AI Integration of BlockShield Project

Parsing Honeypot Traffic and Developing the AI Engine

Nashat Alfarajat – BlockShield AI Development Team

CrossRealms - intern

PCAP File Parsing and Flow-Level Feature Extraction (CICFlowMeter-Style) Using Scapy

To prepare the honeypot network captures for AI model development, the raw .pcap files were parsed into flow-level features using a custom Python script built with Scapy. The goal was to emulate the feature extraction style of CICFlowMeter, producing statistical and protocol-aware attributes for each bidirectional network flow.
Approach

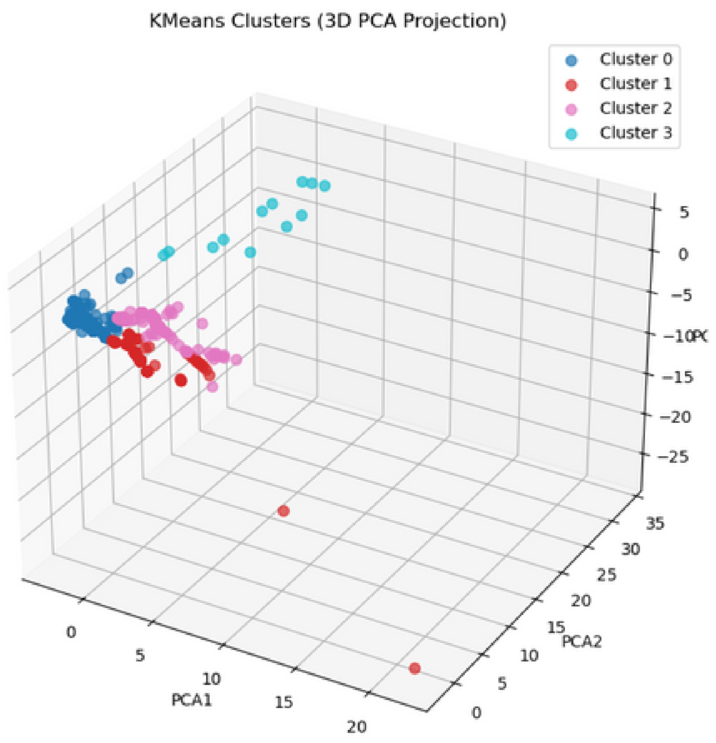
1. Packet Reading – Each .pcap file was read using scapy.rdpcap().
2. Flow Identification – Flows were defined by a 5-tuple: (Source IP, Destination IP, Source Port, Destination Port, Protocol), with reverse traffic aggregated into the same flow record.
3. Direction Separation – Packets were categorized as forward or backward to capture asymmetric behaviors.
4. Feature Computation – For each flow, statistical metrics were calculated, including:
    - Packet size stats: sum, mean, max, min, standard deviation
    - Inter-arrival times (IAT): mean, max, min, std, total
    - Flow duration, packets per second, bytes per second
    - TCP flag counts: FIN, SYN, RST, PSH, ACK, URG, CWE, ECE
    - Active/idle time statistics
    - Bulk transfer stats: average bytes/bulk, packets/bulk, bulk rate
    - Header length, segment size, and down/up ratio
5. Data Output – The resulting features were compiled into a Pandas DataFrame, where each row represents a single flow.

This process transforms unstructured packet captures into a structured, feature-rich dataset suitable for downstream machine learning or AI-based intrusion detection. From approximately 20,000 raw packets, the parsing stage produced 2,087 flow records with 84 features each. The dataset is unlabeled at this stage, meaning no ground-truth classification of flows as malicious or benign is included yet. However, it provides a comprehensive representation of both traffic behavior and protocol-level indicators, forming a solid foundation for subsequent labeling and AI model training to detect malicious activity effectively.

Labeling Approach 1: Clustering-Based Label Assignment

To generate initial labels for the unlabeled flow dataset, an unsupervised clustering approach was applied. The steps were as follows:

1. Feature Selection – A subset of key flow-level features was chosen based on their relevance to traffic behavior and protocol characteristics. Selected features included:
    - **Flow statistics**: Flow Bytes/s, Flow Packets/s, Flow Duration
    - **Packet size metrics**: Min Packet Length, Max Packet Length, Packet Length Mean, Fwd Packet Length Max/Min/Mean, Bwd Packet Length Mean/Max, Average Packet Size
    - **Temporal metrics**: Flow IAT Max, Fwd IAT Max, Idle Mean
    - **Traffic ratio**: Down/Up Ratio
2. Scaling – Features were standardized using StandardScaler to normalize their ranges and improve clustering performance.
3. Clustering Algorithm – K-Means clustering was applied with **k = 4** to partition flows into four groups based on similarity in the selected features.



The K-Means clustering produced four distinct groups with a silhouette score of 0.651, indicating reasonably well-separated clusters. Based on feature patterns and domain knowledge, the clusters were interpreted as follows:

- Cluster 0: Normal background traffic, considered **benign**.
- Cluster 1: High-volume download or streaming flows, usually **benign**, representing **bulk traffic.**
- Cluster 2: Upload-heavy or asymmetric flows, likely **malicious**, possibly indicating data exfiltration or command-and-control communication.
- Cluster 3: Very long, idle, or tiny flows, likely **malicious**, possibly representing **scanning**, beaconing, or probing activity.

These interpretations are based on my own understanding of the traffic characteristics and provide a preliminary labeling for the previously unlabeled dataset.

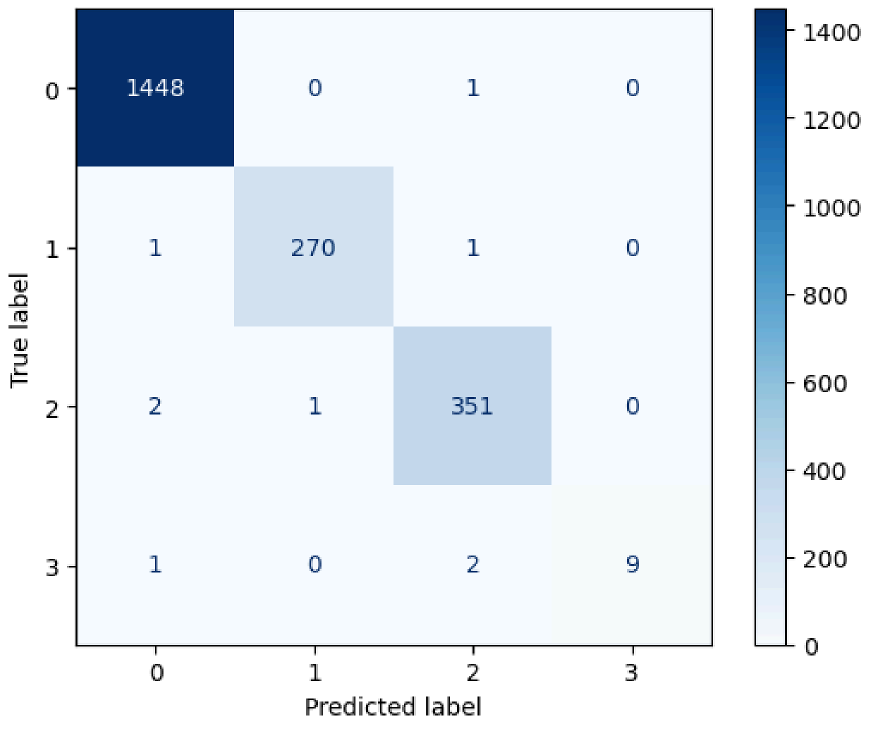
Note: The 3D PCA plot looks messy due to dimensionality reduction, which is expected.

Labeling Approach 2: Pre-Trained Model Prediction

The second labeling method used a pre-trained machine learning model to predict labels for the flow dataset. Trained on the CIC-IDS-2017 dataset, the model achieved 0.99 accuracy on its test set and provides an automated way to assign labels based on learned intrusion patterns. However, the resulting labels are highly imbalanced and not ideal for ML training.

Training a Model Using Cluster-Based Labels

A **Random Forest** classifier was trained using the cluster-based labels. K-fold cross-validation (k=5) produced scores of [0.9976, 0.9976, 0.9952, 0.9976, 0.9904], with a **mean accuracy of 0.996**. The model achieved **precision of 0.9957**, **recall of 0.9957**, and an **F1 score of 0.9956**, demonstrating strong performance in learning the patterns represented by the cluster labels.



Conclusion

The final AI engine can be developed around the Random Forest model trained on the cluster-based labels. Using a larger dataset—either more flows or additional PCAP captures—would help build a stronger, more robust model. Additionally, having a balanced and fully labeled dataset would improve model performance and reliability. While the current clusters provide useful preliminary labels, some could be interpreted more accurately with deeper traffic analysis, further enhancing the quality of the AI engine.