AI-Based Intrusion Detection System for Mobile Networks

Sefa KOC

July 2, 2025

1 Introduction

With the exponential growth of mobile and wireless communication technologies, securing network infrastructures has become more critical than ever. Modern networks face an increasing number of threats, including Denial-of-Service (DoS), Distributed Denial-of-Service (DDoS), port scanning, spoofing, and other cyber-attacks. Traditional rule-based intrusion detection systems often struggle to adapt to these evolving threats due to their static nature and dependence on predefined signatures.

This project presents an **AI-Based Intrusion Detection System (IDS)** designed specifically for mobile networks. The system utilizes machine learning techniques to detect and classify network traffic as either benign or malicious. A key feature of the proposed system is its support for both **real-time detection** from live network interfaces and **historical flow detection** from recorded flows. This dual capability allows for versatile testing, analysis, and validation of intrusion scenarios in both controlled and live environments.

The IDS features a Python-based graphical user interface (GUI) developed with PyQt5. The GUI provides users with interactive control over the system, enabling them to select network interfaces for real-time detection, simulate attack scenarios using historical data, retrain the model, and visualize detection results through dynamic charts and tables.

The project aims to deliver a functional IDS prototype that is both intelligent and interactive. It can perform live monitoring of network traffic while also allowing simulation of historical attacks for evaluation purposes. The combination of machine learning and real-time GUI interaction highlights the system's practicality and educational value in understanding and countering modern cybersecurity threats in mobile network contexts.

2 System Components

2.1 Dataset

The dataset used for this project is the **CIC-IDS2017 Intrusion Detection Evaluation Dataset**, provided by the Canadian Institute for Cybersecurity. This dataset is designed to reflect real-world network traffic and includes both benign and attack flows such as DoS, DDoS, PortScan, Botnet, and Brute Force attacks. It captures packet-level and flow-level network behavior across various days and scenarios.

Each record in the dataset represents a network flow and includes over 80 statistical features extracted from raw traffic using CICFlowMeter. Some of these features include timestamps, IP addresses, packet counts, flow duration, flag counts, inter-arrival times, and various statistical summaries of packet sizes and flow characteristics.

Given the high dimensionality of the dataset, not all features are suitable for real-time analysis. Therefore, feature selection was performed to identify a subset of features that are computationally efficient and relevant for intrusion detection in both real-time and simulated contexts.

2.2 Feature Set

From the original 80+ features available in the CIC-IDS2017 dataset, a carefully selected set of **19 features** was used to train the detection model. These features were chosen based on their availability in both real-time packet analysis and their contribution to model performance.

- Protocol
- Flow Duration
- Total Fwd Packets
- Total Bwd Packets
- Fwd Packet Len Max
- Fwd Packet Len Min
- Fwd Packet Len Mean
- Bwd Packet Len Max
- Bwd Packet Len Min
- Bwd Packet Len Mean
- Flow Bytes/s
- Flow Packets/s
- SYN Flag Count
- ACK Flag Count
- PSH Flag Count
- RST Flag Count
- Fwd Header Len
- Bwd Header Len
- Packet Len Mean

These features provide a balance of statistical, protocol-level, and behavioral attributes, capturing the essence of both normal and malicious flows. They were selected to ensure compatibility with real-time detection using tools like `Scapy`, as well as for simulating historical flows from preprocessed CSV files. This reduced feature space enables efficient model inference and improves the responsiveness of the system.

2.3 Machine Learning Model

The Intrusion Detection System employs an **XGBoost (Extreme Gradient Boosting)** classifier as the core of its detection engine. XGBoost is a powerful ensemble learning method based on gradient boosting that is particularly well-suited for handling structured tabular data. It supports multiclass classification, handles missing data efficiently, and offers robust performance on imbalanced datasets.

XGBoost was selected over other classifiers such as Random Forest or Support Vector Machines due to its superior training speed, high predictive accuracy, and built-in support for feature importance visualization. These advantages make it ideal for security-related classification tasks, where model interpretability and computational efficiency are crucial.

2.4 Training Process

The model is trained using the 19 selected flow-based features extracted from the CIC-IDS2017 dataset. It is saved to disk using `joblib` for reuse in both real-time and historical flow analysis. Additionally, a `LabelEncoder` is used to convert string-based labels into numeric classes, and a `StandardScaler` is used to normalize the features for improved convergence during training.

The training process begins by aggregating and preprocessing the raw CSV files from the CIC-IDS2017 dataset. Records with missing or infinite values are removed to ensure data quality. Only selected attack types and benign traffic are retained to focus on the most relevant classes. The dataset is then shuffled and stratified to maintain class distribution across training and test sets.

Before model training, **20% of the data for each label category is separated and saved into a new CSV file to be used specifically for the historical flow simulation feature of the system.** This subset is excluded from the training process to prevent data leakage and to provide a realistic test environment. The remaining 80% of the data is retained and saved as the updated dataset for model training.

Subsequently, the feature values are standardized using `StandardScaler`, and the class labels are encoded using `LabelEncoder`. The training dataset is then split into 70% training and 30% testing partitions. The XGBoost classifier is trained on this processed data using appropriate hyperparameters.

Post-training, performance evaluation is conducted using precision, recall, F1-score, and a confusion matrix. These metrics are visualized and saved as charts to allow users to analyze the model's strengths and weaknesses. The system also supports retraining the model directly from the GUI, allowing users to update the model with new or additional data without modifying the codebase.

2.5 Used Libraries and Tools

The implementation is entirely based on Python, utilizing a wide range of powerful third-party libraries to handle data manipulation, machine learning, GUI development, and real-time packet analysis.

For data handling and preprocessing, the following libraries are used:

- `pandas` – for data loading, transformation, and CSV operations
- `numpy` – for numerical computations

- `joblib` – for model and object serialization

For machine learning and model evaluation:

- `scikit-learn` – for preprocessing, encoding, evaluation metrics, and model utilities
- `xgboost` – for training the multiclass gradient boosting classifier

For data visualization:

- `matplotlib` and `seaborn` – for generating charts like feature importance, confusion matrix, and label distributions

For the graphical user interface:

- `PyQt5` – for building a responsive and modern desktop GUI with multiple interactive windows

For real-time packet capture and network traffic parsing:

- `scapy` – for accessing live network packets, extracting protocol and header details, and capturing TCP flag information

Additionally, to enable live packet capture, the system requires installation of **Npcap**, a Windows packet capture driver, which can be downloaded from: `https://npcap.com/`

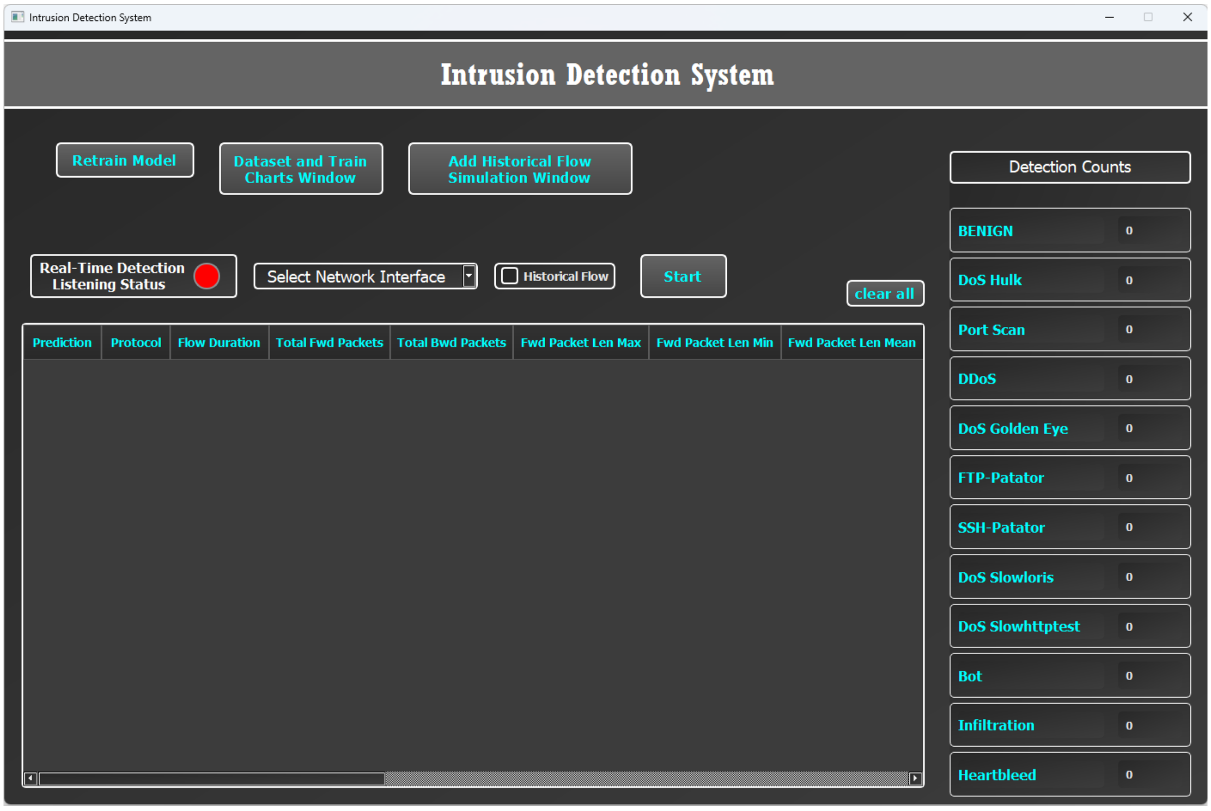
All required Python packages can be installed using the following command:

```
pip install pandas numpy joblib scikit-learn xgboost matplotlib seaborn scapy pyqt5
```

3 Graphical User Interface and System Workflow

The developed system includes an interactive and user-friendly **Graphical User Interface (GUI)** built using `PyQt5`. The GUI is designed to provide intuitive control over all major functions of the Intrusion Detection System, including real-time traffic analysis, historical flow simulation, model retraining, and performance visualization. The application is composed of three main windows: the **Main Window**, the **Historical Flow Simulation Window**, and the **Charts Window**. Each of these plays a specific role in the IDS workflow.

3.1 Main Window



BENIGN	0
DoS Hulk	0
Port Scan	0
DDoS	0
DoS Golden Eye	0
FTP-Patator	0
SSH-Patator	0
DoS Slowloris	0
DoS Slowhttptest	0
Bot	0
Infiltration	0
Heartbleed	0

Figure 1: Main Window

The Main Window serves as the central control panel of the IDS system. Its layout includes the following key components:

- **Retrain Model Button:** Triggers retraining of the model using updated datasets.
- **Open Charts Button:** Opens the Charts Window to visualize model performance metrics.
- **Open Simulation Window Button:** Opens the Historical Flow Simulation Window.
- **Start/Stop Button:** Toggles real-time or historical detection. Disables interface selection and simulation options during active detection.

- **Network Interface ComboBox:** Lists available network interfaces for real-time traffic capture.
- **Historical Flow Checkbox:** Determines whether detection runs on historical flow data or live traffic.
- **Detection Status Indicator:** A colored circular indicator (red/green blinking) shows whether detection is active or not.
- **Clear All Button:** Resets all detection counters and clears the real-time detection table.
- **Detection Counts Panel:** Displays the number of detected flows for each flow type (e.g., BENIGN, DDoS, PortScan).
- **Real-Time Detection Table:** Displays all detected flows with their predicted class and associated features.

When the Start button is pressed, the system either starts sniffing live packets or processes historical flows based on the checkbox selection. Detected flows are shown in a table, and detection counts are updated accordingly. The interface and simulation checkbox are disabled during active detection to maintain state consistency.

3.2 Historical Flow Simulation Window

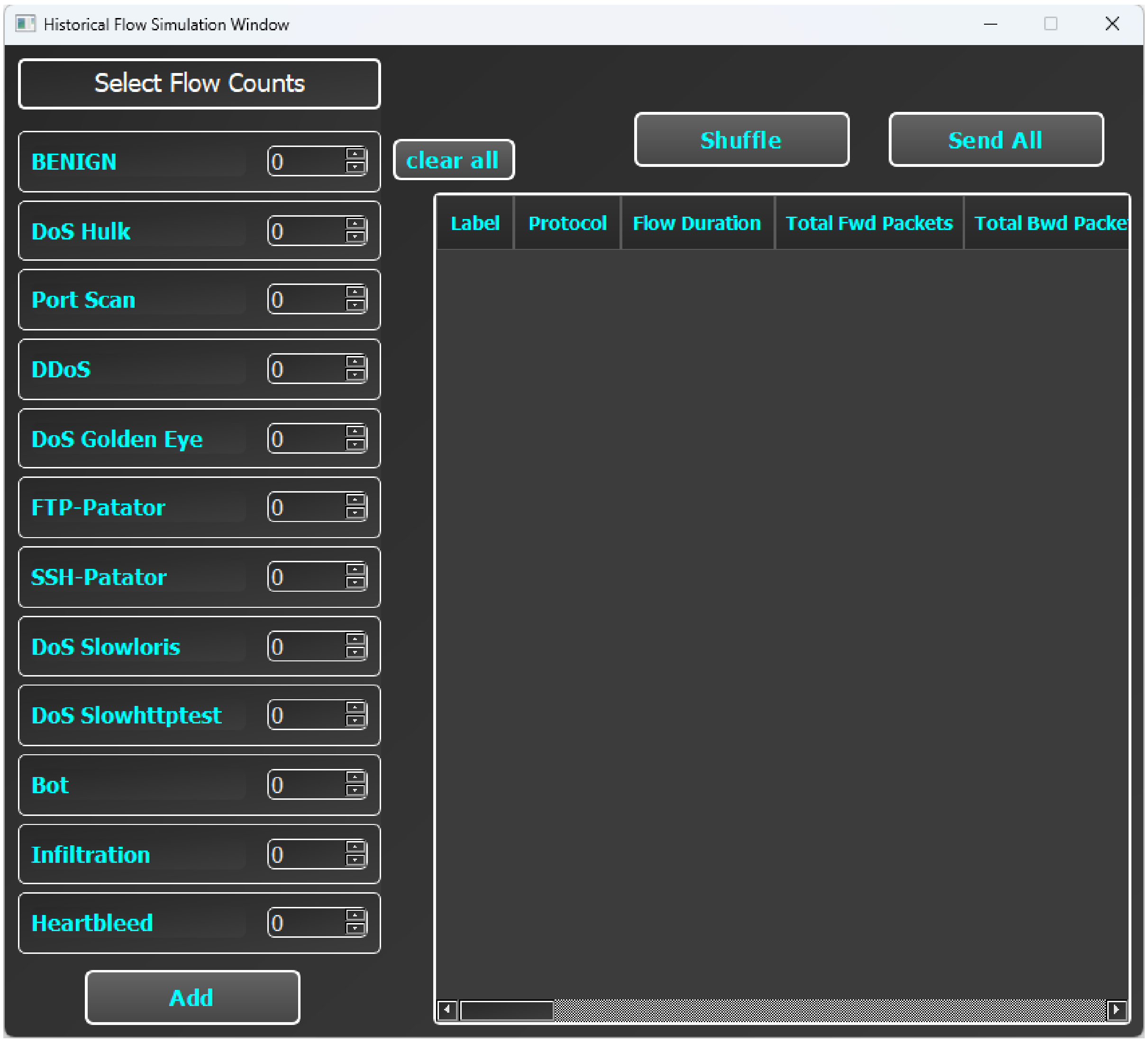


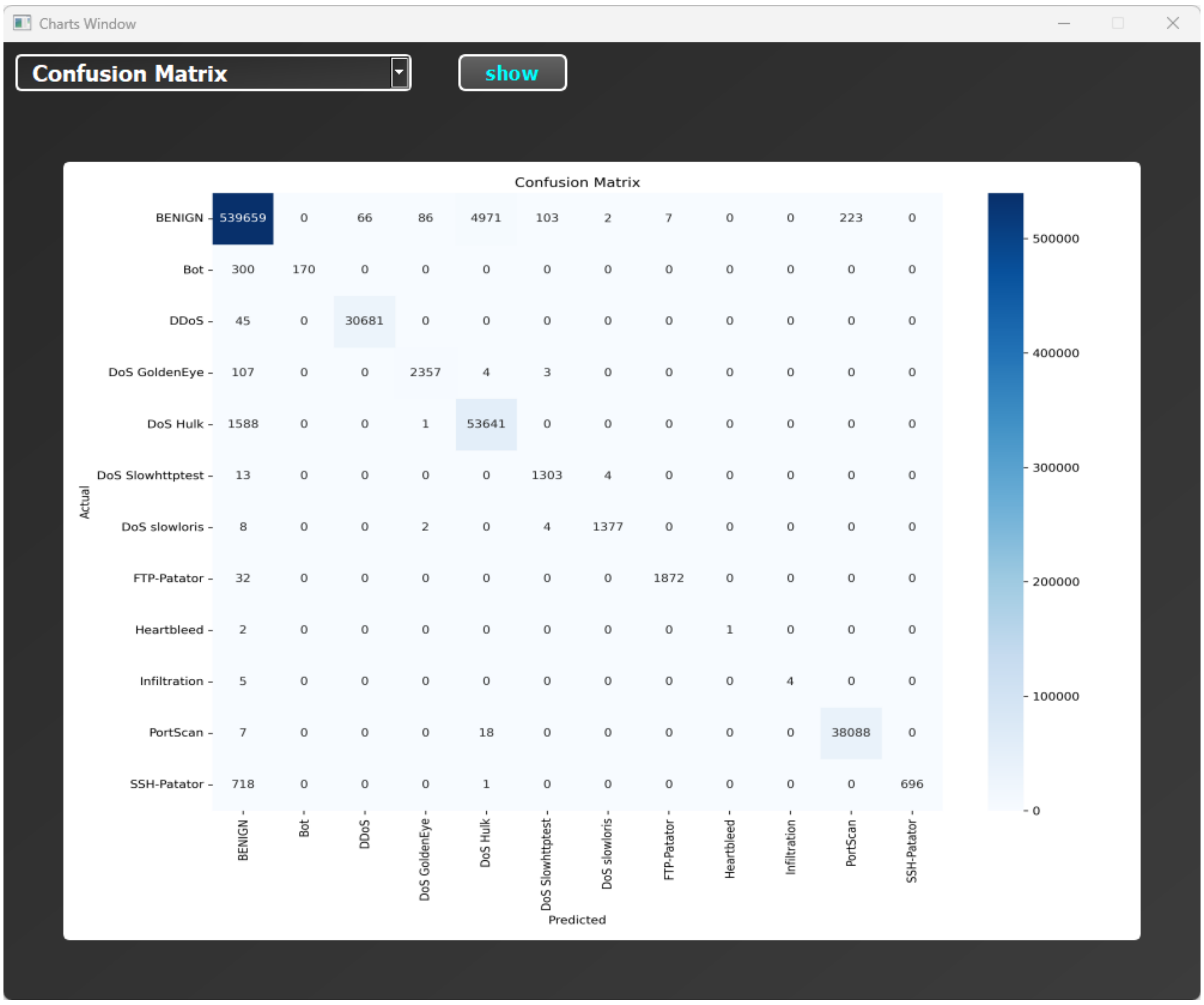
Figure 2: Historical Flow Simulation Window

This window allows users to simulate intrusion scenarios using a predefined dataset. The layout includes:

- **Flow Type Spinboxes:** Each supported label (e.g., BENIGN, DDoS, PortScan) has a spinbox to select the number of flows to simulate.
- **Add Button:** Adds selected flows to a table based on the selected counts.
- **Table View:** Displays all added flows along with their features and labels.
- **Shuffle Button:** Randomly shuffles the order of table rows.
- **Clear All Button:** Clears both the spinboxes and the table.
- **Send All Button:** Appends the listed flows to the `historical.csv` file, which will be used for historical flow detection.

Once the flows are sent, they can be detected when the system is run in Historical Flow mode via the Main Window. Each flow is processed by the trained model, and its predicted class is shown in the real-time detection table.

3.3 Charts Window



Confusion Matrix

	BENIGN	Bot	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS slowloris	FTP-Patator	Heartbleed	Infiltration	PortScan	SSH-Patator
BENIGN	539659	0	66	86	4971	103	2	7	0	0	223	0
Bot	300	170	0	0	0	0	0	0	0	0	0	0
DDoS	45	0	30681	0	0	0	0	0	0	0	0	0
DoS GoldenEye	107	0	0	2357	4	3	0	0	0	0	0	0
DoS Hulk	1588	0	0	1	53641	0	0	0	0	0	0	0
DoS Slowhttptest	13	0	0	0	0	1303	4	0	0	0	0	0
DoS slowloris	8	0	0	2	0	4	1377	0	0	0	0	0
FTP-Patator	32	0	0	0	0	0	0	1872	0	0	0	0
Heartbleed	2	0	0	0	0	0	0	0	1	0	0	0
Infiltration	5	0	0	0	0	0	0	0	0	4	0	0
PortScan	7	0	0	0	18	0	0	0	0	0	38088	0
SSH-Patator	718	0	0	0	1	0	0	0	0	0	0	696

Figure 3: Charts Window

The Charts Window is designed to help users evaluate model performance through visual analytics. It includes a dropdown menu to select various chart types:

- Label Distribution
- Feature Importance
- Test Samples Distribution
- Confusion Matrix
- Classification Report (Precision, Recall, F1-score)
- Accuracy vs Support

These charts are automatically generated during training and saved as image files. When selected, each chart is displayed within the window, allowing users to interpret model behavior and class-specific performance.

3.4 System Workflow

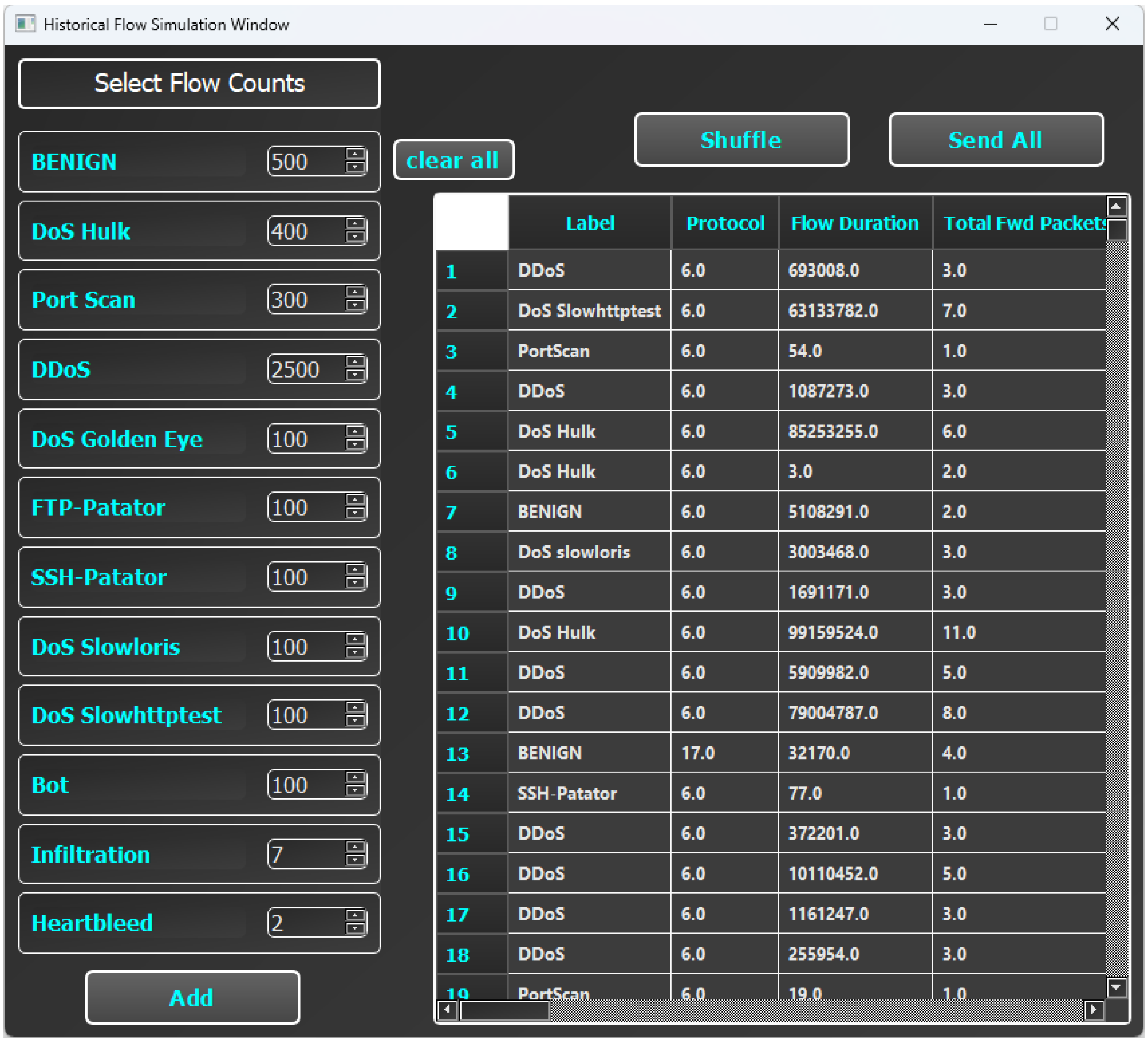
The system workflow follows a modular pipeline:

1. The user optionally retrains the model using new or updated datasets.
2. The user selects either live traffic or historical flow mode.
3. Detection is started. Live packets are captured and converted to flows, or historical flows are loaded from a file.
4. The feature vector is extracted and scaled. The trained model predicts the class of each flow.
5. The flow and prediction are added to the detection table, and detection counts are updated.
6. The user may analyze detection patterns in the table or view model analytics via the Charts Window.

This structured and interactive workflow ensures usability for both analysis and experimentation, making the system a practical tool for intrusion detection in mobile network environments.

3.4.1 Historical Detection Workflow

1- The user selects specific flow types, adds them to the simulation table, optionally shuffles their order, and then clicks the **Send All** button to add them to historical csv file for detection. Historical flows can be added regardless of whether the detection system is actively running or not.

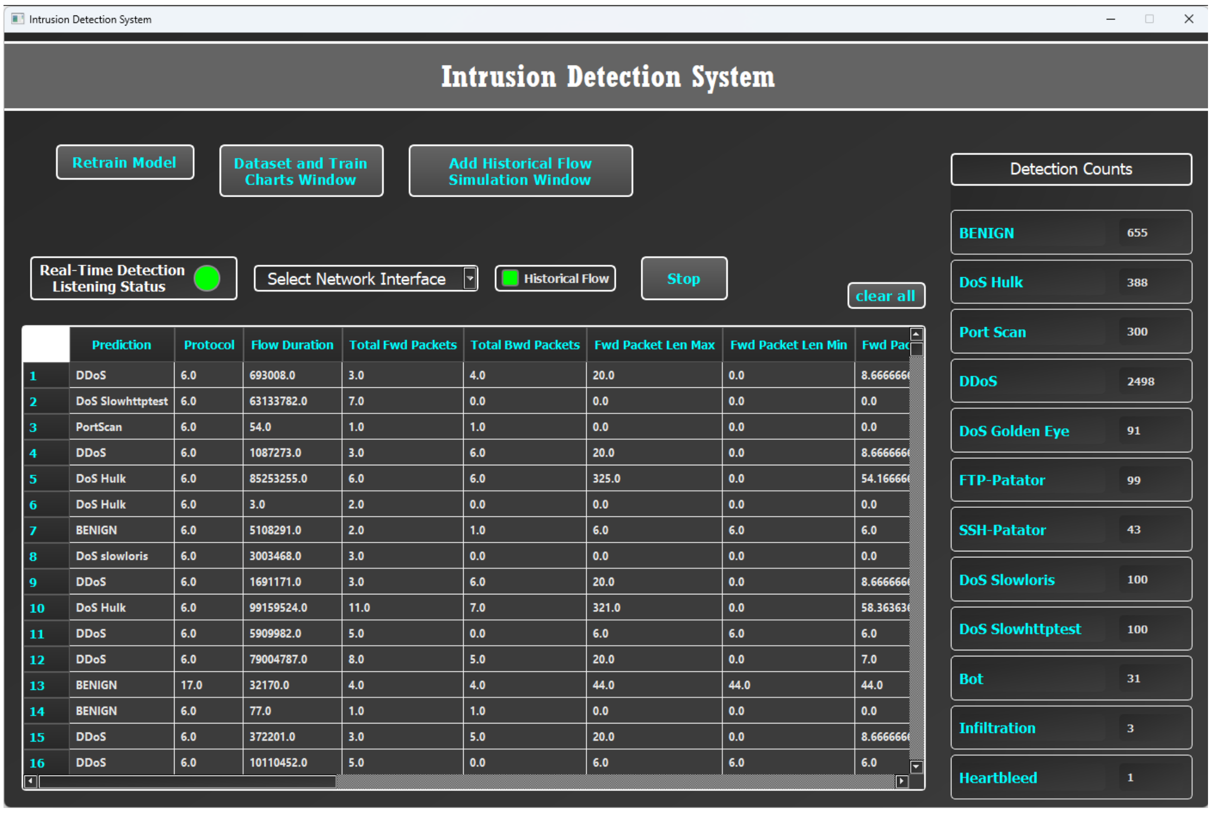


BENIGN	500
DoS Hulk	400
Port Scan	300
DDoS	2500
DoS Golden Eye	100
FTP-Patator	100
SSH-Patator	100
DoS Slowloris	100
DoS Slowhttptest	100
Bot	100
Infiltration	7
Heartbleed	2

	Label	Protocol	Flow Duration	Total Fwd Packets
1	DDoS	6.0	693008.0	3.0
2	DoS Slowhttptest	6.0	63133782.0	7.0
3	PortScan	6.0	54.0	1.0
4	DDoS	6.0	1087273.0	3.0
5	DoS Hulk	6.0	85253255.0	6.0
6	DoS Hulk	6.0	3.0	2.0
7	BENIGN	6.0	5108291.0	2.0
8	DoS slowloris	6.0	3003468.0	3.0
9	DDoS	6.0	1691171.0	3.0
10	DoS Hulk	6.0	99159524.0	11.0
11	DDoS	6.0	5909982.0	5.0
12	DDoS	6.0	79004787.0	8.0
13	BENIGN	17.0	32170.0	4.0
14	SSH-Patator	6.0	77.0	1.0
15	DDoS	6.0	372201.0	3.0
16	DDoS	6.0	10110452.0	5.0
17	DDoS	6.0	1161247.0	3.0
18	DDoS	6.0	255954.0	3.0
19	PortScan	6.0	19.0	1.0

Figure 4: Historical Detection Workflow Simulation Window

2- In the Main Window, when the **Historical Flow** checkbox is selected and the **Start** button is pressed, the system begins processing the queued historical flows. The predicted flows are displayed in the real-time detection table, and the corresponding detection counters are updated. During active detection, both the network interface selection and the historical flow option are disabled to ensure consistent system behavior.


Intrusion Detection System
Intrusion Detection System
Retrain Model
Dataset and Train Charts Window
Add Historical Flow Simulation Window
Real-Time Detection Listening Status
Select Network Interface
Historical Flow
Stop
clear all
Detection Counts

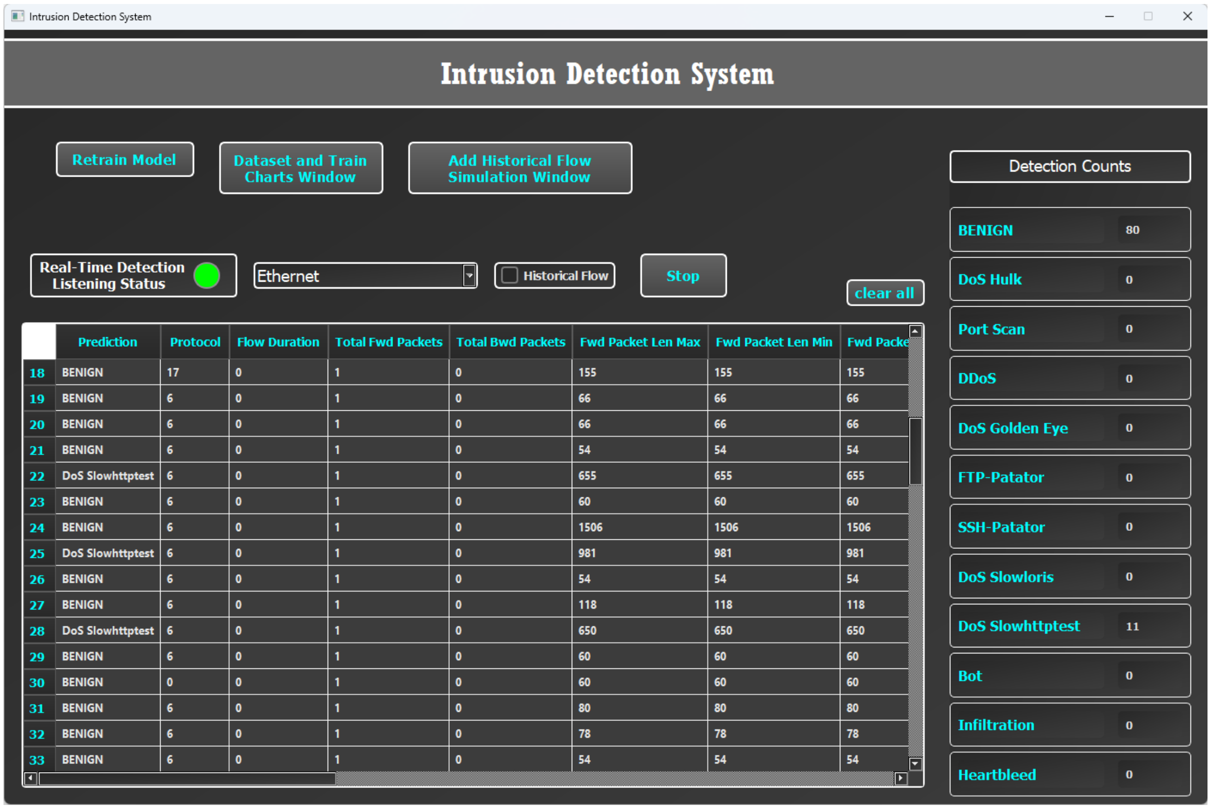
	Prediction	Protocol	Flow Duration	Total Fwd Packets	Total Bwd Packets	Fwd Packet Len Max	Fwd Packet Len Min	Fwd Pac
1	DDoS	6.0	693008.0	3.0	4.0	20.0	0.0	8.666666
2	DoS Slowhttptest	6.0	63133782.0	7.0	0.0	0.0	0.0	0.0
3	PortScan	6.0	54.0	1.0	1.0	0.0	0.0	0.0
4	DDoS	6.0	1087273.0	3.0	6.0	20.0	0.0	8.666666
5	DoS Hulk	6.0	85253255.0	6.0	6.0	325.0	0.0	54.16666
6	DoS Hulk	6.0	3.0	2.0	0.0	0.0	0.0	0.0
7	BENIGN	6.0	5108291.0	2.0	1.0	6.0	6.0	6.0
8	DoS slowloris	6.0	3003468.0	3.0	0.0	0.0	0.0	0.0
9	DDoS	6.0	1691171.0	3.0	6.0	20.0	0.0	8.666666
10	DoS Hulk	6.0	99159524.0	11.0	7.0	321.0	0.0	58.36363
11	DDoS	6.0	5909982.0	5.0	0.0	6.0	6.0	6.0
12	DDoS	6.0	79004787.0	8.0	5.0	20.0	0.0	7.0
13	BENIGN	17.0	32170.0	4.0	4.0	44.0	44.0	44.0
14	BENIGN	6.0	77.0	1.0	1.0	0.0	0.0	0.0
15	DDoS	6.0	372201.0	3.0	5.0	20.0	0.0	8.666666
16	DDoS	6.0	10110452.0	5.0	0.0	6.0	6.0	6.0

BENIGN	655
DoS Hulk	388
Port Scan	300
DDoS	2498
DoS Golden Eye	91
FTP-Patator	99
SSH-Patator	43
DoS Slowloris	100
DoS Slowhttptest	100
Bot	31
Infiltration	3
Heartbleed	1

Figure 5: Historical Detection Workflow Main Window

3.4.2 Real-Time Detection Workflow

In the Main Window, when the Historical Flow checkbox is not selected and a network interface is chosen from the dropdown menu, pressing the **Start** button initiates real-time traffic analysis. The system begins capturing live packets from the selected interface, processes them into flows, and performs prediction using the trained model. Detected flows are appended to the real-time table, and class-specific detection counters are updated accordingly. During active detection, the network interface selector and the Historical Flow checkbox are disabled to prevent inconsistent runtime behavior.


Intrusion Detection System
Intrusion Detection System
Retrain Model
Dataset and Train Charts Window
Add Historical Flow Simulation Window
Real-Time Detection Listening Status
Ethernet
Historical Flow
Stop
clear all
Detection Counts

	Prediction	Protocol	Flow Duration	Total Fwd Packets	Total Bwd Packets	Fwd Packet Len Max	Fwd Packet Len Min	Fwd Packe
18	BENIGN	17	0	1	0	155	155	155
19	BENIGN	6	0	1	0	66	66	66
20	BENIGN	6	0	1	0	66	66	66
21	BENIGN	6	0	1	0	54	54	54
22	DoS Slowhttptest	6	0	1	0	655	655	655
23	BENIGN	6	0	1	0	60	60	60
24	BENIGN	6	0	1	0	1506	1506	1506
25	DoS Slowhttptest	6	0	1	0	981	981	981
26	BENIGN	6	0	1	0	54	54	54
27	BENIGN	6	0	1	0	118	118	118
28	DoS Slowhttptest	6	0	1	0	650	650	650
29	BENIGN	6	0	1	0	60	60	60
30	BENIGN	0	0	1	0	60	60	60
31	BENIGN	6	0	1	0	80	80	80
32	BENIGN	6	0	1	0	78	78	78
33	BENIGN	6	0	1	0	54	54	54

BENIGN	80
DoS Hulk	0
Port Scan	0
DDoS	0
DoS Golden Eye	0
FTP-Patator	0
SSH-Patator	0
DoS Slowloris	0
DoS Slowhttptest	11
Bot	0
Infiltration	0
Heartbleed	0

Figure 6: Real-Time Detection Workflow Main Window

However, in the real-time detection mode, it is not possible to calculate certain flow-level features accurately, such as **Flow Duration**, **Total Backward Packets**, and some statistical attributes related to inter-arrival times or bi-directional communication. This limitation arises because these features typically require observation of a complete flow over time or the capture of response packets, which may not be available in single-directional or short-lived packet streams captured in real-time.

As a result, the feature vectors extracted from live packets are often incomplete or biased toward forward-only characteristics. This leads the model to classify a majority of real-time flows as **BENIGN**, since attack flows often exhibit richer or more complex patterns that rely on fully observed bidirectional traffic. While this approach allows the system to demonstrate real-time functionality, it also highlights a challenge in intrusion detection: accurate real-time classification requires access to full context, which may not always be possible in packet-level monitoring.

Conclusion

In this project, a comprehensive Intrusion Detection System (IDS) was developed that combines both real-time network traffic analysis and historical flow simulation. By leveraging the CIC-IDS2017 dataset, the system was trained using selected flow-based features and an XGBoost classifier to achieve high accuracy in multi-class traffic classification.

The system integrates a user-friendly graphical interface, enabling users to retrain the model, visualize traffic patterns through interactive charts, and simulate network intrusions using historical flow injections. The real-time detection module captures live traffic from the selected network interface and processes it to detect suspicious flows instantly. Additionally, historical simulation allows testing the detection engine against known labeled data to evaluate robustness.

Throughout the development process, careful attention was paid to usability, modularity, and extensibility. The application uses PyQt5 for the GUI, Scapy for real-time packet analysis, and several scientific Python libraries for data processing and model management. One limitation encountered during real-time detection was the inability to extract some advanced flow-level features due to the short lifespan or partial visibility of real-time packet groups. This sometimes caused the model to misclassify potentially malicious flows as benign.

In conclusion, this IDS project demonstrates the effective use of machine learning techniques in network security applications. It offers both a functional detection system and a flexible simulation and evaluation environment, making it suitable for future research, classroom demonstrations, or as a foundation for a production-level security tool.