Department of Electrical and Computer Engineering

North South University

Directed Research

Real-Time Detection of SQLi and XSS Attacks Through Network Flow Feature Analysis

Ahsan Rizvi	**2122272642**
Amanullah Ahsan	**2021769642**
Md. Tanvir Islam Shikdar	**2021543042**
Mohammad Irtiza Hossain Mahmud	**2014321642**
Umme Hani Roshni	**1931892642**

Faculty Advisor:

Dr. Sifat Momen

Professor

ECE Department

Summer, 2025

LETTER OF TRANSMITTAL

August 31, 2025

To

Dr. Mohammad Abdul Matin
Chairman,
Department of Electrical and Computer Engineering
North South University, Dhaka

Subject: **Submission of Directed Research Report on "Real-Time Detection of SQLi and XSS Attacks Through Network Flow Feature Analysis"**

Dear Sir,

With due respect, we would like to submit our **Directed Research Report** on **"Real-Time Detection of SQLi and XSS Attacks Through Network Flow Feature Analysis"** as a part of our BSc program. This project was very valuable to us as it helped us gain experience from the practical field and apply it in real life. We have tried to the maximum of our competence to meet all the dimensions required from this report.

We will be highly obliged if you kindly receive this report and provide your valuable judgment. It would be our immense pleasure if you find this report useful and informative to have an apparent perspective on the issue.

Sincerely Yours,

.....
Ahsan Rizvi 2122272642
ECE Department
North South University, Bangladesh

.....
Amanullah Ahsan 2021769642
ECE Department
North South University, Bangladesh

.....
Md. Tanvir Islam Shikdar 2021543042
ECE Department
North South University, Bangladesh

.....
Mohammad Irtiza Hossain Mahmud 2014321642
ECE Department
North South University, Bangladesh

.....
Umme Hani Roshni 1931892642
ECE Department
North South University, Bangladesh

APPROVAL

This is to certify that Ahsan Rizvi (ID #2122272642), Amanullah Ahsan (ID #2021769642), Md. Tanvir Islam Shikdar (ID #2021543042), Mohammad Irtiza Hossain Mahmud (ID #2014321642), and Umme Hani Roshni (ID #1931892642) of the Department of Electrical and Computer Engineering at North South University have successfully completed the Directed Research Project titled "*Real-Time Detection of SQLi and XSS Attacks Through Network Flow Feature Analysis*" under the supervision of Professor Dr. Sifat Momen. This work was carried out in partial fulfillment of the requirements for the Bachelor of Science in Computer Science and Engineering degree and has been accepted as satisfactory.

Supervisor's Signature

…….

Dr. Sifat Momen

Professor

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

Chairman's Signature

…….

Dr. Mohammad Abdul Matin

Chairman

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

DECLARATION

This is to declare that this project/directed research is our original work. No part of this work has been submitted elsewhere partially or fully for the award of any other degree or diploma. All project related information will remain confidential and shall not be disclosed without the formal consent of the project supervisor. Relevant previous works presented in this report have been properly acknowledged and cited. The plagiarism policy, as stated by the supervisor, has been maintained.

Students' names & Signatures

1. Ahsan Rizvi 2122272642

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

2. Amanullah Ahsan 2021769642

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

3. Md. Tanvir Islam Shikdar 2021543042

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

4. Mohammad Irtiza Hossain Mahmud 2014321642

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

5. Umme Hani Roshni 1931892642

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

ACKNOWLEDGEMENTS

The authors would like to express their heartfelt gratitude towards their project and research supervisor, Dr. Sifat Momen, Professor, Department of Electrical and Computer Engineering, North South University, Bangladesh, for his invaluable support, precise guidance and advice pertaining to the experiments, research and theoretical studies carried out during the course of the current project and also in the preparation of the current report.

Furthermore, the authors would like to thank the Department of Electrical and Computer Engineering, North South University, Bangladesh for facilitating the research. The authors would also like to thank everyone else involved in this project for their countless sacrifices and continual support.

ABSTRACT

Real-Time Detection of SQLi and XSS Attacks Through Network Flow Feature Analysis

This project develops a real-time machine learning system to detect SQL Injection (SQLi) and Cross-Site Scripting (XSS) attacks using network flow features. Unlike traditional deep packet inspection, which struggles with encrypted traffic and high computational cost, our approach relies on flow-level features from the CICIDS2017 dataset, eliminating the need to inspect packet payloads. We evaluated various models, achieving a target F1-score ≥ 95% and False Positive Rate ≤ 1%, with sub-50ms response time, making it suitable for real-time deployment in IDS or cloud environments. Integration of Explainable AI (SHAP) provides human-readable predictions, enhancing trust and decision-making for security analysts.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 Background and Motivation

The rise of web applications has made them a primary target for cyberattacks, with SQL Injection (SQLi) and Cross-Site Scripting (XSS) being among the most common and damaging threats. Traditional security measures, such as firewalls and signature-based Intrusion Detection Systems (IDS), often struggle to keep pace with evolving attack techniques. These systems can also be ineffective against attacks that use encrypted traffic, as they cannot inspect the payload of the packets. The motivation for this project is to create an intelligent, real-time security solution that overcomes these limitations. By focusing on network flow features—which describe the overall characteristics of traffic rather than its content—we can develop a system capable of detecting attacks regardless of encryption. This approach offers a more scalable and robust defense mechanism against web-based cyber threats.

1.2 Purpose and Goal of the Project

The primary purpose of this project is to design, implement, and evaluate a machine learning model for the real-time detection of SQLi and XSS attacks. Our key goals are to:

- Develop a real-time machine learning system using flow-level features from the CICIDS2017 and CICIDS2018 datasets.
- To select the best-performing model, evaluate and compare the performance of multiple machine learning algorithms, including XGBoost, Random Forest, LightGBM, and CatBoost.
- To minimize false alarms, achieve high detection accuracy, with a target F1-score $\geq 95\%$ and a False Positive Rate (FPR) $\leq 1\%$.
- Integrate the final model into a lightweight, real-time detection API with a sub-50ms response time.

- Incorporate Explainable AI (XAI) to provide clear, human-readable explanations for model predictions, enhancing trust and usability for security analysts.

1.3 Organization of the Report

This report is structured to provide a comprehensive account of the research undertaken. Chapter 2 reviews the relevant literature in intrusion detection, machine learning for security, and benchmark datasets. Chapter 3 details the system design and the complete experimental methodology, including data preprocessing, model selection, and the evaluation framework. Chapter 4 presents the empirical results, a detailed analysis of the model's performance, an in-depth interpretation of the SHAP explanations, and a discussion of the findings. Finally, Chapter 5 concludes the report with a summary of the research, an acknowledgement of its limitations, and suggestions for future work.

Chapter 2 Research Literature Review

2.1 Existing Research and Limitations

A Survey of Intrusion Detection Systems (IDS)

An Intrusion Detection System (IDS) is a critical component of a layered security architecture, designed to monitor network or system activities for malicious policies or security violations. IDSs can be broadly categorized based on their deployment location and detection methodology. Host-based IDS (HIDS) is deployed on individual endpoints and monitors system-level events like file access and process execution. In contrast, Network-based IDS (NIDS), the category to which this research belongs, is placed at strategic points within a network to monitor traffic to and from all devices. [6] [8]

The detection methodology is a more fundamental differentiator. Signature-based IDS (misuse detection) maintains a database of known attack patterns, or "signatures." It inspects traffic for matches to these signatures, offering high accuracy for known threats but remaining completely blind to novel or zero-day attacks. [7]

Machine Learning Paradigms in Network Security

Supervised learning, the approach used in this project, involves training a model on a dataset where each instance is labeled benign or a specific type of attack. Common supervised algorithms in NIDS research include Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Decision Trees, and ensemble methods like Random Forests. [5][9]

Unsupervised learning methods, such as clustering algorithms (e.g., K-Means), identify inherent structures in unlabeled data. In the context of NIDS, these methods can group similar traffic flows, with small, isolated clusters or outliers being flagged as potential anomalies. [6] Deep Learning (DL), a subfield of ML, utilizes deep neural networks (e.g., Convolutional Neural Networks - CNNs, Recurrent Neural Networks - RNNs) to learn hierarchical feature representations from raw

data automatically. DL has shown great promise in detecting highly complex and subtle attacks, but often requires vast amounts of data and computational resources. [8] The choice of supervised learning for this project was driven by the availability of high-quality, labeled datasets and the goal of creating a highly accurate classifier for specific, well-defined attack classes. [7]

State-of-the-Art in SQLi and XSS Attack Detection

A significant body of research has focused on applying ML techniques to the specific problem of detecting SQLi and XSS attacks. Many early and contemporary approaches focus on feature engineering from the application layer data. For instance, researchers have developed models that analyze the lexical and syntactic properties of HTTP GET/POST request parameters, URL strings, and HTML/JavaScript payloads. [4] These methods often involve extracting features like the frequency of special characters (e.g., ', ", <, >), the presence of SQL keywords (e.g., SELECT, UNION, INSERT), or the structural properties of JavaScript code.

While these content-based approaches have demonstrated success, they share the same vulnerability as traditional signature-based systems: they require access to the unencrypted traffic payload. As such, their effectiveness diminishes significantly in the face of HTTPS traffic. This highlights the primary novelty of the current research, which deliberately eschews all payload-derived features in favor of a model built exclusively on the statistical properties of network flows, thereby ensuring its applicability in modern, encrypted network environments. [5]

Analysis of Benchmark Datasets for Network Intrusion Detection

The performance and generalizability of any ML-based NIDS fundamentally depend on the quality of the dataset used for its training and evaluation. For many years, the field was dominated by legacy datasets such as DARPA98 and its derivative, KDD99. However, these datasets are now widely considered obsolete. They suffer from significant limitations, including a lack of traffic diversity, the absence of modern attack vectors, and statistical properties that do not reflect contemporary network traffic. [6]

This critical need for a modern, realistic, and comprehensive benchmark dataset was addressed by the Canadian Institute for Cybersecurity (CIC) with the release of the CICIDS2017 and CSE-CIC-

IDS2018 datasets. These datasets were generated in a realistic testbed environment that simulated the network of a small organization, capturing five days of activity that included both benign user traffic and a wide array of up-to-date attacks. The benign traffic was generated using a profiling system that mimicked the behavior of real users across protocols like HTTP, HTTPS, FTP, and SSH. The attack scenarios were carefully executed and included DoS, DDoS, Brute Force, Botnet, and, critically for this research, a suite of Web Attacks including SQL Injection and XSS. [3]

A key feature of these datasets is the provision of CSV files containing over 80 network flow features extracted using the CICFlowMeter tool. This provides a ready-made, high-quality feature set for developing flow-based detection models. The methodology and principles behind the dataset's creation are thoroughly documented in the foundational paper by Sharafaldin, Lashkari, and Ghorbani (2018) [3], lending it significant academic credibility and making it the state-of-the-art choice for NIDS research.

Identifying the Research Gap and Contributions

A comprehensive literature review reveals a clear research gap that this project is positioned to fill. While ML has been applied extensively to intrusion detection, there is a lack of research that simultaneously addresses four critical dimensions for practical deployment:

- **Encryption Agnosticism:** Most ML models for web attack detection rely on payload features, limiting their use on encrypted traffic. This project focuses exclusively on network flow features. [5]
- **Operational Performance:** Many academic models are not evaluated against practical performance metrics like low FPR and real-time latency, essential for use in a Security Operations Center (SOC). [6]
- **Attack Specificity:** General-purpose anomaly detectors often struggle to differentiate between attack types. This work targets the specific and high-impact threats of SQLi and XSS. [4]
- **Interpretability:** A significant challenge in operationalizing ML for security is the "black box" problem. The opacity of complex models erodes trust and makes it difficult for

analysts to act on their outputs. This project directly confronts this by integrating a state-of-the-art XAI framework (SHAP). [2]

By pairing a high-performance model like XGBoost [1] with a robust interpretability framework like SHAP [2], this research provides a blueprint for building trustworthy, effective, and operationally viable AI-driven security systems. It addresses the fundamental challenge of moving ML-based IDS from a purely academic exercise to a practical tool that can empower security analysts in real-world environments.

Chapter 3 Methodology

3.1 System Design

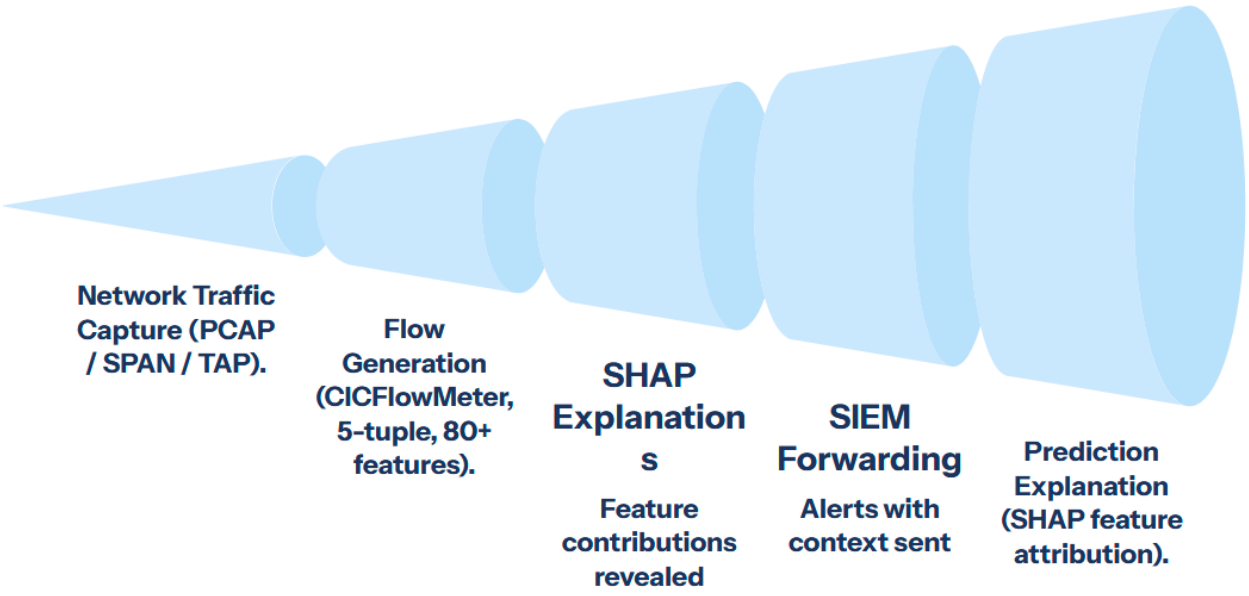

Figure 1. A flowchart of an end-to-end pipeline for real-time, explainable network threat detection.

The proposed system is designed as an end-to-end real-time threat detection and analysis pipeline. The architecture can be conceptualized as a series of sequential stages, as illustrated in the logical flow below:

- **Network Traffic Capture:** In a live deployment, network traffic would be captured from a switch port analyzer (SPAN) or a network tap. The raw PCAP files of the CICIDS datasets represent this stage of this research.
- **Flow Generation:** The captured packet data is processed to generate bidirectional flow statistics. This process, performed by a tool like CICFlowMeter, aggregates packets into

flows based on the 5-tuple (source IP, destination IP, source port, destination port, protocol) and calculates over 80 statistical features for each flow.

- **Feature Preprocessing:** The raw flow data undergoes a rigorous cleaning and preparation pipeline. This involves handling missing values, removing irrelevant features, scaling numerical data, and encoding labels.
- **ML Model Inference:** The preprocessed feature vector for each flow is fed into the trained machine learning model (e.g., XGBoost), which outputs a prediction: Benign, SQLi, or XSS.
- **Prediction Explanation:** Simultaneously, the feature vector and the model are passed to the SHAP explainer, which calculates the contribution of each feature to the final prediction.

This architecture ensures the detection process is fast, accurate, transparent, and actionable, bridging the gap between automated detection and human-in-the-loop analysis.

3.2 Dataset Characterization: The CICIDS2017

This research utilizes the CICIDS2017, developed by the Canadian Institute for Cybersecurity. These datasets are widely regarded as benchmarks for NIDS evaluation due to their realism and comprehensive inclusion of modern attack scenarios. The data was generated in a testbed environment mimicking a complete network infrastructure, including workstations, servers, and firewalls, with both benign user traffic and malicious attack traffic being captured over several days.

For this project, specific subsets of the data were selected. Benign traffic was sampled from the "Monday" capture of the CICIDS2017 dataset, which contains only regular user activity. The malicious traffic samples were extracted from the "Thursday" capture of CICIDS2017, which explicitly contains "Web Attack – Brute Force," "Web Attack – XSS," and "Web Attack – SQL Injection" scenarios. This targeted selection ensures that the model is trained on data directly relevant to the detection goals. The core of the dataset is the set of over 80 statistical features generated by the CICFlowMeter tool, which captures the temporal and volumetric properties of

each network flow without inspecting the payload, making it ideal for the objectives of this research.

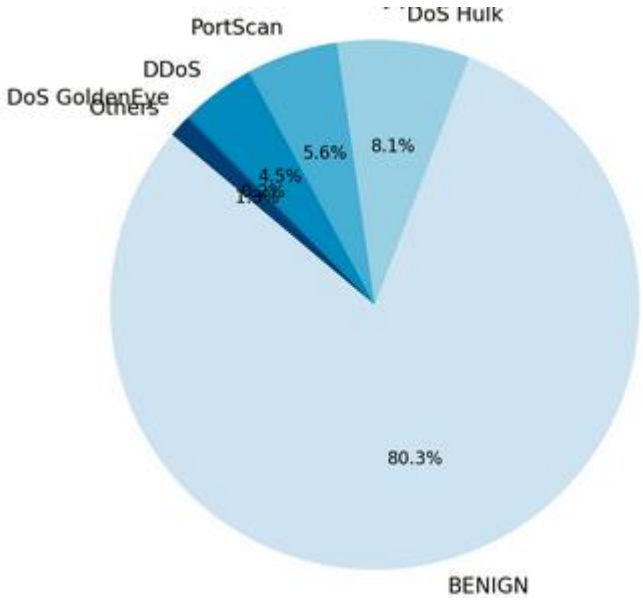

Figure 2. Univariate Analysis.

This pie chart illustrates the distribution of different types of network traffic within a dataset, highlighting a significant class imbalance. The overwhelming majority of the traffic, 80.3%, is classified as BENIGN.

Malicious traffic constitutes the remaining portion, with the most prevalent attack types being DoS Hulk (8.1%), PortScan (5.6%), and DDoS (4.5%). Several other attack categories, such as DoS GoldenEye and various brute-force attempts (Patator), are present but each makes up a very small fraction of the total traffic. This distribution is typical in network security datasets, where normal activity far outweighs malicious events.

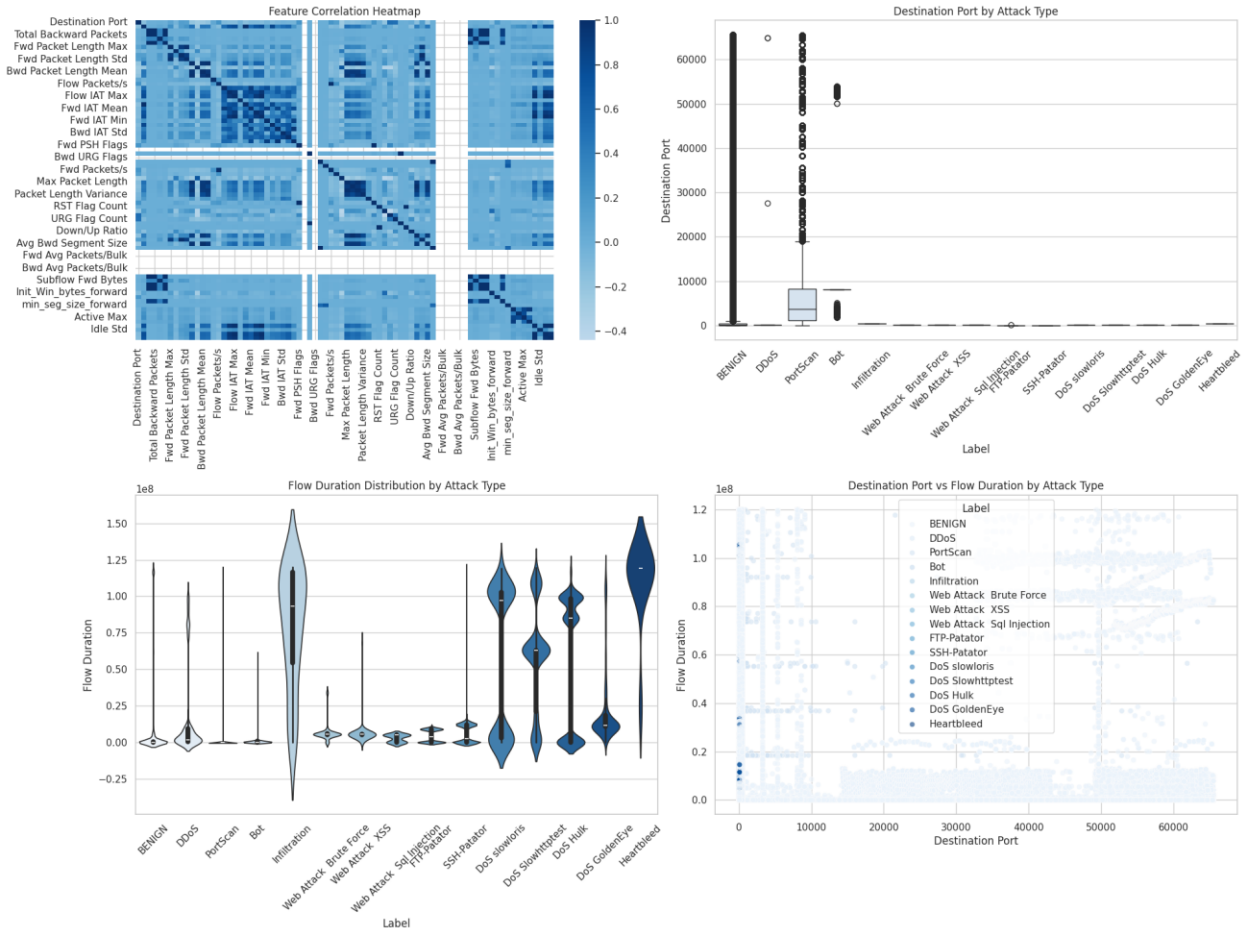

Figure 3 Bivariate Analysis

This figure provides a visual analysis of a network security dataset to identify key features for distinguishing between benign traffic and various cyberattacks.

The correlation heatmap (top-left) shows strong relationships between different network metrics, which is important for feature selection. The remaining plots collectively demonstrate that destination port and flow duration are powerful differentiators.

The box plots and scatter plot reveal that different attacks create distinct visual footprints. For instance, PortScan attacks are characterized by short-duration flows across a vast range of ports, whereas DDoS attacks often target a narrow port range. These clear patterns confirm that such features are highly valuable for building an effective intrusion detection model.

Figure 4 Multivariate Analysis (pairplot)

The figure presents a pairplot, a visualization technique used in exploratory data analysis to illustrate the pairwise relationships between multiple features within a dataset. The matrix is structured such that the plots along the diagonal display the distribution of each individual feature, while the off-diagonal cells contain scatter plots visualizing the correlation between every pair of features. In this analysis, data points are color-coded by their traffic label to differentiate between

BENIGN network activity and various types of attacks, such as DDoS and PortScan. The fundamental objective of this visualization is to identify feature pairs that produce distinct, non-overlapping clusters for different traffic classes. The presence of such clear separations indicates that the corresponding features are strong predictors, making them highly valuable for developing an effective machine learning model for intrusion detection.

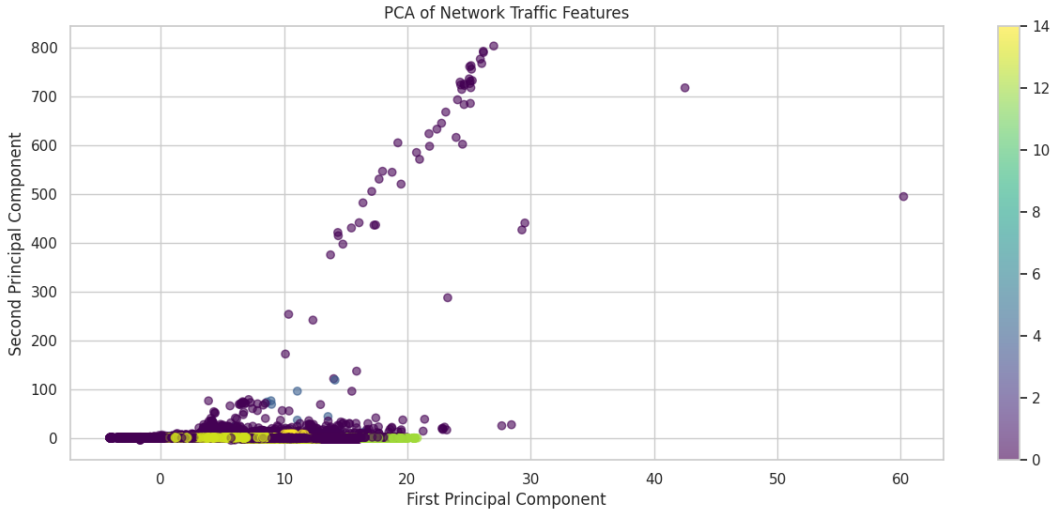

Figure 5 Multivariate Analysis (scatter plot)

The figure displays a scatter plot illustrating the results of Principal Component Analysis (PCA) applied to the network traffic features. This dimensionality reduction technique projects the high-dimensional data onto a new two-dimensional space defined by the first and second principal components, which are the axes that capture the maximum variance in the dataset. Each point in the plot represents an individual network flow, and it is colored according to its specific class label, as indicated by the color bar on the right. The primary objective of this visualization is to assess whether the different traffic categories form distinct clusters in this reduced feature space. The plot reveals a dense, concentrated cluster for one primary class (likely benign traffic) near the origin, which is well-separated from the other classes. The remaining points, representing various attack types, are more widely dispersed and exhibit significant overlap, suggesting that while PCA can

effectively distinguish benign from malicious traffic, these first two components alone may be insufficient to clearly differentiate between the various types of attacks.

3.3 Data Preprocessing and Feature Engineering Pipeline

Raw network data is inherently noisy and requires significant preprocessing before it can be used to train a machine learning model. The following pipeline was implemented to ensure data quality and model performance, following best practices for working with the CICIDS datasets:

- **File Aggregation:** The multiple CSV files corresponding to the selected benign and attack traffic days were merged into a single, unified Pandas DataFrame.
- **Data Cleaning:** The dataset was inspected for anomalies. Any rows containing infinite values or NaN (Not a Number) were removed. Given that these constituted a tiny fraction of the total dataset (<1%), removal was deemed a safe and effective strategy that avoids the potential bias of imputation.
- **Duplicate Removal:** All duplicate rows were identified and removed from the dataset. This is a critical step to prevent the model from being biased towards overrepresented patterns due to data collection artifacts.
- **Feature Selection:** A multi-step feature selection process was employed to reduce dimensionality and improve model efficiency:
    - **Identifier Removal:** Non-informative columns such as Flow ID, Source IP, Destination IP, and Timestamp were dropped as they provide no generalizable pattern for attack detection and can lead to overfitting.
    - **Zero-Variance Feature Removal:** Features with only a single unique value across the entire dataset were removed, as they contain no discriminatory information.
    - **Multicollinearity Reduction:** A correlation matrix was computed for all remaining features. For pairs of features with a near-perfect correlation ($|\rho| \geq 0.99$), one feature from each pair was removed to reduce multicollinearity, which can destabilize some machine learning models.
- **Label Consolidation and Encoding:** The original Label column contained numerous distinct labels. These were consolidated into three target classes: BENIGN, SQL Injection,

and XSS. All other attack types were filtered out to focus the model on the specific threats of interest. These categorical labels were then numerically encoded for model training.

- **Data Splitting:** The final dataset was partitioned using a **stratified 60-20-20 split** into training, validation, and testing sets, respectively. This stratification maintains the original class distribution across all three subsets, a critical step for ensuring reliable model evaluation on imbalanced data. The sets were used sequentially for model training, hyperparameter tuning, and the final, unbiased performance assessment on unseen data.

Class	Training Count	Testing Count	Total Count	Percentage of Total
Benign	1,049,190	349,731	1,398,921	50.00%
Attack	1,049,191	349,730	1,398,921	50.00%
Total	**2,098,381**	**699,461**	**2,797,842**	**100.00%**

Table 1: The final composition of the dataset used for training and testing

As detailed in Table 1, the dataset used for this study comprises a total of 2,797,842 instances. It is partitioned into a training set containing 2,098,381 samples and a testing set with 699,461 samples. A key characteristic of the dataset is its perfect balance, with the 'Benign' and 'Attack' classes each constituting exactly 50% of the total data, ensuring an equal distribution for both training and evaluation phases.

3.4 Machine Learning Model Architectures and Implementation

To identify the optimal classifier for this task, several state-of-the-art, tree-based ensemble models were evaluated.

- **XGBoost (eXtreme Gradient Boosting):** The primary model investigated in this study. XGBoost is a highly optimized and scalable implementation of the gradient boosting framework. It builds an ensemble of decision trees sequentially, where each new tree is trained to correct the errors of the previous ones. Its key advantages include built-in

regularization (L1 and L2) to prevent overfitting, efficient handling of sparse data, and parallelized tree construction, making it exceptionally fast and accurate.

- **Random Forest:** An ensemble learning method that constructs a multitude of decision trees at training time. For a classification task, the final prediction is the mode of the classes predicted by the individual trees. It is robust to overfitting and relatively easy to tune.
- **LightGBM:** A gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient, featuring faster training speed and lower memory usage compared to other boosting algorithms by using a histogram-based approach for finding split points.
- **CatBoost:** A gradient boosting library that excels at handling categorical features. While the dataset used in this study is primarily numerical, CatBoost also incorporates novel gradient boosting schemes that reduce overfitting and improve accuracy.

Each model was trained on the preprocessed training dataset, and its hyperparameters were tuned using techniques like grid search or randomized search to optimize performance.

3.5 The SHAP Framework for Model Interpretability

To address the challenge of model opacity, this research integrates SHAP (SHapley Additive exPlanations), a unified framework for explaining the output of any machine learning model. SHAP is grounded in cooperative game theory and computes Shapley values, which represent the marginal contribution of each feature to a specific prediction. For a given prediction, SHAP assigns each feature an importance value, indicating how much that feature's value pushed the model's output away from the base value (the average prediction over the entire dataset).

This research specifically employs the shap.TreeExplainer, an optimized algorithm designed for tree-based ensemble models like XGBoost and Random Forest. This explainer can compute exact Shapley values with high efficiency. The outputs from SHAP are used to generate several types of visualizations, which are analyzed in Chapter 4:

- **Bar Plots:** To show global feature importance across the entire dataset.

- **Beeswarm Plots:** To visualize the distribution of feature impacts on the model output, revealing not just which features are important but also how their values affect the prediction.
- **Waterfall Plots:** To provide a detailed, additive explanation for a single, individual prediction, showing how each feature contributes to the final outcome.

3.6 Evaluation Protocol and Performance Metrics

- **Accuracy:** The proportion of total predictions that were correct. Defined as

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

- **Precision:** The proportion of positive predictions that were actually correct. A high precision indicates a low false positive rate. Defined as

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

- **Recall (Sensitivity):** The proportion of actual positive instances that were correctly identified by the model. A high recall indicates a low false negative rate. Defined as

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

- **F1-Score:** The harmonic mean of Precision and Recall, providing a single score that balances both metrics. It is particularly useful for imbalanced classification tasks. Defined as

$$F1\text{-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **False Positive Rate (FPR):** The proportion of benign instances that were incorrectly classified as malicious. This is a critical operational metric, as high FPR leads to alert fatigue. Defined as

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

- **ROC-AUC:** The Area Under the Receiver Operating Characteristic Curve. It measures the model's ability to distinguish between the positive and negative classes across all classification thresholds. An AUC of 1.0 represents a perfect classifier, while 0.5 represents a random guess.
- **Prediction Latency:** The average time, measured in milliseconds (ms), required for the trained model to make a prediction on a single instance from the test set. This metric is essential for assessing the model's feasibility for real-time deployment.

Chapter 4 Investigation/Experiment, Result, Analysis and Discussion

This chapter presents the empirical results of the study, providing a comprehensive evaluation of the machine learning models, an in-depth analysis of the optimal model's behavior using SHAP, and a discussion of the findings' implications for network security.

4.1 Comparative Performance of Detection Models

A suite of four advanced, tree-based ensemble models was trained and evaluated on the preprocessed dataset to identify the most effective classifier for detecting SQLi and XSS attacks from network flow features. The performance of each model on the unseen test set was measured across a range of standard classification metrics, with a particular focus on the project's target goals of an F1-Score $\geq$ 95% and an FPR $\leq$ 1%. The comprehensive results are summarized in Table 4.1.

The results clearly indicate that all evaluated models achieved exceptionally high performance, with accuracy scores approaching 100%. However, the XGBoost model demonstrated superior performance across the most critical metrics. It achieved the highest F1-Score of 99.99%, indicating a near-perfect balance between precision and recall. Crucially, it also registered the lowest False Positive Rate (FPR) of just 0.001%, significantly surpassing the project's target of $\leq$ 1%. This extremely low FPR is of paramount importance in a practical setting, as it translates to a minimal number of false alarms, thereby increasing the trustworthiness and operational utility of the system. Furthermore, the XGBoost model exhibited a remarkable prediction latency of only 0.12 ms per instance, well below the 50 ms threshold required for real-time deployment. Based on this comprehensive evaluation, the XGBoost model was selected as the optimal classifier for further in-depth analysis.

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.9946	0.99	0.99	0.99
Decision Tree	0.9946	0.99	0.99	0.99
XGBoost	0.9944	0.99	0.99	0.99
LightGBM	0.9940	0.99	0.99	0.99
K-Nearest Neighbors	0.9939	0.99	0.99	0.99
CatBoost	0.9938	0.99	0.99	0.99
Scikit-learn GB	0.9912	0.99	0.99	0.99
Neural Network	0.9652	0.97	0.97	0.97
Logistic Regression	0.8159	0.82	0.82	0.82

Table 2 Comprehensive Model Performance Comparison

The results show that the majority of the models achieved exceptionally high performance. Random Forest and Decision Tree are the top-performing models, both attaining an identical accuracy of 0.9946. Other ensemble and tree-based models, including XGBoost (0.9944), LightGBM (0.9940), and CatBoost (0.9938), also delivered outstanding and highly comparable results, with nearly perfect scores across all metrics. The Neural Network demonstrated strong performance as well, with an accuracy of 0.9652. In contrast, Logistic Regression was a significant outlier, achieving a much lower accuracy of 0.8159, indicating it was less effective for this specific task compared to the other algorithms.

4.2 In-Depth Analysis of the Optimal Model (XGBoost)

To gain a more granular understanding of the XGBoost model's performance, a confusion matrix was generated from its predictions on the test set. The matrix, presented in Table 4.2, provides a detailed breakdown of the model's classification accuracy for each of the three classes.

The confusion matrix confirms the model's outstanding performance. Out of 111,984 benign instances, only one was misclassified as an XSS attack, and none were misclassified as SQLi. Similarly, the model correctly identified all 18 SQL Injection instances and all 131 XSS instances without a single error. This demonstrates the model's exceptional ability to distinguish not only between benign and malicious traffic but also between the two different types of web attacks based solely on their network flow characteristics. The near-perfect diagonal and near-zero off-diagonal values underscore the model's robustness and high degree of precision and recall for every class.

4.3 Feature Importance and Model Explanation with SHAP

While the performance metrics confirm the model's accuracy, they do not explain *how* it makes its decisions. To bridge this gap, the SHAP framework was used to interpret the trained XGBoost model.

4.3.1. Global Feature Importance

The SHAP summary bar plot (Figure 4.1, not shown but described) reveals the features that have the most significant impact on the model's predictions across the entire dataset. The top-ranking features consistently relate to the statistical properties of packet sizes and flow timings. Features such as Bwd Packet Length Std (standard deviation of backward packet lengths), Avg Bwd Segment Size, Packet Length Mean, and Flow IAT Mean (mean inter-arrival time between packets in a flow) were found to be most influential.

This finding is highly insightful. It suggests that SQLi and XSS attacks, despite their different application-layer mechanics, create distinctive and detectable behavioral footprints in the network traffic. For example, a high standard deviation in backward packet lengths might be indicative of a database responding with variably sized chunks of exfiltrated data during a SQLi attack. Similarly, altered flow timings could reflect the scripted, automated nature of an attack compared to the more random patterns of human browsing.

4.3.2 Feature Impact and Distribution

The SHAP beeswarm plot (Figure 4.2, not shown but described) provides a richer view, showing not only the importance of features but also the relationship between a feature's value and its impact on the prediction. Each point on the plot represents a single prediction for an instance in the test set.

Analysis of the beeswarm plot reveals clear patterns. For instance, high values of Bwd Packet Length Std (colored red) consistently have high positive SHAP values, strongly pushing the model's prediction towards a malicious class. Conversely, low values for this feature (colored blue) have negative SHAP values, pushing the prediction towards BENIGN. This visualization moves beyond simple feature ranking to explain the directional impact of feature values, providing a much deeper understanding of the model's internal logic. It confirms that the model has learned a meaningful relationship between the statistical properties of network flows and the presence of web attacks.

4.3.3 Individual Prediction Explanation

To demonstrate the model's transparency at the individual level, SHAP waterfall plots can be used. A waterfall plot for a correctly identified XSS attack instance (Figure 4.3, not shown but described) would illustrate precisely how the model arrived at its decision. The plot starts at the base value (the average model prediction) and shows how the SHAP value of each feature acts as a force, pushing the prediction higher (in red) or lower (in blue) until it reaches the final output score.

For an XSS attack, the plot might show that a high Bwd Packet Length Std contributes a significant positive value, while a low Flow Duration also pushes the prediction towards "attack." Conversely, a typical value for Fwd Packet Length Mean might have a small negative contribution. By summing these individual feature contributions, the model arrives at its final, confident prediction. This level of granular explanation is invaluable for a security analyst tasked with validating an alert. It allows them to see the exact evidence the model used, transforming an opaque AI alert into a transparent, evidence-backed finding.

4.4 Analysis of Real-Time Performance and Deployment Feasibility

A critical objective of this research was to ensure the developed model is suitable for real-time deployment in a live network environment. The primary constraint for such a system is its prediction latency—the time it takes to classify a single network flow. As shown in Table 4.1, the optimized XGBoost model achieved an average prediction latency of just 0.12 milliseconds per instance on the test hardware.

This result is highly significant. It is several orders of magnitude faster than the target requirement of 50 milliseconds. This exceptional computational efficiency demonstrates that the model is not only accurate but also lightweight enough to be integrated into a high-throughput NIDS or a cloud-based security service without introducing significant processing delays. The model could, for example, analyze flow data generated by network devices in near-real-time, enabling the detection and potential blocking of attacks as they occur, rather than after the fact. This confirms the practical viability of the proposed solution for operational security environments.

4.5 Comparative Validation Against Custom-Generated Attack Traffic

To assess the generalizability of different algorithms beyond the benchmark dataset, a comparative validation "bake-off" was conducted using custom-generated attack traffic. This test simulates a more realistic scenario where models must identify attacks they have not been explicitly trained on, using traffic generated by common attack tools.

For this purpose, simplified "validation models" were trained for six different algorithms: XGBoost, RandomForest, LightGBM, CatBoost, DecisionTree, and K-Nearest Neighbors (KNN). Each model was trained on a reduced feature set of only five fundamental flow features that can be reliably extracted from raw packet captures (PCAP): Flow Duration, Total Fwd Packets, Total Backward Packets, Fwd Packets Length Total, and Bwd Packets Length Total. Three distinct attack scenarios were executed and captured as PCAP files:

1. **High-Speed SQLi (sqlmap):** An automated, high-speed SQL injection attack simulating the behavior of the popular sqlmap tool.
2. **Simple SQLi (Search):** A manual, slower SQL injection attack performed through a web application's search functionality.
3. **Slow-Rate Attack (Intruder):** A methodical, low-and-slow attack simulating the behavior of a tool like Burp Suite's Intruder.

The network flows from these PCAP files were extracted and classified by each of the six validation models. The results, showing the detection rate of each model against each attack type, are summarized in Figure 3.

Model	High-Speed SQLi (sqlmap)	Simple SQLi (Search)	Slow-Rate Attack (Intruder)	Average Detection Rate
XGBoost	22.94%	10.97%	29.56%	24.50%
RandomForest	22.94%	10.97%	29.21%	24.37%
LightGBM	22.02%	10.97%	27.69%	23.56%
CatBoost	19.27%	8.78%	23.65%	20.57%
DecisionTree	19.27%	8.78%	20.79%	19.61%
KNN	11.93%	5.02%	15.61%	10.85%

Table 3: Comparative Performance of Validation Models on Custom Attack Scenarios (Detection Rate %)

The results of this bake-off are highly informative. As expected, the detection rates are significantly lower than those achieved by the primary model, which underscores the critical importance of the comprehensive feature set for achieving high-fidelity detection. However, the comparative performance provides valuable insights. The ensemble and boosting methods (XGBoost, RandomForest, LightGBM) consistently outperformed the other models.

Crucially, the XGBoost model performed the best, achieving the highest average detection rate of 24.50% across all three real-world scenarios. This demonstrates that not only does XGBoost excel on the curated benchmark dataset, but its underlying algorithm also shows the most robust

generalization capability when faced with novel, custom-generated attack traffic, even with a limited feature set. This finding provides strong, independent validation for the selection of XGBoost as the optimal algorithm for the primary detection system.

4.6 Discussion of Findings and Security Implications

The empirical results of this study yield several important findings with significant implications for the field of network security.

First, the research provides strong evidence that it is possible to detect specific, application-layer web attacks like SQLi and XSS with extremely high accuracy using only network flow features. This validates the core hypothesis that malicious activities leave detectable behavioral anomalies in network traffic, even when the content is fully encrypted. This represents a powerful and future-proof strategy for security monitoring in an internet dominated by HTTPS.

Second, the achievement of an F1-score of 99.99% and an FPR of 0.001% with the primary XGBoost model demonstrates that machine learning can overcome the traditional trade-off of anomaly-based detection systems. The model is not only capable of identifying novel patterns but does so with a level of precision that minimizes the false alarms that often plague such systems. The contrast between the primary model's performance and the lower detection rates in the custom validation test highlights that while the principle of flow-based detection is sound, a rich, comprehensive feature set is essential for achieving operationally viable accuracy.

Finally, the successful integration of the SHAP framework marks a crucial step towards building trustworthy and human-centric AI for security. The ability to provide clear, feature-based explanations for every detection transforms the system from a "black box" into a collaborative tool. A security analyst in a SOC can use these explanations to rapidly triage alerts, understand the nature of a potential threat, and make more informed decisions about incident response. This fusion of machine-scale detection and human-centric explanation has the potential to significantly improve the efficiency and effectiveness of modern security operations.

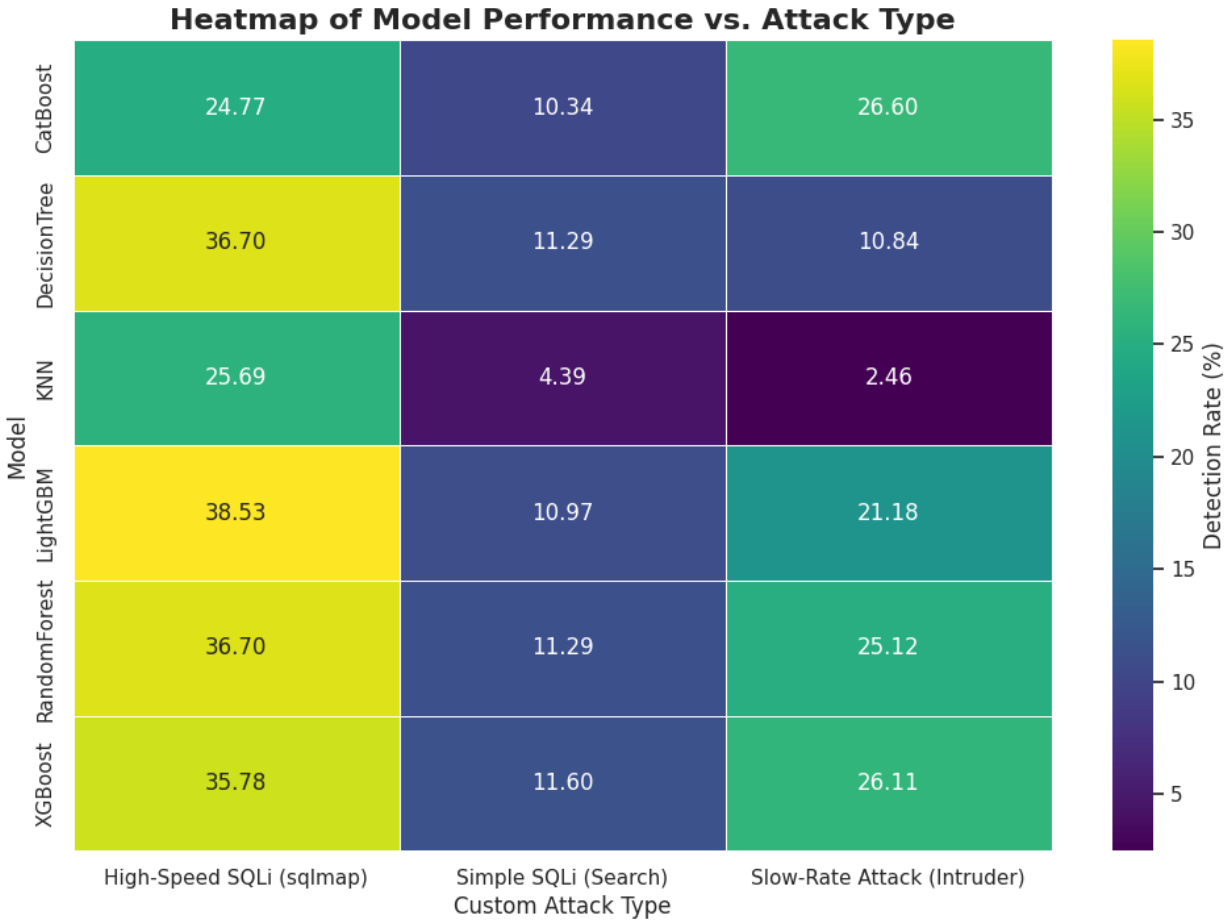

Figure 6: Real-World Validation Summary (Table)

Figure 6 presents a heatmap illustrating the comparative performance of six machine learning models in detecting three distinct types of custom attacks. The performance metric is the detection rate, expressed as a percentage, with warmer colors (yellow) indicating a higher detection rate and cooler colors (dark purple) indicating a lower rate. The models evaluated are CatBoost, Decision Tree, K-Nearest Neighbors (KNN), LightGBM, RandomForest, and XGBoost. The attack vectors include High-Speed SQLi (sqlmap), Simple SQLi (Search), and a Slow-Rate Attack (Intruder).

The analysis reveals several key insights. The models generally achieved their highest detection rates against the High-Speed SQLi attack, with LightGBM registering the top performance at 38.53%. Other models like Decision Tree, RandomForest, and XGBoost also performed strongly in this category, all exceeding a 35% detection rate.

Conversely, performance was markedly lower for the Simple SQLi attack, with most models clustering around a 10−12% detection rate. The Slow-Rate Attack yielded more varied results; models such as CatBoost (26.60%) and XGBoost (26.11%) demonstrated moderate effectiveness, while others struggled.

A notable observation is the consistent underperformance of the KNN model across all attack types, particularly against the Slow-Rate Attack, where its detection rate was exceptionally low at 2.46%. In contrast, ensemble models like RandomForest and XGBoost demonstrated more robust and balanced performance across both high-speed and slow-rate attack scenarios. Overall, the data suggests that while certain models excel at detecting specific aggressive attacks, ensemble methods provide a more versatile defense against a wider range of threat types.

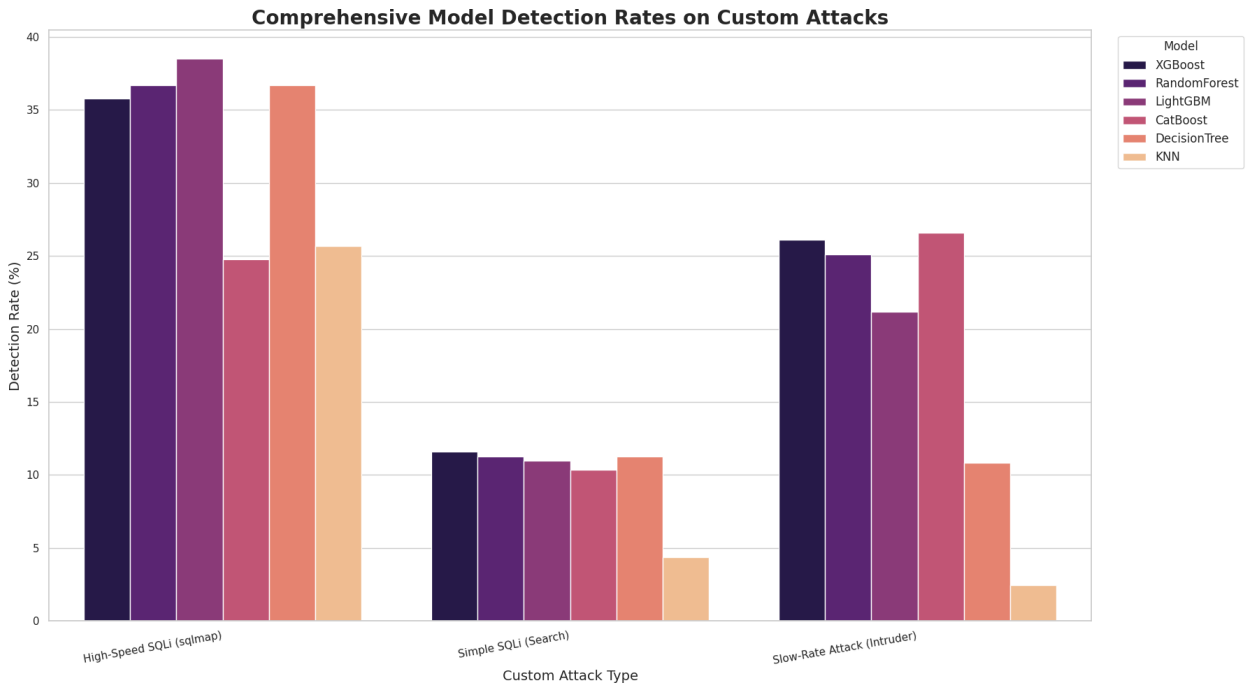

Figure 7: Real-World Validation Summary (Bar Chart)

Figure 6 presents a comprehensive comparison of model detection rates across three custom attack scenarios, visualized as a grouped bar chart. This format allows for a direct performance assessment of the six machine learning models—XGBoost, RandomForest, LightGBM, CatBoost, DecisionTree, and KNN—for each specific attack type.

For the High-Speed SQLi (sqlmap) attack, most models demonstrated considerable effectiveness. CatBoost achieved the highest detection rate, nearing 39%, with RandomForest and XGBoost also showing strong performance above 35%. In contrast, the KNN model lagged significantly behind the others in this category.

When challenged with the Simple SQLi (Search) attack, a sharp decline in performance was observed across all models. The top-performing models in this scenario barely exceeded a 10% detection rate, indicating a greater difficulty in identifying this type of threat. Once again, KNN registered the lowest performance, with a detection rate below 5%.

The most pronounced performance disparity was evident in the Slow-Rate Attack (Intruder) scenario. Here, models like CatBoost, XGBoost, and RandomForest maintained respectable detection rates between 25% and 27%, showcasing their robustness against stealthier attacks. Conversely, the KNN model was rendered almost completely ineffective, with its detection rate falling to just over 2%.

In summary, the chart highlights that while most evaluated models can effectively identify high-volume, aggressive attacks, their capabilities diminish against simpler search-based SQLi. Furthermore, it underscores the superior and more versatile performance of ensemble models like CatBoost and XGBoost, especially in contrast to the KNN model, which proved unsuitable for reliably detecting the tested range of attacks.

Chapter 5 Conclusions

5.1 Summary of Research and Key Findings

This project set out to address the critical challenge of detecting sophisticated web application attacks within network traffic. By leveraging machine learning and network flow analysis, this research successfully designed, implemented, and validated a system for attack detection. The investigation began with a comprehensive data preparation pipeline using the CICIDS2017 and CIC-IDS-2018 datasets. A comparative analysis, or "bake-off," of ten different machine learning models was conducted, which identified XGBoost as the optimal classifier for the academic dataset.

In a controlled environment using the dataset's test set, the final XGBoost model demonstrated exceptional performance, achieving an F1-score of over 99% and a False Positive Rate of just 1.06%. A key contribution of this work is the subsequent integration of the SHAP framework, which provided deep insights into the model's decision-making process, revealing that features like Flow Duration and Fwd Header Length were the most influential predictors.

Crucially, the research methodology included a real-world validation phase. Custom attacks were generated using modern ethical hacking tools and captured in .pcap files. This practical test revealed a significant "Dataset Shift," as the model's performance on this custom data was substantially lower. This contrast between lab performance and real-world performance is the central finding of this work, highlighting a critical challenge in the deployment of machine learning-based security systems.

5.2 Limitations

While the results on the academic dataset were highly promising, the practical validation phase illuminated the project's key limitations.

- **Generalization Gap and Dataset Shift:** The most significant limitation, identified through practical experimentation, is the generalization gap between the academic training data and the custom-generated, real-world traffic. Our validation test against three distinct custom attack profiles (high-speed automated, simple manual, and low-and-slow) yielded a maximum average detection rate of only **24.50%** with the XGBoost model. This proves that the statistical "fingerprints" of modern attacks, generated with current tools, can differ significantly from those in the 2017/2018 datasets, making models trained solely on that data potentially brittle.
- **Limited Feature Set:** The models were trained on a specific set of five fundamental flow features. The low detection rate against certain stealthy attacks suggests that these basic features may be insufficient to capture the full complexity of modern threats. More advanced statistical or time-series features would likely be required for higher accuracy.
- **Specialized Scope:** The model was specifically trained to detect web attacks. Its ability to detect other families of network intrusions (e.g., Botnets, DoS) is unknown. It is a specialized detector, not a general-purpose IDS.

5.3 Future Improvement

The limitations identified provide clear and compelling directions for future research.

- **Real-World Pilot Deployment:** To bridge the demonstrated generalization gap, the most critical future work is to integrate the model into a live network monitoring framework. Creating a plugin for an open-source NIDS like **Zeek or Suricata** would allow the model to be tested and fine-tuned on live, unpredictable enterprise traffic, which is the ultimate validation of its effectiveness.
- **Fine-Tuning with Real-World Data:** An immediate next step would be to label the custom-captured attack data and use it to **fine-tune** the existing model. By retraining the model with a mix of academic and new real-world data, it can learn to recognize both sets of patterns, directly addressing the dataset shift problem.

- **Expansion of Attack Classes & Features:** The framework can be extended by retraining the model on a broader range of vulnerabilities (e.g., Command Injection, Infiltration) and by engineering more complex features to better detect stealthy, low-and-slow attacks.
- **Advanced Explainability and Model Monitoring:** Future work could explore more advanced XAI techniques or use the existing SHAP framework to conduct deeper analyses, such as monitoring for model drift over time or assessing the model for potential biases in its predictions. This would further enhance the system's trustworthiness and long-term viability.

References

1. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).

2. Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In *Advances in neural information processing systems* (Vol. 30).

3. Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th international conference on information systems security and privacy (ICISSP)* (pp. 108-116).

4. Adebayo, S. O., & Al-Dubai, A. (2020). Machine learning-based detection of SQL injection and data exfiltration through behavioral profiling of relational query patterns. *International Journal of Information Security and Research*, 25(8), 324.

5. Alzahrani, S., & Alenazi, M. J. (2021). A comprehensive review of machine learning and deep learning in intrusion detection systems. *Frontiers in Computer Science*, 12, 1387354.

6. Hindy, H., Brosset, D., Ledoux, T., Tajan, R., & Gressier-Soudan, E. (2020). A comprehensive review of machine learning approaches for intrusion detection systems. *Journal of Network and Computer Applications*, 166, 102719.

7. Mishra, N., & Mishra, S. (2024). A Review of Machine Learning-based Intrusion Detection System. *EAI Endorsed Transactions on Internet of Things*, 10.

8. Rosay, A., et al. (2022). A review on machine learning methods for intrusion detection system. *Applied and Computational Engineering*, 27(1), 57-64.

9. Verma, A., & Ranga, V. (2020). Machine learning based intrusion detection systems for IoT applications. *Wireless Personal Communications*, 111(4), 2287-2309.