**1 Introduction**

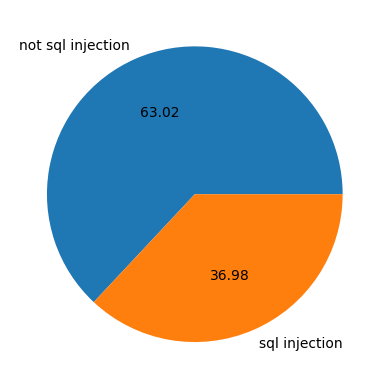
This project focuses on detecting SQL Injection attacks using various machine learning and deep learning models. We implemented five models: **SVM**,**Logistic Regression**, **MLP**, **RNN**, and **LSTM**. These models were trained on a labeled dataset to distinguish between malicious and normal SQL queries.

The goal is to compare their performance and establish a baseline before evaluating a more advanced model based on **BERT**. This report summarizes the implementation and results of the six models for validation.

2 **Dataset Description**

This study uses two separate datasets for training and testing purposes:

* **Training Dataset (SQLIV3\_cleaned2.csv)**:
  + Contains **30,614 SQL queries**.
  + Each entry is composed of a Sentence (the SQL query) and a Label (0 for benign, 1 for SQL Injection).
  + Label distribution:
    - **Normal (Label = 0)**: 19,268 queries
    - **Malicious (Label = 1)**: 11,346 queries



* **Testing Dataset (sqliv2\_utf8.csv)**:
  + Contains **33,760 SQL queries**.
  + Structured in the same format with Sentence and Label columns.
  + Label distribution:
    - **Normal (Label = 0)**: 22,305 queries
    - **Malicious (Label = 1)**: 11,455 queries

**(SQLIV3\_cleaned2.csv)**

Before training, both datasets were cleaned by removing duplicate queries. These queries were then vectorized using appropriate text processing techniques such as **TF-IDF** or **token embeddings**, depending on the model.

**3.1 Support Vector Machine (SVM)**

We implemented a Support Vector Machine (SVM) classifier using a **linear kernel** with a regularization parameter **C=1**. Prior to training, the SQL queries were transformed into numerical features using **TF-IDF vectorization**. This allowed the model to capture the importance of individual tokens in detecting malicious patterns.

The model was trained on the cleaned training dataset and evaluated using the separate test dataset.

**Results**:

* Accuracy: **98.78%**

The SVM model achieved excellent performance, showing a very high precision, which indicates its strong ability to correctly identify SQL Injection attempts while minimizing false positives. These results establish a solid traditional machine learning baseline for comparison with deep learning and transformer-based approaches.

**3.2 Regression logestic**

We implemented a logistic regression model using the scikit-learn library to detect SQL injection attacks from a labeled dataset of text samples. The dataset was first cleaned by removing duplicates and unnecessary characters, then transformed into numerical vectors using the TF-IDF technique, which captures the importance of words across the dataset. The logistic regression algorithm was trained to classify inputs as either SQL injection or benign queries, and we optimized the model by tuning parameters and evaluating performance on both training and validation sets.

After preprocessing, the dataset contained 11,424 entries labeled as SQL injections (label = 1). On the training set, the logistic regression model achieved an overall accuracy of 98.12% and a precision of 99.73%, reflecting a strong generalization capacity and a remarkably low false positive rate. These results highlight the robustness of the model in text classification tasks related to cybersecurity. When evaluated on the test set, it successfully detected 10,892 malicious queries, resulting in a detection accuracy of 95.34%, confirming its reliability as a baseline method for identifying SQL injection attacks using natural language processing techniques.

**3.3Multilayer Perceptron (MLP)**

To evaluate the effectiveness of neural networks in detecting SQL injection attacks, two Multilayer Perceptron (MLP) architectures were implemented using the Keras Sequential API. The goal was to assess how hidden layers affect the classification performance.

This model consists of three hidden layers with decreasing neuron sizes and ReLU activations, followed by a sigmoid output layer for binary classification.

**Architecture**:

* **Input dimension**: 3000 (TF-IDF vector size)
* Dense(512), activation = ReLU
* Dense(256), activation = ReLU
* Dense(128), activation = ReLU
* Dense(1), activation = Sigmoid

**Training Configuration**:

* Optimizer: SGD (learning rate = 0.01)
* Loss function: Binary Crossentropy
* Epochs: 27
* Batch size: 32
* Validation split: 20%

**Results**:

* **Final training loss**: 0.0304
* **Final training accuracy**: 99.40%
* **Test accuracy (on SQLi samples)**: 98.78%
* **Number of SQL injection queries detected**: 11,314 out of 11,455

**3.4 simple Neural Network**

A simpler baseline model was built with only an input and output layer (i.e., no hidden layers). The objective was to serve as a reference point for measuring the impact of adding hidden layers.

**Architecture**:

* Dense(1), input\_dim = 3000, activation = Sigmoid

**Training Configuration**:

* Optimizer: SGD (learning rate = 0.01)
* Loss function: Binary Crossentropy
* Epochs: 27
* Batch size: 32
* Validation split: 20%

**Results**:

* **Final training loss**: 0.3927
* **Final training accuracy**: 92.00%
* **Test accuracy (on SQLi samples)**: 83.10%
* **Number of SQL injection queries detected**: 9,518 out of 11,455

**3.5 RNN**

We implemented a deep learning model using a SimpleRNN architecture to detect SQL injection attacks. The dataset was cleaned by removing duplicates and minimally preprocessing the text to preserve SQL-specific characters. Sentences were tokenized without filtering out symbols, and sequences were padded based on the 95th percentile of sentence length. The model was built using embedding and two stacked SimpleRNN layers, followed by dense layers to enhance learning capacity.

The model achieved a high test accuracy of 99.33%, demonstrating strong performance in identifying malicious SQL queries. Specifically, it successfully detected 11,353 out of 11,424 SQL injection samples in the test set, resulting in a detection rate of **99.43%.** These results highlight the model's effectiveness in recognizing patterns specific to SQL injection attacks.

**3.6 LSTM**

We implemented an LSTM-based deep learning model to detect SQL injection attacks. The dataset was cleaned by removing duplicates and applying minimal text preprocessing. After tokenization and padding, the model was built with embedding layers, LSTM layers, and dropout to prevent overfitting. It was compiled using the Adam optimizer and binary cross-entropy loss.

The model achieved a test accuracy of 99.06%, with a **99.51%** detection rate for SQL injection attacks, correctly identifying 11,362 out of 11,424 spam queries. This highlights its effectiveness in detecting and preventing SQL injection vulnerabilitie

