SQLi & XSS Threat Detector

Problem Statement : AI/ML For Networking

Modern networks face increasing challenges in monitoring and securing traffic due to the exponential growth of data, encrypted communication, and sophisticated cyber threats. Traditional rule-based security measures and deep packet inspection (DPI) techniques are becoming less effective in detecting and classifying threats, especially in encrypted traffic. Manual intervention in network traffic classification is inefficient, leading to delayed threat detection and security vulnerabilities. To address these issues, AI-driven solutions can analyze traffic patterns, detect anomalies, classify applications, and enhance security in real-time, ensuring adaptive and intelligent network defense.

Our Solution :

To identify SQL Injection (SQLi) and Cross-Site Scripting (XSS) attacks, we suggest an AI-based traffic analysis system. It analyses real-time HTTP requests and URL patterns using two machine learning models that have been independently trained. The system classifies possible threats, automatically extracts features from traffic data, and logs attacks with pertinent information.

To help users comprehend threats that have been identified and receive prompt responses to enquiries about security, an intelligent chatbot has been integrated. There is no manual labour involved in any part of the process, from data analysis to threat classification and logging.

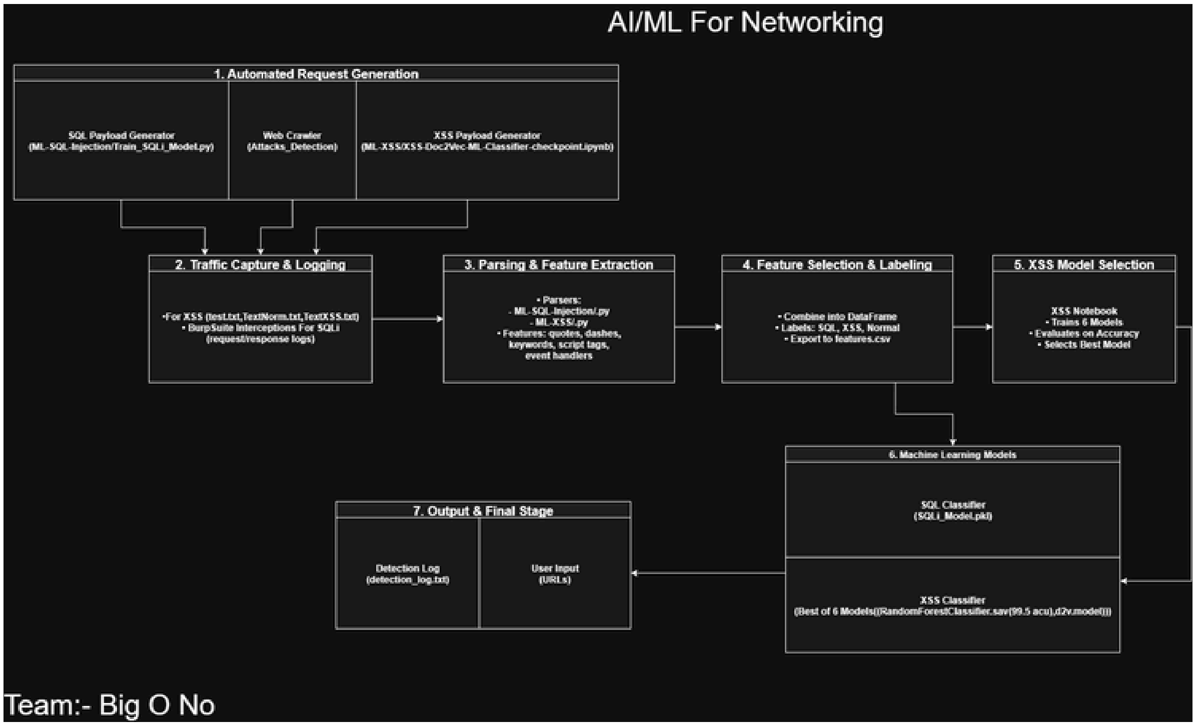
For contemporary network security requirements, this hybrid solution is scalable and effective since it provides precise, flexible, and real-time defence against frequent online threats.

Team Name : Big O No

Team Members :

- Rohith Reddy Y
- Manish Siva
- Vishnupriya T

Architecture Diagram :



Drive Link

How the code works:

1. User Input :
   - The user enters a URL or payload.
1. Preprocessing :
   - The input is cleaned and converted into a format suitable for the models.
1. Threat Detection:
   - The input is analyzed by two trained models one for SQLi and one for XSS.
   - Each model checks if the input matches known attack patterns.
4. Result Output :
   - The system shows whether the URL is vulnerable.
   - It specifies if the threat is SQLi, XSS, or safe.
5. AI Chatbot Support:
   - An integrated chatbot to understand the results and ask security-related questions.

File Structure:

```
AI-ML_FOR_NETWORKING/
├──── Attacks_Detection/
│     ├──── Data/
│     │     ├──── Demo1_log_parser_good_and_bad.csv
│     │     ├──── Demo2_log_parser_good_and_bad.csv
│     │     └──── Good_and_Bad_requests.csv
│     ├──── Models/
│     │     ├──── d2v.model
│     │     ├──── RandomForestClassifier.sav
│     │     └──── SQLi_Model.pkl
│     ├──── Templates/
│     │     └──── index.html
│     ├──── .env
│     ├──── app.py
│     └──── detection_log.txt
├──── ML-SQL-Injection/
│     ├──── Log/
│     │     ├──── bad_requests.log
│     │     └──── good_requests.log
│     ├──── Good_and_Bad_requests_1.csv
│     ├──── Good_and_Bad_requests.py
│     └──── Train_SQLi_Model.py
├──── ML-XSS/
│     ├──── .ipynb_checkpoints/
│     │     └──── XSS-Doc2Vec-ML-Classifier-checkpoint.ipynb
│     ├──── lib/
│     │     ├──── DecisionTreeClassifier.sav
│     │     ├──── GaussianNB.sav
│     │     ├──── KNeighborsClassifier.sav
│     │     ├──── MLPClassifier.sav
│     │     ├──── SVC.sav
│     │     ├──── test.txt
│     │     ├──── testNORM.txt
│     │     └──── testXSS.txt
│     └──── ML-XSS_Model_Testing.py  < --  Corrected location
├──── venv/
├──── .gitignore
├──── LICENSE
├──── README.md
└──── requirements.txt
```

The Models and Datasets:

We trained two separate supervised learning models using labeled datasets:

- SQLi Detection Model: Trained on a dataset containing both malicious and benign SQL queries.
- XSS Detection Model: Trained on a dataset with various XSS payloads and clean text.

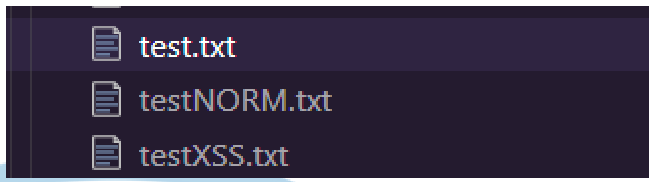
Datasets Source: Open-source security datasets and custom inputs from simulated attacks.
Preprocessing Techniques: Text normalization, tokenization, and TF-IDF vectorization.
Models Used: We experimented with Logistic Regression, SVM, and Random Forest classifiers.

XSS

Models

```
Training Classifier #1 DecisionTreeClassifier
Training Classifier #2 SVC
Training Classifier #3 GaussianNB
Training Classifier #4 KNeighborsClassifier
Training Classifier #5 RandomForestClassifier
Training Classifier #6 MLPClassifier
```

Datasets



Accuracy Score of Trained Models:

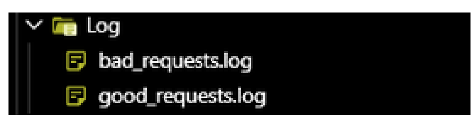
```
Training Classifier #1 DecisionTreeClassifier
Training Classifier #2 SVC
Training Classifier #3 GaussianNB
Training Classifier #4 KNeighborsClassifier
Training Classifier #5 RandomForestClassifier
Training Classifier #6 MLPClassifier
```

```
Acccuracy Score #1: 98.5%
Acccuracy Score #2: 98.2%
Acccuracy Score #3: 94.8%
Acccuracy Score #4: 96.0%
Acccuracy Score #5: 99.5%
Acccuracy Score #6: 99.5%
```

SQLi

Models

Datasets



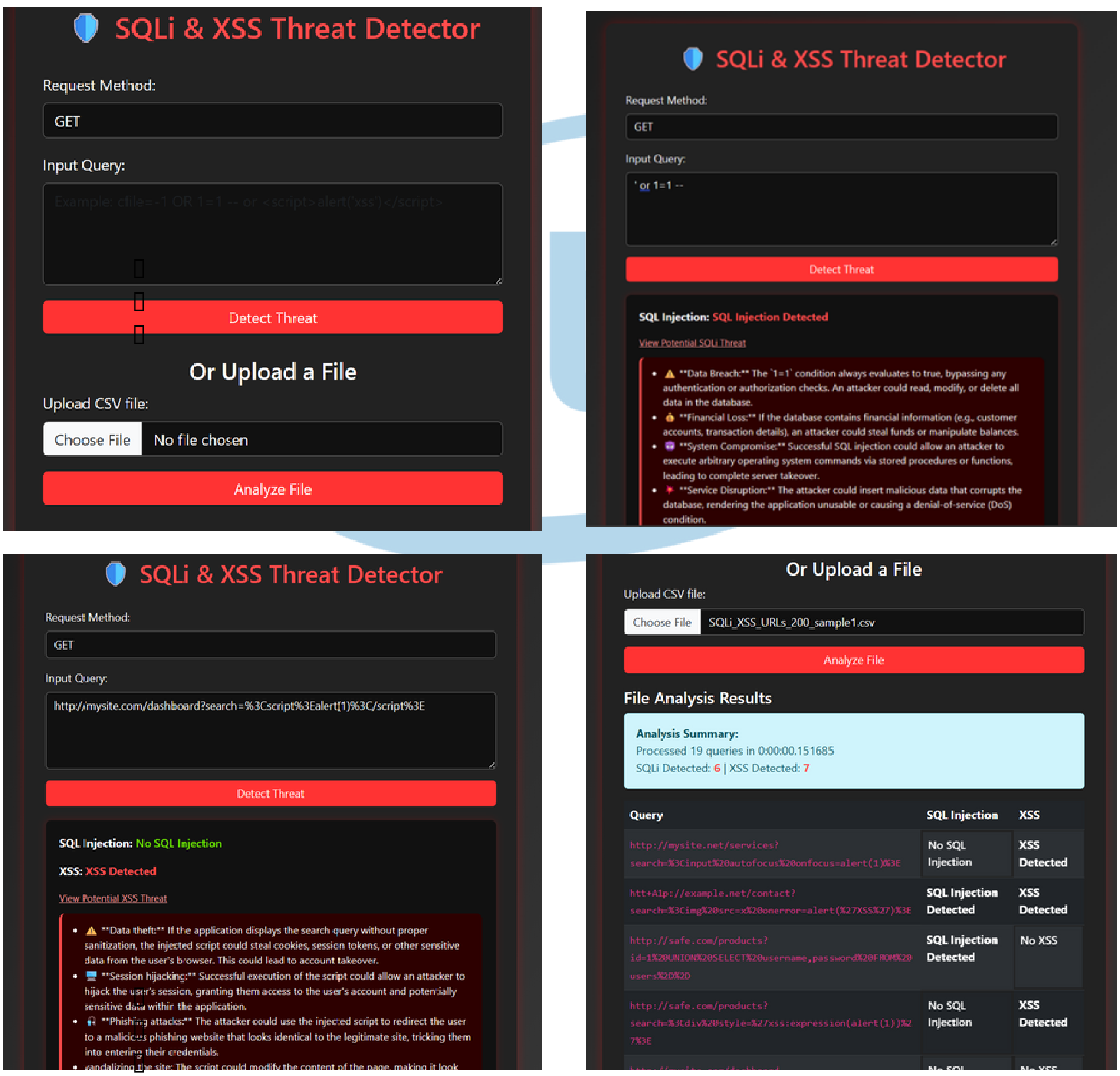
To identify the most effective classifier for detecting XSS attacks, we experimented with six different machine learning models:

- Decision Tree Classifier
- Support Vector Classifier (SVC)
- Gaussian Naive Bayes (GaussianNB)
- K-Nearest Neighbors (KNeighborsClassifier)
- Random Forest Classifier
- Multi-Layer Perceptron Classifier (MLPClassifier)

Each model was trained on our processed datasets and evaluated based on its accuracy score. As shown in the results, the Random Forest Classifier and MLPClassifier achieved the highest accuracy of 99.5%, making them the most suitable models for our use case. The SVC and Decision Tree classifiers also performed well with accuracy scores above 98%. These results highlight the reliability of AI-driven models in detecting malicious input patterns with high precision.

To build a reliable detection system for SQL Injection (SQLi) attacks, we collected HTTP request logs from two sources: good_requests.log containing normal traffic and bad_requests.log containing malicious SQLi payloads. These files were processed using a custom Python script (Good_and_Bad_requests.py) that extracted relevant features and converted the data into a structured CSV file (Good_and_Bad_requests_1.csv). This dataset was then used to train a machine learning model using the Train_SQLi_Model.py script. The trained model was saved as SQLi_Model.pkl and is now used to classify and detect SQLi threats in incoming HTTP requests. This pipeline enables the system to automatically recognize injection patterns based on previously learned behavior, providing accurate and real-time SQLi detection.

Project Output:



Links:

Github Repository

Project Video

Project Architecture Diagram

Conclusion:

In this project, we developed an AI-powered system to detect SQL Injection (SQLi) and Cross-Site Scripting (XSS) attacks in real-time network traffic. By combining automated data preprocessing, supervised learning models, and a user-friendly interface, the system effectively identifies and classifies malicious inputs with high accuracy.
For SQLi detection, we processed HTTP request logs into structured datasets and trained a dedicated model (SQLi_Model.pkl) capable of recognizing harmful patterns based on past data.
For XSS detection, we experimented with six different machine learning models and found that the Random Forest Classifier and MLPClassifier delivered the best results, achieving an accuracy of 99.5%.
All detected threats are automatically logged into the detection_log.txt file, ensuring that every malicious request is recorded for further analysis and auditing. Additionally, the integration of an intelligent chatbot provides real-time threat explanations and enhances user interaction without manual intervention.
Overall, our AI-driven solution demonstrates a scalable and adaptive approach to modern network security, offering precise detection, automated analysis, and real-time defense against common web-based attacks.