Developing a Network Monitoring Tool

Charles Bowen-Rayner

Earlham College

Abstract

Machine learning is gaining prominence for network monitoring, yet current tools are often complex to understand and use. This project attempts to address this by developing an algorithm for network anomaly detection using libpcap and the random forest algorithm. This approach provides real-time anomaly detection by analyzing past network traffic from real-life datasets and employing machine learning techniques. Through various tested methods, the effectiveness of identifying various network anomalies will be analyzed. This study will highlight the potential of integrating libpcap and machine learning for scalable and adaptable network security solutions, contributing to improved threat detection in modern computing environments.

Dataset

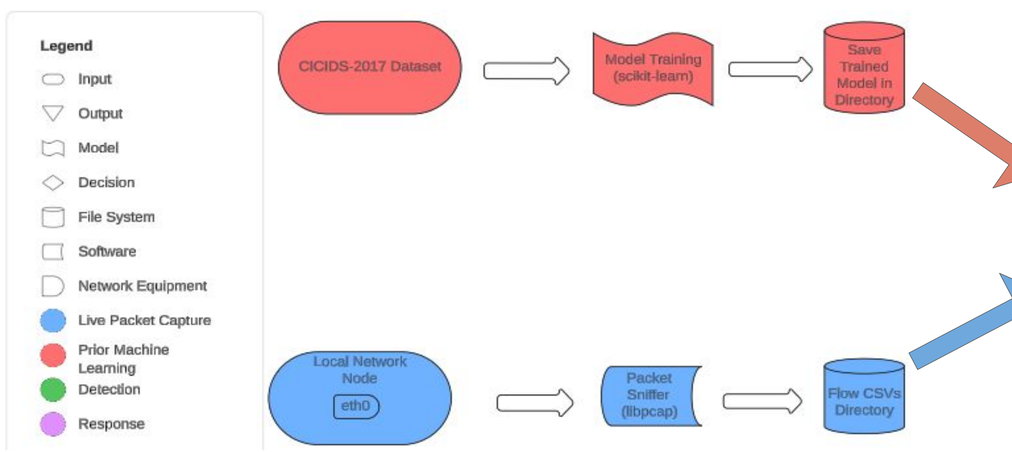
Comprehensive network intrusion detection dataset created by the Canadian Institute for Cybersecurity (CICIDS-2017), containing realistic network traffic data for various attack scenarios and benign behavior

- For this project the 15 classes were combined into 9 classes
  - Benign, Dos, DDos, Port Scan, Brute Force, Web Attack, Bot, Infiltration, Heartbleed

- Addressing class imbalance problem
  - Applied SMOTE to synthetically balance the dataset based on targeted sampling proportions
  - Next, assigned higher training weights to minority classes

Model Training

- Selected Random Forest (RF) classifier research showing strong performance for intrusion detection tasks with the CICIDS2017 dataset
- Feature Selection
  - Pearson Correlation Matrix: Reduced feature set from 71 to 32 by filtering out highly correlated features
  - Recursive Feature Elimination (RFE): Selected 10 most informative features
- Hyperparameter Optimization:
  - Ran RF model with the 10 selected features choosing settings with best accuracy and minimal complexity
- Final mean cross-validation accuracy: **0.96833**
- Test set accuracy: **0.97458**

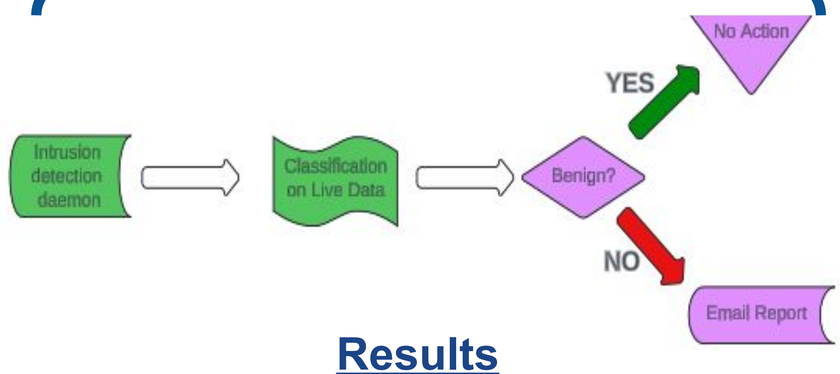


Packet Capture

- Used libpcap in C to monitor network traffic on two nodes from a CS cluster which were designated as source and destination
- Traffic Generation
  - iperf3 for benign UDP/TCP traffic
  - Metasploit and Nmap for malicious traffic
- Flow-Based Monitoring Setup:
  - Packets were aggregated into flows using timeouts: 600 seconds for UDP and, for TCP, either 600 seconds or a FIN packet
- CSV Logging
  - The 10 selected features from model training were captured from the packets and stored in a .csv file to be sent for detection

Detection and Response

- Flow Detection
  - A Python daemon continuously monitors a directory for new flow files.
- Trigger Script Execution
  - If a new flow is detected, the daemon passes it as an argument to a bash script
- Malicious Traffic Classification
  - The classification script runs the flow data through a trained machine learning model.
  - The model analyzes the flow features and outputs a classification result
- Notification System
  - If the flow is classified as malicious:
    - An email notification is automatically sent containing the flow data, time-stamp and classification



Results

During live testing, it became evident that the model had difficulty accurately predicting the generated flows. In certain instances, it incorrectly labeled them as benign, while in others, it misclassified the attacks due to their features being similar. To address the benign misclassification, I implemented a threshold for the probability of the second-highest predicted class. This threshold was determined through multiple test iterations, resulting in the most accurate predictions within the testing environment.

Future Work

Give the packet capture tool the ability to identify common attack patterns which will allow it to more accurately predict DoS, Port Scan and others. On top of that, I also want to fine-tune the model by labelling attack flows I created to see if it improves real world performance.