**1 Introduction**

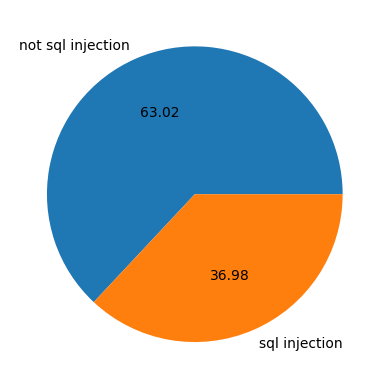
This project focuses on detecting SQL Injection attacks using various machine learning and deep learning models. We implemented five models: **SVM**,**Logistic Regression**, **MLP**, **RNN**, and **LSTM**. These models were trained on a labeled dataset to distinguish between malicious and normal SQL 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: **99.45%**
* Precision: **99.73%**

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). The model correctly detected 10,892 of these cases, achieving a detection accuracy of **95.34%** on the test set. On the training set, it reached an overall accuracy of **98.12%** and a precision of **99.73%**, indicating strong generalization and a very low false positive rate. These results demonstrate the effectiveness of logistic regression in text classification for cybersecurity tasks, making it a reliable baseline model for detecting SQL injection attacks through natural language processing.

**3.3Multilayer Perceptron (MLP)**

A Multilayer Perceptron (MLP) model was implemented using the Keras Sequential API. The architecture consists of three hidden layers with ReLU activation functions, and a final output layer with a sigmoid activation function for binary classification. Dropout layers were included after each hidden layer to reduce overfitting.

**Model architecture**:

* Input size: 3000 (TF-IDF features)
* Hidden layers:
  + Dense(512) → ReLU → Dropout(0.5)
  + Dense(256) → ReLU → Dropout(0.5)
  + Dense(128) → ReLU → Dropout(0.5)
* Output layer: Dense(1) → Sigmoid
* Optimizer: Stochastic Gradient Descent (SGD)
* Learning rate: 0.01
* Loss function: Binary Crossentropy
* Epochs: 27
* Batch size: 32

**Training results**:

* Final Training Loss: **0.0373**
* Final Training Accuracy: **99.23%**

After training, the model was evaluated on the separate test dataset. It correctly detected **11,307** out of **11,453** malicious queries.

**Test Accuracy**: **98.73%**

**3.4 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 **99.38%.** These results highlight the model's effectiveness in recognizing patterns specific to SQL injection attacks.

**3.5 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 **99.46%** 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 vulnerabilities.