

**INOVATIVE PRODUCT DEVELOPMENT-I**  
**Report on**  
**SQL INJECTION AVOIDANCE FOR PROTECTED DATABASE**  
**WITH ASCII**

**Submitted by**

B.Sai Ruchitha  
20RH1A6904

S.Sushma  
20RH1A6947

S.Anjali  
20RH1A6948

*Under the Esteemed Guidance of*  
**Dr.S.Pradeep**  
**Associate Professor of CSE**

*In partial fulfillment of the Academic Requirements for the Degree of*

**BACHELOR OF TECHNOLOGY**  
**Computer Science and Engineering - IOT**



**MALLA REDDY ENGINEERING  
COLLEGE FOR WOMEN**

**MALLA REDDY ENGINEERING COLLEGE FOR WOMEN**

**(Autonomous Institution - UGC- Govt. of India)**

**Accredited by NBA & NAAC with 'A' Grade**

**NIRF Indian Ranking, Accepted by MHRD, Govt. of India**

**Band National Ranking by ARIIA, Accepted by MHRD, Govt. of India**

**Affiliated to JNTUH, Approved by AICTE, ISO 9001:2015 Certified Institution**

**Maisammaguda, Dhulapally (post), Secunderabad, Telangana**

**December-2022- 2023**

# **MALLA REDDY ENGINEERING COLLEGE FOR WOMEN**

**(Autonomous Institution - UGC- Govt. of India)**



**Accredited by NBA & NAAC with 'A' Grade**

**NIRF Indian Ranking, Accepted by MHRD, Govt. of India**

**Band National Ranking by ARIIA, Accepted by MHRD, Govt. of India**

**Affiliated to JNTUH, Approved by AICTE, ISO 9001:2015 Certified Institution**

**Maisammaguda, Dhulapally (post), Secunderabad, Telangana**

**December-2022- 2023**

---

## **CERTIFICATE**

This is to certify that An Innovative Product Development – I entitled “SQL injection avoidance for protected database with ASCII” being submitted by

**B.Sai Ruchitha                      20RH1A6904**

**S.Sushma                              20RH1A6947**

**S.Anjali                                20RH1A6948**

In partial fulfillment for the award of degree of **BACHELOR OF TECHNOLOGY** in **Computer Science and Engineering –IOT**, **Jawaharlal Nehru Technological University, Hyderabad** during the academic year 2022-2023 is a report of bonafide piece of work, undertaken by him/her under the supervision of the undersigned.

**Approved and supervised by**

**Dr. S. Pradeep**

**Head of IOT and Associate Professor,**

**Dept of CSE-IOT.**

# MALLA REDDY ENGINEERING COLLEGE FOR WOMEN



(Autonomous Institution - UGC- Govt. of India)

Accredited by NBA & NAAC with 'A' Grade

NIRF Indian Ranking, Accepted by MHRD, Govt. of India

Band National Ranking by ARIIA, Accepted by MHRD, Govt. of India

Affiliated to JNTUH, Approved by AICTE, ISO 9001:2015 Certified Institution

Maisammaguda, Dhulapally (post), Secunderabad, Telangana

December-2022- 2023

---

## Department of Computer Science and Engineering-IOT

### DECLARATION

We hereby declare that An Innovative Product Development –I Entitled “**SQL injection avoidance for protected database with ASCII** “ done by ‘**B.Sai Ruchitha (20RH1A6904), S.Sushma (20RH1A6947) , S.Anjali (20RH1A6948)** students of **Bachelor of Technology in Computer Science and Engineering –IOT** Submitted to **Malla Reddy Engineering College for Women(Autonomous), Maisammaguda, Secunderabad** Affiliated to **Jawaharlal Nehru Technological University, Hyderabad (JNTUH)** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

**Date:08-12-2022**

**1) B.Sai Ruchitha (20RH1A6904)**

**2) S.Sushma (20RH1A6947)**

**3) S.Anjali (20RH1A6948)**

## ACKNOWLEDGEMENT

We feel honored and privileged to place our warm salutation to our college **Malla Reddy Engineering College for Women(Autonomous)** and **Department of Computer Science and Engineering-IOT** which gave us the opportunity to have expertise in engineering and profound technical knowledge.

We would like to deeply thank our Honourable Minister of Telangana State **Sri. Ch. MALLA REDDY Garu**, Founder Chairman MRGI, the largest cluster of institutions in the state of Telangana for providing us with all the resources in the college to make our project successful.

We wish to convey gratitude to our Principal **Dr. Y. MADHAVEE LATHA**, for providing us with the environment and mean to enrich our skills and motivating us in our Endeavour and helping us to realize our full potential.

We would like to thank **Prof. A. RADHA RANI**, Director of Computer Science and Engineering & Information Technology for encouraging us to take up a project on this subject and motivating us towards the Project Work.

We express my sincere gratitude to **Dr. S.PRADEEP**, Head of CSE- IOT and Associate Professor, Department of Computer Science and Engineering for inspiring us to take up a project on this subject and successfully guiding us towards its completion.

I would like to thank our Internal Guide **Dr. S. PRADEEP**, Head of CSE- IOT and Associate Professor, Department of Computer Science and Engineering and all the Faculty members for their valuable guidance and encouragement towards the completion of our project work.

**With Regards and Gratitude**

- |                   |              |
|-------------------|--------------|
| 1) B.Sai Ruchitha | (20RH1A6904) |
| 2) S.Sushma       | (20RH1A6947) |
| 3) S.Anjali       | (20RH1A6948) |

## LIST OF FIGURES

Fig 1.1: The state of SQL injection attacks	4
Fig 1.2: The top 10OWASPmostcritical Web application security risks	4
Fig 2.3: System Architecture	9
Fig 2.4: Class diagram for the system	10
Fig 2.5: Use case diagram of the system	10
Fig 5.1: testing process	17
Fig 6.1: Opening screenshot of the project	18
Fig 6.2: Register the user	18
Fig 6.3: Login with user	19
Fig 6.4: If it is error	19

# INDEX

•	<b>ABSTRACT</b>	<b>1</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>2-7</b>
	1.1 SQL Injection	2
	1.2 Types of SQL Injection	3-4
	1.3 Web application	5
	1.4 Problem Definition	6
	1.5 Project purpose	7
	1.6 Project features	7
<b>2.</b>	<b>SYSTEM DESIGN</b>	<b>8-10</b>
	2.1 System development methodology	8
	2.2 Model phases	8
	2.3 System Architecture	9
	2.4 Classes Designed for the system	9
	2.5 Use case diagram	10
<b>3</b>	<b>IMPLEMENTATION</b>	<b>11-13</b>
	3.1 Algorithm	12-13
<b>4</b>	<b>Attack Details</b>	<b>14-15</b>
	4.1 Attack detection	14
	4.2 Attack prevention	15
<b>5.</b>	<b>TESTING</b>	<b>16-18</b>
	5.1 Unit Testing	16
	5.2 Validation testing	17
	5.3 System testing	17
	5.4 Testing of initialization and UI components	18
<b>6.</b>	<b>INTERPRETATION OF RESULT</b>	<b>19-20</b>
<b>7.</b>	<b>CONCLUSION</b>	<b>21</b>
<b>8.</b>	<b>FUTURE ENHANCEMENTS</b>	<b>22</b>
<b>9.</b>	<b>REFERENCES</b>	<b>23</b>

## ABSTRACT

SQL injection (SQLi) is an application security weakness that allows attackers to control an application's database – letting them access or delete data, change an application's data-driven behavior & other undesirable things. These weaknesses occur when an application uses untrusted data, such as data entered into web form fields, as part of a database query. When an application fails to properly sanitize this untrusted data before adding it to a SQL query, an attacker can include their own SQL commands which the database will execute.

In this project we develop a SQLi attack detection and prevention system. The proposed solution is based on SQLi Signature detection and Anomaly detection approaches. The patterns of SQLi attacks are stored in database. When a web form either through POST or GET request is received, it is first sent to SQLi signature detection module to match if the request parameters has any SQL query patterns and if any pattern found, the FORM request is not processed and user is redirected to Failure response page.

The proposed system of detection and prevention of SQLi is tested against a E-commerce web application. The E-commerce web application will have Forms for searching products, prices etc. SQLi attack is launched by typing SQL queries in these Forms and see the response of the server. The server response with and without proposed SQLi detection and prevention system will be demonstrated in this project.

The project will be implemented as Web Framework Glass Fish web server. The coding will be done using PHP. The attack signatures are kept in a file, so new signatures can be added and the system can be improvised.

The attack signatures are stored in encrypted way (Using AES), so that it is not possible for any attacker to modify the attack signatures.

This project will be very useful online shopping and e-commerce websites with Forms. Leakage of sensitive information in database due to SQL injection attack is prevented by using the proposed solution.

# CHAPTER-1

## INTRODUCTION

### 1.1 SQL Injection

SQL injection is an attack in which SQL code is inserted or appended into application/user input parameters that are later passed to a back-end SQL server for parsing and execution. Any procedure that constructs SQL statements could potentially be vulnerable, as the diverse nature of SQL and the methods available for constructing it provide a wealth of coding options. The primary form of SQL injection consists of direct insertion of code into parameters that are concatenated with SQL commands and executed. A less direct attack injects malicious code into strings that are destined for storage in a table or as metadata. When the stored strings are subsequently concatenated into a dynamic SQL command, the malicious code is executed.

When a Web application fails to properly sanitize the parameters which are passed to dynamically created SQL statements (even when using parameterization techniques) it is possible for an attacker to alter the construction of back-end SQL statements. When an attacker is able to modify an SQL statement, the statement will execute with the same rights as the application user; when using the SQL server to execute commands that interact with the operating system, the process will run with the same permissions as the component that executed the command (e.g., database server, application server, or Web server), which is often highly privileged. A successful SQL injection exploit can read sensitive data from the database, modify database (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system.



## 1.2 Types of SQL Injection types and attacks:

- **Inband SQLi**: Error based Union
- **Inferential/Blind SQLi**:      Boolean based blind  
   Time based blind
- **Out of band SQLi**
- **Inband**: when an attacker is able to use the same communication channel to both launch the attack and gather results.
- **Error based**: Relies on error messages thrown by the database server to obtain information about the structure of the database
- **Union-based**: It is an in-band SQL injection technique that leverages the UNION SQL operator to combine the results of two or more SELECT statements into a single result which is then returned as part of the HTTP response.
- 
- **Blind sql**: No data is actually transferred via the web application and the attacker would not be able to see the result of an attack unlike in-band. Instead, an attacker is able to construct the database structure by sending payloads, observing the web application's response and the resulting behavior of the database server.
- 
- **Out-of-band sql**: It occurs when an attacker is unable to use the same channel to launch the attack and gather results
- Attacks:
- ->And/or attack
- ->Union attack
- ->Hexa decimal attack
- ->String concatenation attack

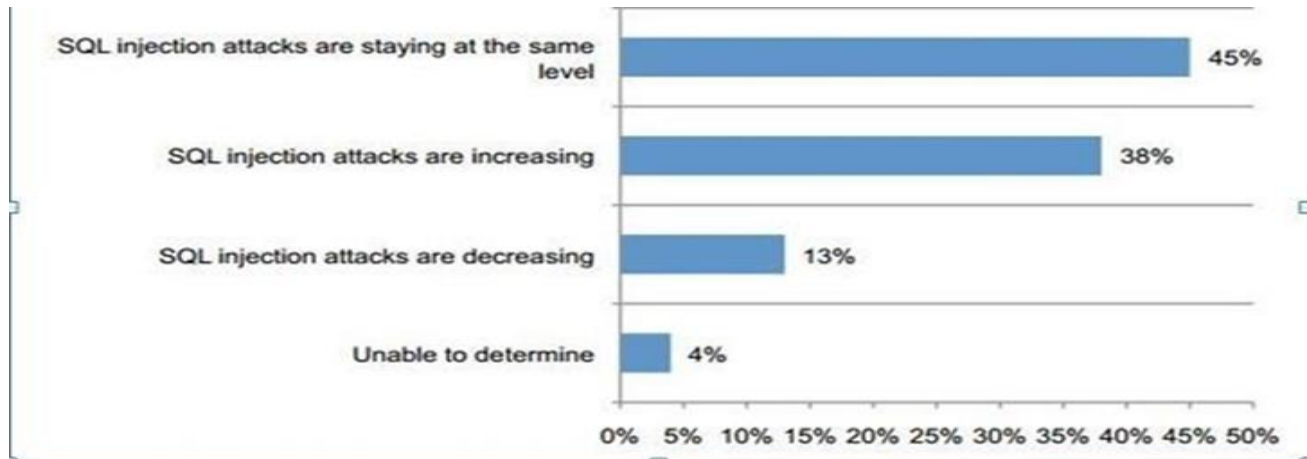


Fig 1.1: The state of SQL injection attacks

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	⊗	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	⊗	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Fig 1.2: The top 10OWASPmostcritical Web application security risks

### **1.3 Web Application and Vulnerability**

Advances in web technologies coupled with a changing business environment, mean that web applications are becoming more prevalent incorporate, public and Government services today. Although web applications can provide convenience and efficiency, there are also a number of new security threats, which could potentially pose significant risks to an organization's information technology infrastructure if not handled properly. The rapid growth in web application deployment has created more complex distributed IT infrastructures that are harder to secure. For more than a decade, organizations have been dependent upon security measures at the perimeter of the network, such as firewalls, in order to protect IT infrastructures.

However, now that more and more attacks are targeting security flaws in the design of web applications, such as injection flaws, traditional network security protection may not be sufficient to safeguard applications from such threats. These threats originate from non-trusted client access points, session-less protocols, the general complexity of web technologies, and network-layer insecurity.

With web applications, client software usually cannot always be controlled by the application owner. Therefore, input from a client running the software cannot be completely trusted and processed directly. An attacker can forge an identity to look like a legitimate client, duplicate a user's identity, or create fraudulent messages and cookies. In addition, HTTP is a session-less protocol, and is therefore susceptible to replay and injection attacks. Hypertext Transport Protocol messages can easily be modified, spoofed and sniffed. As such, organizations must understand and be fully aware of the threats to properly implement appropriate defensive strategies. Additional security controls, both technical and administrative, may be required to reinforce the protection of vital infrastructure to the deployment of web applications.

## 1.4 Problem Definition

Given the perspective of time passed since web applications entered the commercial market, SQL injection is hardly a new threat. The problem has been described by many security professionals and hackers, and the information is widely spread on the Internet. Many attempts have also been made in order to find countermeasures that can contend with and overcome SQL injection threats. In addition, the countermeasures constitute new solutions regarding application layer vulnerabilities in general and SQL injection threats in particular. It even seems to be a somewhat common assumption among writers to think that protecting web applications from SQL injection is an easy task, as long as you have an understanding of the SQL injection threat. Never the less corporations, security professionals and hackers continue to announce that SQL injection vulnerabilities are inherent in web applications and reports of compromised applications are frequently published. This clearly indicates that there still seems to exist a lack of awareness, knowledge and respect of SQL injection threats inherent in web applications among security professionals. One reason might be that software development companies and third party vendors do not take a structured security approach when developing web applications. Another reason is that software developers compete in introducing software to the market.

Several commercial and open source tools were used to detect SQL Injection vulnerabilities in a set of vulnerable web application. The results for penetration testing tools and static code analysis tool were analyzed separately and then compared to better understand the strengths and weaknesses of each approach. A key observation is that different tools implementing the same approach frequently report different vulnerabilities in the same piece of code. Although the results of this study cannot be generalized, they highlight the strengths and limitations of both approaches and suggest that future research on this topic is of utmost important. Finally, results show that web application programmers should be very careful when selecting a vulnerability detection approach and when interpreting the results provided by automated tools.

## 1.5 Project Purpose

---

The objective of this project is to make the database on the E-Commerce website secure by preventing SQL injection attacks while firing queries to database. The system uses SQL Injection intrusion detection and prevention mechanism and encryption technique (AES-Advanced Encryption Standard) .The attack signatures are stored in encrypted way, so that it is not possible for any attacker to modify the attack signatures to keep the data safe and secure.

The server response with and without proposed SQLi detection and prevention system will be demonstrated in this project.

This purpose of this project is that it will be very useful online shopping and e-commerce websites with Forms. Leakage of sensitive information in database due to SQL injection attack is prevented by using the proposed solution. It is more secure, lesser vulnerable to attacks and more robust. The data present in web application is also secured tightly, so data cannot be hacked, modified ,deleted or stolen.

## **1.6 Project Features**

The basic features of this project is based on SQLi Signature detection and Anomaly detection approaches. The patterns of SQLi attacks are stored in database. When a web form either through POST or GET request is received, it is first sent to SQLi signature detection module to match if the request parameters has any SQL query patterns and if any pattern found, the FORM request is not processed and user is redirected to Failure response page.

When there is no SQLi pattern, the user behaviour is collected in terms number of times here requested in a period of time, different source ip address it is requesting web pages and based on any abnormal behaviour threshold, the user is denied access to web page for certain duration.

The proposed system is tested against a E-commerce web application, which will have forms for searching products, prices etc .SQLi attack is launched by typing SQL queries in these forms and see the response of the server.

## CHAPTER-2

### SYSTEM DESIGN

System design of the project is given in detail in this chapter.

#### 2.1 System development methodology

System development method is a process through which a product will get completed or a product gets rid from any problem. Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps which is used for product progress. The development method followed in this project is waterfall model.

#### 2.2 Model phases

The waterfall model is a successive programming improvement process, in which advance is seen as streaming relentlessly downwards (like a waterfall) through the periods of Requirement start, Analysis, Design, Implementation, Testing and upkeep.

**Prerequisite Analysis:** This stage is worried about gathering of necessity of the framework. This procedure includes producing record and necessity survey.

**Framework Design:** Keeping the prerequisites at the top of the priority list the framework details are made an interpretation of into a product representation. In this stage the fashioner underlines on: - calculation, information structure, programming design and soon.

**Coding:** In this stage developer begins his coding with a specific end goal to give a full portrayal of item. At the end of the day framework particulars are just changed over into machine coherent register code

**Usage:** The execution stage includes the genuine coding or programming of the product. The yield of this stage is regularly the library, executables, client manuals and extra programming documentation

**Testing:** In this stage all projects (models) are coordinated and tried to guarantee that the complete framework meets the product prerequisites. The testing is worried with check and approval.

**Support:** The upkeep stage is the longest stage in which the product is upgraded to satisfy the changing client need, adjust to suit change in the outside environment, right mistakes and over sights before hand undetected in the testing stage, improve the proficiency of the product.

## 2.3 System Architecture

The System architecture is shown below.

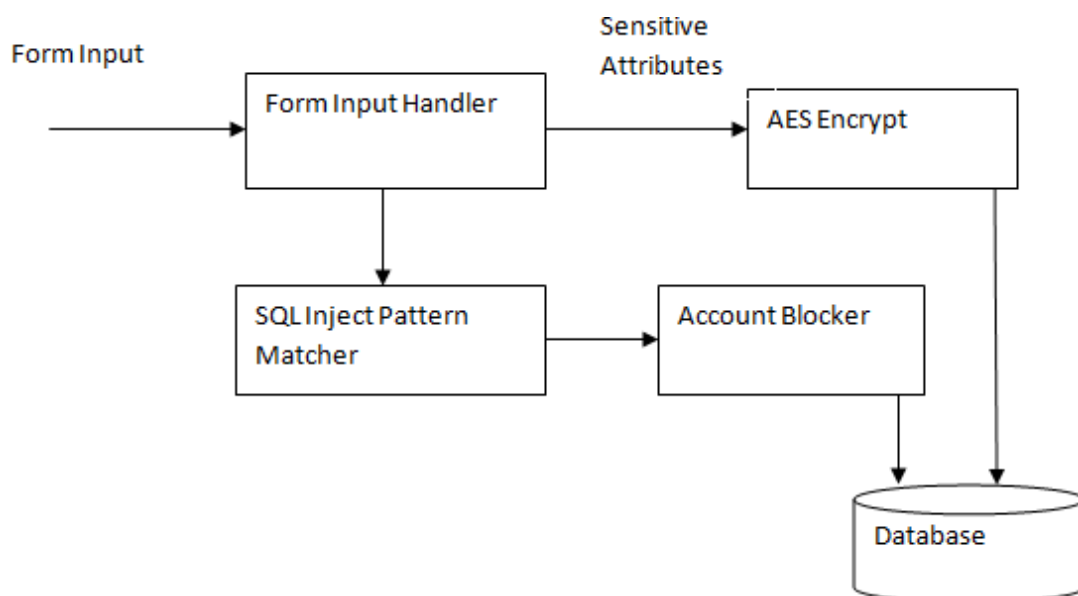
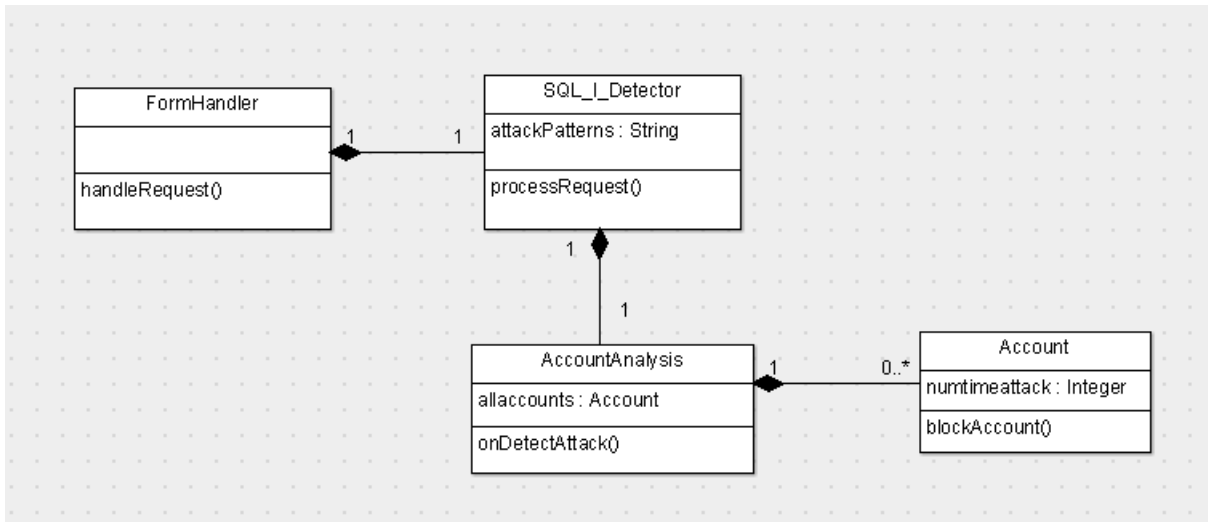


Fig 2.3: System Architecture

## 2.4 Classes Designed for the system

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

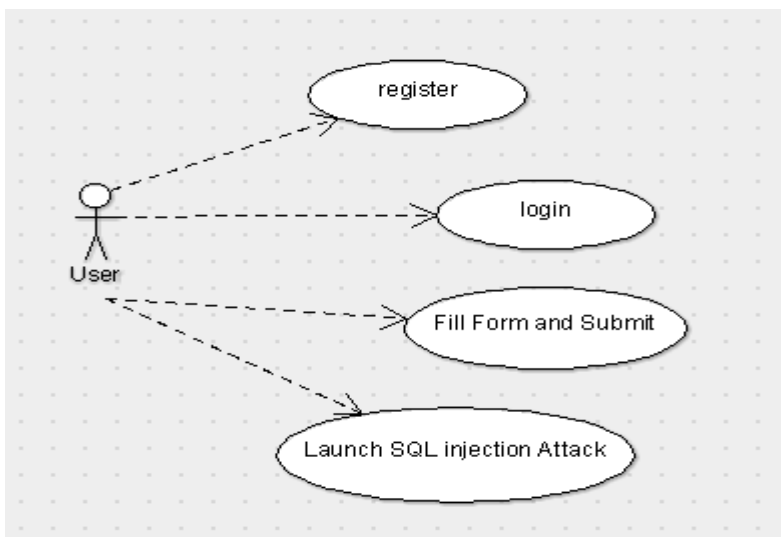
The class diagram is shown below. The class diagram has the following classes:



**Fig 2.4: Class diagram for the system**

## 2.5 Use case diagram of the system

A use case diagram is a type of behavioral diagram created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.



**Fig 2.5: Use case diagram of the system**



## CHAPTER-3

### IMPLEMENTATION

#### Language used for implementation

Execution stage ought to consummately outline plan archive in a suitable programming dialect keeping in mind the end goal to accomplish the essential last and right item. Frequently the item contains blemishes and gets demolished because of erroneous programming dialect decided for execution. In this anticipate, for execution reason PHP is picked as the programming dialect. Few explanations behind which PHP is chosen as a programming dialect can be sketched out as takes after:-

**Platform Independence:** PHP compilers don't deliver local item code for a specific stage but instead 'byte code' directions for the PHP Virtual Machine (JVM). Making PHP code deal with a specific stage is then just an issue of composing a byte code translator to reenact a JVM. What this all methods is that the same assembled byte code will run unmodified on any stage that backings PHP.

**Objects Orientation:** PHP is an unadulterated item situated dialect. This implies everything in a PHP system is an article and everything is slipped from a root object class.

**Rich Standard Library:** One of PHP's most appealing elements is its standard library. The PHP environment incorporates many classes and strategies in six note worthy practical zones:-

Language Support classes for cutting edge dialect elements, for example, strings, exhibits, strings, and exemption taking care of.

- Utility classes like an arbitrary number generator, date and time capacities, and holder classes.
- Input/yield classes to per use and compose information of numerous sorts to and from an assortment of sources.
- Networking classes to permit between PC correspondences over a neighborhood system or the Internet.

### 3.1 Algorithm

SQLI Attack Detection

AlgorithmInput : form request ,

account

userDontprocessform  $\leftarrow$  false

Foreach param in request

    Res  $\leftarrow$  Look up for attack signature in param

    if Res is True

        Send error

        responseDontprocessform  $\leftarrow$  true

        continue; Break;

    End

End

If DontProcessform == true

    Update numtimes attacked = numtimes+1 in

    DB If numtimes > 5

        Block the account

    userEnd

End

If

Dontprocessform=falsefp

aram issensitive

Eparam $\leftarrow$ AESEncrypt(param)St

roreEparamin DB

End

End

## CHAPTER-4

### ATTACK DETAILS

SQL injection attack launched for two modes

1. Updating the data base through attack
2. Getting the sensitive values from database through attack.

Update attack is implemented through providing a INSERT query as a form argument.

Eg.1);insert into product values('4.0',200,0,'Garbage new movie'

In this query , first the parameter for the Form 1 is passed and in addition a update operation is launched via insert query.

Getting the sensitive values is implemented through providing a SELECT query as form argument

Eg2)

2.0'UNIONselectpassfromuserprofile where userid='sam

Here union of select query is passed a form argument, so that execution on the server will return the sensitive results

## 4.1 Attack Detection

Attack is detected by parsing the form parameters and looking for any side effect

queries. Patterns like;

insert Union select

Are defined in the code.

The parameters passed in Form are parsed and matched if any known patterns of SQL attack (defined above) is present in the form parameter. If found, it is detected as SQL attack.

## 4.2 Attack Prevention

When SQL injection attack pattern is found in the Form parameters, the processing is blocked and number of times of attack is counted. If a user launches attack more than threshold times, the account is blocked.

## CHAPTER-5

### TESTING

This section details the testing carried out on the project code. The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or 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 and does not fail in an unacceptable manner. There are various types of test. Each test type addresses specific testing requirement.

#### 5.1 Unit Testing

Since we have followed OOPS, each class is a unit, so unit test cases are written for each class and tested.

<b>Classes integrated</b>	<b>Tests done</b>	<b>Remarks</b>
Class: <b>Form Handler</b>	Class tested for handling input on Forms	Success
Class: <b>SQL_ID detector</b>	Class tested to check if SQL injection attack is detected	Success
Class: <b>Account Analysis</b>	Class tested to check whether abnormal behaviour on account is detected	Success

## 5.2 Validation testing

At the culmination of integration testing, software is completed and assembled as a package. Interfacing errors are uncovered and corrected. Validation testing can be defined in many ways. Here the testing validates the software function in a manner that is reasonably expected by the customer. Validation is a Quality assurance process of establishing evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements.

## 5.3 SYSTEMTESTING

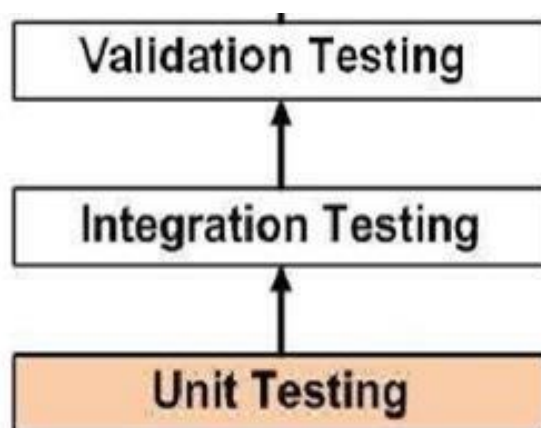
System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable system(s).

System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a system Requirement Specification (SRS).

System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s)



**Fig 5.1: testing process**

## **5.4 TESTING OF INITIALIZATION AND UI COMPONENTS**

### **Testcase for database connection setup**

Serial Number of TestCase	TC 01
Module Under Test	DATABASE Connection
Description	When the client program is executed, it tries to connect to DATABASE(SQLserver) using the data source and catalogue.
Output	If the connection details are correct, the DATABASE is connected. If the connection details are incorrect, an exception is thrown.



## CHAPTER-6

### INTERPRETATION OF RESULT

The following snapshots define the results or outputs that we will get after step by step execution of all the modules of the system.

**Fig 6.1: Opening screenshot of the project**

[SQL Injection](#) [Home](#)

## SQL Injection and Prevention



**SQL Injection**  
Username:  
  
Password:


**Inject using Example 1**  
Username:   
Password:

**Inject using Example 2**  
Username:   
Password:

**Fig 6.2: Register the user:**

[SQL Injection](#) [Home](#)

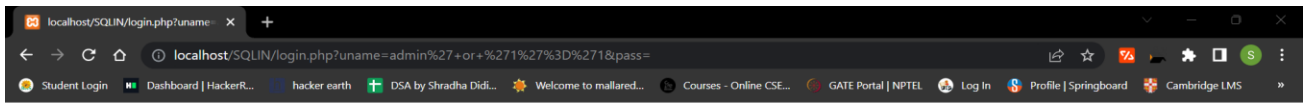
## SQL Injection and Prevention



**SQL Injection**  
Username:  
  
Password:

**Inject using Example 1**  
Username:   
Password:

**Inject using Example 2**  
Username:   
Password:

**Fig 6.3: Login with user:**

**Connection is successful**

**Fig 6.4:If it is not user:**

**Connection failed.**

Wrong user credentials

## CHAPTER-7

### CONCLUSION

#### CONCLUSION:

In this project we developed a SQL injection attack detection and prevention system and implement edit on an E-commerce web applications it which sells movies/DVDs. The proposed solution is based on SQLi Signature detection and Anomaly detection approaches. The patterns of SQLi attacks are stored in database. When a web form either through POST or GET request is received, it is first sent to SQLi signature detection module to match if the request parameters has any SQL query patterns and if any pattern found, the FORM request is not processed and user is redirected to failure response page.

The attack signatures are stored in encrypted way(Using AES) , so that it is not possible for any attacker to modify the attack signatures.

When there is no SQLi pattern, the user data is collected, that is the login details(username and password) and also the purchase details (the product user has purchased), the different source of IP addresses he is requesting to access the web page and based on any abnormal behaviour threshold, the user is denied access to web page after certain duration, after a certain number of times they has been detected that they are attacking.

So the proposed system provides a complete security for the website we hold ,as the database cannot be corrupted nor modified nor can they obtain the details or passwords of user.

## CHAPTER-9

### REFERENCES

- [1] Technical report,Wikipedia.[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection).
- [2] The essentials of filters. Online document,  
Oracle.<http://www.oracle.com/technetwork/PHP/filters-137243.html>.
- [3] File:sql-injection-detection-wp.pdf  
Website:<http://www.f5.com/pdf/whitepapers/sql-injection-detection-wp.pdf>Retrieved 2009-10-01
- [4] Servlet and jsp filters.Online document,Sun Microsystems and Prentice Hall.  
<http://www.moreservlets.com/>.
- [5] Sql injection are your web applications vulnerable? Online document,HP.<https://products.spidynamics.com/asclabs/sqlinjection.pdf>.
- [6] Writing secure web applications. Online documentation, Advosys Consulting Inc.,june 2002. <http://advosys.ca/papers/web/61-web-security.html>.
- [7] The critical need to secure the web in your company. Anosterman research white paper,Webroot,Inc.,Jan2010. <http://www.ostermanresearch.com/whitepapers/or0210a.pdf>.
- [8] Owaspssecurecoding practices quick reference guide.Online document,Open Web ApplicationSecurityProject,2010.  
<https://www.owasp.org/index.php/Category:OWASPCodeReviewProject>.
- [9] The ten most critical web application security risk.Online document,Open Web Application Security Project, 2010. [https://www.owasp.org/index.php/Category:OWASP\\_Top\\_Ten\\_Project](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project).
- [10] Acunetix web vulnerability scanner.Technical report, Acunetix,2011.  
<http://www.acunetix.com/vulnerability-scanner/>.