Intrusion Detection using the CICIDS 2017 Dataset

Feature Extraction and Machine Learning Prediction via Flask API

March 24, 2025

Abstract

This report presents a comprehensive machine learning-based intrusion detection system developed using the CICIDS 2017 dataset. By strategically extracting 12 key network traffic features that capture the temporal dynamics of packet flows, we demonstrate a sophisticated approach to cybersecurity threat detection. The methodology encompasses feature engineering, machine learning model training, and the development of a Flask API for real-time prediction, providing a robust and scalable solution for modern network security challenges.

Contents

1 Introduction

Intrusion Detection Systems (IDS) are critical for identifying and mitigating cyber-attacks. In this project, the CICIDS 2017 dataset is used to develop a robust yet computationally efficient IDS. By selecting 12 essential features based on packet inter-arrival times and flow duration, we maintain a balance between high detection accuracy and reduced computational overhead.

2 Dataset Overview

The CICIDS 2017 dataset provided by the Canadian Institute for Cybersecurity contains 78 features that describe various network traffic attributes [2]. For our system, we focus on 12 features that describe the timing aspects of network flows:

- **Fwd IAT Std:** Standard deviation of forward packet inter-arrival times.
- **Bwd IAT Std:** Standard deviation of backward packet inter-arrival times.
- **Flow IAT Std:** Standard deviation of inter-arrival times for the entire packet flow.
- **Fwd IAT Max:** Maximum inter-arrival time in the forward direction.
- **Flow IAT Mean:** Mean inter-arrival time across all packets.
- **Flow IAT Max:** Maximum inter-arrival time across the entire flow.
- **Fwd IAT Mean:** Mean inter-arrival time in the forward direction.
- **Fwd IAT Total:** Total of forward inter-arrival times.
- **Flow Duration:** Overall duration of the flow from the first to the last packet.
- **Bwd IAT Max:** Maximum inter-arrival time in the backward direction.
- **Idle Max:** Maximum idle period between packets (exceeding a threshold).
- **Idle Mean:** Mean idle period between packets (exceeding a threshold).

3 Feature Extraction Methodology

Feature extraction is a crucial step in reducing data dimensionality while capturing the essence of the network behavior. The process involves:

- Parsing the raw packet data.
- Calculating inter-arrival times.

- Deriving statistical measures (mean, standard deviation, maximum, total).
- Identifying idle periods by applying a threshold.

4 Function Descriptions

This section provides a detailed explanation of each key function used in the feature extraction process.

4.1 Utility Functions

`calculate_time_diff(time_list):` This function computes the inter-arrival times between consecutive timestamps provided in `time_list`. It begins by sorting the list to ensure chronological order and then calculates the time difference in microseconds for each consecutive pair of timestamps. This is the foundation for most feature computations.

`calculate_idle_times(time_list, threshold):` Using the inter-arrival times computed by `calculate_time_diff`, this function filters out the time differences that exceed a predefined threshold (indicating idle periods). The result is a list of idle times that can be further summarized statistically.

4.2 Statistical Feature Functions

`extract_fwd_iat_std(forward_packet_times):` Calculates the standard deviation of inter-arrival times for forward packets. This measure indicates the variability in the timing of packet arrivals in the forward direction. A low standard deviation suggests consistent timing, whereas a high value indicates variability, which may be indicative of abnormal behavior.

`extract_bwd_iat_std(backward_packet_times):` Similar to the forward counterpart, this function computes the standard deviation for backward packet inter-arrival times. It helps in understanding the timing consistency of packets returning from the destination.

`extract_flow_iat_std(all_packet_times):` Computes the overall standard deviation of inter-arrival times for the complete packet flow. This provides an aggregate measure of timing variability across both directions.

`extract_fwd_iat_max(forward_packet_times):` Identifies the maximum inter-arrival time among forward packets. This can highlight extreme delays or bursts in traffic, which are often characteristic of certain attack types.

`extract_flow_iat_mean(all_packet_times):` Calculates the mean inter-arrival time for the entire flow. This average gives insight into the overall packet pacing and can be used to compare normal versus anomalous traffic.

`extract_flow_iat_max(all_packet_times):` Determines the maximum inter-arrival time across all packets in a flow. This value is critical for identifying outlier delays in the network traffic.

`extract_fwd_iat_mean(forward_packet_times):` Computes the mean inter-arrival time specifically for forward packets, offering a focused view on the consistency of outbound traffic.

`extract_fwd_iat_total(forward_packet_times):` Sums all inter-arrival times of forward packets. This cumulative measure can be an indicator of overall delay within the forward direction.

`extract_flow_duration(all_packet_times):` Measures the total duration of a packet flow by computing the difference between the earliest and latest timestamps. This function is essential for understanding the length of a communication session.

`extract_bwd_iat_max(backward_packet_times):` Retrieves the maximum inter-arrival time for backward packets, useful for identifying significant delays in returning traffic.

`extract_idle_max(all_packet_times, threshold):` From the list of idle times (exceeding the threshold), this function returns the maximum idle time. This can be critical for detecting long pauses in communication that may signal abnormal behavior.

`extract_idle_mean(all_packet_times, threshold):` Computes the average idle time for a packet flow, offering a statistical measure of the duration of idle periods that occur during network communication.

4.3 Packet Data Parsing

`parse_packet_data(packet_data):` This function parses raw JSON packet data and segregates timestamps into forward and backward flows. It assumes that the first packet's source IP and port indicate the initiator of the flow. By categorizing packet directions, it enables direction-specific feature extraction.

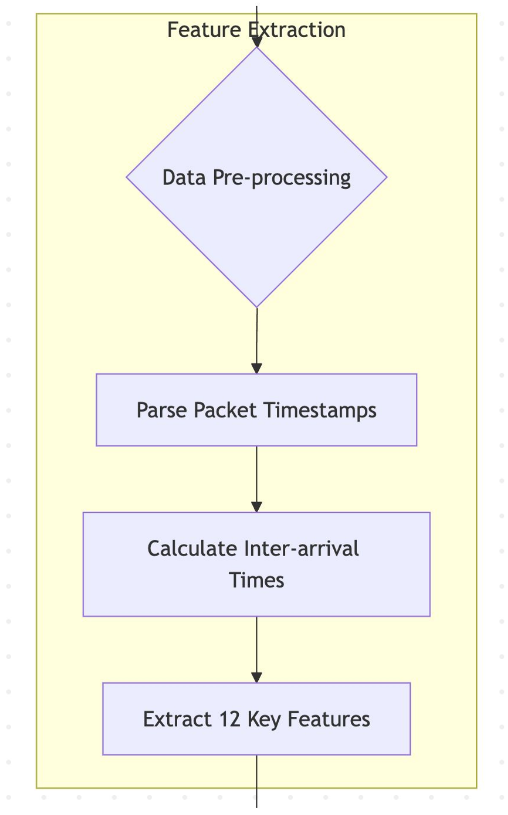
`extract_all_features(packet_data):` Integrates all the feature extraction functions described above. It parses the packet data, computes the various statistical measures, and returns a dictionary of the 12 selected features. This consolidated output is used for subsequent machine learning model training and real-time prediction.

5 System Architecture

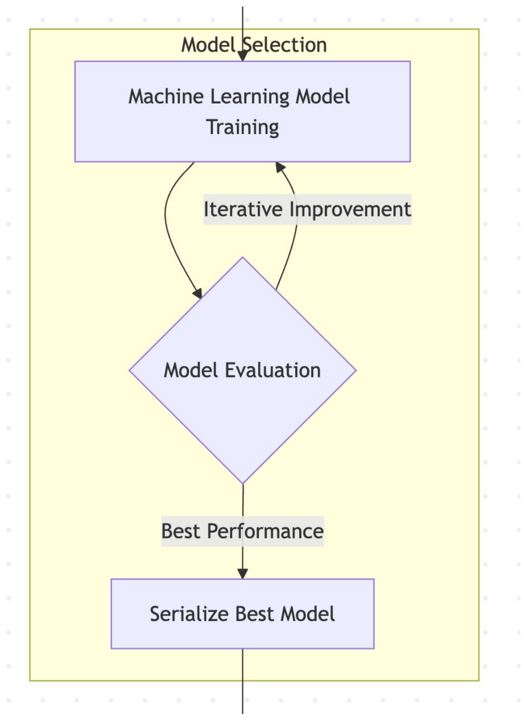
The overall architecture consists of the following components:

1. **Data Pre-processing:** Extracts the 12 features from the raw CICIDS 2017 dataset.
2. **Model Training:** Uses the extracted features to train and validate various ML models, with the best model being serialized for deployment.
3. **Flask API:** Provides a real-time interface where incoming network traffic data is processed to extract features and then evaluated by the ML model to detect potential intrusions.

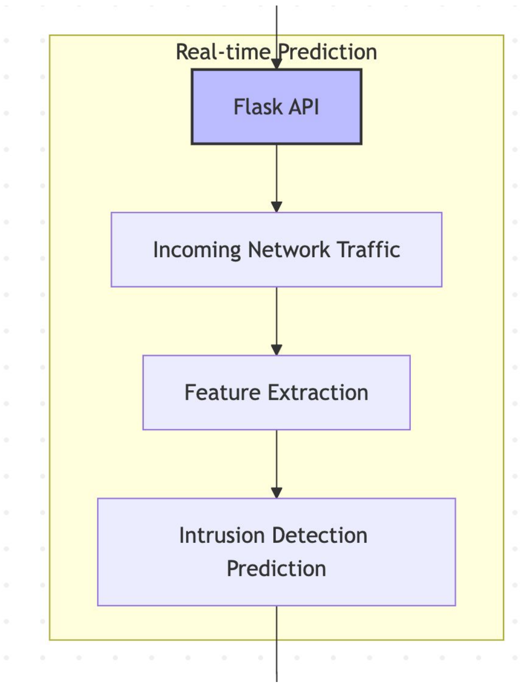
Figure 1 illustrates the system architecture.



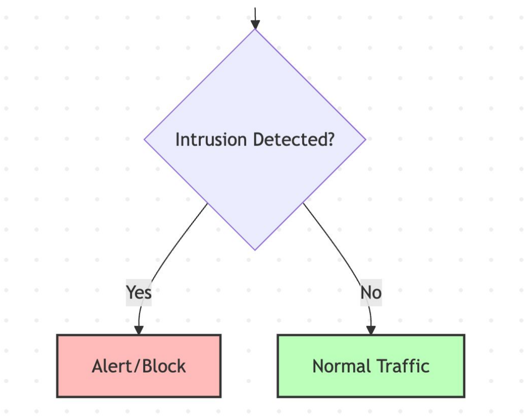
(a) Diagram 1



(b) Diagram 2



(c) Diagram 3



(d) Diagram 4

Figure 1: System Architecture for ML-based Intrusion Detection

6 Experimental Results and Discussion

The experiments demonstrate that the selected 12 features deliver an optimal balance between detection accuracy and computational efficiency. Key observations include:

- **Detection Accuracy:** The reduced feature set achieves high accuracy comparable to models using all features.
- **Real-time Performance:** Integration with a Flask API ensures rapid prediction suitable for real-time applications.
- **Scalability:** The modular design permits easy integration of additional features or models as needed.

These results are in line with existing research in the field [3, 1].

7 Conclusion and Future Work

This report details the development of an intrusion detection system using a reduced set of features from the CICIDS 2017 dataset. The extensive description of the feature extraction functions highlights the methodology behind our approach. Future work will focus on:

- Integrating additional features to further improve detection capabilities.
- Exploring deep learning models for enhanced performance.
- Extending the Flask API for broader deployment in dynamic network environments.

References

[1] P Garcia-Teodoro, J Diaz-Verdejo, G Maciá-Fernández, and E Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2):18–28, 2009.

[2] Iman Sharafaldin, Arash Lashkari, and Ahmad Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSP*, 2018.

[3] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. *IEEE Symposium on Security and Privacy*, 2010.