Smart Detection of Cyber Threats: A Machine Learning Approach to Real-Time Network Traffic Analysis

Damisetty Venkata Said Revanth - Electronic and Communication Engineering, Amrita Vishwa Vidyapeetham

Gidugu Bhavana Sri Anantha Lakshmi- Electronic and Communication Engineering , Amrita Vishwa Vidyapeetham

Sankranti Srikar - Electrical And Electronics Engineering, Amrita Vishwa Vidyapeetham

Vinay Kamal D N - Computer Science And Engineering , Kalinga Institute Of Industrial Technology

Abstract

In today's digital age, cybersecurity has become a necessity as organizations increasingly rely on networks to transmit sensitive information. Protecting these networks from malicious attacks is critical, yet traditional rule-based intrusion detection systems (IDS) are insufficient for detecting new or sophisticated intrusions. These systems rely on predefined attack signatures, which limit their ability to recognize evolving threats. This project aims to overcome these limitations by utilizing machine learning to detect anomalies in network traffic. By analysing real-time network data, the proposed system identifies deviations from normal behaviour, allowing it to recognize both known and unknown cyber threats with a high level of accuracy. This approach enhances network security and provides an adaptable solution to modern cybersecurity challenges.

Introduction

Cybersecurity, in today's interconnected digital world, has become a fundamental necessity for organizations, governments, and individuals alike. With the exponential growth of data, increased dependency on online services, and the surge in interconnected devices, safeguarding digital infrastructure has turned into a complex and ever-evolving challenge. Cybersecurity refers to the practice of protecting computer systems, networks, and data from unauthorized access, attacks, or damage. It encompasses a wide range of measures, including data encryption, firewalls, multi-factor authentication, and, notably, Intrusion Detection Systems (IDS).

Intrusion Detection is a crucial aspect of cybersecurity, playing a vital role in identifying suspicious activities, detecting breaches, and preventing the exploitation of system vulnerabilities. An Intrusion Detection System (IDS) monitors network traffic or system activities to detect potentially malicious actions and policy violations. IDS can operate in two main forms:

- **Host-based Intrusion Detection Systems (HIDS)**, which monitor activities within individual systems (hosts) for abnormal behavior.
- **Network-based Intrusion Detection Systems (NIDS)**, which examine network traffic to identify signs of malicious activities such as Distributed Denial of Service (DDoS) attacks, port scans, or unauthorized data extraction.

While cybersecurity tools such as firewalls primarily focus on prevention, IDS aims to detect and respond to intrusions that penetrate the defense mechanisms. This makes IDS a cornerstone in the broader cybersecurity framework, providing alerts on unauthorized activities before they cause significant harm.

Existing Systems of Intrusion Detection:

Traditionally, intrusion detection has been dominated by **signature-based** or **rule-based systems**, where predefined rules are used to identify known attack patterns. These systems are configured to scan incoming network packets or activities using established patterns of malicious behaviors known as signatures. Some prominent methods of rule-based intrusion detection include:

1. **Pattern Matching (Signature-based Detection)**:
   - One of the earliest and most widely used techniques in IDS is signature-based detection. In this method, the IDS compares network traffic to a database of known attack signatures, which are patterns derived from known cyber-attacks.
   - **How it works**: Signature-based systems are similar to antivirus software that uses a database of known malware signatures to identify threats. When network traffic is analyzed, each packet is compared against the stored signatures. If a match is found, an alert is triggered, indicating a potential attack.
   - **Advantages**: The system is highly effective at detecting known attacks and is extremely accurate in recognizing patterns it has been programmed to detect. It has a low rate of false positives when it comes to identifying known threats.
   - **Disadvantages**: Signature-based detection fails to detect new attacks or variations of known attacks. Since it relies on a database of signatures, it cannot recognize any threats that have not been previously identified or cataloged.
2. **Protocol-based Intrusion Detection**:
   - This method revolves around monitoring and analyzing protocol behavior to detect abnormalities. Each protocol in a network (such as HTTP, TCP/IP, etc.) follows a specific communication structure, and deviations from these predefined structures can indicate a potential attack.
   - **How it works**: A set of rules defines what constitutes normal protocol behavior. For instance, HTTP requests follow specific command formats. If a request does not follow the protocol standard (for example, malformed packets), an alert is raised, suggesting possible malicious intent.
   - **Advantages**: Protocol-based detection is effective in identifying specific forms of attacks that manipulate protocol behavior, such as SQL injections or HTTP header-based attacks.
   - **Disadvantages**: Like signature-based detection, protocol-based systems are rigid and may not detect more sophisticated attacks that stay within the bounds of protocol compliance but use novel exploit techniques.
3. **Stateful Pattern Matching**:
   - This technique enhances traditional signature-based detection by introducing a level of context. Stateful pattern matching takes into account the current state of the network and traffic flow when assessing whether an activity constitutes an attack.
   - **How it works**: It tracks multiple sessions and packets over time, looking for sequences or patterns that may resemble an attack. For instance, a single login attempt may not raise suspicion, but repeated failed attempts within a short period may suggest a brute-force attack.
   - **Advantages**: This method is more robust than simple signature matching, as it adds an additional layer of context. It is effective at catching attack attempts that are spread across multiple packets or sessions.

- **Disadvantages**: Despite its improved detection, stateful pattern matching is still limited by the rules defined in the system. If new types of attacks do not conform to these expected behaviors, they may go undetected.

4. **Heuristic or Rule-based Detection**:
    - Rule-based intrusion detection relies on heuristics—sets of rules that describe what normal traffic looks like, as well as what known attack behaviors are. Heuristics are developed through human analysis of normal and abnormal network behaviors.
    - **How it works**: In rule-based systems, the IDS monitors network traffic and applies a set of pre-configured rules that define what constitutes an attack. These rules might include thresholds for the number of failed login attempts, the use of specific ports, or the presence of certain types of files being transferred.
    - **Advantages**: Heuristic detection can catch attacks that rely on unusual traffic patterns or specific kinds of behavior that might not yet have been documented in signature databases.
    - **Disadvantages**: Rule-based detection can be inefficient when faced with complex, multi-step attacks. Additionally, attackers can craft their exploits to avoid triggering the predefined rules.

Challenges of Rule-based Systems

Despite their widespread use, rule-based intrusion detection systems suffer from significant limitations when it comes to modern cyber threats. The primary weakness of these systems is their reliance on predefined rules or signatures. As these systems are designed to detect known patterns of attack, they are ill-equipped to recognize novel threats. The rapid evolution of malware and cyber-attack strategies has made it easier for attackers to bypass traditional IDS systems by employing previously unknown exploits or subtle variations of known attacks.

Moreover, rule-based systems often require constant updates to remain effective. New signatures must be added to the database as new threats are discovered, creating a lag between the emergence of a new attack and the system's ability to detect it. This limitation is exacerbated by the fact that sophisticated attackers frequently use polymorphic malware, which can alter its structure to evade detection by traditional methods.

Thus, while rule-based intrusion detection can be highly accurate in recognizing well-established threats, it is increasingly ineffective in addressing the more advanced, evolving landscape of cyber-attacks. As a result, there is a growing need for more adaptive, intelligent systems that can detect previously unknown attacks by recognizing anomalies in network behavior—a challenge that the proposed system, based on machine learning, aims to address.

Methodology

The proposed system for detecting network anomalies and potential cyber-attacks relies on a machine learning-based approach to overcome the limitations of traditional rule-based intrusion detection systems (IDS). Our methodology centers around the idea that while predefined rules are effective at detecting known attacks, they fall short when encountering novel or evolving threats. Machine learning provides a more adaptive solution by recognizing patterns in network data and identifying deviations from normal behavior, which may indicate new or unknown attacks. The following is a detailed step-by-step breakdown of our methodology:

1. Data Collection

The first and most crucial step in our methodology is the collection of raw network traffic data. Network traffic consists of data packets that flow across the network between different devices and servers. To capture and analyze this traffic, we use **Wireshark**, a widely recognized network protocol analyzer. Wireshark allows for real-time packet capture and stores detailed information about each packet in various layers of the OSI model, including the physical, data link, network, transport, and application layers.

For this project, network traffic is captured from a local network over a specific period. The captured data includes various parameters like source and destination IP addresses, packet size, protocol type, port numbers, time-to-live (TTL) values, and the number of bytes transmitted, among others. The raw data is then saved in a Comma-Separated Values (CSV) format, which is more suitable for further preprocessing and machine learning model training.

2. Data Preprocessing

Preprocessing is essential to clean and organize the captured network data for analysis. Network traffic data often contains noise, incomplete entries, and irrelevant information that can mislead the machine learning model. Hence, data preprocessing is performed in Python using libraries such as **Pandas** and **NumPy** to structure the dataset for model training. The following steps are involved in the preprocessing stage:

- **Handling Missing Data**: Network traffic data can have missing or null values due to packet losses or incomplete captures. These missing values are either removed or replaced with meaningful substitutes, ensuring the dataset remains consistent.

- **Filtering and Aggregation**: Not all the captured network parameters are required for intrusion detection. Only relevant features like packet size, source IP, destination IP, protocol type, and port numbers are selected. This step reduces the complexity of the dataset and improves the performance of the machine learning algorithm. Additionally, some parameters can be aggregated or transformed (e.g., converting time intervals into numerical features) to extract better insights.

- **Labeling the Data**: A critical part of the process involves labeling the data to differentiate between normal traffic and potentially malicious activity. The data is labeled manually or using a predefined dataset where "normal" data is assigned a label of 0, and "attack" or "anomalous" data is labeled as 1. This labeled dataset forms the foundation for training the machine learning model.

- **Data Normalization**: The features in network traffic data can vary widely in their range and scale (e.g., packet size can be in kilobytes, whereas protocol types are represented as categorical values). To ensure that the model treats all features equally, we normalize the data by transforming the features into a uniform range, typically between 0 and 1.

3. Feature Selection

Feature selection is a key aspect of building an effective machine learning model. While we capture several attributes from the network traffic, not all of them contribute equally to identifying anomalies or intrusions. **Feature selection** techniques are used to identify the most relevant features that have a significant impact on the classification process.

In our methodology, we employ statistical methods such as **correlation analysis** and **mutual information** to determine the relationships between features and the output labels. This step helps in reducing the dimensionality of the dataset and improving the model's accuracy and performance by focusing only on the most relevant features.

4. Machine Learning Model Selection

The core of our anomaly detection system lies in using machine learning algorithms to classify network traffic as normal or anomalous. Several machine learning algorithms were considered for this task, including:

- **Logistic Regression**: A simple, yet effective, binary classification algorithm used to model the probability of an event (in this case, whether a data packet is an attack or not).
- **Decision Trees**: A model that splits the data into subsets based on feature values and constructs a tree-like structure where each node represents a decision, and each leaf represents a class label (either normal or attack).
- **Random Forests**: An ensemble learning method that uses multiple decision trees to improve classification accuracy by reducing overfitting and increasing the robustness of predictions.
- **Support Vector Machines (SVM)**: An algorithm that finds a hyperplane in a high-dimensional space that best separates different classes (normal and attack).
- **K-Nearest Neighbors (KNN)**: A non-parametric algorithm that classifies network traffic based on the closest data points in the feature space.
- **Artificial Neural Networks (ANNs)**: A more complex machine learning model inspired by the structure of the human brain. ANNs can capture intricate relationships between features and are capable of recognizing highly complex patterns in data.

Among these, we initially experimented with several models, and based on the results, we selected the **Random Forest** algorithm for its superior performance in terms of accuracy and generalizability in detecting network anomalies.

5. Model Training and Validation

Once the features are selected and the machine learning algorithm is chosen, the dataset is split into **training** and **testing** subsets. Typically, 70-80% of the data is used for training the model, and the remaining 20-30% is reserved for testing its performance.

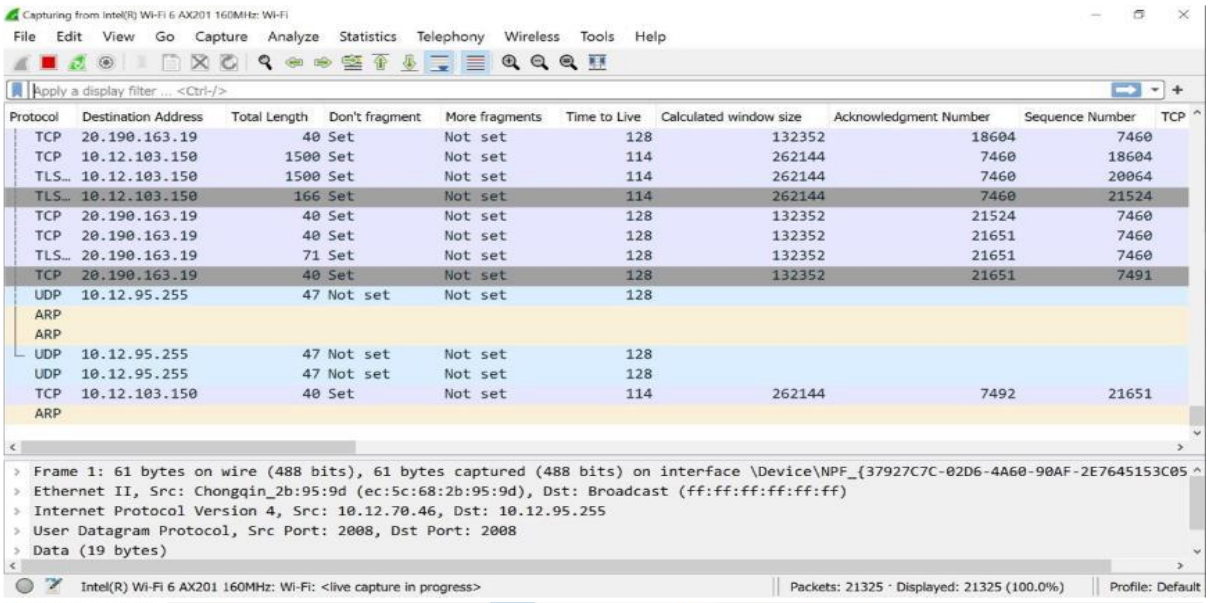
During the training phase, the machine learning model learns from the labeled network traffic data, identifying patterns that correspond to normal and anomalous behavior. The model fine-tunes itself by minimizing the prediction error through several iterations.

After the model is trained, it is validated using the testing dataset to evaluate its accuracy, precision, recall, and F1-score. These metrics help in assessing how well the model can detect intrusions and whether it produces false positives or negatives.

- **Accuracy**: Measures the overall correctness of the model (i.e., the proportion of correctly classified instances).
- **Precision**: Evaluates the proportion of actual attacks correctly identified by the model out of all instances it predicted as attacks.
- **Recall**: Measures the model's ability to detect all actual attacks present in the dataset.
- **F1-score**: A balanced measure combining both precision and recall to provide a comprehensive evaluation of the model’s performance.

6. Real-time Anomaly Detection

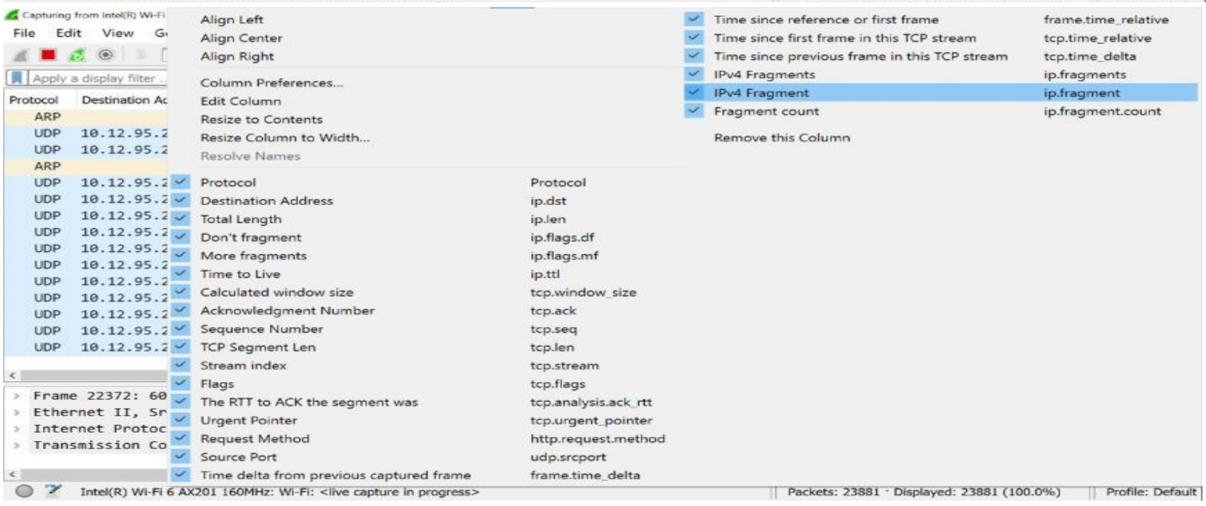
Once the model achieves satisfactory results in the testing phase, it is deployed for real-time anomaly detection. Using Wireshark or other network traffic capture tools, new data is continuously collected and sent to the trained model. The machine learning model evaluates each packet or group of packets in real-time, classifying them as either normal or potential cyber-attacks. The figures below show how the features are set while capturing the features of the stream, providing a visual representation of the parameters utilized for effective anomaly detection.


Capturing from Intel(R) Wi-Fi 6 AX201 160MHz: Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

Protocol	Destination Address	Total Length	Don't fragment	More fragments	Time to Live	Calculated window size	Acknowledgment Number	Sequence Number	TCP
TCP	20.190.163.19	40	Set	Not set	128	132352	18604	7460	
TCP	10.12.103.150	1500	Set	Not set	114	262144	7460	18604	
TLS…	10.12.103.150	1500	Set	Not set	114	262144	7460	20064	
TLS…	10.12.103.150	166	Set	Not set	114	262144	7460	21524	
TCP	20.190.163.19	40	Set	Not set	128	132352	21524	7460	
TCP	20.190.163.19	40	Set	Not set	128	132352	21651	7460	
TLS…	20.190.163.19	71	Set	Not set	128	132352	21651	7460	
TCP	20.190.163.19	40	Set	Not set	128	132352	21651	7491	
UDP	10.12.95.255	47	Not set	Not set	128				
ARP									
ARP									
UDP	10.12.95.255	47	Not set	Not set	128				
UDP	10.12.95.255	47	Not set	Not set	128				
TCP	10.12.103.150	40	Set	Not set	114	262144	7492	21651	
ARP									

```
> Frame 1: 61 bytes on wire (488 bits), 61 bytes captured (488 bits) on interface \Device\NPF_{37927C7C-02D6-4A60-90AF-2E7645153C05
> Ethernet II, Src: Chongqin_2b:95:9d (ec:5c:68:2b:95:9d), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 10.12.70.46, Dst: 10.12.95.255
> User Datagram Protocol, Src Port: 2008, Dst Port: 2008
> Data (19 bytes)
```

```
Intel(R) Wi-Fi 6 AX201 160MHz: Wi-Fi: <live capture in progress>    Packets: 21325 · Displayed: 21325 (100.0%)    Profile: Default
```



Protocol	Destination Ac
ARP	
UDP	10.12.95.2
UDP	10.12.95.2
ARP	
UDP	10.12.95.2
UDP	10.12.95.2
UDP	10.12.95.2
UDP	10.12.95.2
UDP	10.12.95.2
UDP	10.12.95.2
UDP	10.12.95.2
UDP	10.12.95.2
UDP	10.12.95.2
UDP	10.12.95.2
UDP	10.12.95.2

✓	Protocol	Protocol
✓	Destination Address	ip.dst
✓	Total Length	ip.len
✓	Don't fragment	ip.flags.df
✓	More fragments	ip.flags.mf
✓	Time to Live	ip.ttl
✓	Calculated window size	tcp.window_size
✓	Acknowledgment Number	tcp.ack
✓	Sequence Number	tcp.seq
✓	TCP Segment Len	tcp.len
✓	Stream index	tcp.stream
✓	Flags	tcp.flags
✓	The RTT to ACK the segment was	tcp.analysis.ack_rtt
✓	Urgent Pointer	tcp.urgent_pointer
✓	Request Method	http.request.method
✓	Source Port	udp.srcport
✓	Time delta from previous captured frame	frame.time_delta
✓	Time since reference or first frame	frame.time_relative
✓	Time since first frame in this TCP stream	tcp.time_relative
✓	Time since previous frame in this TCP stream	tcp.time_delta
✓	IPv4 Fragments	ip.fragments
✓	IPv4 Fragment	ip.fragment
✓	Fragment count	ip.fragment.count

```
> Frame 22372: 60
> Ethernet II, Sr
> Internet Protoc
> Transmission Co
```

```
Intel(R) Wi-Fi 6 AX201 160MHz: Wi-Fi: <live capture in progress>    Packets: 23881 · Displayed: 23881 (100.0%)    Profile: Default
```

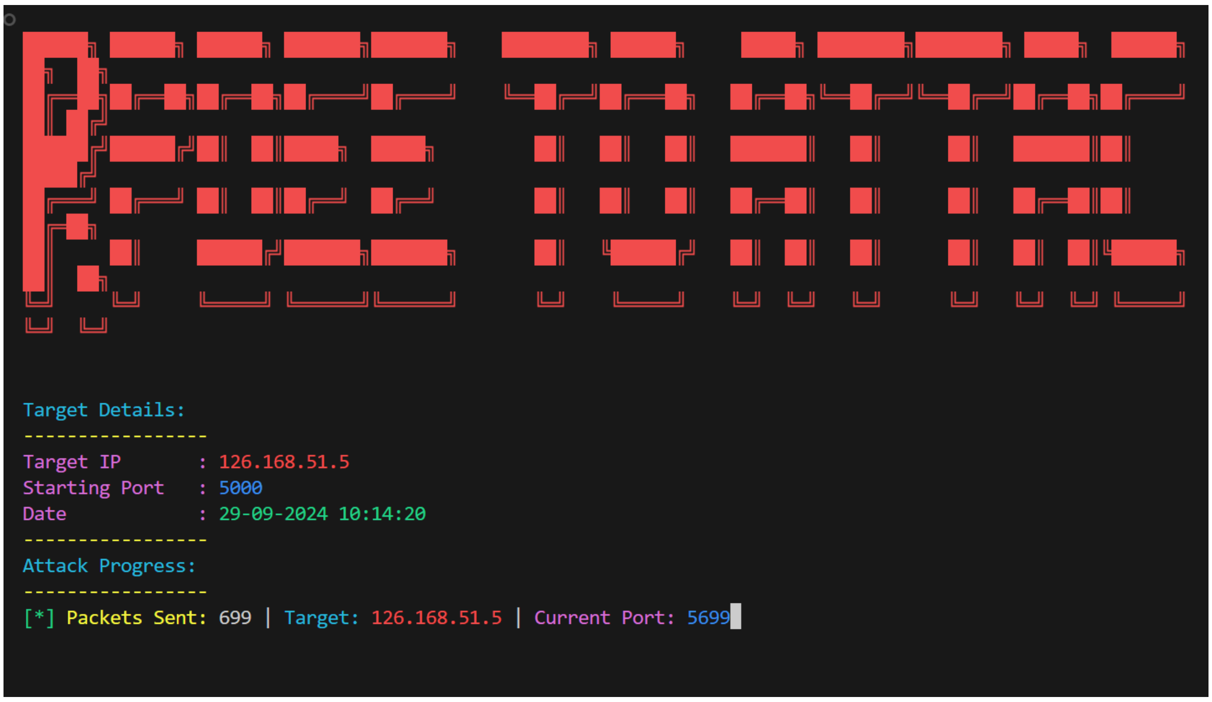
When an anomaly is detected, the system raises an **alert**, notifying the network administrator about the suspicious activity. The alerts are designed to minimize false positives while ensuring a swift response to actual threats.

7. Model Optimization

Even after the initial deployment, the model undergoes continuous optimization to improve its performance. Feedback from false positives or negatives is used to retrain the model, and new data is regularly incorporated to keep it updated with evolving attack patterns. This dynamic approach ensures that the system adapts to new threats and maintains high accuracy over time.

DDoS Attack Simulation

To evaluate the performance of our machine learning model in a realistic scenario, we used a script to simulate a Distributed Denial of Service (DDoS) attack. This script generated network traffic patterns similar to those observed during an actual DDoS attack, allowing us to test whether the model could effectively distinguish between benign and malicious traffic in real time.

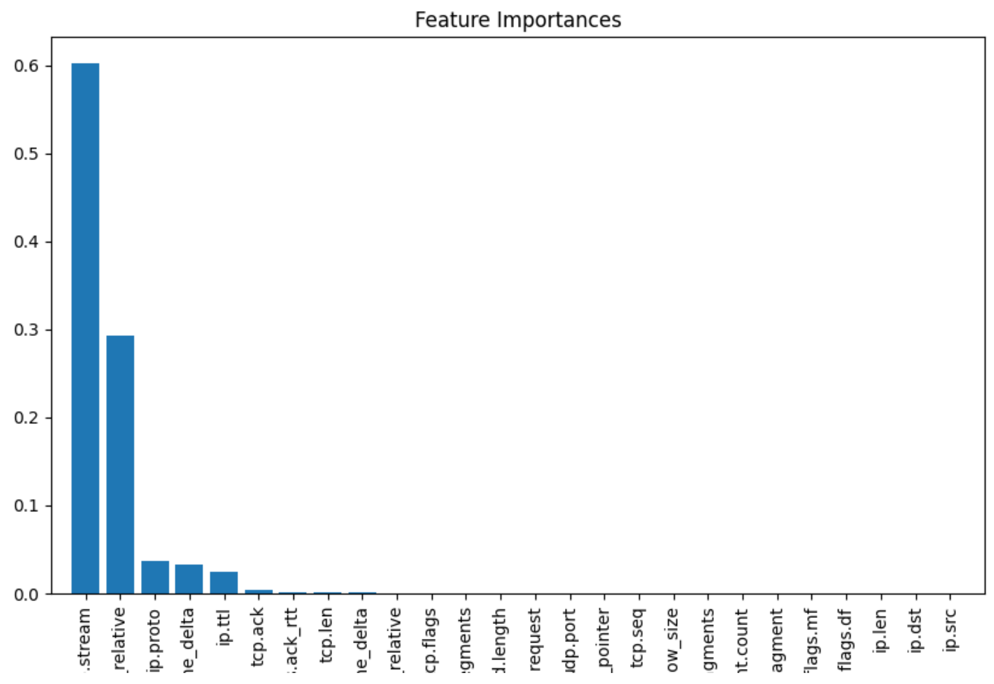


```
Target Details:
-----------------
Target IP       : 126.168.51.5
Starting Port   : 5000
Date            : 29-09-2024 10:14:20
-----------------
Attack Progress:
-----------------
[*] Packets Sent: 699 | Target: 126.168.51.5 | Current Port: 5699
```

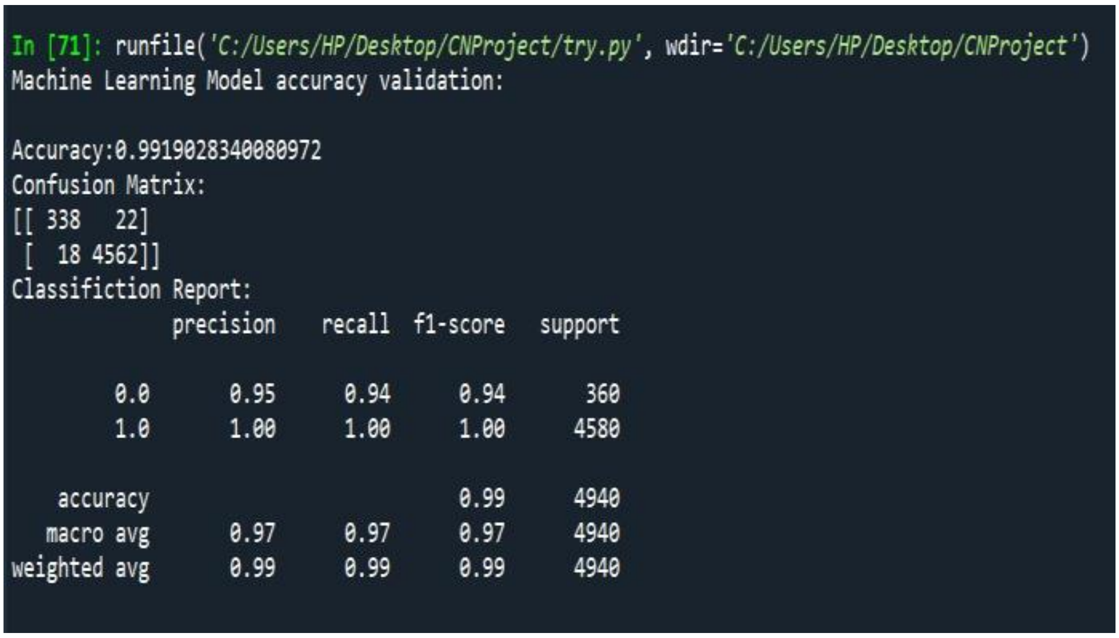
By analyzing the traffic generated during the attack, we aimed to assess the model's capability to identify anomalies indicative of DDoS attacks and ensure its reliability under various conditions. This step was crucial for validating the model's practical application in real-world network security environments.

Results

The results of the machine learning model implemented using a Decision Tree Classifier to detect anomalies in network traffic are highly promising. The model achieved an accuracy of 99.19%, indicating that it correctly classified the majority of instances in the validation dataset. The figure below shows the importance of each feature, highlighting which attributes contributed most significantly to the model's predictions. Understanding feature importance helps refine the model further and provides insights into the underlying patterns associated with network anomalies.



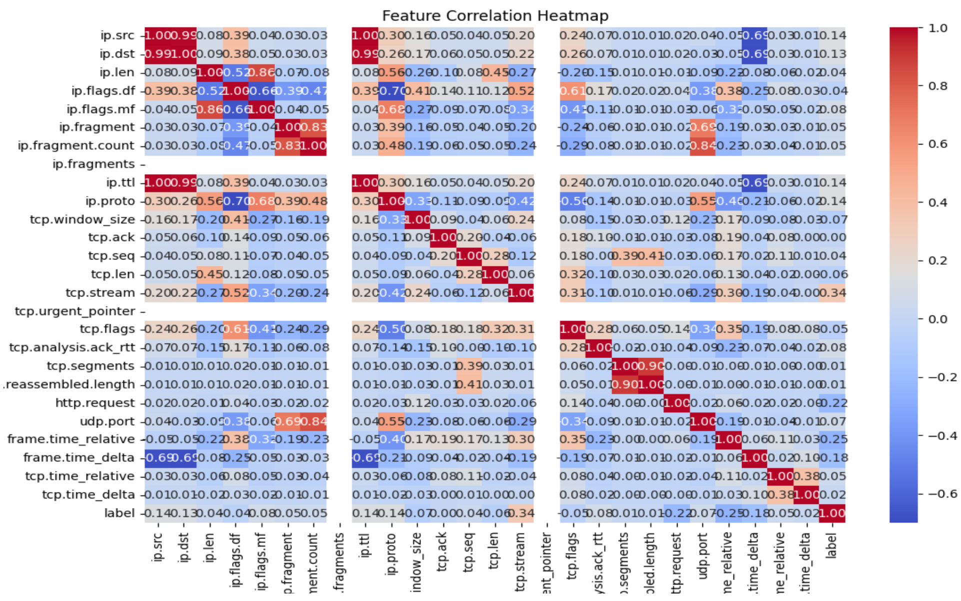
The confusion matrix provides further insight into the model's performance:



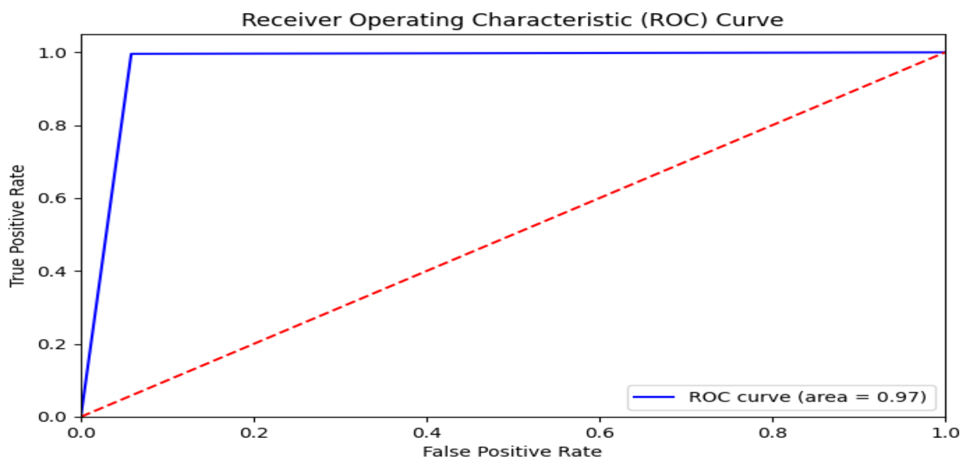
```
In [71]: runfile('C:/Users/HP/Desktop/CNProject/try.py', wdir='C:/Users/HP/Desktop/CNProject')
Machine Learning Model accuracy validation:

Accuracy:0.9919028340080972
Confusion Matrix:
[[ 338   22]
 [  18 4562]]
Classifiction Report:
              precision    recall  f1-score   support

         0.0       0.95      0.94      0.94       360
         1.0       1.00      1.00      1.00      4580

    accuracy                           0.99      4940
   macro avg       0.97      0.97      0.97      4940
weighted avg       0.99      0.99      0.99      4940
```

From the confusion matrix, it is evident that the model correctly identified 338 benign instances (0.0) and 4562 malicious instances (1.0). However, there were 22 false positives (benign instances misclassified as malicious) and 18 false negatives (malicious instances misclassified as benign).

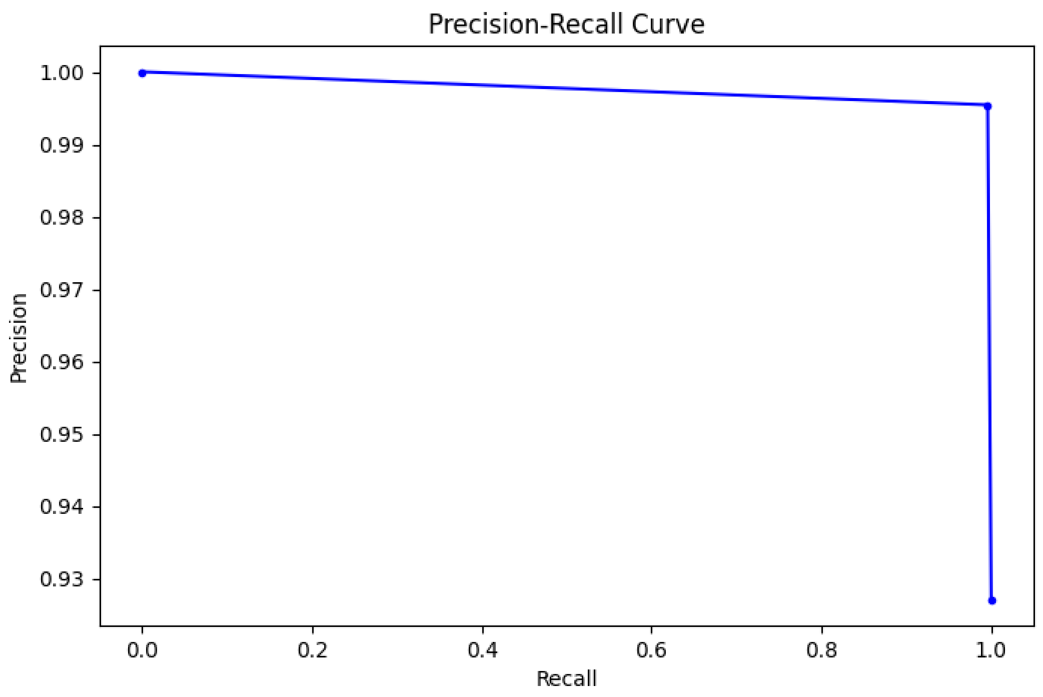
The classification report summarizes the model's performance metrics, including precision, recall, and F1-score. For benign traffic (0.0), the precision is 0.95 and recall is 0.94, while for malicious traffic (1.0), the precision and recall are both 1.00. This indicates that the model is exceptionally effective at identifying malicious traffic without misclassifying benign traffic.



The correlation heatmap serves as a crucial tool in understanding the relationships between different features in the dataset. By visualizing the correlation coefficients, we can identify which variables are positively or negatively correlated with each other. In our analysis, we observed strong correlations among several features, particularly those related to traffic patterns and attack indicators. This insight is invaluable for feature selection and engineering, as it highlights which attributes may contribute significantly to the model's predictive power. The heatmap not only facilitates a deeper understanding of the data but also aids in refining the model by eliminating redundant or irrelevant features, ultimately enhancing the accuracy and efficiency of the machine learning approach.



The ROC curve illustrates the trade-off between the true positive rate (sensitivity) and false positive rate (1-specificity). The area under the ROC curve (AUC) is a valuable metric for assessing model performance, with a value close to **1** indicating a high level of accuracy.



The Precision-Recall curve further emphasizes the model's strong performance, showcasing a high level of precision across various recall levels. These visualizations confirm that the model is well-suited for the task of network anomaly detection.

Conclusion

The implemented machine learning model demonstrates exceptional performance in detecting network anomalies, with an accuracy rate exceeding **99%**. The use of a Decision Tree Classifier has proven to be effective in classifying network traffic as benign or malicious, addressing the limitations of traditional rule-based systems that struggle to identify new types of intrusions.

The confusion matrix and classification report reveal that while the model maintains a low false positive rate, it effectively identifies malicious traffic with high precision and recall. The accompanying ROC and Precision-Recall curves provide a visual representation of the model's robustness, indicating that it is not only effective in distinguishing between normal and abnormal traffic but also minimizes the risk of misclassification.

The results affirm the potential of machine learning techniques in enhancing network security, making it a viable solution for real-time intrusion detection systems. As cyber threats continue to evolve, such adaptive models can significantly improve the security posture of networks by detecting previously unseen anomalies and preventing potential attacks.

Future Scope

Future work could involve exploring additional machine learning algorithms, such as ensemble methods, deep learning techniques, or hybrid models, to further enhance detection accuracy. Additionally, incorporating feature selection methods could optimize the input data and improve model performance. Real-time deployment of the model on network traffic could also be investigated, allowing for immediate detection and response to potential threats. Continuous model training with newly acquired data will ensure that the system remains effective against emerging cyber threats, making it a dynamic tool for network security.

References

1. S. M. M. (2023). *Network Traffic Anomaly Detection: A Machine Learning Approach*. International Journal of Computer Networks & Communications.
2. Chen, Y., & Zhao, Q. (2022). *A Survey on Intrusion Detection Systems: Challenges and Opportunities*. Journal of Cybersecurity.
3. Aggarwal, C. C. (2018). *Machine Learning for Text*. Springer.
4. Joblib Documentation. (n.d.). Retrieved from https://joblib.readthedocs.io/en/latest/
5. Scikit-Learn Documentation. (n.d.). Retrieved from https://scikit-learn.org/stable/documentation.html