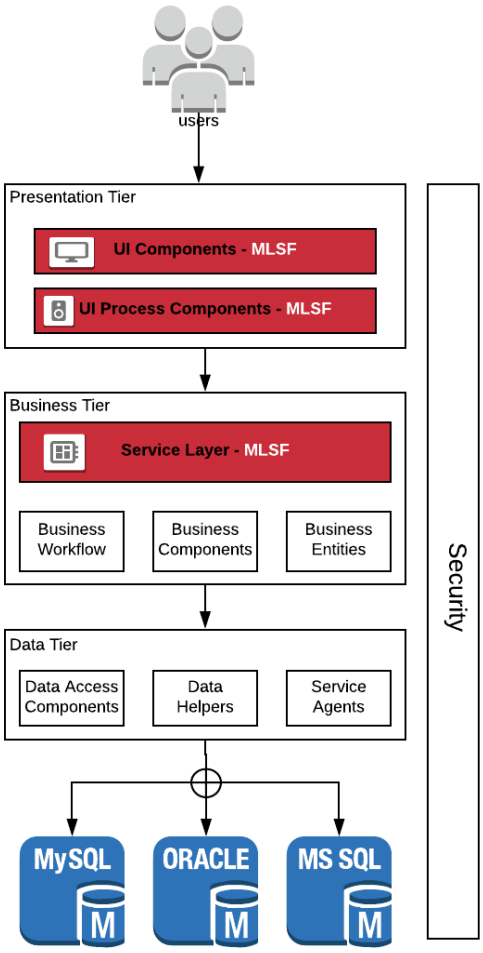

Figure 3.1 MLSF embedded in cloud application

4 Implementation

Multi-layer security framework is lightweight and incorporated security mechanism in cloud application which can be used for designing, development and migrate cloud application to enhance the security of the application. As this framework is not depending on any specific programming language or framework, this can be developed with any technology stack. For development and demonstration purpose cloud-based vulnerable application has been developed and multi-layer security framework is embedded into it to improve its defensiveness.

4.1 Development tools and technologies

The J2EE framework is widely used for financial as well as business applications due to the excellent features of java and high popularity. For demonstration purpose, MLSF has been developed in a J2EE framework which includes HTML5 as UI component, JavaScript as UI Process Components to form cloud applications presentation tier i.e. view of the application. Business tier is composed of java/servlet and we have introduced service layer separately i.e. at the controller. MySQL is holding a database of the cloud application which is connected to the application through JDBC i.e. model. This application is hosted on Platform as a Service (PaaS) provided by Microsoft

Azure [28]. Development of J2EE cloud application popular integrated development environment eclipse and MySQL Workbench has been used and deployment has been done with the help of Azure plugins for eclipse. Here are the tools and technologies used for development purpose:

Layer of cloud application		Technology used
Presentation tier	UI Component layer	HTML
	UI Process Component layer	JavaScript
Business tier	Service layer	Servlets / Java
	Business component / workflow / entities	Servlets / Java
Data tier	Data helpers	Java
	Data access component / service agents	JDBC

Table 4.1 Development technologies

As shown in the below process flowchart, multi-layer security framework checks for vulnerability at three different layers. First layer i.e. at UI component layer it checks for special characters, at second layer i.e. UI process layer it checks for the malicious script and at third layer i.e. service layer is validating for special characters as well as malicious keyword. If and only if user input is valid and passes all three-validation framework allow it to flow towards data tier of the application otherwise framework rejects user input.

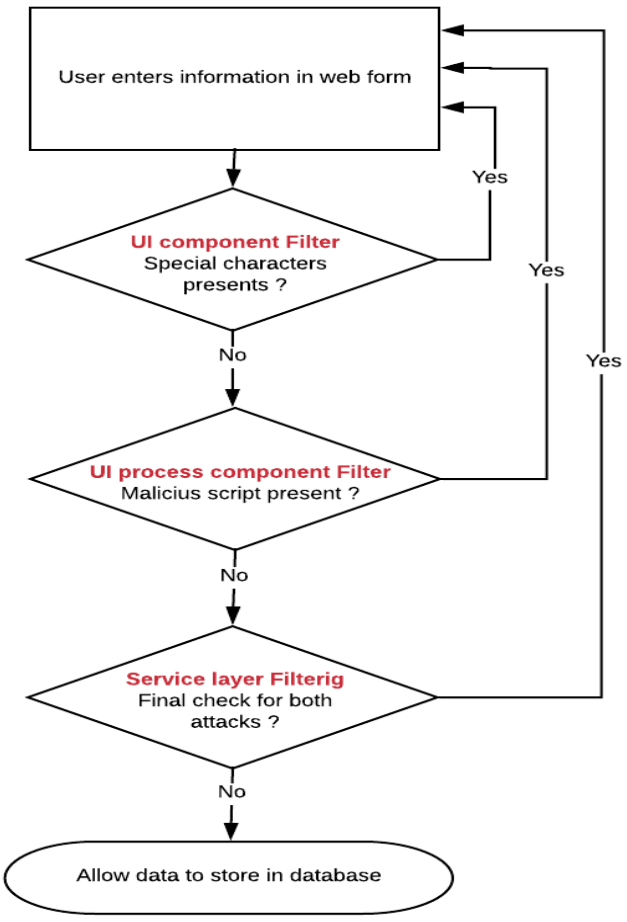

Figure 4.1 Process flowchart of MLSF

5 Evaluation

Generally, Instruction Detection System can be evaluated by generating traffic with the help of another IDS or with the help of vulnerability exploitation tool which attacks the targeted system with attacks [12][29]. Similar approaches have been evaluated with vulnerability exploitation tool especially with SQLMap [12][30].an MLSF is evaluated by using vulnerability exploitation tool which is used to attack and analyse on cloud application developed in J2EE. Apart from that, MLSF has been evaluated for performance of the application which shows the overhead of framework on a cloud application which helps to prove its feathery outlook.

5.1 Vulnerability testing and Content policy analysis

SQLMap is an open source penetration testing tool which automates the process of abusing SQL Injection vulnerabilities is used as vulnerability exploitation tool [30]. SQLMap is installed on attacking machine to target two cloud application i.e. vulnerable cloud application and MLSF embedded vulnerable cloud application. In order to guarantee similar cloud application database of both web application kept same and with the same configuration i.e. server version, java version etc. For evaluation purpose below scenarios has been considered:

1) **Scenario 1:** WebApp1 which is a vulnerable cloud application attacked by using vulnerability exploitation tool. The goal of this scenario is to know the vulnerability of the cloud application.
2) **Scenario 2:** WebApp2 which is vulnerable cloud application with MLSF i.e. the same code base but MLSF has been incorporated in the same has been attacked by using vulnerability exploitation tool as well as with content security policy evaluator. The objective of this scenario is to understand improved defensiveness of cloud application.

5.2 Improved defensiveness

Table 5.1 illustrates a comparison of vulnerable cloud application with MLSF embedded vulnerable application on basis of two user input fields. After using MLSF, various types of attacks have been not only effectively detected but also defended with high accuracy. In particular, various types of attacks have been automated by SQLMap on both applications specifically on two input fields. Thus, Multi-layer security framework embedded cloud application seems to be more secure.

Type of attack	Scenario 1		Scenario 2	
	Email	Password	Email	Password
Boolean-based blind	×	✓	×	×
Error-based	✓	✓	×	×
AND/OR time-based blind	✓	✓	×	×
UNION query	✓	×	×	×

Table 5.1 SQL Injection attacks prevented by Multi-layer Security Framework

Symbol	Associated value
×	Type of attack is not present in the given input field
✓	Type of attack is present in the given input field

Table 5.2 Values associated with symbols

5.3 Overhead of framework

The previous section clearly shows improved defensiveness of the cloud application after using multi-layer security framework. In order to evaluate the impact of multi-layer security framework on cloud application response time in seconds has been measured using JMeter, firstly in absence of framework and then in the presence of multi-layer security framework [32]. JMeter is open source Apache tool which can be used for measuring and analyzing the performance of a variety of web-

based services. Here is the comparison of cloud application without MLSF and with MLSF which shows the overhead of framework in normal condition.

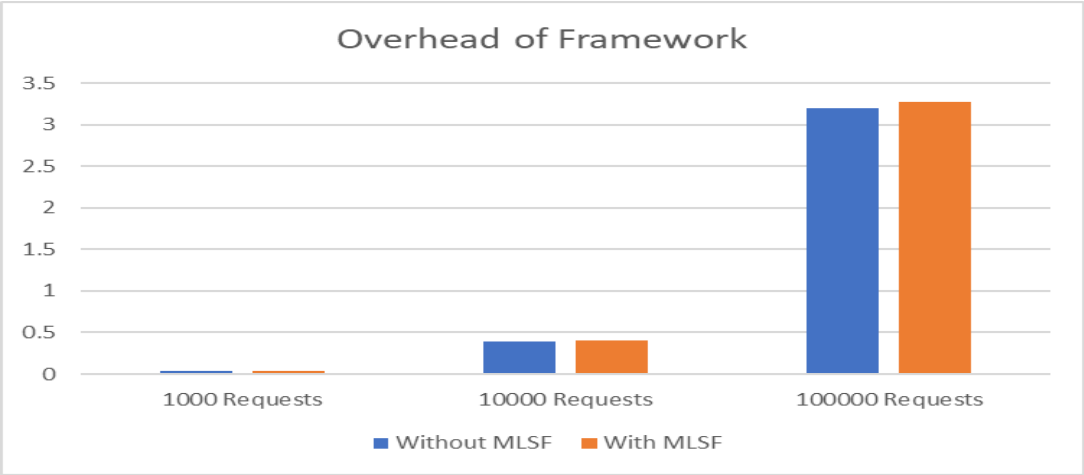

Chart 5.2 Overhead of framework

5.4 Comparative analysis

This section compares the performance of multi-layer security framework in attacking condition also we have compared this analysis with existing aspect based defensive system [8]. In order to automate attacking condition with performance malicious HTTP requests have been fired from JMeter. For each scenario, different load has been used i.e. 1000 requests, 10000 requests and 100000 requests with both the attacks. To know overhead of the framework we have created below scenarios:

Scenario 1: HTTP request with SQL Injection attack

Scenario 2: HTTP request with cross-site scripting attack

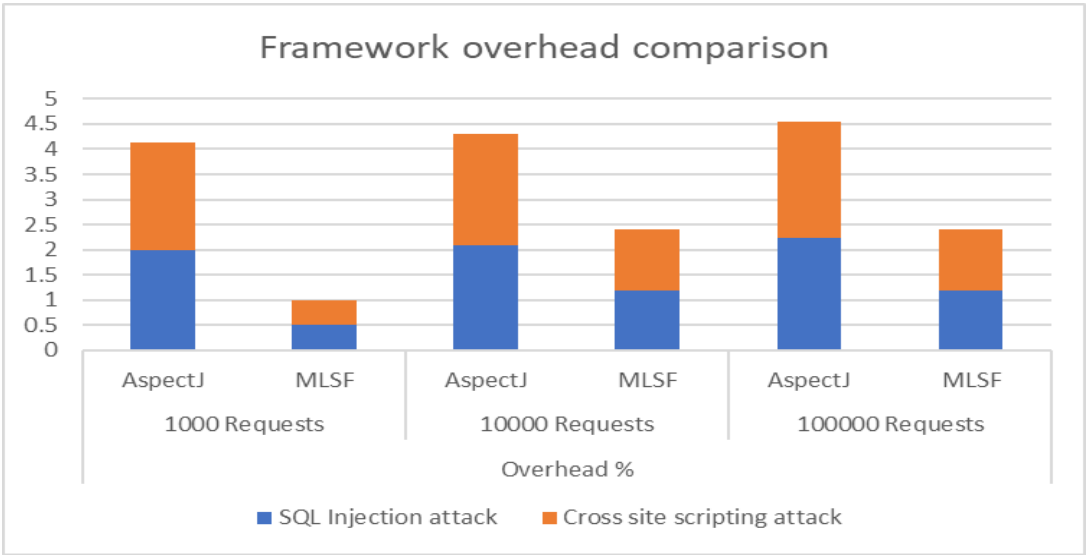

Chart 5.3 Performance Evaluation and comparison

This analysis clearly shows that aspect based defensive system is validating, comparing, encoding, and rejecting dangerous script however multi-layer security is rejecting malicious information so it not only makes cloud application self-defensive but also it improves performance of the application in attacking situation.

6 Conclusion and Future Work

The primary mechanism used to attack database is through SQL Injection in which dynamic queries are injected on cloud or internet. Cross-site scripting can change behavior of the application or can access secured information from a cloud application. Therefore, there is need for detection and mitigation algorithm based on network traffic which leads increase in the security budget of a cloud application. Multi-layer security framework provides integrated, detection and mitigation mechanism for cloud application which makes cloud application self-defensive as well as controls the security budget. Moreover, it is designed to ensure vulnerable code does not enter into the cloud application and data tier would not have to deal with corrupt data which may entirely or partially affect sensitive user information. MLSF proposes security pattern and malicious script analyzer which are programming language independent and makes cloud application self-defensive for SQL Injection and cross-site scripting attacks. Evaluation clearly shows that MLSF can counteract multiple types of SQL Injection as well as cross-site scripting attacks which makes cloud application self-defensive.

The proposed framework is simple, lightweight and can be used to develop self-defensive cloud application as well as an application which is connected to the database without depending on specific programming language. Future work should be carried out for other cyber-attacks which can also defend with this framework.

References

[1] Chouhan, P.K., Yao, F., Sezer, S., 2015. Software as a service: Understanding security issues, in: 2015 Science and Information Conference (SAI). Presented at the 2015 Science and Information Conference (SAI), pp. 162–170. https://doi.org/10.1109/SAI.2015.7237140

[2] Barclays: 97 percent of data breaches still due to SQL injection | Security | Techworld [WWW Document], n.d. URL https://www.techworld.com/news/security/barclays-97-percent-of-data-breaches-still-due-sql-injection-3331283/ (accessed 12.16.18).

[3] Manipulating_SQL_Server_Using_SQL_Injection.pdf, n.d.

[4] A proposed approach for preventing cross-site scripting - IEEE Conference Publication [WWW Document], n.d. URL https://ieeexplore.ieee.org/document/8355356 (accessed 12.16.18).

[5] About The Open Web Application Security Project - OWASP [WWW Document], n.d. URL https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project (accessed 12.16.18).

[6] Category:OWASP Top Ten Project - OWASP [WWW Document], n.d. URL https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project (accessed 12.16.18).

[7] Halfond, W.G.J., Orso, A., 2006. Preventing SQL Injection Attacks Using AMNESIA, in: Proceedings of the 28th International Conference on Software Engineering, ICSE '06. ACM, New York, NY, USA, pp. 795–798. https://doi.org/10.1145/1134285.1134416

[8] Kajo-Mece, E., Kodra, L., Vrenozaj, E., Shehu, B., 2012. Protection of web applications using aspect oriented programming and performance evaluation. CEUR Workshop Proceedings 920, 46–50.

[9] Gould, C., Su, Z., Devanbu, P., 2004. JDBC checker: a static analysis tool for SQL/JDBC applications, in: Proceedings. 26th International Conference on Software Engineering. Presented at the Proceedings. 26th International Conference on Software Engineering, pp. 697–698. https://doi.org/10.1109/ICSE.2004.1317494

[10] AProSec: an Aspect for Programming Secure Web Applications - IEEE Conference Publication [WWW Document], n.d. URL https://ieeexplore.ieee.org/document/4159905 (accessed 12.16.18).

[11] Abstracting application-level web security [WWW Document], n.d. URL https://dl.acm.org/citation.cfm?id=511498 (accessed 12.16.18).

[12] Yassin, M., Ould-Slimane, H., Talhi, C., Boucheneb, H., 2017. SQLIIDaaS: A SQL Injection Intrusion Detection Framework as a Service for SaaS Providers, in: 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud). Presented at the 2017

IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), pp. 163–170. https://doi.org/10.1109/CSCloud.2017.27

[13] A multilevel system to mitigate DDOS, brute force and SQL injection attack for cloud security - IEEE Conference Publication [WWW Document], n.d. URL https://ieeexplore.ieee.org/document/8279028 (accessed 12.16.18).

[14] A novel three-tier SQLi detection and mitigation scheme for cloud environments - IEEE Conference Publication [WWW Document], n.d. URL https://ieeexplore.ieee.org/document/8167160 (accessed 12.16.18).

[15] Nguyen, K.K., Hoang, D.T., Niyato, D., Wang, P., Nguyen, D.N., Dutkiewicz, E., 2018. Cyberattack detection in mobile cloud computing: A deep learning approach [WWW Document]. undefined. URL /paper/Cyberattack-detection-in-mobile-cloud-computing%3A-A-Nguyen-Hoang/f798c7946b98c26b91f0176eede18e86f2d0af0c (accessed 12.16.18).

[16] Application of Hidden Markov Model in SQL Injection Detection - IEEE Conference Publication [WWW Document], n.d. URL https://ieeexplore.ieee.org/document/8029993 (accessed 12.16.18).

[17] Wang, K., Hou, Y., 2016. Detection method of SQL injection attack in cloud computing environment, in: 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC). Presented at the 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), pp. 487–493. https://doi.org/10.1109/IMCEC.2016.7867260

[18] Detection of SQL injection and XSS attacks in three tier web applications - IEEE Conference Publication [WWW Document], n.d. URL https://ieeexplore.ieee.org/document/7860069?denied= (accessed 12.16.18).

[19] Xiao, Z., Zhou, Z., Yang, W., Deng, C., 2017. An approach for SQL injection detection based on behavior and response analysis, in: 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN). Presented at the 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), pp. 1437–1442. https://doi.org/10.1109/ICCSN.2017.8230346

[20] Vijayalakshmi, K., Leema, A.A., 2017. Extenuating web vulnerability with a detection and protection mechanism for a secure web access, in: 2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN). Presented at the 2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN), pp. 1–4. https://doi.org/10.1109/ICSCN.2017.8085652

[21] D'silva, K., Vanajakshi, J., Manjunath, K.N., Prabhu, S., 2017. An effective method for preventing SQL injection attack and session hijacking, in: 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT). Presented at the 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT), pp. 697–701. https://doi.org/10.1109/RTEICT.2017.8256687

[22] Dubey, R., Gupta, H., 2016. SQL filtering: An effective technique to prevent SQL injection attack, in: 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO). Presented at the 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp. 312–317. https://doi.org/10.1109/ICRITO.2016.7784972

[23] Appiah, B., Opoku-Mensah, E., Qin, Z., 2017. SQL injection attack detection using fingerprints and pattern matching technique, in: 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS). Presented at the 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 583–587. https://doi.org/10.1109/ICSESS.2017.8342983

[24] Use of Query tokenization to detect and prevent SQL injection attacks - IEEE Conference Publication [WWW Document], n.d. URL https://ieeexplore.ieee.org/document/5565202 (accessed 12.16.18).

[25] Assad Taha, T., Karabatak, M., 2018. A proposed approach for preventing cross-site scripting. pp. 1–4. https://doi.org/10.1109/ISDFS.2018.8355356

[26] Ismail, O., Etoh, M., Kadobayashi, Y., Yamaguchi, S., 2004. A proposal and implementation of automatic detection/collection system for cross-site scripting vulnerability, in: 18th

International Conference on Advanced Information Networking and Applications, 2004. AINA 2004. Presented at the 18th International Conference on Advanced Information Networking and Applications, 2004. AINA 2004., pp. 145-151 Vol.1. https://doi.org/10.1109/AINA.2004.1283902

[27] Kamtuo, K., Soomlek, C., 2016. Machine Learning for SQL injection prevention on server-side scripting, in: 2016 International Computer Science and Engineering Conference (ICSEC). Presented at the 2016 International Computer Science and Engineering Conference (ICSEC), pp. 1–6. https://doi.org/10.1109/ICSEC.2016.7859950

[28] Microsoft Azure Cloud Computing Platform and Services [WWW Document], n.d. URL https://azure.microsoft.com/en-in/ (accessed 12.16.18).

[29] Probst, T., Alata, E., Kaâniche, M., Nicomette, V., 2015. Automated Evaluation of Network Intrusion Detection Systems in IaaS Clouds, in: 2015 11th European Dependable Computing Conference (EDCC). Presented at the 2015 11th European Dependable Computing Conference (EDCC), pp. 49–60. https://doi.org/10.1109/EDCC.2015.10

[30] Advanced automated SQL injection attacks and defensive mechanisms - IEEE Conference Publication [WWW Document], n.d. URL https://ieeexplore.ieee.org/document/7868248 (accessed 12.16.18).

[31] Apache JMeter - Apache JMeterTM [WWW Document], n.d. URL https://jmeter.apache.org/ (accessed 12.16.18).

[32] Eclipse Download and Installation Instructions [WWW Document], n.d. URL https://www.ics.uci.edu/~pattis/common/handouts/pythoneclipsejava/eclipsejava.html (accessed 12.19.18).

[33] Help - Eclipse Platform [WWW Document], n.d. URL https://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.stardust.docs.wst%2Fhtml%2Fwst-integration%2Fconfiguration.html (accessed 12.19.18).

[34] How do I manually download and install Java for my Windows computer? [WWW Document], n.d. URL https://java.com/en/download/help/windows_manual_download.xml (accessed 12.19.18).

[35] MySQL :: MySQL 5.5 Reference Manual :: 2.3 Installing MySQL on Microsoft Windows [WWW Document], n.d. URL https://dev.mysql.com/doc/refman/5.5/en/windows-installation.html (accessed 12.19.18).

[36] selvasingh, n.d. Create a Hello World web app for Azure using Eclipse [WWW Document]. URL https://docs.microsoft.com/en-us/java/azure/eclipse/azure-toolkit-for-eclipse-create-hello-world-web-app (accessed 12.19.18).

Appendix

A. Project Configuration

MLSF has been developed in the J2EE framework which includes three layers of application i.e. presentation layer, business tier and data tier. Below is the technology stack with their versions used for development of vulnerable cloud application [32-35].

Layer of cloud application		Technology used	Version
Presentation tier	UI Component layer	HTML	5
	UI Process Component layer	JavaScript	9
Business tier	Service layer	Servlets / Java	8.0.11
	Business component / workflow / entities	Servlets / Java	8.0.11
Data tier	Data helpers	Java	JDK 8
	Data access component / service agents	JDBC	8.0.11

Table 1.1 Project and programming language versions

B. Development Tool Configuration

For the development of multi-layer security framework, firstly vulnerable cloud application has been developed then MLSF has been embedded into it to improve defensiveness of cloud application. For the development of vulnerable application maven project has been created which needs below tool set:

Java Development IDE	Eclipse Oxygen (v 3.7.1)
Database Development IDE	MySQL Workbench (v 8.0)
Java Build Tool	Maven
Development Server	Tomcat server (v 9.0)
Java Development Kit	JDK 8

Table 2.1 Development tools

C. Run Configuration

To host and run MLSF embedded vulnerable cloud application locally you can follow below steps:

1. Download Project

Self-defensive cloud application project has been made open source for cloud community on GitHub under "**GNU General Public License v3.0**". Researchers can download it from below link:

https://github.com/akashhande/Multi-Layer-security-framework

Researcher can unzip the file and copy it in eclipse workspace in his local machine. Generally, eclipse workspace is present in below path:

C:\Users\user-name\eclipse-workspace

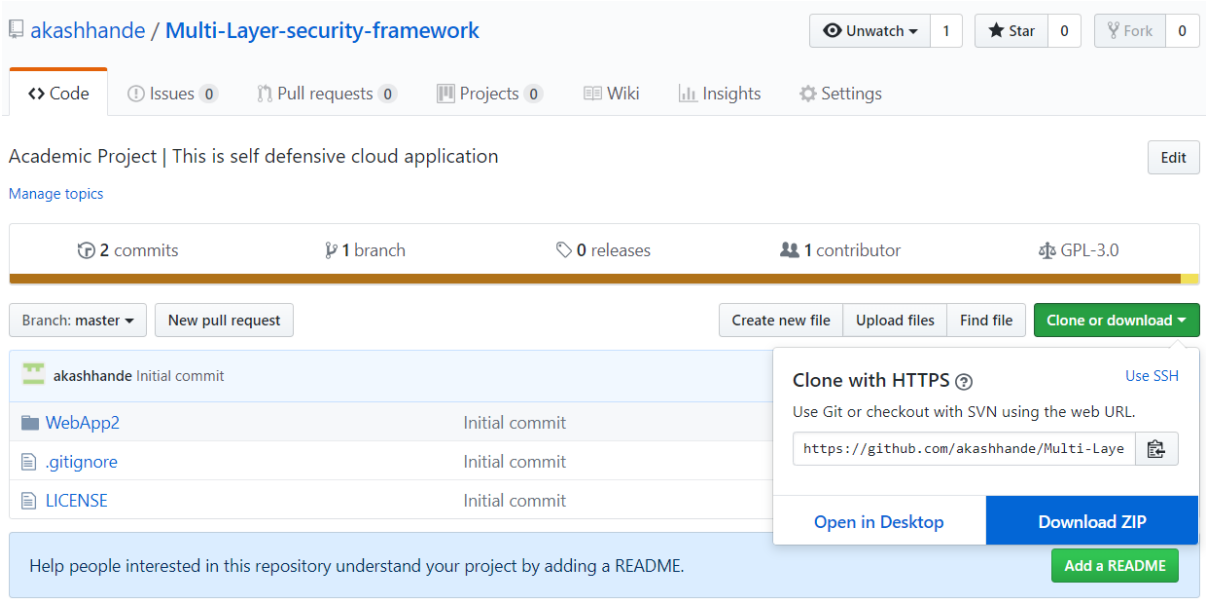

Figure 3.1 GitHub download

2. Configure locally

Open eclipse and import existing project from workspace with below steps:

File => Import => Existing Projects into workspace => Next => Select root directory => Browse => select WebApp2 => hit OK

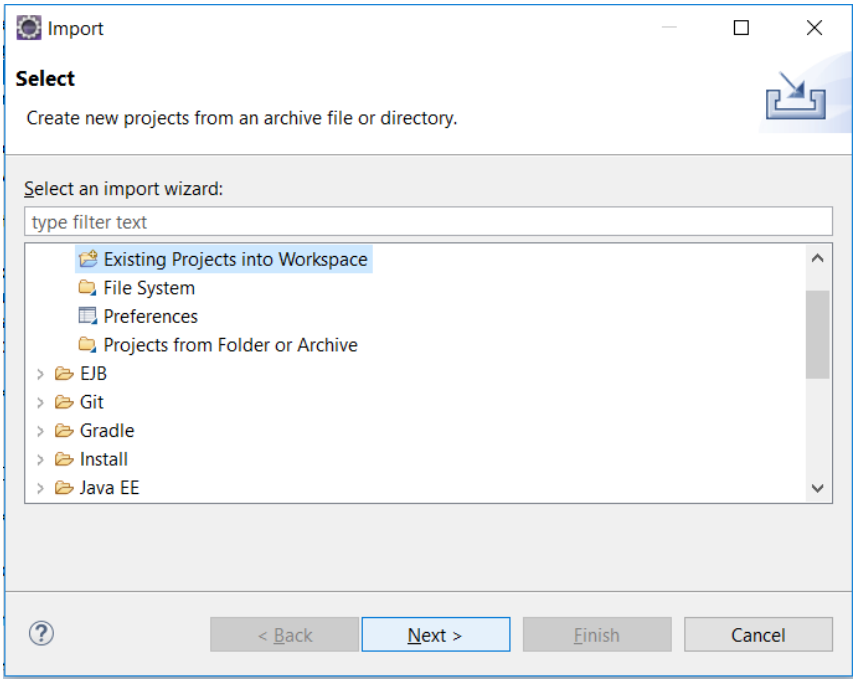

Figure 3.2 Import project from workspace

After import you can see below directory structure in project explorer panel of eclipse:

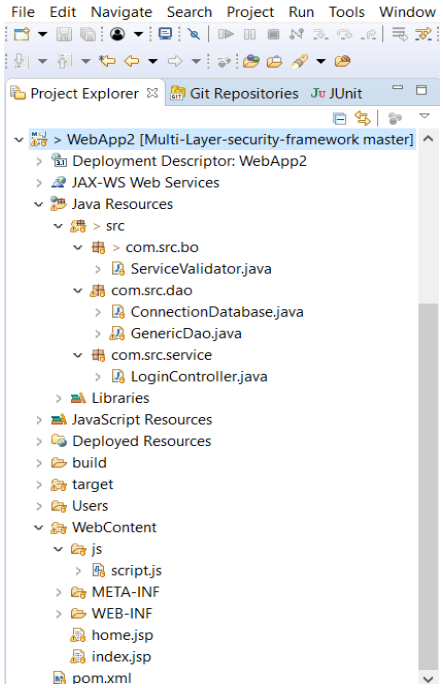

Figure 3.3 Project structure from eclipse

3. Run on the local environment

To run self-defensive cloud application, follow below steps:

- Right click on WebApp2 => Run As => Maven Install
- Right click on WebApp2 => Run As => Run on server => Finish

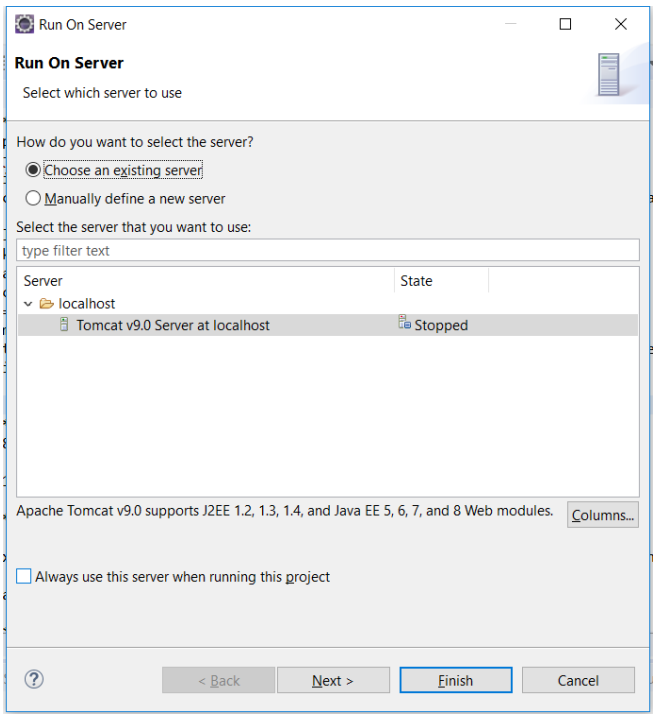

Figure 3.4 Run project on local environment

4. Application UI

Application UI looks like below and application URL for local environment is http://localhost:8080/WebApp2/

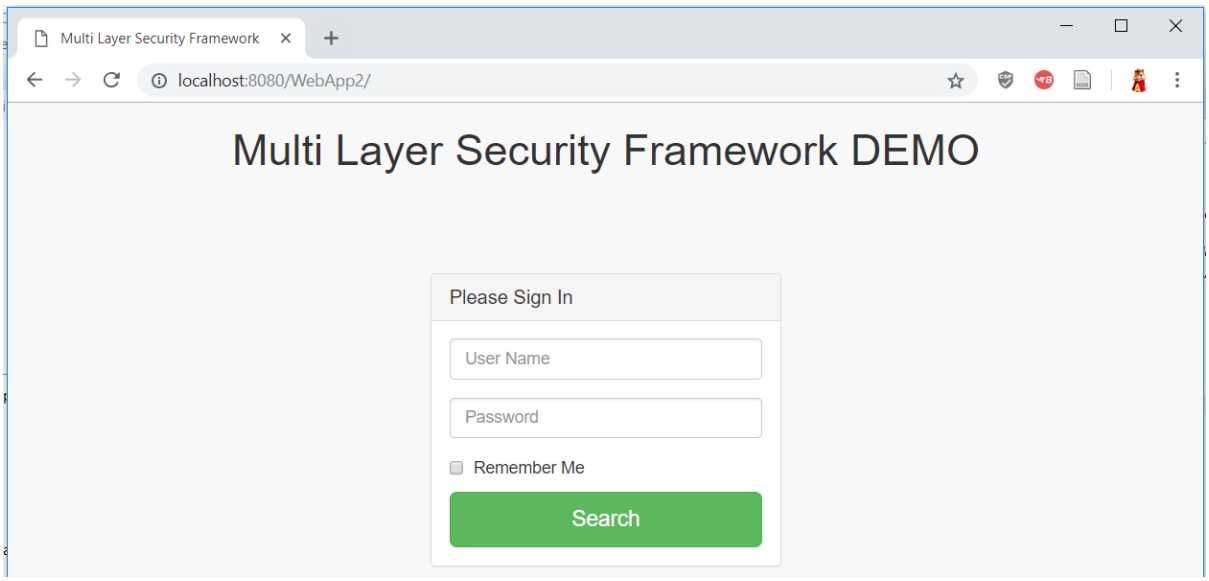

Figure 3.5 Application UI

D. Deployment Configuration

Developed secured application can be hosted on Azure PaaS platform from eclipse IDE with the help of Azure plugin [36]. For deployment Right click on project => select Azure => click on Publish to Azure Web App.

Figure 4.1 Deploy project on Azure

You must select deployment configuration such as App Service, pricing tier, location, instance size etc. Then you can deploy app on Azure platform. Installed apps can be explored via built-in Azure explorer from eclipse.

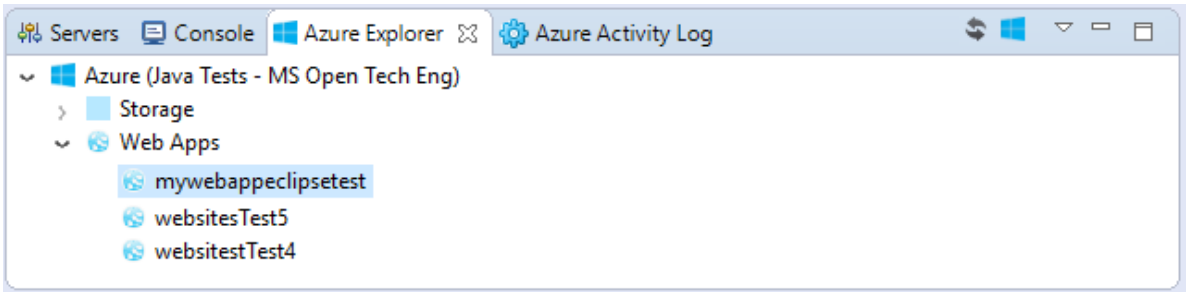

Figure 4.2 Application UI