

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology in Information Technology

By
ROHIT V
REG. NO. 38120068



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

NOVEMBER 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI
CHENNAI- 600119
www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of ROHIT V (**Reg. No: 38120068**) who carried out the project entitled "**Machine Learning Approach For Prevention Of SQL Injection**" under my supervision from June 2021 to November 2021.

Internal Guide

Name of the Guide

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **ROHIT.V** hereby declare that the project report entitled **Machine Learning Approach For Prevention Of SQL Injection** done by me under the guidance of **Dr. AJITHA** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D, Dean**, School of Computing, **Dr. R. Subhashini, M.E., Ph.D. Head of the Department of Information Technology** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. AJITHA M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



CERTIFICATE OF COMPLETION

CSW-I-26-081121 is given to

ROHIT V

For completing the Cyber Security training that includes basics of Cyber Security, Pentesting, Networking, and WebApp.

Aaron V Sandeen

AARON SANDEEN
CEO - CSW

N. Vengatesh

VENGATESH NAGARAJAN
MENTOR



TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No
	ABSTRACT	i
	LIST OF FIGURES	ii
	LIST OF TABLES	iii
	LIST OF ABBREVIATIONS	iv
1	INTRODUCTION	1
	1.1 General	1
	1.2 Objective	2
	1.3 Scope Of The Object	3
	1.4 Organization Of The Thesis	3
2	LITERATURE SURVEY	4
	2.1 General	4
	2.2 Existing System	4
	2.2.1 Literature Survey On The SQLi Attacks	5
	2.3 Issue In The Existing System	6
	2.4 Proposed System	7
	2.5 Summary	7
3	SYSTEM DESIGN	8
	3.1 General	8
	3.2 System Architecture	9
	3.3 Functional Architecture	9
	3.4 Modular Design	10
	3.4.1 URL Intercept Engine	11
	3.4.2 Context Free Grammar For SQLi Attacks	12
	3.4.3 Pattern Classification Through Machine Learning	13
	3.5 System Requirements	14

	3.6 Summary	14
4	SYSTEM IMPLEMENTATION	15
	4.1 General	15
	4.2 Overview Of The Platform	15
	4.2.1 Python	15
	4.2.2 Python Flask	16
	4.2.3 HTML	17
	4.2.4 Cascading Style Sheets	18
	4.2.5 Microsoft SQL server Management Studio	19
	4.3 Module Implementation	20
	4.3.1 Implementation Of Frontend And Backend	21
	4.3.2 Implementation Of Pattern Generation	22
	4.3.3 Detection And Prevention	23
	4.4 Summary	23
5	SYSTEM TESTING	24
	5.1 General	24
	5.2 Test Cases	26
	5.2.1 Login Validation	27
	5.2.2 Login Without SQL Injection Detection	28
	5.2.3 Login With SQL Injection Detection	28
	5.3 Performance Measures	28
	5.4 Performance Analysis	29
	5.5 Summary	31

6	CONCLUSION ND FUTURE WORK	32
	6.1 Conclusion	32
	6.2 Future Work	32

ABSTRACT:

In today's scenario, web application firewalls are an essential protection mechanism for online software system. In Internet age, the most critical security risk of vulnerable web applications is SQL Injection attacks. With the increasing threats of SQL Injections, Web Application Firewall (WAF) has to be updated and tested regularly to prevent attackers from easily attacking them. As technology grows, the number of attackers who intend to attack the applications find numerous new ways to enter into the system. Thus, the existing systems find it difficult to cope up with the new hackers with new technologies to completely save the system. In the existing WAF, the white box testing and static analysis approach needs access to source code. The model based testing requires more sets of rules. The black box testing is not efficient for detecting SQL injection attacks. Machine learning is an application of artificial intelligence that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. The concept of collaborating machine learning with web application firewalls increases the efficiency of the existing system. The approach used in this paper is Unsupervised Learning Technique. The algorithm used for Unsupervised learning technique is the k-means algorithm which is used for clustering problems. The flow of the system can be given as follows. The end user makes a request in the Web application, the values of the request are extracted and sent to the SQL injection detector, which provides two layers of security. In the first layer of security, patterns are generated using CFGs for low level attacks. The second layer of security for high level attacks is trained using Unsupervised Learning Algorithm.

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
3.1	System architecture	8
3.2	Functional architecture	9
3.3	Workflow of URL Intercept Engine	9
3.4	Workflow of Context Free Grammar for SQLi Attacks	10
3.5	Workflow of Pattern Classification through Machine Learning	14
4.1	Implementation of the frontend	19
4.2	Backend - Database server	20
4.3	Rules for pattern generation	21
4.4	Possible attack patterns	21
4.5	Detection by the CFG function	22
4.6	Detection by Clustering	23
5.1	Performance analysis based on time take to cluster	30
5.2	Performance analysis based on accuracy	31

LIST OF TABLES

TABLE NO	TITLE	PAGE NO
5.1	Login Validation	24
5.2	Login without SQL injection detection	26
5.3	Login with SQL injection detection	27

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
AMNESIA	Analysis and Monitoring for Neutralizing SQL Injection Attacks
BSD	Berkeley Software Distribution
CFG	Context Free Grammar
CSS	Cascading Style Sheets
EA	Evolutionary Algorithm
FCM	Fuzzy C-Means
FP	False Positive
FN	False Negative
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
NLP	Natural Language Processing
NLTK	Natural Language ToolKit
SQLi	Structured Query Language injection
SSMS	SQL Server Management Studio
TDD	Test Driven Development
TP	True Positive

CHAPTER 1

INTRODUCTION

1.1 GENERAL

A web application firewall (WAF) is used in HTTP applications as application firewalls. It applies a collection of rules to HTTP conversation. Generally, these rules cover common attacks like cross-site scripting (XSS) and SQL injection. While proxies typically shield clients, WAFs shield servers. A WAF is deployed to shield a selected internet application or set of internet applications. A WAF may be thought as a reverse proxy.

WAFs might be available in the form of an appliance, server plugin or filter and can also be custom-built to an application. The hassle to perform this customization may be insignificant because, if there is any need to change the application, the WAFs should also be maintained according to the change.

Web applications with high security requirements are commonly protected by WAFs. In a overall system architecture, a WAF is placed in front of the web application that has to be protected. Every request that is sent to the web application is examined by the WAF before it reaches the web application. The WAF hands over the request to the web application only if the request complies with the firewall rule set.

Since the threat of cyber attacks are growing day by day, the WAFs are getting complicated. Also, manually testing and maintaining the principles is a issue. Therefore, automatic testing techniques for WAFs are crucial to stop malicious requests from reaching internet applications and services.

SQLi attacks have never lost it's trend and always possess a major threat to the web applications of various domains.

PROBLEM DESCRIPTION

To provide a web application firewall which provides a two layer security component to detect and prevent SQL injection attacks. SQL Injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. SQL Injection can also be used to add, modify, and delete records in the database.

1.2 OBJECTIVE

The objectives are described as follows:

- To extract the values passed in the request through URL
- Use Context free grammar to generate the attack patterns and detect attack
- Use Machine Learning Algorithm and prevent attack

1.3 SCOPE OF THE PROJECT

The scope of this project is to utilize the significant development in technology for monitoring many kinds of web application. It helps to detect and prevent SQL injection attack using machine learning techniques. This project effectuates optimal use of k-means clustering for efficient detection of SQL attack.

liyaa is a component on the server side which can also be called as the SQL injection detector. It provides a two layer security so as to have a safe and secure way of communication of the user with the server without affecting its database.

1.4 ORGANIZATION OF THE THESIS

The report consists of 6 chapters, the contents of which are described below: Chapter 1 is the introduction that explains about the basic information of the system. Chapter 2 is a literature survey that elaborates on the research works on existing systems. Chapter 3 describes the system design. Chapter 4 gives details regarding the system implementation. Chapter 5 describes the different test case scenarios for the modules described by the proposed system and the performance analysis of the proposed system. Chapter 6 provides the conclusion, which summarizes the efforts undertaken in the proposed system, and states findings and shortcomings in the proposed system.

CHAPTER 2

LITERATURE SURVEY

2.1 GENERAL

Literature survey is the documentation of a comprehensive review of the publishers and unpublished work from the secondary sources data in the areas of specific interest to the researcher. It is a survey of all the techniques used so far to summarize and analyze user reviews.

Many research works are carried out in firewalls mainly for detecting SQL Injection since it became a major security threat. Some of the works include use of certain algorithms to test and detect attacks on firewall, detection and prevention of SQLi attacks. The recently carried works in the field SQLi has been described in detail below.

2.2 EXISTING SYSTEM

Existing System contains a search-based approach that combines machine learning and EAs to automatically test the attack detection capabilities of WAFs. The approach automatically generates a diverse set of attacks, sends them to a WAF under test, and checks if they are correctly identified. By incrementally learning from the tests that are blocked or bypassing the firewall, this approach selects tests that exhibit string patterns with high bypassing probabilities (according the machine learning) and mutates them using an attack grammar designed to generate new and hopefully successful attacks. Identified bypassing attacks can be used to learn path conditions, which characterize

successful attack patterns. With such a set of bypassing attacks and path conditions that characterize them, a security expert can fix or fine-tune the WAF rules in order to block imminent SQLi attacks. In the attacker–defender war, time is vital. Being able to quickly learn and anticipate more attacks that can circumvent a firewall in a timely manner, is very important to secure business data and services.

2.2.1 Literature survey on testing for SQLi Attacks

Various techniques are proposed within the literature to observe SQLi attacks based on a range of approaches, including white-box testing[7], static analysis[14], model-based testing[7], and black-box testing[1]. However, such techniques present some limitations that may adversely impact their practical pertinency as well as their vulnerability detection capability. As an example, white-box testing techniques and static analysis tools need access to source code[6], which might not be potential once coping with third-party elements or industrial appliances, and are linked to specific programming languages[26]. Model-based testing techniques require models expressing the protection policies or the implementation of WAFs and the net application under test[7], that are typically not offered or terribly troublesome to manually construct. Black-box testing ways don't need models or the source code however they're less effective in sleuthing SQLi vulnerabilities. Indeed, comprehensive reviews on black-box techniques[1],[6] have unconcealed that many sorts of security vulnerabilities (including SQLi attacks) remain mostly undiscovered and, thus, warrant more analysis.

Static Analysis and Runtime Monitoring[12] for analyzing the query that is to be executed on the server side which helped in prevention of Static and Dynamic SQL Injection attacks. A learning approach[9] generates attacks from a grammar and learns which pattern an attack cannot contain in order to escape detection. A framework[12] automatically test if a approach is correctly imposed by a firewall. Therefore, the framework generates a set of approaches as well as test traffic and checks whether how well the firewall handles the generated traffic according to the generated approach. Structural coverage condition of approaches[8] under test are defined and developed a test generation technique based on impulsion solving that tries to maximize structural coverage. Analysis for Monitoring and Neutralizing SQL Injection Attacks(AMNESIA)[5] does the required String analysis and also we don't need any Source code adjustments here. One of the major issues in this approach is that automatic detections is not available. Detection Rates are less than 75% which makes this approach unreliable.

2.3 ISSUES IN THE EXISTING SYSTEM

Currently there is no automatic detection and prevention of SQLi Attacks. The existing system makes the detection based on certain set of rules. The system checks the query with each and every rule and then it detects for attack. If a certain type of rule is not maintained in the set and if that attack is invoked, then the system will allow the query to be executed since that rule is not present in the rule set. It is also difficult to maintain the set of rules and test them. Also we need to define more complex rules for huge applications.

2.4 PROPOSED SYSTEM

The system for preventing SQL injections using machine learning helps in thwarting the exploitation of any database in the server side. In this system, a component is developed that resides in the server side. Before executing the request made by the client, our system will check for the SQL injection attack patterns that are appended with the values passed to the application server. There are two levels of security in this system, the first one being the patterns generated by the context free grammar rules and comparing the values with the pattern generated by the defined rules for SQL attacks. If the values are not detected in the first level it is sent to the second level of security that includes the machine learning algorithm(k-means) that clusters the pattern based on the given data set and categorizes the new data into one of those clusters and prevents the injection.

2.5 SUMMARY

The literature survey has covered information regarding the existing SQL Injection Attack Detection. The issues and challenges have been identified in related work. This chapter has highlighted the proposed model

CHAPTER 3

SYSTEM DESIGN

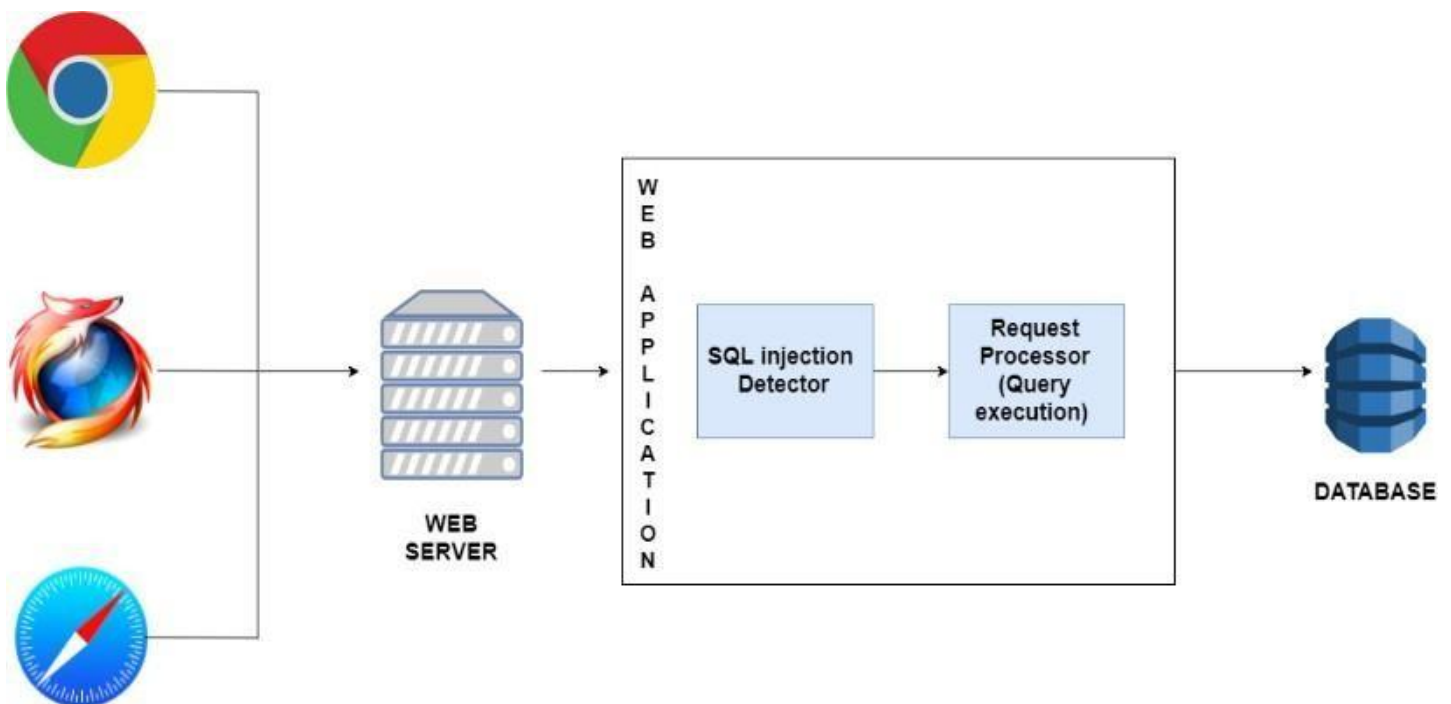
3.1 GENERAL

The main purpose of the design phase is to plan solution to the problems specified by the requirement document. The design phase takes as input the requirement in Requirements Analysis stage. This phase is the step in moving from problem domain to the solution domain. The design of the system is perhaps the most critical factor affecting the quality of the software, and has a major impact in the later phases, particularly in testing and maintenance. The output of this phase is the functional design document. The design documents created for the proposed system consists of the system architecture and the data flow diagram. This is also known to be top-level design which identifies the modules in the system, and the way in which the data transfer takes places between the modules.

This chapter deals with design documents created for the proposed system which consists of a functional architecture and activity diagram etc. During detailed design, the internal logic of all the modules specified in the system design is decided. The system architecture deals with the components to be used in the proposed system which explains its interaction with one other. The data flow diagram describes the graphical representation of how the data flows between the different modules in the system. The data flow diagram gives a very clear picture of the flow of data among the working modules. These design documents created are used as a continuous reference point for further system development and coding.

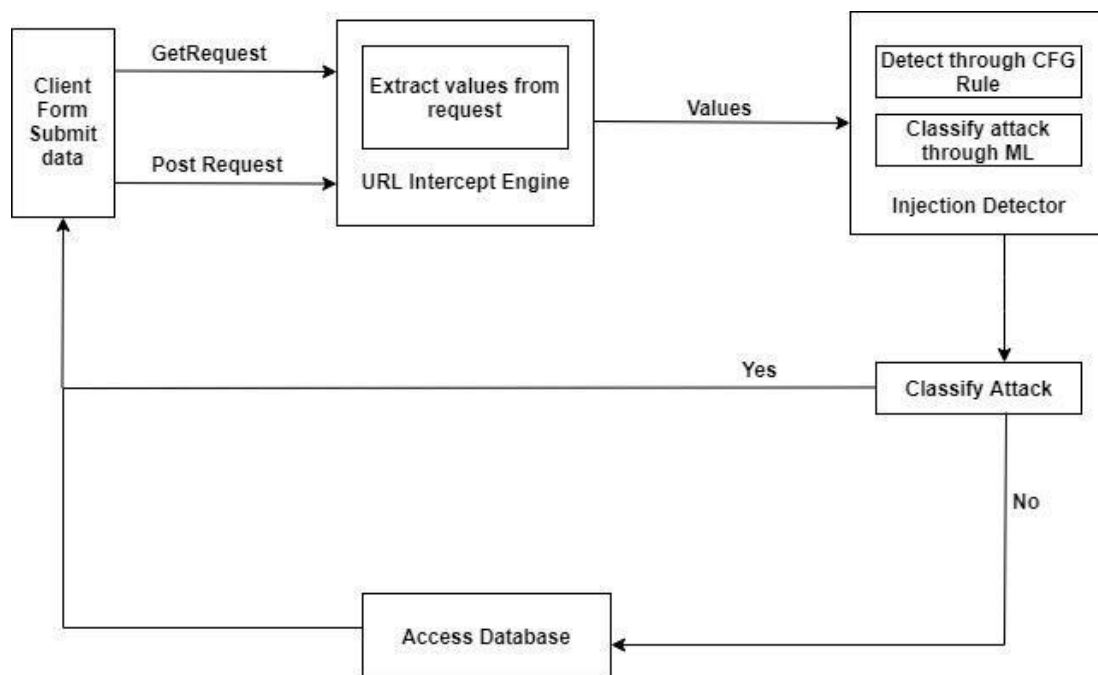
3.2 SYSTEM ARCHITECTURE

The web browser is used by the client to send necessary requests to the web server. The web browser sends the values provided by the client to the web server where a security component known as the SQL Injection Detector resides. It tests the values passed by the user for SQLi Attacks. If the values are valid, then it is sent to the Request Processor which processes the request and executes the query. Finally, the database is accessed and the request of the client is processed and executed.



3.3 FUNCTIONAL ARCHITECTURE

The following diagram in the figure 3.2 is the functional architecture of the project. It tells us about the complete flow of the project with all the minute details with utmost details. This architecture also highlights the various algorithms such as Context Free Grammar, K- means clustering which we have made use in this. First the client submits the data in the form. The request is sent to the URL Intercept Engine where the values are extracted from the request. The values are then passed to the Injection Detector which checks for pre-defined patterns using CFG and then classifies attack using K-means clustering. If there is no attack, the database is accessed with the help of the values and then the response is given to the client. If there is a attack, the access is prevented and the attack type is found.



3.4 MODULAR DESIGN

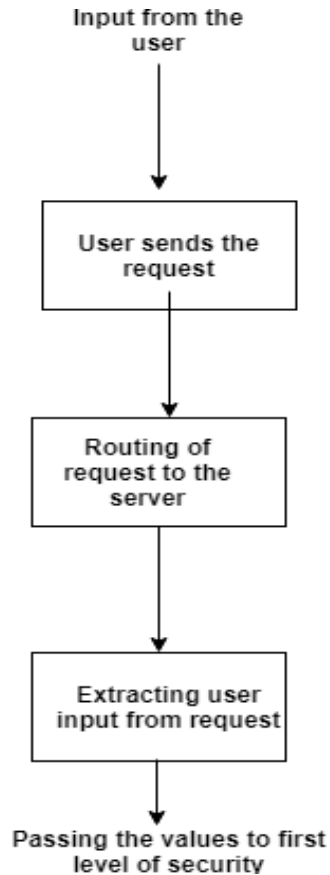
Modular structure defines structure of the overall module. Modularity is a general system concept typically defined as a degree to which a system's components may be separated and recombined.

The proposed system consists of the following modules:

- URL Intercept Engine
- Context-Free Grammar for SQLi Attacks
- Classify pattern through Machine Learning

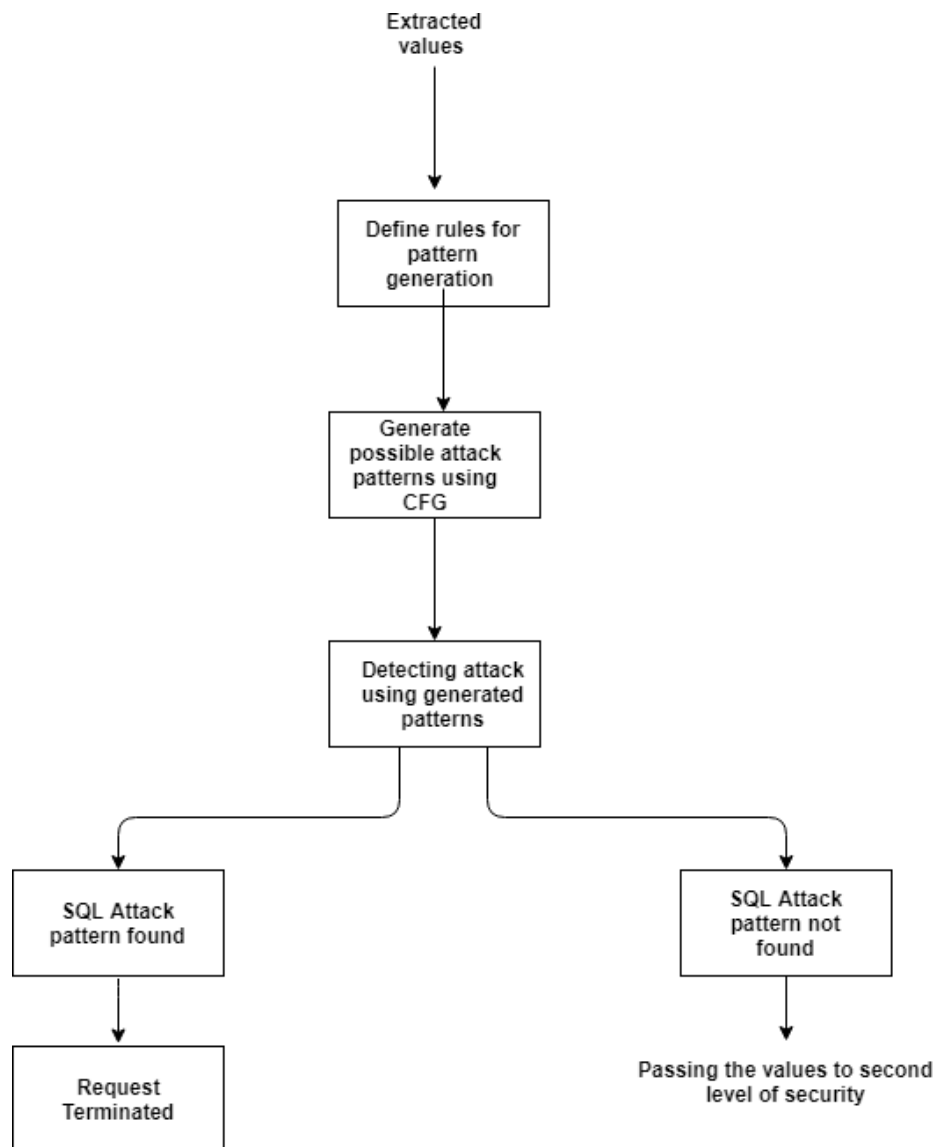
3.4.1 URL Intercept Engine

The text fields in the web applications are the most vulnerable part where most of the SQL injection attacks happen. URL (Uniform Resource Locator) and is defined as the global address of documents and other resources on the World Wide Web. We all use URLs to visit webpages and other resources on the web. The URL is an address that sends users to a specific resource online, such as a web page, video or other document or resource. The end users or hackers who are intended to affect the database of an organisation, enter some malicious SQL queries in the text field which can be appended to the SQL query, already defined in the server side, thus, affects the database. So the first module extracts the values given in the text fields and sends the values to pattern checking algorithm.



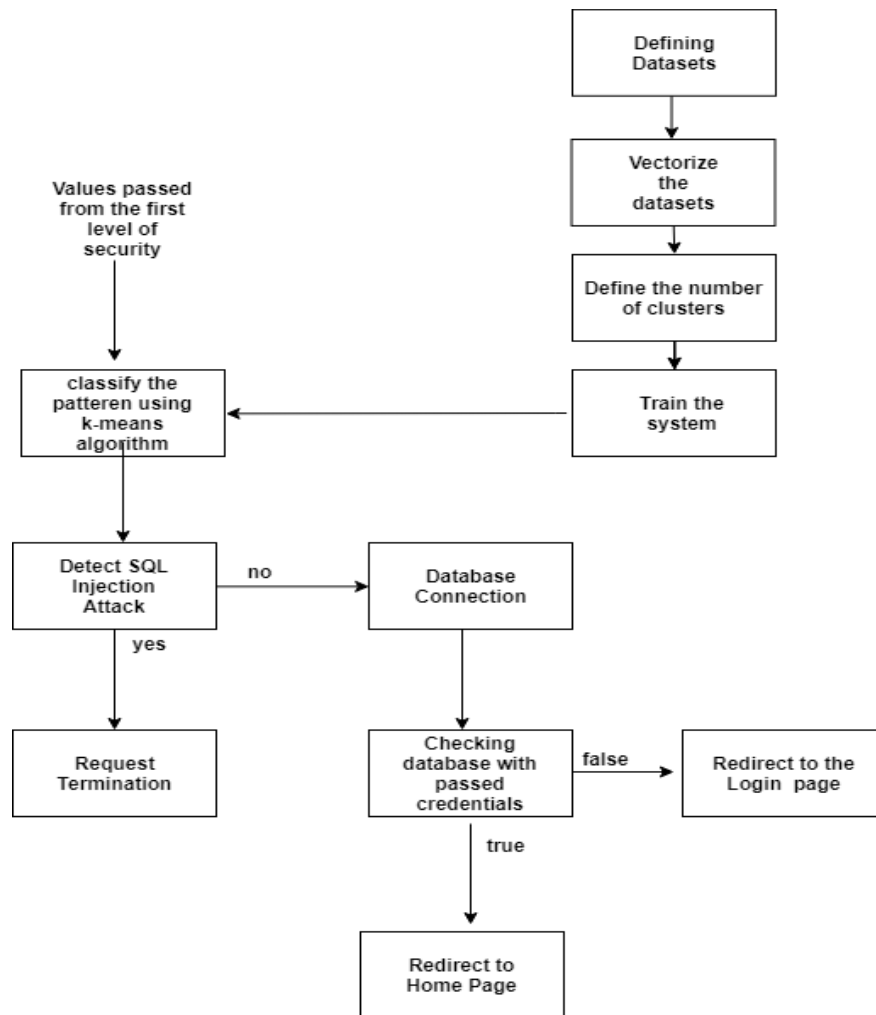
3.4.2 Context Free Grammar for SQLi Attacks

The second module is the first level of security from the SQL injection attacks. This module consists of context free grammar rules for the SQL injection attacks that generate the SQL injection attack pattern. The value extracted from the first module is checked with the rule that generates the different attack patterns. If the value extracted matches the attack pattern generated by the Context free grammar rules, the value is passed to the second level security that determines whether the value entered is malicious or not. Even if the value entered does not match the pattern generated by the Context free grammar rules, the value is passed to the second level security. These rules generate a pattern until no terminals are left.



3.4.3 Pattern Classification through Machine Learning

This module is the second level of security provided to the system. It consists of unsupervised machine learning algorithm that classifies the different types of attack into different clusters and tests the value passed to it by checking the fitness of the value to the cluster. If it matches with the cluster associated with any attack pattern, the system prevents the request from executing. If not, allows the request to execute the query and access the database.



3.5 SYSTEM REQUIREMENTS

Hardware Specification

Laptop specs : 8GB Ram

Server : Microsoft Windows Server

Hard disk : 1TB

Software Specification

Operating System : Windows 7 or higher, Linux, MAC

OS IDE : IntelliJ IDEA 2018.3.5

SDK : Python 3.7

Language : HTML, Python

Framework : Flask

3.6 SUMMARY

This chapter gives the overview of system design and its importance in software life cycle. The functional architecture gives the entire functionality of the proposed system along with its modular structure and its interactions between the modules.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 GENERAL

This chapter describes the implementation details of the project. It describes various modules of the proposed system, tools and platforms, which were used. Implementation details are also mentioned and snapshots of output for each module are displayed. The following presents the input and output models used while implementing the proposed system.

4.2 OVERVIEW OF THE PLATFORM

4.2.1 Python

Python is a general purpose and high level programming language. We can use Python for developing desktop GUI applications, websites and web applications. Also, Python, as a high level programming language, allows us to focus on core functionality of the application by taking care of common programming tasks. The simple syntax rules of the programming language further makes it easier for us to keep the code base readable and application maintainable.

4.2.2 Python Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program. The microframework Flask is based on the Werkzeug and Jinja2

Werkzeug

Werkzeug is a utility library for the Python programming language, in other words a toolkit for Web Server Gateway Interface (WSGI) applications, and is licensed under a BSD License. Werkzeug can realize software objects for request, response, and utility functions. It can be used to build a custom software framework on top of it and supports Python 2.6, 2.7 and 3.3.

Jinja

Jinja, also by Ronacher, is a template engine for the Python programming language and is licensed under a BSD License. Similar to the Django web framework, it provides that templates are evaluated in a sandbox.

4.2.3 HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML elements are the building blocks of HTML pages. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

4.2.4 Cascading Style Sheets

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content. CSS also has rules for alternate formatting if the content is accessed on a mobile device. The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable. The CSS specifications are maintained by the World Wide Web Consortium (W3C).

4.2.5 Microsoft SQL Server Management Studio

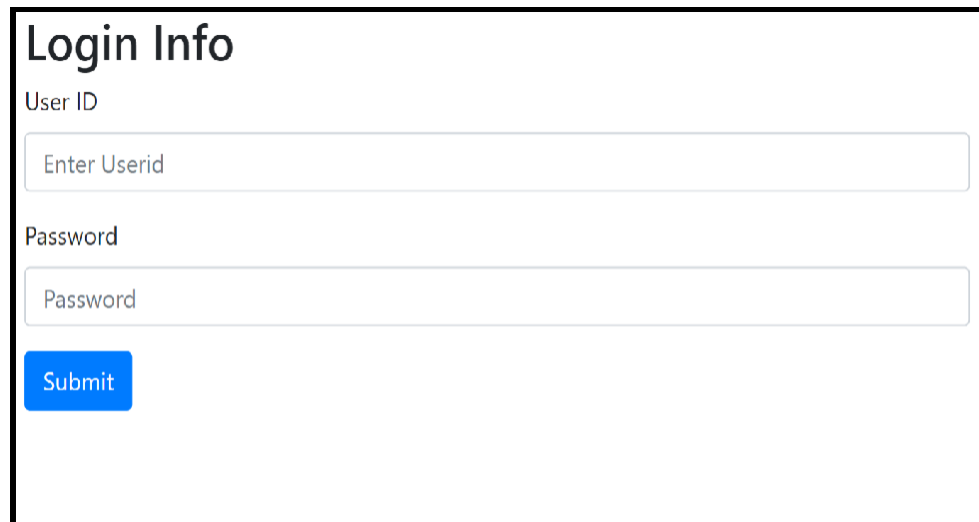
SQL Server Management Studio is a workstation component\client tool that will be installed if we select workstation component in installation steps. This allows us to connect to and manage our SQL Server from a graphical interface instead of having to use the command line. In order to connect to a remote instance of an SQL Server, we will need this or similar software. It is used by Administrators, Developers, Testers, etc.

SQL Server Management Studio (SSMS) is a software application first launched with Microsoft SQL Server 2005 that is used for configuring, managing, and administering all components within Microsoft SQL Server. The tool includes both script editors and graphical tools which work with objects and features of the server. A central feature of SSMS is the Object Explorer, which allows the user to browse, select, and act upon any of the objects within the server. It also shipped a separate Express edition that could be freely downloaded, however recent versions of SSMS are fully capable of connecting to and manage any SQL Server Express instance. Microsoft also incorporated backwards compatibility for older versions of SQL Server thus allowing a newer version of SSMS to connect to older versions of SQL Server instances. Starting from version 11, the application was based on the Visual Studio 2010 shell, using WPF for the user interface. Versions after 18 are based on the Visual Studio 2017 Isolated Shell.

4.3 MODULE IMPLEMENTATION

4.3.1 Implementation of the Frontend and the Backend

The Figure 4.1 represents the Frontend of the application where user enters the login credentials and it helps the user to easily send the request to server where the validation process takes place.



The image shows a web form titled "Login Info". It has two text input fields. The first is labeled "User ID" and contains the placeholder text "Enter Userid". The second is labeled "Password" and contains the placeholder text "Password". Below these fields is a blue button with the text "Submit".

Figure 4.1 Implementation of the frontend

The Figure 4.2 represents the database server where the credentials of all the users are stored and used for validation purpose. The username and password are stored in the database where username is the primary key of the database table.

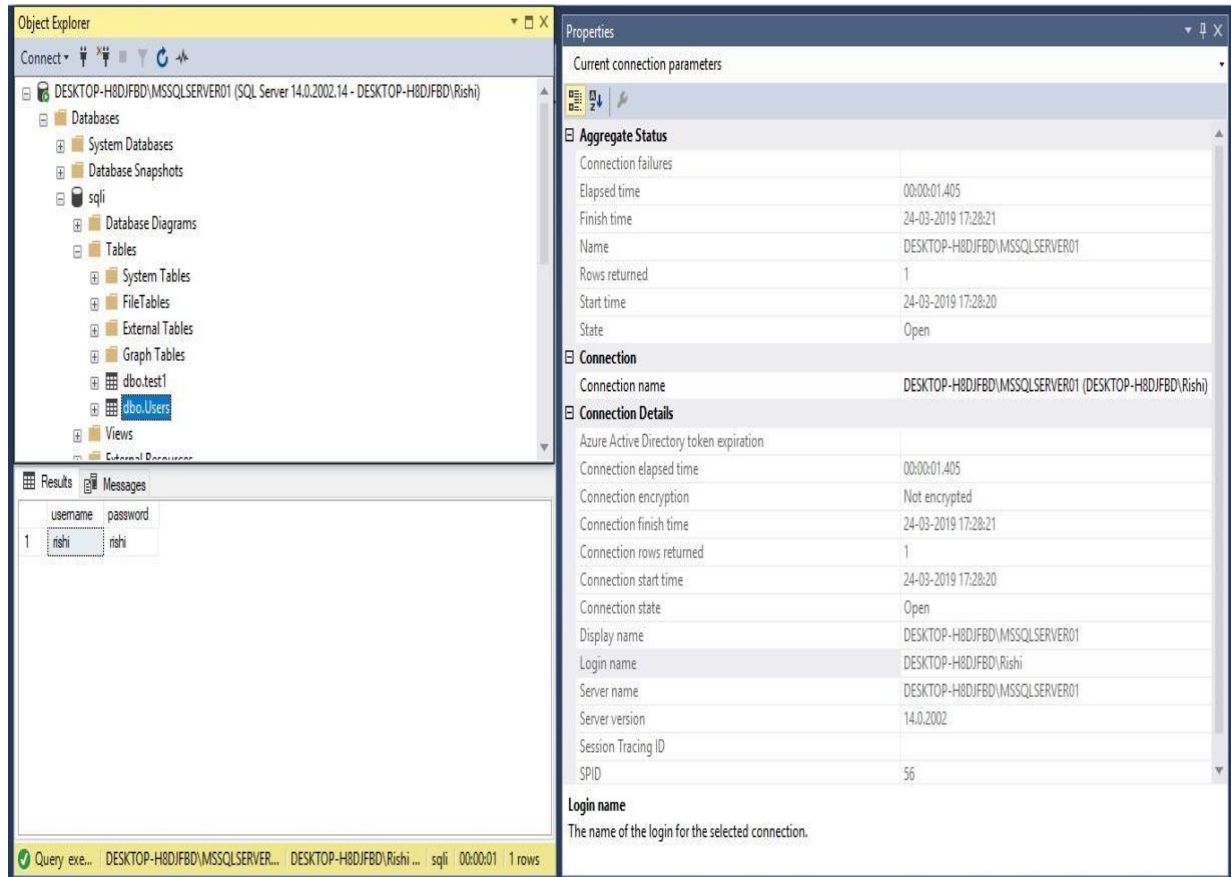


Figure 4.2 Backend - Database server

4.3.2 Implementation of Pattern Generation

The Figure 4.3 represents the rules defined for the random generation of the SQL injection attack pattern that can be injected along with the user input. These rules are utilised by the CFG function that generates the pattern. The rules are defined to generate three type of SQL Injection attacks namely Boolean Attack, Piggy Backed Attack and Union Attack. The user input is extracted from the request and then passed to the rules from which the possible attack patterns are generated by the function that can be possibly appended to the user input.

```

rules = {
  "SQLAttack": [
    ["booleanAttack"],
    ["commentAttack"],
    ["piggyAttack"],
    ["unionAttack"]
  ],
  "unionAttack": [
    [userid.partition(' ')[0], "union", "select"],
    [userid.partition(' ')[0], "select"]
  ],
  "piggyAttack": [
    [userid.partition(' ')[0], "semicolon", "SHUTDOWN", "semicolon", "--"] ,
    [userid.partition(' ')[0], "semicolon", "drop", "table", table_name, "semicolon", "--"],
    [userid.partition(' ')[0], "table", table_name, "semicolon", "--"] ,
    [userid.partition(' ')[0], "SHUTDOWN", "semicolon", "--"] ,
    [userid.partition(' ')[0], "drop", "table", table_name, "semicolon", "--"],
    [userid.partition(' ')[0], "semicolon", "SHUTDOWN", "--"] ,
    [userid.partition(' ')[0], "semicolon", "drop", "table", table_name, "--"],
    [userid.partition(' ')[0], "table", table_name, "--"] ,
    [userid.partition(' ')[0], "SHUTDOWN", "--"] ,
    [userid.partition(' ')[0], "drop", "table", table_name, "--"]
  ],
},

```

Figure 4.3 Rules for pattern generation

The Figure 4.4 represents the output of the possible attack pattern that can be appended along with the user input is generated by the rules defined as shown in Figure 4.3 .These patterns are compared with the user input and the attack is detected.

```

inputunionselect
input'--
input'--
inputunionselect
input'--
inputor'2'='2'--
input('3'='3')--
input'--
inputunionselect
input'--
inputselect
input;droptabledbo.test;--
input0-0--
inputselect

```

Figure 4.4 Possible attack patterns

4.3.3 Detection and prevention

The Figure 4.5 shows the detection of the SQL Injection generated by the CFG patterns and also the SQL query that is to be executed by the SQL server is also generated . The Figure 4.5 shows all type of attacks that can be appended with the username is generated and finally it matches with the one of the attack patterns generated by the CFG function.

```
dsj'or'3'='3'--  
dsj'unionselect  
dsj'unionselect  
dsj'select  
dsj'unionselect  
dsj'or'0'='0'--  
dsj'tabledbo.test--  
dsj'unionselect  
dsj'or1=1--  
pattern found in cfg  
SELECT * FROM Users WHERE username = 'dsj' or 1=1--' AND passwor  
127.0.0.1 - - [25/Mar/2021 00:05:15] "GET /processLogin?userid=d
```

Figure 4.5 Detection by the CFG function

The Figure 4.6 shows the detection of attack pattern by the machine itself as the machine is learned by a Unsupervised Learning Technique using KMeans Clustering Algorithm. Using NLTK the stop words such as and, or, from, to etc are detected and eliminated and only the meaningful words are used for teaching the machine. Based on the cluster, it detects the attack and prevents the request to be executed.

```
[nltk_data] Downloading package stopwords to
[nltk_data]      C:\Users\Rishi\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Prediction
table dbo.test --
rishi';drop table dbo.test -- : piggy based attack, cluster: [1
found attack pattern
127.0.0.1 - - [25/Mar/2021 00:24:13] "GET /processLogin?userid=r
```

Figure 4.6 Detection by Clustering

4.4 SUMMARY

This chapter brought out the analysis of each technology used to develop this project. This chapter also brought out the basic system implementations such as about the platforms, languages and tools used here. And the screenshots of different modules implemented using those platforms, languages and tools.

CHAPTER 5

SYSTEM TESTING

5.1 GENERAL

Testing is the process of trying to find out every conceivable fault or weakness in a work product. It provides a way to check functionality of components, subassemblies, assemblies and a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations.

5.2 TEST CASES

5.2.1 TEST CASE - 1: Login Validation

The user is allowed to input the data. If the username and password is valid, the web application will proceed to next page.

Table 5.1 Login Validation

TEST CASE ID	TEST DESCRIPTION	TEST INPUT	EXPECTED RESULT	ACTUAL RESULT
1.	Check login with registered user details	Valid registere d credential s	Should open the Home Page	Home page opened

2.	Check login with registered user details	Invalid registered credentials	should return the Login Page	Same Login Page opened
----	---	--------------------------------------	------------------------------------	------------------------------

5.2.2 TEST CASE - 2: Login without SQL injection detection

The user is allowed to input the data along with the attack pattern. Even though the login credentials are incorrect the user will be signed in and also at the same time database is affected.

Table 5.2 Login without SQL injection detection

TEST CASE ID	TEST DESCRIPTION	TEST INPUT	EXPECTED RESULT	ACTUAL RESULT
1.	Check login using user details with attack	Boolean Attack pattern	Should open the Home Page with wrong credentials	Home Page opened
2.	Check login using user details with attack	Piggyback Attack	Should open the Home Page with wrong credentials and database should be affected	Home Page opened and Database affected

5.2.3 TEST CASE - 3: Login with SQL injection detection

The user is allowed to input the data along with the attack pattern. If the login credentials are incorrect the user will be shown the Login Page again. Also if the login credentials contains malicious pattern along with them, it is detected and prevented and returns the Login Page.

Table 5.3 Login with SQL injection detection

TEST CASE ID	TEST DESCRIPTION	TEST INPUT	EXPECTED RESULT	ACTUAL RESULT
1.	Check login using user details with attack	Boolean Attack pattern	Attack should be detected and return the Login Page	Attack detected and Login Page returned
2.	Check login using user details with attack	Piggyback Attack	Attack should be detected and protect the database and also return the Login Page	Attack detected Login Page returned and Database protected

5.3 PERFORMANCE MEASURES

5.3.1 Accuracy

The accuracy of a test is its ability to differentiate the types of attacks correctly. To estimate the accuracy of a test, we should calculate the proportion of true positive and true negative in all evaluated cases.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

Where True positive (TP) = the number of attack pattern correctly

identified False positive (FP) = the number of attack pattern incorrectly

identified True negative (TN) = the number of attack pattern correctly
identified

False negative (FN) = the number of attack pattern incorrectly identified

5.3.2 Response Time

It is the time, a system or a functional unit takes to react to a given input. In other words, it is the length of time taken for a system or a person to react to a given stimulus or event.

Response Time = Reporting period / Check frequency interval

5.3.3 Time Complexity

The time requirement for K-means is calculated as

$$O(I * K * m * N)$$

Where I is the number of iterations required, n is the number of attributes, m is the number of points, K is the number of clusters and I is often small and can usually be safely bounded, as most of the changes occur in the first few iterations.

5.4 PERFORMANCE ANALYSIS

The performance of the proposed system can be measured by Accuracy of the algorithm and time taken to cluster the datasets. The algorithm of proposed system is being compared with the different clustering algorithms and the accuracy and time taken to cluster the data is measured.

5.4.1 Time taken to cluster

The performance of the proposed system based on the time taken to cluster the datasets of different clustering algorithms are depicted as shown in the Figure 5.1. The Bar graph has been created based on the time taken by three different clustering algorithms.

The vertical axis consists of the time taken to cluster the data in seconds and the horizontal axis consists of the number of clusters

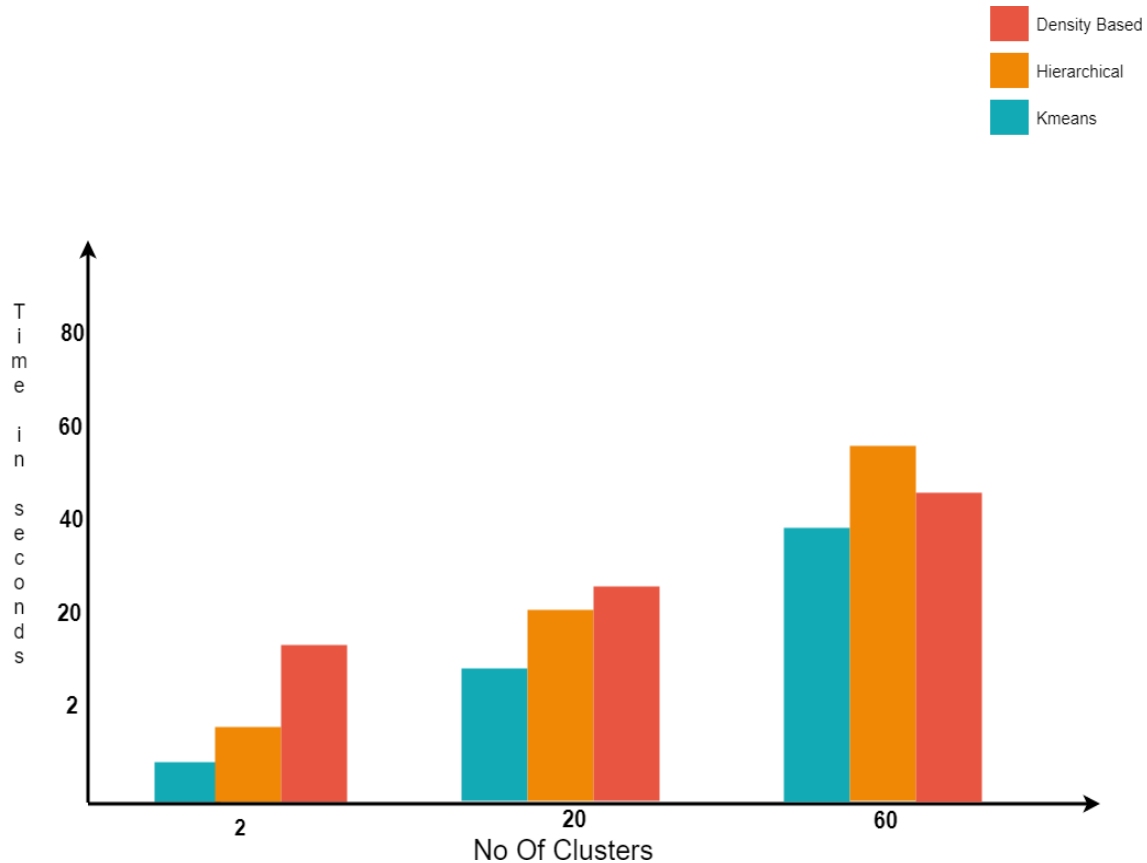


Figure 5.1 Performance analysis based on time taken to cluster

5.4.2 Accuracy

The performance of the proposed system based on accuracy of various algorithms can be depicted as shown in the Figure 5.2. The Bar graph has been created based on the accuracy from three different algorithms .The vertical axis consists of the accuracy percentage and the horizontal axis consists of number of clusters.

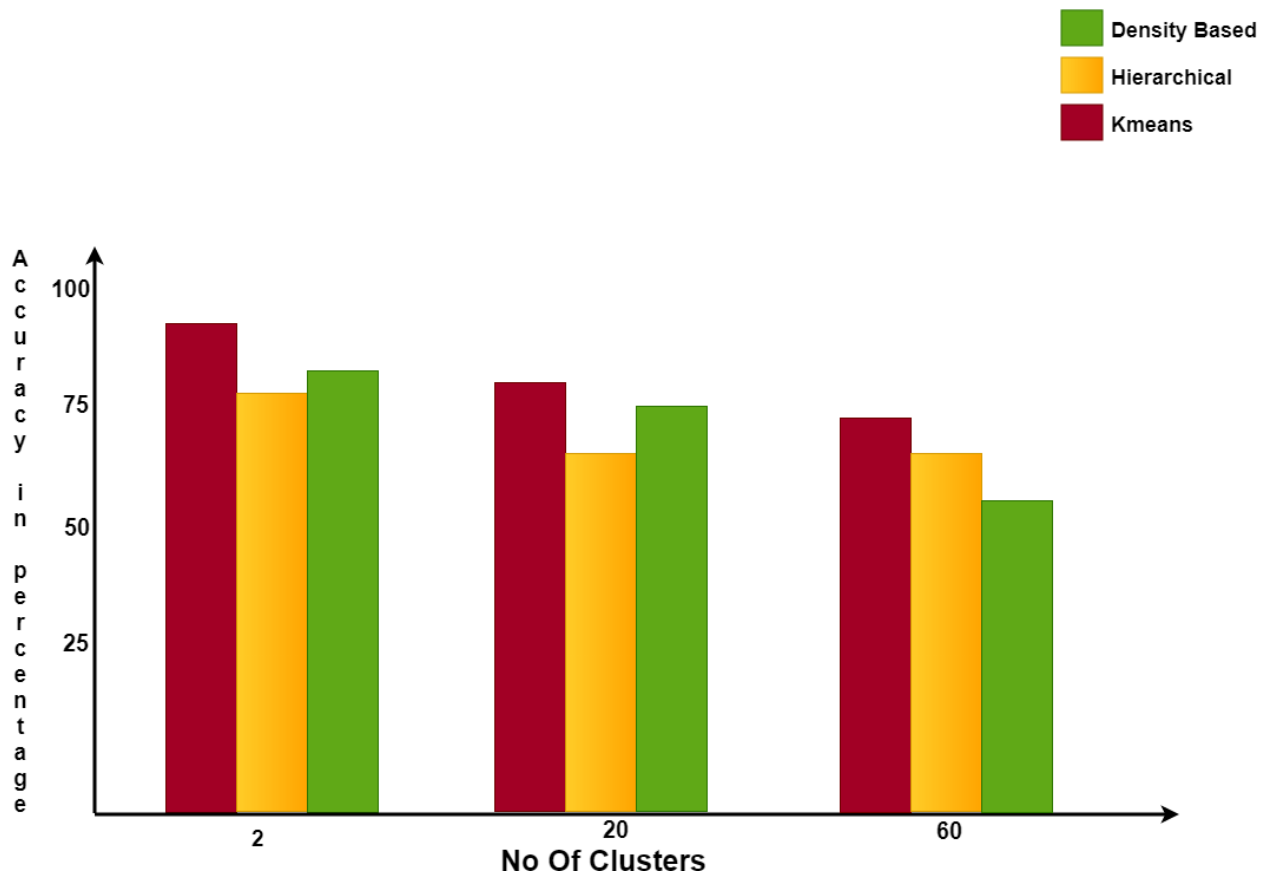


Figure 5.2 Performance analysis based on accuracy

5.5 SUMMARY

This chapter brought out the general description of the different testing process applicable to the entire project development. It also considers the performance analysis of a system which proved to improve the level of user satisfaction compared to the existing system.

CHAPTER 6 CONCLUSION

AND FUTURE WORK

6.1 CONCLUSION

This chapter concludes all the ideas that were discussed till now in the proposed system. The system proposed uses a machine learning approach to detect and prevent SQL injection attacks via user requests in a web application. There are three modules in the system, namely, URL Intercept Engine, Context-Free Grammar for SQLi Attacks and Classify pattern through Machine Learning. At first, the values entered by the user on the client side is extracted using URL Intercept Engine which is then passed to the first level of security. The first level of security detects the pattern using the Context-Free Grammar rules. The patterns are passed to the Machine Learning Algorithm which classifies the value to clusters based on the pattern. If the value is found malicious by the implemented security then the request made by the client will be revoked. If not malicious, the request will be executed normally.

6.2 FUTURE WORK

Future enhancements of this project can include the prevention of Cross- Site Scripting(XSS) and Path Traversal Attacks in web applications using machine learning techniques.

XSS enables attackers to inject client-side scripts into web pages viewed by other users. A XSS vulnerability may be used by attackers to bypass access controls such as the same-origin policy.

A path traversal attack (also known as directory traversal) aims to access files and directories that are stored outside the web root folder. It should be noted that access to files is limited by system operational access control (such as in the case of locked or in- use files on the Microsoft Windows operating system). This attack is also known as “dot-dot-slash”, “directory traversal”, “directory climbing” and “backtracking”.

Both these attacks are major threats to web applications. The use of machine learning techniques for the detection and prevention of cross-site scripting and path traversal attacks can be developed using the efficient use of technology.

