A MACHINE LEARNING-BASED PACKET SNIFFER FOR DETECTION AND CLASSIFICATION OF THE DENIAL OF SERVICE ATTACK PACKETS AT THE NETWORK LAYER

Kipkorir Peacemark

PA201/S/17787/22

A Research Thesis Submitted in Partial Fulfillment for the Requirement of the Award of

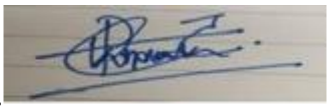
The Master of Science Degree in Information Technology of

Kirinyaga University.

May, 2025.

DECLARATION

This research thesis is my original work and has not been presented for any degree award in any learning institution.

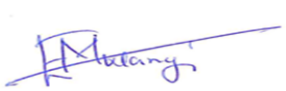
Sign …**... Date…11/6/2025…**

Kipkorir Peacemark

PA201/S/17787/22

APPROVAL

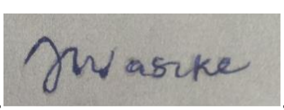
This research thesis has been submitted for examination with our approval as the student's supervisors.

Sign … **……Date……11th June, 2025..…**

1. Dr. Ephantus Mwangi, PhD

   School of Pure and Applied Sciences

   Kirinyaga University

Sign……**……Date………..11th June, 2025..………**

2. Dr. Jotham Wasike, PhD

   School of Pure and Applied Sciences

   Kirinyaga University

ACKNOWLEDGEMENT

I am eternally grateful to the Almighty God, whose infinite wisdom has guided me through every step of this academic journey, granting me strength, perseverance, and inspiration. I sincerely appreciate my lecturers Dr. Ephantus Mwangi and Dr. Jotham Wasike who continually give me insights on how to implement the research study, with guidelines correctly, and always insist on seeing that the research study is rightfully executed, from documentation to testing and implementation. I am also grateful to Appkings Solutions Limited for giving me ample time to attend studies.

DEDICATION

I dedicate this thesis to my wife and daughter; your understanding and encouragement has pushed me throughout this journey. To my beloved parents, your unwavering support, sacrifices, and encouragement have been the cornerstone of my achievements; I am forever grateful for your love and guidance. To my esteemed supervisors, your invaluable mentorship, insightful feedback, and unwavering dedication have shaped my scholarly endeavors and enriched my intellectual growth. To my dear friends, your camaraderie and encouragement have been a source of joy and motivation throughout this endeavor; your belief in me has fueled my determination to excel. This work stands as a testament to the profound impact each of you has had on my life and academic pursuits.

ABSTRACT

The research study was on modelling a packet sniffer utilizing machine learning techniques to identify denial of service (DOS) attack packets at the network layer of the OSI model. Cyber threats and attacks have continued to evolve in complexity and sophistication, posing significant risks to the network infrastructure and sensitive data's availability, confidentiality, and integrity. The necessity for sophisticated methods to improve network security is highlighted by the fact that conventional methods frequently fail to identify and mitigate these attacks. The overall purpose of the research study was to capture and interpret packets transmitted over a local area network to detect and capture the DOS threats within the Open Systems Interconnection Model (OSI) network layer. This layer is prone to several attacks for instance, denial-of-service, routing protocol attacks, Port scanning and enumeration, and fragmentation-based attacks. However, in this study, we delved into detecting and capturing the denial of service threats at the third layer of the OSI model in a local area network. Some examples of DOS attacks are UDP flood which sends a significant quantity UDP (User Datagram Protocol) packets to the targeted systems and thereby exhausting network resources, ICMP flood which transmits a significant quantity of Internet Control Message Protocol (ICMP) packets to overwhelm network devices, SYN flood which takes advantage of the TCP three-way hand-shake procedure by sending a lot of SYN requests without carrying out the necessary handshake, using server resources and blocking valid connections. Essential components extracted from Ethernet frames comprise TCP segments, ICMP packets, IPv4 packets, and associated flags. IPv4, a crucial protocol in Internet communication, enables routing and logical addressing, forming the Internet's backbone. The Internet Control Message Protocol (ICMP) facilitates error reporting and the interchange of operational information inside the Internet Protocol suite. There are header and data sections in a packet. Data bytes are sent plus a header that TCP added to the data to make up a TCP segment. Even though internet-based data transmission protocols have expanded, traditional network security measures are frequently insufficient to combat the dynamic environment of cyber threats that target networks used for data transfer. This deficiency emphasizes the need for innovative technologies to improve network security. Although antivirus programs, intrusion detection systems, and firewalls are crucial barriers against malicious attacks, they frequently fall short in detecting and halting more crafty attacks that evade their protection. The study sought to bridge this gap by providing an automated machine learning-based packet sniffer that can identify and categorize network risks. To address the research objectives thoroughly, experimental research methodology was used. This study employed the CICIDS2018 dataset. The LightGBM model was successfully trained and implemented for the task of detecting DoS attacks. We used the CICIDS2018 dataset, which provided labeled network traffic data containing both normal and attack (DoS) instances. The model was trained to classify traffic as either normal or a DoS attack based on various network features. The model's performance was evaluated using several metrics to demonstrate its ability to accurately detect threats at the network layer in a local area network including sensitivity, specificity, and accuracy. The AUC (Area Under the Curve) was particularly high, which indicated that the model was able to effectively differentiate between normal traffic and DoS attacks. Additionally, the F1-score, precision, and recall were balanced, suggesting that the model was capable of identifying attacks while minimizing false positives and false negatives. The model was successful in meeting its primary objective of detecting DoS attacks from network traffic. The performance metrics indicate that LightGBM is a strong candidate for the task, achieving a high AUC and a well-balanced F1-score. This suggested the model achieved good generalization capabilities, and it can effectively distinguish between normal traffic and DoS attack traffic in most cases. The main contribution of this work is the development of a LightGBM-based machine learning model for detecting DoS attacks using the CICIDS2018 dataset. The model's ability to classify network traffic as normal or malicious will aid in enhancing network security by automating the detection of such attacks in LANs. The model will henceforth serve as a foundational step for building more advanced intrusion detection systems, especially for environments where DoS attacks are prevalent.

Table of Contents

List of Tables

List of figures

Acronyms

NIC – Network Interface Card

ICMP - Internet Control Message Protocol

HTTP - Hypertext Transfer Protocol

IDS - Intrusion Detection Systems

NIDS - Network Intrusion Detection Systems

BTLSM - Bidirectional Long Short-term memory

SDAE - Stacked De-noising Auto-encoder

LSTM - Long Short-Term Memory

CNN - Convolutional Neural Network

ODP - Optimal Dual Paths

HSR - High-availability Seamless Redundancy

PMT - Pruning Multicast Traffic

EPL - Enhanced Port-Locking

RDP - Ring-based Dual Paths

DVP - Dual Virtual Paths

FPGA - Field Programmable Gate Arrays

DSRM - Design science research methodology

LightGBM – Light gradient boosting model

Abbreviations

KDD Cup - Knowledge Discovery and Data Mining Cup

CHAPTER ONE: INTRODUCTION

1.1 Introduction

This chapter provides a clear outline for the next chapters of the thesis by establishing the background, problem statement, suggested solution, objectives, rationale, assumptions made throughout the study, limitations, significance of the study, and scope of the research study.

1.2 Background of the Study

Network security is now a top priority for businesses of all sizes in the interconnected society of today. As networks continue to expand and become more complex, the potential for security breaches and cyber threats increases (Julian *et al*., 2014). To effectively protect network infrastructure and sensitive information, organizations must employ advanced techniques and technologies that go beyond traditional security measures. One such approach is the use of advanced packet sniffer analysis (Tariq *et al*., 2023).

Packet sniffers are tools that capture and analyze network traffic, allowing organizations to monitor and examine the data packets flowing through their networks. These tools provide valuable insights into network behavior, identifying potential vulnerabilities, anomalies, and malicious activities (Chiradeep, 2022). By analyzing packet-level information, Companies can obtain a extensive comprehension of the security posture of their network and make informed decisions to enhance network security.

Among the many threats facing these systems, Denial-of-Service (DoS) attacks remain one of the most disruptive and persistent. A DoS attack seeks to render a service or network unavailable to legitimate users by overwhelming it with malicious traffic, thereby degrading performance or causing complete service outages (Kumar & Saxena, 2020).

Traditionally, network administrators have relied on rule-based Intrusion Detection Systems (IDS) and firewalls to detect and block such malicious traffic. However, as attack techniques become more sophisticated and dynamic, static and signature-based detection methods struggle to adapt to unknown or evolving threats. This has led to the rise of Machine Learning (ML)-based solutions, which offer the ability to learn patterns from data and generalize beyond pre-defined attack signatures (Somula et al., 2021).

One crucial component in network security monitoring is packet sniffing, which involves capturing and analyzing packets traversing a network. Packet sniffers enable deep inspection of traffic and can be used to extract statistical features such as packet sizes, time intervals, and protocol types. When combined with machine learning, these features can be leveraged to detect anomalous behavior indicative of a DoS attack. However, implementing an effective detection system requires not only accurate modeling but also careful handling of large-scale, imbalanced, and often noisy data.

Recent datasets such as CICIDS2018, developed by the Canadian Institute for Cybersecurity, have addressed some of the limitations of earlier benchmarks by providing realistic, labeled network traffic including various types of DoS attacks. This dataset, along with tools like

CICFlowMeter for feature extraction, has enabled researchers to explore and develop data-driven security models that can better identify and classify malicious activities in network traffic (Sharafaldin et al., 2018).

Given these advancements, there is a growing need to design systems that not only detect DoS attacks at the network layer but also classify them into specific categories, enabling more targeted responses. This study proposes a machine learning-based packet sniffer that leverages flow-based features to detect and classify DoS attacks using algorithms such as LightGBM, which is known for its high speed and accuracy on large-scale datasets (Ke et al., 2017).

However, traditional packet sniffers often fall short in addressing the evolving and sophisticated nature of cyber threats. To overcome these limitations, advanced packet sniffers make utilization of cutting-edge technologies such as machine learning, artificial intelligence, and behavioral analytics. With the help of these cutting-edge capabilities, organizations can better prevent and mitigate cyberattacks by being able to quickly detect and address network security issues (Ramanpreet *et al*., 2023).

Enhancing network security through advanced packet sniffer analysis offers several benefits. Firstly, it provides organizations with a proactive approach to security, enabling them to recognize any risks and take appropriate action before they have a substantial negative impact. By continuously monitoring network traffic, unauthorized access attempts can be detected by organizations, malware infections, exfiltration of data, and other malicious activities (Vanlalruata & Jamal, 2023).

Secondly, advanced packet sniffer analysis allows for faster incident response and remediation. Through real-time packet-level data analysis, security teams are able to promptly determine the origins and impact of security issues, enabling them to take immediate action to mitigate risks and minimize potential damage (Alexander & Linda, 2022).Moreover, it assists in identifying network vulnerabilities and weaknesses. By analyzing network traffic patterns and anomalies, organizations can identify areas where security measures can be strengthened, such as patching vulnerabilities, adjusting access controls, or implementing encryption protocols (Craig, 2023).

The increasing complexity of network security threats necessitates the use of advanced packet sniffer analysis as a proactive and effective security measure. Organizations may strengthen their network security posture, detect and address issues in real-time, and pinpoint areas for improvement by utilizing technologies like machine learning, artificial intelligence, and behavioural analytics. The adoption of advanced packet sniffers empowers organizations to always be aware of potential cyber threats, safeguarding their network infrastructure and preventing illegal access to critical data or compromise (Ramanpreet *et al.*, 2023).

By focusing on the detection and classification of DoS attacks at the network layer, this research aims to contribute toward more adaptive, scalable, and intelligent intrusion detection systems. Such systems are vital for maintaining the integrity, availability, and resilience of modern network infrastructures.

1.3 Statement of the Problem

In the current interconnected and digitized environment, network security is of utmost importance for organizations across various sectors. Cyber threats and attacks continue to evolve in complexity and sophistication, posing significant risks to the network infrastructure and sensitive data's availability, confidentiality, and integrity (Raymond, 2023). The necessity for sophisticated methods to improve network security is highlighted by the fact that conventional methods frequently fail to identify and mitigate these attacks (Zheng *et al*., 2022).

The network layer is the third tier in the OSI model. Attacks like a denial-of-service, routing protocol attacks, port-scanning and enumeration, and fragmentation-based attacks are common against this layer. However, in this study, we delved into detection and capturing of the denial of service threats at the OSI model's network layer in a local area network. Some examples of DOS attacks are UDP flood which sends a significant quantity of packets over the User Datagram Protocol (UDP) to the target systems and thereby exhausting network resources, ICMP flood which delivers a significant volume of packets using the Internet Control Message Protocol (ICMP) to overwhelm network devices, SYN flood which takes advantage of the TCP three-way hand-shake procedure by sending a lot of SYN requests without carrying out the necessary handshake, using server resources and blocking valid connections.

As internet usage continues to grow, the threat landscape for networked systems becomes increasingly complex and dangerous. Among the most persistent threats are Denial-of-Service (DoS) attacks, which disrupt the availability of network services by overwhelming systems with illegitimate traffic (Mirkovic & Reiher, 2004). These attacks often target the network layer, making it a critical point for early detection and response. Despite the existence of traditional security mechanisms such as firewalls and signature-based intrusion detection systems (IDS), these solutions often fail to detect novel or evolving DoS attacks due to their static nature and reliance on predefined rules (Buczak & Guven, 2020).

Packet sniffers are widely used tools for monitoring and analysing network traffic, but they typically lack intelligent, automated capabilities for identifying malicious activity in real time. They rely heavily on manual configuration and expert interpretation, which can lead to delayed responses or missed threats (Beale et al., 2007). As network traffic volumes increase and attack methods become more sophisticated, there is a growing need for adaptive, automated solutions.

Machine learning (ML) provides a promising avenue for addressing this gap. ML models can learn from historical data, recognize patterns, and generalize to detect previously unseen attack types with minimal human intervention (Buczak & Guven, 2020). However, integrating ML techniques into real-time packet sniffing systems—specifically for the purpose of detecting and classifying DoS attacks at the network layer—remains a challenging and underexplored area. The key difficulties lie in efficiently capturing and processing large volumes of traffic data, extracting meaningful features, and maintaining high detection accuracy while minimizing false positives.

The issue at hand was how little use traditional network security methods have in combating the constantly changing cyberthreat environment. Firewalls and intrusion detection systems are two examples of conventional security methods. As gatekeepers, firewalls filter incoming and outgoing data according to pre-established rule and is set to stop illegal access and lessen the possibility of destructive activity. Intrusion detection systems (IDS) on the other hand, keep an eye on network traffic in real time and are able to spot and stop suspicious activity or possible security breaches before they have a chance to cause serious damage. Some of the obstacles in effectively defending against current cyber threats by these Conventional approaches to network security are discussed below.

Firstly, in response to the widespread denial-of-service assaults conventional security techniques include intrusion detection/prevention systems (IDS/IPS) and firewalls may find it difficult to scale successfully. These assaults have the potential to overload network resources, making it impossible for these devices to instantly inspect and filter traffic.

The second problem with the traditional systems was Resource Exhaustion. Denial-of-service (DoS) attacks frequently seek to deplete network resources, including memory, CPU, bandwidth, and connection state tables. The volume of malicious traffic may overwhelm traditional security systems, resulting in resource depletion, network performance deterioration, or total service failure.

Detection Challenges was the third issue with the conventional systems. Real-time DoS attack detection can be difficult for conventional security solutions. It can be challenging for standard detection algorithms to discern between malicious and legitimate traffic since attackers may use a variety of evasion techniques to conceal their operations or disperse attack traffic over several sources.

Zero-Day Attacks were the fourth issue with traditional systems: to detect and lessen threats, traditional security solutions usually rely on rule-based rules or signature-based detection. Nevertheless, these defenses may fail to identify zero-day DoS assaults, which take use of undiscovered vulnerabilities or attack paths, leaving networks open to exploitation.

When trying to identify DoS attacks, traditional security techniques may produce false positives or false negatives. When legitimate traffic is mistakenly classified as harmful, it might cause false positives, which can cause needless disruption or filtering of genuine services. However, false negatives occur when malicious traffic goes undetected, providing attackers with the chance to effectively launch denial-of-service assaults without assistance.

Several times, sophisticated and resource-intensive methods, including traffic filtering, rate limitation, or traffic redirection, are needed to mitigate DoS assaults. Conventional security measures might not be as flexible or automated as they need to be in order to respond quickly to changing attack plans and put in place efficient mitigation techniques.

As a result of the above, organizations have over time faced an increased risk of security breaches, and service disruptions, potentially leading to financial losses, reputational damage, and legal liabilities (Abuya, 2020).

To overcome these challenges, organizations needed to adopt an advanced machine learning packet sniffer analysis model as a means of enhancing network security. Advanced packet sniffers offer capabilities that go beyond the traditional approaches by capturing, analyzing, and interpreting network traffic at a granular level. These tools leverage technology including Behavioural analytics, machine learning, and artificial intelligence to identify and address anomalous or malicious activities in real-time (Rounsavall, 2020).

The lack of widespread adoption and utilization of advanced packet sniffer analysis hinders organizations' ability to effectively enhance their network security. This problem can be attributed to several factors, including limited awareness and understanding of the benefits and functionalities of advanced packet sniffers, perceived complexity in implementation and management, and potential concerns about privacy and legal compliance (Julian *et al*., 2019). Addressing this problem is crucial for organizations to safeguard their networks and remain ahead of the always changing threat landscape and sensitive information. By enhancing network security through advanced packet sniffer analysis, organizations can proactively detect and mitigate denial of service (DOS) threats, respond quickly to security incidents, identify network vulnerabilities, and strengthen their overall security posture (Tariq *et al.*, 2023)

Therefore, creating a machine learning-based packet sniffer for identification and categorization of denial-of-service attack packets at the network layer was the purpose of this research. This study aimed to look into the technical capabilities, benefits, challenges, and implementation considerations of advanced packet sniffers, providing organizations with valuable insights and guidance for adopting and leveraging these tools effectively. By addressing the problem of limited efficacy of conventional network security measures and encouraging the implementation of advanced packet sniffer analysis, this research contributed to the advancement of network security practices and assists organizations in protecting their critical assets and maintaining a secure and resilient network infrastructure.

Therefore, this research addresses the problem of how to design and implement a machine learning-based packet sniffer that can detect and classify network-layer DoS attacks in real time. The goal is to build a lightweight and efficient system capable of adaptive learning and accurate decision-making without compromising the performance or reliability of the network.

1.4 Proposed Solution

To address the limitations of traditional network security measures and enhance network security, the proposed solution was to implement a machine learning-based packet sniffer for identifying and categorizing the denial of service attacks packets at the network layer as a proactive and effective security measure. Through the application of cutting-edge technology such as machine learning, artificial intelligence, and behavioral analytics, organizations can acquire more profound understanding of network traffic, identify denial-of-service incidents, and promptly address possible security risks.

Key components of the proposed solution include the designing, developing and deployment of an advanced packet sniffer tool that can capture and analyze network data in real time. This program analyzed trends, spot anomalies, and pinpointed possible security vulnerabilities in network traffic data using machine learning techniques. Furthermore, data collection and storage procedures were implemented to assure the preservation of comprehensive records for future study and investigation.

The implementation of the machine learning-based packet sniffer analysis tool involved configuring advanced filters, rules, and alerts to monitor network traffic continuously. Machine learning models were trained on historical data to distinguish typical network behavior from suspicious or malicious activities. When anomalies or security threats are detected, the tool triggers an alert for further investigation and mitigation.

To ensure effective utilization of advanced packet sniffer analysis tools, organizations should also invest in skilled personnel or training programs. Security analysts and network Administrators must possess the knowledge and abilities required to interpret the output of the packet sniffers, analyze network behavior patterns, and address security-related issues promptly. Lastly, organizations should establish incident response procedures and protocols that outline the actions that should be performed in case of a security breach or threat detection. This includes defining roles and responsibilities, establishing communication channels, and coordinating actions to lessen the effects of security incidents.

1.5 Objectives

1.5.1 General Objectives

The main objective of this study was to develop a machine learning-based packet sniffer that can detect and classify attacks at the network layer.

1.5.2 Specific Objectives

1. To analyze the existing literature on Machine Learning-Based Packet Sniffer with an aim of identifying the optimal parameters that can be applied to design a machine learning based packet sniffing tool that can detect denial of service attack packets at the network layer.
2. To design and develop a machine learning based Packet sniffing model that can automate the detection and classification of the denial of service attack packets at the network layer.
3. To validate the efficacy and efficiency of the proposed machine learning-based packet sniffer in detecting and classifying the DOS attack packets through the extensive experimentation and comparative studies with existing solutions.

1.6 Research Questions

1. What are the optimal parameters of existing machine learning-based packet sniffers for detecting attack packets that cause a denial of service (DOS) at the network layer?

2. How can machine learning algorithms be effectively utilized in the design and development of a packet sniffing model capable of automated detection and classification of DOS attack packets at the network layer?

3. How does the proposed machine learning-based packet sniffer perform in terms of effectiveness, efficiency, and accuracy in detecting and classifying DOS attack packets at the network layer?

1.7 Justification of the Study

The threat landscape in the digital sphere has changed significantly in recent years. (Moatsum *et al.*, 2022). The sophistication of cyberattacks has increased, putting both individuals and companies at serious risk (Yuchong *et al.*, 2021). The attacks are motivated by a variety of factors, including espionage, critical infrastructure disruption, financial gain, and data theft (Raymond, 2023). Consequently, network security has emerged as a critical concern for organizations, governments, and individuals looking to safeguard sensitive data and priceless assets (Moatsum *et al.*, 2022).

The increasing dependence on internet-connected systems and services has made the availability and integrity of network infrastructure a critical concern. Denial-of-Service (DoS) attacks, particularly those targeting the network layer, pose a significant threat to organizations and users by overwhelming resources and rendering services unavailable (Mirkovic & Reiher, 2004). Traditional security systems such as firewalls and signature-based intrusion detection systems are insufficient in responding to modern, dynamic attack techniques. These tools often rely on predefined rules, making them ineffective against zero-day attacks and traffic anomalies that deviate from known patterns (Buczak & Guven, 2020).

Given the limitations of static detection mechanisms, there is an urgent need for intelligent, adaptive solutions capable of analyzing large volumes of real-time traffic to identify malicious behavior. Machine learning has shown great potential in enhancing network security due to its ability to detect both known and unknown attack patterns through pattern recognition and data-driven modeling (Sommer & Paxson, 2010). Recent advances in ML algorithms enable not only the detection of anomalies but also the classification of attack types, providing more granular insights into network threats (Shone et al., 2018).

Integrating machine learning techniques into a packet sniffer provides a proactive defense mechanism that goes beyond passive monitoring. Unlike traditional sniffers, an ML-enhanced sniffer can autonomously analyze packet-level data, detect anomalies, and initiate appropriate countermeasures. This study is particularly justified because it seeks to bridge the gap between passive traffic monitoring tools and intelligent intrusion detection systems by building a real-time, machine learning-based packet sniffer focused specifically on DoS attacks at the network layer.

Furthermore, the proposed system can benefit both academia and industry by contributing a scalable, modular, and intelligent network defense solution. It addresses the growing demand for cybersecurity systems that are not only reactive but also predictive, reducing response times and improving overall resilience against evolving cyber threats.

As modern networks continue to expand in complexity, scale, and usage, they have become increasingly vulnerable to cyberattacks—particularly Denial-of-Service (DoS) attacks, which are among the most prevalent and disruptive forms of cybercrime (Mirkovic & Reiher, 2004). These attacks target the network layer, flooding systems with illegitimate traffic and severely affecting the availability of services. The rise of critical applications—such as financial services, healthcare systems, and government platforms—being hosted online further amplifies the potential consequences of such disruptions. In this context, developing robust, intelligent, and proactive defense mechanisms has become not just an academic interest but a practical necessity.

Conventional network security tools, including firewalls and signature-based intrusion detection systems, struggle to keep pace with the evolving nature of DoS attacks. These systems are typically reactive, static, and often incapable of detecting zero-day attacks or polymorphic threats (Buczak & Guven, 2020). Additionally, traditional packet sniffers like Wireshark or Tcpdump, while effective for manual analysis, lack autonomous decision-making capabilities and are not well-suited for real-time threat detection in high-speed network environments (Beale et al., 2007). This creates a critical gap between passive traffic monitoring and active, intelligent intrusion detection.

Machine learning (ML) provides a powerful solution by enabling systems to learn traffic patterns, detect anomalies, and adapt to new threats without constant human intervention (Sommer & Paxson, 2010). ML models can classify traffic as benign or malicious based on a range of features such as packet size, frequency, flags, and timing, significantly improving both detection accuracy and response time (Shone et al., 2018). Moreover, the integration of ML with real-time packet sniffing tools has the potential to create autonomous network defense systems capable of functioning with minimal latency, even under high traffic loads.

This study is particularly justified in today's environment, where low-latency, high-throughput, and real-time detection are not just desirable but essential for maintaining service reliability. By focusing specifically on DoS attack detection and classification at the network layer, the research addresses a high-risk vulnerability zone where early detection can mitigate widespread impact.

Additionally, the use of lightweight, efficient ML algorithms ensures that the system can be deployed in resource-constrained environments, such as IoT networks and edge computing platforms, which are increasingly targeted by attackers (Sangster et al., 2009).

From a broader perspective, this work contributes to the growing body of research focused on proactive cybersecurity solutions. The ability to not only detect but also classify different types of DoS attacks provides more actionable intelligence for system administrators and can support automated incident response systems. The system developed in this study can serve as a framework for future research into intelligent intrusion prevention, anomaly detection, and adaptive network security, thereby advancing both academic knowledge and practical applications in cybersecurity.

By implementing advanced packet sniffer analysis, organizations can enhance network security by real-time monitoring and the early detection of DOS attack packets at the network layer, minimizing the time window for potential attacks and reducing the risk of data breaches or network compromises.

1.8 Limitations of the Study

The research only involved organizations that that leverage network in their day to day operations. A sample institution was surveyed out as a representation of all other institutions that utilize network technologies in their operations due to time and financial limitations.

1.9 Assumptions of the study

It was believed that the characteristics and findings displayed from the sampled institution reflect the characteristics held by all other institutions of utilizing the network.

This study is based on several underlying assumptions that influence the design, implementation, and evaluation of the proposed machine learning-based packet sniffer for detecting and classifying Denial-of-Service (DoS) attacks at the network layer.

I. Availability and Quality of Network Traffic Data

It is assumed that the datasets used for training and testing the machine learning models contain sufficient, accurate, and representative samples of both normal and DoS attack traffic. The effectiveness of any supervised ML model depends heavily on the quality and diversity of the training data (Ring et al., 2019). It is also assumed that the features extracted from packet headers are adequate to differentiate between benign and malicious traffic without requiring payload inspection.

II. DoS Attacks Have Detectable Patterns

The study assumes that DoS attacks at the network layer exhibit detectable statistical or behavioral patterns that distinguish them from legitimate traffic. Machine learning models rely on the assumption that these patterns are learnable and can be generalized to unseen data (Shone et al., 2023).

III. Feature Relevance and Stability

It is assumed that the selected features (e.g., packet size, protocol type, source/destination IP and port, packet frequency) are relevant for classification and do not vary significantly across different environments or time periods. Stability of features is important to ensure that the trained model remains effective when deployed in real-world settings (Buczak & Guven, 2024).

IV. Real-Time Classification is Feasible

The study assumes that it is computationally feasible to process, analyze, and classify packets in real time using lightweight machine learning algorithms. This includes the assumption that packet capture, feature extraction, and classification can be performed with minimal latency on standard or moderately equipped hardware (Sommer & Paxson, 2020).

V. Limited Encryption at the Network Layer

It is assumed that packet headers, which reside at the network layer (e.g., IP, ICMP), remain unencrypted and accessible for inspection. Since this study does not focus on deep packet inspection or payload analysis, the visibility into header fields is critical for feature extraction and attack detection (Beale et al., 2022).

VI. Static Environment During Testing

The training and evaluation phases are assumed to occur in a controlled environment with minimal noise, traffic variability, or interference. While real-world networks are more dynamic, a controlled setting is assumed to ensure consistent results during the experimental phase of the study (Sangster et al., 2019).

These assumptions provide the foundation upon which the system is designed and tested. They also define the scope and limitations of the study and should be revisited when deploying the system in diverse or more complex network environments.

1.10 Significance of Study

1. Network plays a critical role across all disciplines. Having a technology that ensures the security of any transaction across systems on the network was so essential.
2. The sole purpose of implementing a packet sniffing technology was to improve security of the packets travelling across the network.
3. Upon conclusions of the research, the results contributed to the corpus of knowledge on Network security across all disciplines worldwide
4. The empirical evidence generated from evaluation of the packet sniffing technology provided information on the solution's efficacy to policy-makers and IT industry stakeholders. It also act as a manual for establishing rules governing the use of networks, which are essential for daily operations.

1.11 Scope of the Study.

The research targeted a representative organization that utilize networks in their daily operations as a result of financial and time constraints. The study was be subject to the availability, access and quality of the sampled institutional data.

This study focuses on the development and evaluation of a machine learning-based system for the detection and classification of Denial-of-Service (DoS) attacks at the network layer using packet sniffing techniques. The research is limited to analyzing and classifying network traffic data collected through sniffers using supervised machine learning models, specifically focusing on flow-based features relevant to network behavior.

The study employs the CICIDS2018 dataset, which provides a rich collection of labeled network traffic data representing both normal activity and various DoS attack scenarios. Only network-layer DoS attacks—such as HTTP Flood, ICMP Flood, UDP Flood, and SYN Flood—are considered. Application-layer attacks, social engineering, insider threats, or attacks involving encrypted payloads are beyond the scope of this research.

The machine learning models examined include LightGBM, Random Forest, and Support Vector Machines (SVM), with an emphasis on model performance metrics such as accuracy, precision, recall, and F1-score. Feature extraction is performed using flow-based attributes derived from packet captures, while preprocessing includes standard techniques like data cleaning, normalization, and balancing (e.g., SMOTE for minority classes).

The scope does not extend to the deployment of a real-time intrusion detection system in a production environment. Instead, it focuses on offline training and evaluation of models using pre-recorded datasets. Furthermore, this research does not address the mitigation or prevention of attacks—only their detection and classification.

The implementation is carried out in a controlled simulation using Python-based machine learning libraries such as Scikit-learn and LightGBM. Network traffic is analyzed in a post-capture mode rather than through a live monitoring interface. The study, therefore, aims to provide a proof-of-concept system capable of identifying DoS patterns in network traffic, which can later be integrated into real-time intrusion detection frameworks.

CHAPTER TWO: LITERATURE REVIEW

2.0 Introduction

The chapter evaluated and critiqued relevant research on machine learning-based packet sniffers for computer network traffic monitoring and analysis. The chapter is organized as follow; - literature on existing models, literature on the design and development of packet sniffer models, literature on validation on the packet sniffer models, conceptual framework, followed by the literature review chapter's conclusion.

2.1 Literature on Existing models

2.1.1 Wireshark

Wireshark is a network packet analyzer. It records local network activity and retains the information for later offline analysis (Singanamalla, 2022). Wireshark is used for network troubleshooting, software development, research, analysis, and protocol invention (Combs, 2023). Wireshark is one free tool for intercepting data being transferred back and forth (Porup, 2020). The network card is put into an unselective mode, which enables it to receive all incoming packets. (Porup, 2020). Even while Wireshark is frequently used to delve deeply into basic TCP/IP connection problems, it can analyze hundreds of protocols in real time and allow decrypting many of those protocols (Porup, 2020). It is used for software testing and network troubleshooting since it can scan packet contents and drill down.

Wireshark uses Berkeley Packet Filters, a microprogramming language that is executed at runtime and run on packets that have been captured by programs like tcpdump and Wireshark (OpenLogic, 2023).

Wireshark is used for port scanning, (Lee et al., 2023). One method that is commonly used to find servers that are susceptible is port scanning. One way to view the process of port scanning is as a hostile internet search for open doors that allow hackers to access machines. One method that is commonly used to find servers that are susceptible is port scanning. Port scanning, according to Lee et al. (2023), is the act of sending a message to a port and watching for a response. It may be thought of as a hostile internet search for open doors that allow hackers to access systems. The answer received shows the port status and provides the necessary information to initiate further attacks.t al., 2023) and It entails putting a message on a port and waiting for a reply. The received answer shows the port status and provides the necessary information to initiate further attacks (Ndatinya et al., 2021).

Network performance might be a major worry for network administrators, according to Tuli (2023). Numerous factors affect how well the network performs. Network performance problems include: - bottlenecks, retransmissions or packet loss, slow Internet, and high bandwidth use. A thorough grasp of network protocols is necessary in order to troubleshoot a network. The researcher was inspired by this to conduct a study in which he used Wireshark, a program that captures network traffic for multiple devices, to examine the performance and integrity of a network in the context of a home network. Following that, Basic statistical tools and sophisticated tools for many performance metrics in Wireshark were used to examine the collected traffic. The researcher's conclusion demonstrated that Wireshark offers thorough information about network issues and performance, as well as how to use it with other tools by giving a list of events and issues that

Wireshark has found on the network. Wireshark won't alert you when someone on your network engages in weird or forbidden behavior. To address this concern the proposed machine learning packet sniffer tool will seek to detect and raise an alert for any malicious act or data packets across the network.

2.1.2 SCRUB-tcpdump

SCRUB-tcpdump is an application that builds up on the well-known tcpdump program to facilitate simple packet trace data handling while also utilizing anonymization to prevent sensitive data Users of SCRUB-tcpdump have the ability to permute the data, add noise, or remove all information (black marker) in order to anonymize sensitive fields to various desired levels (Yurcik et al., 2019). Options for multi-level anonymization in every field provide organizations the ability to choose anonymization schemes that are more closely aligned with actual sharing environments between parties due to the fact that several firms have varying security policies and privacy needs (Yurcik et al., 2019). It is crucial to have multiple levels of anonymization because there isn't a single, effective anonymization method (Yurcik et al., 2019).

SCRUB-tcpdump handles I/O using the libpcap engine, enabling input from a capture log file or from a live capture. Progressively, the system reads, parses, and sends each packet individually. One can add extensions directly to tcpdump or combine Libpcap (as a standalone library) with a standalone SCRUB-tcpdump anonymizer (Yurcik et al., 2019). Since libpcap forms the basis of tcpdump, libpcap is ideally suited for extending the capabilities of the command line, allowing users to use tcpdump for file processing in addition to during the capture process, and linking directly into tcpdump source (Yurcik et al., 2019). The list of anonymization and the input file procedures to be carried out are specified in the command line options used to run the system. The input file is loaded, The controller block is assigned to the anonymization method list, and the required pcap filters are applied (Yurcik et al., 2019). Selected fields in a sequential packet are anonymized before being stored in memory (Yurcik et al., 2019). This process keeps going until the user stops it or there are no more packets (Yurcik et al., 2019). The pcap header is initially checked during the anonymization process, and if any pcap-specific data (such as timestamps) is chosen to be anonymized, the required routine is then invoked (Yurcik et al., 2019). Packet fields are enclosed and may be dependant, thus a tiered approach is used that anonymizes data at each layer and then explores the next layer until all packet fields are exhausted (Yurcik et al., 2019).

There is also a high level of complexity while configuring the anonymization settings to safeguard privacy while preserving data utility especially in the case of SCRUB-tcpdump. It requires a high level of expertise and similarly a lot of keenness while at it. Furthermore, there is no anonymization tool that can guarantee complete privacy. There is always a risk of re-identification, especially when anonymized data is integrated with other data sources.

The drawback of SCRUB-tcpdump is that it ignores packets that have an invalid checksum. It is therefore not particularly helpful when handling network packets that have incorrect checksums. It requires the usage of particular equipment. It doesn't show any internal details—just the data that was meant to be conveyed. It is unable to identify any fake IP addresses in the packet.

2.1.3 K-means clustering

According to Shanker et al. (2020), there are several algorithms, different protocols, and intrusions in any communication system. "K-means clustering" is the feature extraction method they proposed. In their study, they used Python and the stream, as well as its implementation and applications, utilizing the Knowledge Discovery and Data Mining Cup (KDD-cup) dataset and the K-Means technique. In addition to the employed technique, they created a web-based framework that was utilized to examine a real network traffic packet in order to spot network attacks. The web page-based approach employed a database to gather and test the information identifying new attacks, as opposed to a static file, to test the network assault. This gave the K-Means machine learning methodology the information it needed to detect and categorize the attacks through a comparative examination. They calculated the degree of similarity between the attack groups using the K-means algorithm. The suggested machine learning-based packet sniffer aims to address the issues of detecting more real-time environment datasets and categorizing attacks into distinct groups utilizing additional data mining methods.

2.1.4 Network Intrusion Detection Systems

Mehra Gupta and Guruji (2019) state that a number of studies have employed data mining techniques in intrusion detection systems. They emphasize further that these studies have concentrated on anomaly detection IDS and found associations between attributes, omitting to take into account the parameters of network intrusion detection systems (NIDS) such as detection rate and search efficiency. Inspired by these results, they suggested ways to enhance NIDS performance by taking into account both the detection rate and search efficiency criteria. Their suggested approach performs better than the prior systems, achieving an accuracy of 82.43% compared to 89.56% based on the metric rate of detection. But the false alarm frequency for the prior system and the suggested system is 0.1%. By taking traffic content into account for mining, the proposed machine learning-based packet sniffer aims to increase network intrusion detection and detection rates as well as reduce false alarm rates.

2.2 Design and Developing of Packet Sniffing Model for Detecting and Classification of Network Attacks

Su et al. (2020) proposed a novel model termed BAT-MC through the two phases of learning Bidirectional Long Short-Term Memory (BLSTM) and focusing on time series features for intrusion detection using the NSL-KDD dataset. The BLSTM layer, which is utilized to extract features from each packet's traffic bytes, connects the forward and backward LSTMs. Any data packet has the ability to produce a packet vector. The BAT-MC approach's performance is investigated using the KDDTest+ and KDDTest-21 datasets. The results of the experiments conducted on the NSL-KDD dataset demonstrate that the accuracy of the BAT-MC model is relatively high. These evaluations against a standard classifier show how promising the BAT-MC model's results are when compared to those of other deep learning-based methods currently in use. However, its detection performance distinguishes normal traffic from attack traffic. The proposed machine learning-based packet sniffer will seek to improve this early work in distinguishing different the different attack traffic.

Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) attacks are among the most prevalent forms of cyber threats that target the availability of services by overwhelming networks with malicious traffic. These attacks often exploit vulnerabilities at the network layer (Layer 3 of the OSI model) by flooding targets with excessive ICMP, UDP, or SYN packets, thereby exhausting bandwidth or server resources (Mirkovic & Reiher, 2024). DoS attacks not only degrade system performance but can also result in substantial economic losses and service downtime for enterprises and critical infrastructure.

Mirkovic and Reiher (2024) provide a taxonomy of DoS attack strategies and mitigation techniques, emphasizing the importance of early detection mechanisms at the packet level. Similarly, Douligeris and Mitrokotsa (2024) suggest that multilayered defense strategies, particularly those incorporating intelligent traffic analysis, are essential to counter the sophistication of modern DoS variants. 1.6.1 Denial-of-Service Attacks and Network Vulnerabilities

Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) attacks remain persistent threats to network infrastructure due to their capacity to disrupt service availability. These attacks commonly target the network layer, where attackers flood targets with large volumes of traffic using protocols such as TCP SYN, UDP, and ICMP (Zargar, Joshi, & Tipper, 2013). The increasing accessibility of botnets and low-cost attack tools has made launching DoS attacks easier than ever before, resulting in an upsurge in both frequency and sophistication.

According to Kreutz et al. (2021), with the rise of programmable networks and software-defined networking (SDN), the attack surface for DoS has expanded further, requiring smarter and faster defensive solutions that can dynamically adapt to network changes. Therefore, a shift toward intelligent and adaptive intrusion detection is both timely and necessary.

Packet sniffing has long served as a cornerstone of network monitoring and traffic analysis. Tools like Wireshark and Tcpdump offer deep visibility into network packets but are designed for manual inspection and lack autonomous decision-making (Beale et al., 2007). Intrusion Detection Systems

(IDS) like Snort and Suricata introduce rule-based detection mechanisms but still depend on signature libraries, making them ineffective against novel or zero-day attacks (Axelsson, 2020).

To bridge the gap, researchers have proposed incorporating automated classification systems into packet sniffers. This enhances their capacity to not just monitor traffic but to proactively identify suspicious patterns and potential threats. Real-time sniffing paired with machine learning is a promising avenue for elevating traditional monitoring tools into intelligent IDS platforms.

Machine learning (ML) techniques have emerged as powerful alternatives to traditional, signature-based IDS. ML algorithms are capable of learning from historical data and identifying anomalies, thereby detecting both known and unknown threats (Buczak & Guven, 2020). Supervised methods like Random Forests, K-Nearest Neighbors (KNN), and Support Vector Machines (SVM) have been extensively studied for classifying malicious traffic.

More recently, deep learning models such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks have been used to capture spatiotemporal dependencies in network traffic (Kim et al., 2020). However, these models often require high computational resources, which can hinder their use in real-time detection scenarios unless properly optimized.

Shone et al. (2018) proposed a hybrid deep learning framework using autoencoders for feature reduction and deep neural networks for classification, achieving high accuracy on benchmark datasets such as NSL-KDD. While promising, such models must be validated on modern traffic datasets to ensure practical relevance.

The accuracy of ML-based detection is heavily influenced by the features used to train models. Effective feature selection minimizes computational complexity and improves model performance. Studies by Tsai et al. (2009) and Ring et al. (2019) emphasize that selecting discriminative features—such as packet size, flow duration, and protocol flags—is crucial for distinguishing between benign and malicious behavior.

Unsupervised and semi-supervised learning approaches, which rely on clustering and anomaly detection, are increasingly employed to overcome the lack of labeled data in real-world settings. These models can detect abnormal patterns without requiring exhaustive labeling, which is often impractical for large-scale networks.

Dataset quality plays a fundamental role in evaluating the performance of ML-based IDS. The NSL-KDD, CICIDS2021, and UNSW-NB15 datasets are among the most widely used in recent research. However, many researchers point out limitations in their realism and variability. For instance, McHugh (2020) critiques KDD-based datasets for being outdated and overly synthetic, while more recent datasets like CICIDS2021 aim to replicate real-world network environments more accurately (Sharafaldin, Lashkari, & Ghorbani, 2018).

To build more practical systems, researchers like Sangster et al. (2009) advocate for simulated environments and competitions (e.g., CTF) that generate realistic labeled traffic to train and validate ML models effectively.

Real-time deployment of ML-based IDSs introduces several challenges. These include latency, throughput, memory usage, and adaptability to changing traffic patterns. In edge and cloud environments, lightweight models such as Decision Trees or ensemble techniques like XGBoost are preferred due to their speed and lower resource consumption (Chen & Guestrin, 2020).

Furthermore, explainability and interpretability of ML decisions are gaining attention. As Binns et al. (2018) point out, deploying "black box" models in critical infrastructure requires mechanisms for administrators to understand and trust detection outputs—hence the growing interest in Explainable AI (XAI) in cybersecurity.

Machine learning (ML) has emerged as a powerful approach to enhancing intrusion detection systems (IDS). Unlike traditional rule-based systems, ML can model complex relationships in data and detect anomalies or attacks even when they deviate from previously observed behavior (Buczak & Guven, 2024). Supervised algorithms such as Decision Trees, Random Forests, and Support Vector Machines (SVM) have been widely applied to classify network traffic into normal or malicious classes.

Shone et al. (2023) proposed a deep learning framework using autoencoders and deep feedforward neural networks to improve detection accuracy and reduce false positives. Their results show that deep learning outperforms traditional ML approaches, especially in recognizing subtle attack patterns in high-dimensional datasets.

In their review, Sommer and Paxson (2020) argue for the cautious use of ML in IDS, pointing out that real-world deployment demands robust models that can handle noisy, imbalanced, and encrypted traffic. Nonetheless, they affirm that with proper feature selection and training, ML remains a viable tool for real-time intrusion detection at the packet level.

Paul and Saha (2020) denotes that end users of a computer system submits data as well as applications for the cluster, A data cluster is made up of several computers connected to a network, and when processing data, numerous processes operate on various nodes and exchange data. The standard network protocols are used for the data exchange. The researcher noted that as a result of poor hardware or operating system health, one or more computers may not perform well during processing. Additionally, certain computers may be subject to known network attacks like denial-of-service (DoS), while other computers may be subject to unknown attacks. All of these factors might lower the cluster's performance.

Nsaif and Rhee (2019) underlined that since every message in the HSR networks has to be duplicated, the High-availability Seamless Redundancy (HSR) protocol used a lot more network capacity than the Ethernet standard (IEEE 802.3). Upon realizing this, the researchers sought to introduce an approach, the Pruning Multicast Traffic (PMT) approach, that was geared to lessen needless redundant multicast traffic in High-availability Seamless Redundancy (HSR) networks by restricting how the network's traffic is sent to the rings where those members are connected to that traffic.

The PMT technique's simulated analysis demonstrate a greater traffic efficiency in comparison to the regular High-availability Seamless Redundancy protocol. When compared to the standard HSR protocol, the PMT provides a traffic reduction percentage of between more than 50% and less than 80%. Because the traffic does not pass via every network element, the PMT approach also provides a certain level of traffic security for the supplied traffic. Nonetheless, nodes forward unicast frames from one port to another port in accordance with the normal PMT protocol's forwarding rule, unless they have already sent the same message in that route. A frame injected into the ring by a node will not be forwarded hence the objective for this current research project to develop a more flexible machine learning packet sniffer tool to enhance utilization of network bandwidth.

Tien et al., (2020) presented a unique algorithm, dubbed optimal dual pathways (ODP), to handle an issue of degradation of network performance resulting from superfluous redundant traffic that consumes a considerable percentage of the network Bandwidth. The goal of the ODP algorithm is to construct dual routes for each connection pair of nodes. The dual pathways are determined based on the ideal metrics of the links and have no common nodes between them, to avoid a single point of failure. These pathways are used to forward the unicast frames between each connection pair of nodes instead of duplicating and forwarding frames randomly to all portions of the network. The ODP algorithm showed a traffic reduction of 73% compared to the regular HSR protocol. This finding suggested that the ODP frees up a lot of the network bandwidth and increases network traffic performance, therefore allowing more apps to function in the network. The challenge of the ODP algorithmic approach is that it works only on a network where the nodes are linked by cables.

The proposed machine learning packet sniffer tool will seek to address this by having the ability to operate optimally on wireless networks and for all traffic types, that is, both unicast and multicast broadcast.

In 2019, Tien and Rhee presented a cutting-edge technique to reduce and filter HSR unicast traffic in connected-ring networks. This method, known as "filtering HSR traffic" (FHT), aims to remove circulating traffic for each ring in connected-ring networks and filter HSR unicast traffic. As a result, FHT dramatically lowers connected-ring networks' unicast network traffic. FHT's traffic performance was investigated, assessed, and contrasted with that of the method of port locking (PL) and the conventional HSR protocol by the researchers. Based on analytical and simulation results, FHT reduced network unicast traffic by around 82% compared to standard HSR and roughly 56% compared to the PL technique, improving network traffic performance and freeing up network bandwidth for our sample network. FHT is a productive method for cutting notably increased unicast traffic in HSR networks. However, it has no solution for the Multicast traffic hence the motivation to carry on with this research study of developing a machine learning packet sniffer analysis tool in a way to operate in all types of network traffics.

Ring-based dual pathways (RDP), a unique dual routes-based technique, was introduced by Tien, Kim, and Rhee (2018) to drastically cut down on redundant unicast traffic on networks with seamless redundancy that are high availability. In HSR networks, RDP creates two channels for unicast frame forwarding between nodes. RDP creates dual pathways for every connection pair of HSR rings, in contrast to other dual paths-based techniques like dual virtual paths (DVP), which create dual paths for every pair of HSR terminal nodes. Therefore, compared to the current dual pathways-based systems, RDP greatly reduces the number of established dual paths, which in turn minimizes the overhead in detecting and building dual paths as well as the memory space necessary to store these paths. The traffic performance analysis was validated using a series of simulations. According to analytical and simulation data, RDP improved network traffic performance and freed up network bandwidth for the sample networks by reducing network unicast traffic by 80–88% when compared to the standard HSR protocol. It is important to do this research in an effort to address the problem of having an approach that only functions for unicast types of network traffic because RDP can only handle unicast types of network data.

2.3 Validating the Machine Learning Packet Sniffer Analysis Tool's Effectiveness and Efficiency.

Grossi, Alfonsi, Prandini and Gabrielli (2023) describes the testing of a stateless packet sniffer that has been implemented on FPGAs, that is, field programmable gate arrays and supports up to one gigabit per second of data transport. The designed system computes the principal frame fields (Source and destination ports, IP addresses, MAC addresses), analyzes Ethernet frames of different types (ARP, IP, UDP, TCP, ICMP), and assesses potential threats of the received data based according to a set of guidelines that the user Preprints. The packet sniffer was tested twice: once in controlled conditions using an ad hoc packet generator to create groups of Ethernet frames, as well as once more in situations involving actual online traffic by linking the device to the internet. The current research project's objective is to develop a machine learning packet sniffer tool that will have an increased speed and will also have the ability to work at data rates of 10 Gigabits /s and 100 Gigabits /s. this will form basis for validation of the proposed tool with respect to its effectiveness and efficiency.

In order to communicate the outcomes of the continuous sniffing process, Dharmesta et al. (2020) developed a computer network protocol traffic learning application that makes use of Scapy and natural language. Students specializing in Computer and Network Engineering were given a questionnaire to complete in order to gauge how well the application translated the sniffer procedure using natural language. After the questionnaire was distributed, the results were calculated using a Likert scale, and it was discovered that the initial smelling process results had a Likert scale value of 37%. Conversely, the translated sniffer results yield a 73% value. However translation to natural language suffers one major blow and that is, it is only suited where all people working in an organization understands that one natural language. The proposed Machine learning packet sniffer tool is expected to be effective and efficient by being able to translate the results from the original language to a universal language across several geographical borders.

Qadeer and Zahid (2020) concentrated on the fundamentals of packet sniffer operation, Linux platform development, and intrusion detection applications of the program. The researchers talked about how to effectively identify and deal with intrusion detection software that may be present on the network. Using this packet sniffer they had constructed themselves, the researchers examined the bottleneck scenario that had emerged in the network. Prior to creating this native software, the researchers examined how other sniffer programs operated. These findings provided as the basis for the development of our own sniffer program. This developed packet sniffer worked quite well only that it ran on Linux platform. The proposed machine learning packet sniffer tool will be made effective and efficient by adding capabilities such as filtering packets using a filter table, removing questionable information from network traffic, making the packet sniffer application platform independent, and collecting and presenting network statistics.

Mohammed *et al*., (2021) ventured into a research study to determine which ineffective network-based apps are using up precious bandwidth on Ahmadu Bello University's Using Zaria as a case study, this research project aimed to investigate the internet traffic networks of Ahmadu Bello University (ABU). The use of bandwidth monitoring software and a packet sniffer (using Wireshark Version 165, SVR Rev 40429) as a gateway between the University network system and the internet was allotted 15 minutes per day, 30 minutes per week, and 2 hours per month during the course of the 90-day monitoring period. MATLAB, a tool for charting network data, was used to further evaluate the data (packets) that were acquired. The major setback for this research is that it is not possible to decrypt encrypted traffic using Wireshark. Additionally, it is not difficult to spoof IPv4 packets. It is not really possible for Wireshark to determine if an IP address it discovers in a captured packet is legitimate or not. Therefore, should the proposed machine learning packet sniffer tool must have the ability to tell real IP addresses from spoofed ones. It will also ensure that it captures data packets and make sense out of them and finally interpreting the results it finds.

2.4 Conceptual Framework & Methodology

A synopsis of how users and the system interact is provided by the conceptual framework.

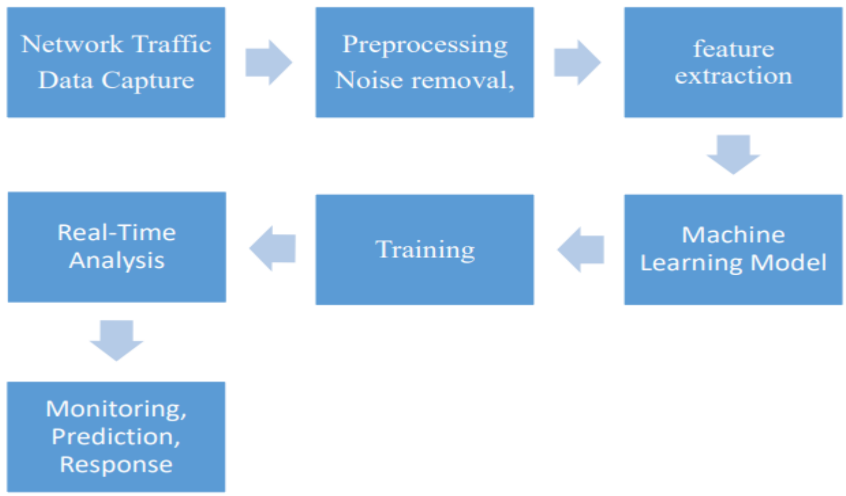


Figure 2. 1 The machine learning packet sniffer model's conceptual framework

The figure 1 above, visualizes the flow of data and processes within the framework. This framework explains the key processes in the rolling out of the machine learning-based packet sniffer to improve network security.

a) Data Capture: involves capturing network traffic data from various sources such as routers, switches, or network nodes.

b) Preprocessing: includes cleaning and preparing the captured data by removing noise, filtering irrelevant packets, and standardizing data format.

c) Feature Extraction: Extraction of relevant attributes or attributes from the preprocessed data, such as protocol type, timestamp, source and destination IP addresses, and packet size.

d) Machine Learning Model: involves training the machine learning model using the extracted features to classify network traffic into different categories, for instance, normal traffic or suspicious behavior.

e) Training: Training the machine learning model using labeled data, where each packet has a known classification label attached to it (e.g., normal or malignant).

f) Real-Time Analysis: involves real-time implementation of the trained model environment to analyze incoming network traffic. Evaluate each packet and predict its classification based on learned patterns and features.

g) Anomaly Detection: Detect anomalies or unusual behavior in network traffic that deviates from normal patterns. Identify potential security threats or abnormal network conditions.

h) Response Mechanism: Trigger automated response mechanisms, such as alerting network administrators, blocking suspicious traffic, or initiating further investigation, when malicious activity is detected.

i) Feedback Loop: involves continuous collection of feedback from the network environment to help update the machine learning model with new data and retraining the model periodically to adapt to changes in network behavior and emerging threats.

j) Evaluation and Validation: Analyze the machine learning packet sniffer's performance using measures like false positive rate, accuracy, precision, and recall. Verify the system's efficacy and identify areas that need improvement.

2.5 Conclusion

Institutions and individuals desire to get it right the level of security of their resources, data and information in their computing infrastructure. This desire led the researcher to propose a model that provides a new tool that span beyond the human focus of the reviewed models. The proposed tool when applied to a computer network environment hoped to detect and capture various DOS attack packets at the network layer in a local area network. In addition, the literature review process showed that the reviewed models, when applied to a computer networked environment, they resulted in notable shortcomings. This included platform dependency of the existing models, packet loss problems, they have a single point of view, limited categorization of attacks, not to mention complexity and the need for a high level of expertise to operate among many other challenges as highlighted above.

Therefore, in conclusion from the layout observation and this chapter's analysis, it was eminent enough to proceed the proposed research study.

CHAPTER THREE: METHODOLOGY

3.1 Introduction

The implementation technique, which involved using a machine learning-based packet sniffer to detect and classify denial-of-service attack packets at the Local Area Network's network layer, was covered in this chapter. In the network layer, we focused on the IPV4 and IPV6 address protocols. These addresses were fundamental for routing and addressing traffic at the OSI model's network layer. The focus of this research was centered on developing a model to for detecting and classifying of the denial-of-service attacks at the OSI model's network layer in a local area network. Some examples of DOS attacks are UDP flood which sends a significant quantity of packets over User Datagram Protocol (UDP) to the target systems and thereby exhausting network resources, ICMP flood which Transmits a substantial quantity of Internet Control Message Protocol (ICMP) packets to overwhelm network devices, SYN flood which takes advantage of the TCP three-way hand-shake procedure by sending a lot of SYN requests without carrying out the necessary handshake, using server resources and blocking valid connections. The use of machine learning packet sniffers models for detection and classification of DOS attack packets at the network layer greatly improved network security posture, improve threat detection capabilities, and help enterprises stay ahead of cyber threats in an increasingly complex and dynamic threat landscape. The research methodology applied in this study was an experimental research methodology. The model was implemented in Python and employed the CICIDS2018 dataset downloaded from Kaggle.

3.2 Dataset Description

The CICIDS2018 dataset designed by the Canadian Institute for Cybersecurity (CIC), contained network traffic data collected from various network environments, including LANs. They contain normal traffic and many forms of attacks, including those targeting Layer 3 protocols, making them ideal for training a network security model.

It is one of the most comprehensive and realistic benchmark datasets available for evaluating network intrusion detection systems. It was created to address the limitations of older datasets such as KDD99 and NSL-KDD, particularly in terms of modern attack vectors, realistic traffic patterns, and high-quality labeling (Sharafaldin, Lashkari, & Ghorbani, 2018).

Data collection for CICIDS2018 was conducted in a controlled environment that simulated real-world network infrastructure. The network consisted of a diverse mix of devices including workstations, servers, and attacker machines, all communicating over a testbed hosted on Amazon Web Services (AWS). The dataset spans 10 days of activity and includes both benign and malicious traffic. Attacks were launched in a time-controlled manner to simulate realistic intrusion scenarios across different days.

The dataset comprises both raw packet capture files (PCAPs) and pre-processed CSV files. The CSV files are generated using CICFlowMeter, a tool that extracts more than 80 flow-level features from raw traffic data. These features include basic attributes like flow duration, protocol type, and packet size, as well as statistical metrics such as inter-arrival times, flow byte rates, and packet counts. Each traffic flow is labeled as either benign or belonging to a specific attack type, enabling supervised learning applications.

CICIDS2018 incorporates a wide variety of contemporary attack types, including Distributed Denial-of-Service (DDoS), Brute Force, Botnet, Infiltration, Heartbleed, and various web-based exploits. Notable attack variants such as HOIC, LOIC, Slowloris, and GoldenEye are included to reflect the evolving nature of cyber threats. This diversity makes the dataset particularly useful for evaluating the effectiveness of machine learning models across multiple threat categories.

One of the dataset's key strengths lies in its realism; the traffic was generated by simulating actual user behavior using automation tools that mimic web browsing, email access, and file transfers. However, it also presents certain challenges. The class distribution is highly imbalanced, with some attack types occurring far less frequently than others. This poses a difficulty for classifiers and necessitates the use of data balancing techniques such as oversampling, undersampling, or synthetic data generation.

In terms of scale, CICIDS2018 is extensive—containing over 16 million traffic flows and totaling several hundred gigabytes when including raw PCAPs. Due to its richness, it has become a popular choice in cybersecurity research, particularly in the evaluation of machine learning and deep learning-based intrusion detection systems. Nevertheless, researchers have noted some caveats, including labeling inconsistencies and occasional artifacts from simulation that may lead to overly optimistic performance if not properly accounted for.

Despite these limitations, CICIDS2018 remains a valuable benchmark for developing and testing intelligent intrusion detection models. Its comprehensive nature and inclusion of modern attack vectors make it a superior alternative to legacy datasets, and its widespread adoption has helped standardize performance comparisons across research studies.

The dataset was used to train, validate, and test the model to evaluate trends, detect anomalies, and identify potential security vulnerabilities in network traffic data.

The purpose of employing the CICIDS2018 dataset was to use the dataset to develop create a model with comprehension and categorization of network traffic data. The model's performance was evaluated using several measures that demonstrate its ability to accurately identify threats at the network layer in a local area network, such as sensitivity, specificity, and accuracy.

https://www.kaggle.com/datasets?search=CICIDS2018+dataset is the URL to the dataset.

3.3 Tools, Libraries and Frameworks

Using the CICIDS2018 dataset, the researcher created a machine learning model for threat detection or other uses by combining a number of tools and libraries for data preparation, model development, training, evaluation, and deployment.

They include: -

i. **Python**: - Python is frequently used in machine learning and data research because of its ease of use, adaptability, and huge libraries. Python was used to perform data preprocessing, model creation, evaluation, and deployment. It was used as the major programming language.

ii. **Jupyter Notebook**: - Jupyter Notebook offered a data exploration interactive environment, experimentation, and machine learning model building on the CICIDS2018 dataset.

iii. **Pandas and NumPy**: Pandas is a Python package for data processing and analysis, whereas NumPy is a foundational package for scientific computing in Python. Pandas and

NumPy are crucial tools for data preprocessing, manipulation, and transformation tasks, allowing to handle structured data more effectively, such as the CICIDS2018 dataset.

iv. **Scikit-learn**: is a common Python machine-learning framework that offers simple-to-use tools for data mining and analysis. Scikit-learn is a great tool for creating and evaluating machine learning models on the CICIDS2018 dataset since it offers a variety of tools and methods for classification, regression, clustering, dimensionality reduction, and model evaluation.

v. **TensorFlow or PyTorch**: TensorFlow and PyTorch are two popular deep learning frameworks that offer complete tools for developing and training neural network models. TensorFlow or PyTorch can be used to create deep learning models, including convolutional and recurrent neural networks (RNNs and CNNs), to detect threats using the CICIDS2018 dataset.

vi. **Matplotlib and Seaborn**: Matplotlib and Seaborn: Matplotlib is a Python charting toolkit; Seaborn is a Matplotlib-based statistical data visualization library. Matplotlib and Seaborn are effective tools for visualizing data distribution, linkages, and trends in the CICIDS2018 dataset.

vii. **Scapy**: Scapy is a strong interactive packet manipulation software and Python library. Scapy was used to analyze and manipulate network traffic data at the packet level in the CICIDS2018 dataset, particularly if raw packet captures need to be preprocessed.

3.4 Datatypes

While dealing with the CICIDS2018 dataset for threat detection in network traffic OSI model's network layer, the researcher used the following types of data: -

i. **Numerical data** - includes continuous or discrete numerical values representing various features or attributes of network traffic data. The CICIDS2018 dataset included numerical data such as packet sizes, flow durations, byte counts, and numerical representations of protocol types, IP addresses for the source and destination, and port numbers.

ii. **Categorical data** - are discrete categories or labels for various attributes in a dataset. The CICIDS2018 dataset contained categorical data such as protocol names (e.g., TCP, UDP), attack types (e.g., DDoS, brute force), and binary labels denoting normal or malicious traffic.

iii. **Text Data** - includes data such as domain names, URLs, or payload content. Text data required particular preprocessing techniques such as tokenization, stemming, or vectorization before it being utilized as input characteristics for machine learning models.

iv. **Temporal data** - refers to timestamps or time-related information connected with network traffic occurrences. This comprises timestamps showing the start and end times of network flows, as well as time intervals between packets.

v. **Spatial data** - represents geographical information such as IP addresses or network locations.

vi. **Image Data** - includes picture data created from visual representations of network traffic or network diagrams.

3.5 Data Acquisition and Analysis

Gathering and analyzing data were fundamental steps in the implementation of the proposed machine learning packet sniffer model that were used to detect network threats at the network layer in a LAN using the CICIDS2018 dataset. The dataset was downloaded from the official website of the University of New Brunswick's Canadian Institute for Cybersecurity (CIC). There was need to familiarize with the structure and contents, feature's meaning, data format, and any related documentation or metadata of the dataset.

To familiarize with the crucial insights into the characteristics in the dataset, effective data analysis and visualization was adopted prior to implementation (Waskom, 2021). In implementing the machine learning packet sniffer model to detect network threats in a LAN using the CICIDS2018 dataset, data analysis had a vital role in understanding the dataset, identifying patterns, and preparing the data for model training. This analysis is essential for building accurate and effective predictive models that helped enhance network security and protect against various types of threats and attacks at OSI model's network layer.

3.6 Data Cleaning and Preprocessing

Ensuring the quality and dependability of a dataset for analysis involves finding and correcting flaws, inconsistencies, and inaccuracies. This phase entailed cleaning up and preprocessing the raw dataset to help remove missing values, outliers, and inconsistencies (Makara, 2023). It also aids in converting categorical variables to numerical representations using techniques such as one-hot encoding. Normalize or standardize numerical properties to ensure that scaling was constant.

During the implementation a machine learning packet sniffer model to detect network risks in a LAN using the CICIDS2018 dataset, data cleaning was critical in ensuring the dataset's suitability for model training. It involved the following tasks.

Identifying and handling Missing Values: - the extent of missing-ness was analyzed by identifying any missing values in the CICIDS2018 dataset, represented as NaN, NULL, or other placeholders. To handle the missing values, the researcher chose between performing imputation, that is, replacing missing values with a calculated value, such as, mean, median or mode according to the distribution of the feature or having the removal of the columns containing missing values if the columns are not critical to the model training or doing prediction of the missing values based on other features in the dataset using machine learning models.

Identifying duplicate records: - Any duplicate records or observations in the dataset had to be removed to ensure data integrity.

Correcting Data Errors: - this involved identifying and correcting the inconsistencies and data errors existing in the dataset. It involved standardizing formats, correcting typos and addressing outliers in the dataset.

Handling inconsistent data: - the discrepancies in the data set were identified and resolved by cross-validation that is, comparing values of different features to identify the inconsistency and using domain knowledge to resolve the inconsistencies based on the dataset context.

Normalize or Standardize Data: - Normalizing or standardizing numerical features had to be done to ensure they are on a similar scale. This helped prevent bias and skewness in the model and improve its performance.

Verify Data Integrity: - Verifying the integrity of the cleaned dataset was done by performing sanity checks and validation tests.

Label Encoding: - In order to make the dataset compatible with machine learning models that require numerical inputs, label encoding is a preprocessing technique that was used to transform categorical data into numerical values.

3.7 Model Training

The researcher proposed to use the light gradient boosting machine (LigthGBM). Before the model training, cross-validation was performed to tune the model parameters and assess the model's effectiveness using hypothetical data. By taking this action, you may ensure that the model does not overfit and that it fits new data properly. First, the data was divided using the train_test_split() function from sci-kit-learn into training and validation datasets. This division took the proportions of 70% of the dataset for training and 30% formed the validation dataset. This kind of proportion facilitated learning of all the patterns and relationships within the dataset during the training. The machine learning packet sniffer tool was trained using the training dataset, enabling the tool to learn patterns and relationships within the dataset while the validation dataset was used to evaluate trained modes to assess their generalization performances. The LightGBM model was trained using the prepared dataset and configured parameters. During the training, the LightGBM model iteratively built decision trees to minimize the specified objective function (such as, binary classification, multiclass classification), regression using the gradient-based optimization techniques.

3.8 Light Gradient Boosting Machines (LightGBM)

The researcher proposed to use the LightGBM model because it's optimized for speed, accuracy, and scalability, memory usage and is also well-suited for large-scale datasets and high-dimensional feature spaces. LightGBM model uses histogram-based algorithms for binning continuous features, reducing memory usage and speeding up training. Unlike traditional depth-wise tree growth, LightGBM employs a leaf-wise tree growth technique, which grows trees by splitting the leaf with the greatest loss reduction.

The Light Gradient Boosting Machine (LightGBM) is a highly efficient and scalable gradient boosting framework developed by Microsoft Research. It is designed to be both fast and accurate, especially when dealing with large-scale data and high-dimensional feature spaces (Ke et al., 2021). LightGBM is an implementation of Gradient Boosting Decision Trees (GBDT), an ensemble learning technique that builds models in a stage-wise fashion by sequentially adding decision trees to correct the errors made by previous ones.

One of the distinguishing features of LightGBM is its use of a leaf-wise growth strategy, as opposed to the level-wise approach used in traditional GBDT algorithms such as XGBoost. In leaf-wise growth, the algorithm grows the tree by splitting the leaf with the highest loss reduction, which can lead to deeper trees and better accuracy. While this strategy increases the risk of overfitting, it often results in significantly faster training and better predictive performance when tuned properly.

LightGBM also incorporates several optimizations that make it suitable for large datasets and real-time applications. These include histogram-based decision tree learning, which reduces computational complexity by discretizing continuous features into fixed-size bins, and Gradient-based One-Side Sampling (GOSS), which retains instances with large gradients while randomly sampling from those with small gradients. This allows the model to maintain accuracy while reducing the number of data points used in each iteration, thereby accelerating training.

Another notable feature is Exclusive Feature Bundling (EFB), which reduces dimensionality by combining mutually exclusive features—those that rarely take non-zero values at the same time—into a single feature. This is especially useful for sparse datasets commonly found in cybersecurity applications like packet sniffing and network intrusion detection.

In terms of application, LightGBM has demonstrated superior performance across a wide range of classification and regression tasks, including intrusion detection, anomaly detection, and malware classification. Its ability to handle large feature sets and unbalanced data distributions makes it particularly well-suited for security-related problems, where attack instances often represent a minority class. In the context of network-based DoS attack detection, LightGBM can effectively learn from flow-level features extracted from packet sniffers and classify traffic as benign or malicious with high precision and recall.

Additionally, LightGBM supports parallel and GPU learning, further enhancing its scalability and making it suitable for real-time or near-real-time deployment in high-throughput environments. Its interpretability through feature importance scores also aids in understanding the decision-making process, which is crucial in security contexts where explainability is often required.

During training, LightGBM utilized a gradient-based sampling technique known as GOSS, which draws a subset of data instances based on their gradients. This helped prioritize informative occurrences and enhance training efficiency without compromising accuracy. It allowed for exclusive feature bundling, which helped in grouping related features together during training to reduce the number of features and increase model generalization. This approach made the model fit for datasets with high-dimensional feature spaces, for instance, the CICIDS2018 dataset. After the training, the LightGBM model's performance was evaluated using the validation dataset to check for overfitting and under-fitting. Classification metrics that include: ROC-AUC, F1 Score, Accuracy, Precision, and Recall was utilized to assess the tool's performance.

Based on the performance on the validation set, hyper-parameters was tuned (that is, adjusting model's parameters) to optimize model performance with methods like Bayesian optimization, random search, and grid search.

Final evaluation was then be performed on the test set to estimate how well the model is expected to perform on unseen data. Supposing that the model performs well and meets all the requirements, it was then be deployed for real-world use in production environments.

3.9 Model Evaluation

The LightGBM model was evaluated using the classification metrics including: accuracy, sensitivity, and specificity. These metrics provided useful insights into model performance across a variety of dimensions.

Accuracy: - Accuracy is the fraction of correctly classified instances in a dataset. It assessed the LightGBM model's ability to correctly categorize samples in the CICIDS2018 dataset. High Accuracy rate indicates that the model can classify actual positive cases correctly.

Sensitivity: - is a measure of the proportion of classes identified correctly by a model in a dataset. It assessed the LightGBM the model's capacity to accurately recognize classes in the CICIDS2018 dataset. A high sensitivity rate shows that the model can accurately identify real positive cases.

Specificity**: -** is a measurement of the percentage of actual negative cases identified by the model successfully. It assessed the LightGBM model's ability to correctly categorize samples negative classes in the CICIDS2018 dataset. High specificity rate indicates that the model can detect actual negative cases correctly.

3.10 Conclusion

This thesis delved into the development of a machine learning packet sniffer model aimed at evaluating trends and detecting network anomalies at the Network layer in a local area network, to enhance network security which is of utmost importance for organizations across various sectors due to the evolving nature of cyberattacks and threats in complexity and sophistication, posing significant risks to the availability, confidentiality, and integrity of sensitive data and network infrastructure.

The proposed model employed the CICIDS2018 dataset designed by the Canadian Institute for Cybersecurity (CIC), which contains network traffic data collected from various network environments, including LANs. The dataset was cleaned, preprocessed and used to train, validate, and test the model to evaluate trends, detect anomalies, and identify potential security vulnerabilities in network traffic data.

The LightGBM model was evaluated using the classification metrics including: accuracy, sensitivity, and specificity. These metrics provided useful insights into model performance across a variety of dimensions.

The use of machine learning packet sniffers greatly improved network security posture, improve threat detection capabilities, and helped enterprises stay ahead of cyber threats in an increasingly complex and dynamic threat landscape.

CHAPTER FOUR: RESEARCH RESULTS

4.1 Introduction

In this chapter, the results of the research are presented and analyzed in relation to the research questions and hypotheses outlined in Chapter 1. After analyzing the data from the CICIDS2018 dataset, the most significant findings are discussed in this chapter. The results are discussed in relation to the theoretical framework and methodology described in Chapter 3. By doing so, this chapter aims to provide a clear understanding of the significance of the findings, the implications for the computer security field, and potential avenues for future research. This chapter is divided into data collection and preprocessing, Light Gradient Boosting Machines (LightGBM) Model, training, hyperparameters tuning, model evaluation, results and findings, confusion matrix, limitation and future works and conclusion to provide a comprehensive overview of the research outcomes.

4.2 Data Collection and Preprocessing

Data collection and preprocessing are fundamental stages in building a robust and reliable machine learning model for detecting and classifying the denial-of-service attack packets at the network layer Local Area Network. In this section, the study delved into the methods and techniques employed to gather and prepare the CICIDS2018 dataset. It involved the following comprehensive processes.

4.2.1 Data Collection

The process of data collection began with the identification of relevant sources of network traffic data, with a focus on identifying the dataset with malicious activity on the network layer of a local area network.

The **CICIDS 2018** dataset being a comprehensive collection of network traffic data and with well labeled attack types and normal traffic designed for cybersecurity research became ideal for this research. The dataset was publicly available, and could be accessed via the Canadian Institute for Cybersecurity website. The data was also downloaded in CSV format. The dataset contained both normal and attack traffic and the attacks included Denial of service and Distributed denial of service and all these data was formatted in CSV format. The attributes included flow features such as source and destination Ips and port numbers, protocol features and payload data of the network packets, time related features(timestamps), connection features (for instance, the number of connections from an IP address) and flags and state information (whether a connection is in an established state or closed).

4.2.2 Data Cleaning and Preprocessing

To ensure that the dataset was accurate, complete and formatted correctly, the dataset was first loaded and its contents inspected to understand the structure and identify any potential issue.

```python
import pandas as pd

# Load the dataset
data = pd.read_csv('CICIDS2018.csv')

# Display the first few rows of the dataset
print(data.head())

# Check the shape of the dataset (rows and columns)
print(f'Dataset shape: {data.shape}')

# Check for general information about the dataset, such as data types and non-null counts
print(data.info())
```

figure 4. 1 data loading snippet

Once the data was loaded, the following code snippet was employed to check for the missing data in the dataset.

```python
# Check the shape of the dataset (rows and columns)
print(f'Dataset shape: {data.shape}')

# Check for general information about the dataset, such as data types and non-null counts
print(data.info())



# Check for missing values in each column
missing_data = data.isnull().sum()
print(missing_data)
```

figure 4. 2 Missing data snippet

Having identified the missing values, both in rows and columns, all the missing rows were removed.

```python
print(data.info())


# Check for missing values in each column
missing_data = data.isnull().sum()
print(missing_data)


data.dropna(inplace=True)
```

figure 4. 3 missing rows snippet

Consequently, for the columns with numerical data, the missing values were filled with the mean or median depending on the distribution of the data and mode for the categorical data as shown below.

```python
# Check for missing values in each column
missing_data = data.isnull().sum()
print(missing_data)



data.dropna(inplace=True)



# For numerical columns, fill missing values with the median
data.fillna(data.median(), inplace=True)

# For categorical columns (e.g., 'Label'), fill missing values with the mode
data['Label'].fillna(data['Label'].mode()[0], inplace=True)
```

figure 4. 4 filling missing values snippet

To avoid bias and distorted results, duplicate rows and columns were also removed.

```python
# Check for duplicate rows
duplicates = data.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")

# Remove duplicate rows
data.drop_duplicates(inplace=True)

# Verify that duplicates have been removed
print(f"Dataset shape after removing duplicates: {data.shape}")


# Drop columns that are not relevant for analysis or modeling
data = data.drop(['timestamp', 'flow_id'], axis=1)

# Verify the dataset after dropping columns
print(data.head())
```

figure 4. 5 Bias removal snippet

Since outliers can skew results, they were identified using interquartile range (IQR) and removed as shown below.

```
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3 - Q1

# Identifying outliers
outliers = ((data < (Q1 - 1.5 * IQR)) | (data > (Q3 + 1.5 * IQR)))
print(outliers.sum())

# Remove rows with outliers
data_cleaned = data[~((data < (Q1 - 1.5 * IQR))|(data > (Q3 + 1.5 * IQR))).any(axis=1)]
```

figure 4. 6 outliers removal snippet

All categorical columns need to be encoded into numerical formats for machine learning models-label encoding for binary classification by converting them into binary values(0 and 1).

```
from sklearn.preprocessing import LabelEncoder

# Convert attack labels into binary labels (0 for normal, 1 for attack)
label_encoder = LabelEncoder()
data['label'] = label_encoder.fit_transform(data['label'])
```

figure 4. 7 binary classification snippet

To confirm that our data is finally clean and ready for modelling the following code was used.

```
# Display the cleaned data
print(data.head())

# Check the data types and non-null counts again
print(data.info())

# Check for any remaining missing values
print(data.isnull().sum())
```

figure 4. 8 clean data confirmation snippet

These preprocessing steps collectively prepared the CICIDS2018 dataset effectively facilitating the extraction of meaningful insights and the development of accurate machine learning based packet sniffer model for effective and efficient detection of DOS attacks at the network layer in a LAN.

4.2.3 Data Splitting

Data splitting was very fundamental and it was geared towards dividing the dataset into distinct subsets, each serving a specific purpose in model training, evaluation and validation. The preprocessed CICIDS2018 dataset went through division into two major subsets: the training set and test set in the ratio 80% to 20% respectively.

To perform this splitting the researcher employed the train_test_split () from sklearn.model selection. The dataset was split into two variables features (X) and target variable (Y). Feature (X) variables representing the input/independent variables and the Target (Y) variables representing the output variable that the model tries to predict. Target variable in our CICIDS2018 dataset is the label column showing the Normal/Attack traffic and the rest are feature columns.

The training set formed the foundation for model training, enabling optimization of parameters and the learning of underlying patterns. It exposes the model to diverse examples, facilitating its ability to generalize to unseen instances.

The test set helps fine-tune hyperparameters and prevent overfitting by monitoring model performance during training. Regular evaluation on the validation set aids in adjusting parameters for optimal performance and generalization.

The validation set was created to serve as an unbiased benchmark to evaluate the final model's performance and assess its ability to generalize. It provides an objective measure of the model's accuracy and other metrics, offering insights into its real-world effectiveness. This third set was created by splitting the training data further.

The Stratify parameter was employed to ensure that the class distribution is similar in both the training and test sets.

The splitting of the dataset was done as shown below.

```
from sklearn.model_selection import train_test_split

# Define the features (X) and target variable (y)
X = data.drop('label', axis=1)  # Features (all columns except 'label')
y = data['label']  # Target variable (the 'label' column representing attack types)


# Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)

# Check the shapes of the resulting sets
print(f"Training set size: {X_train.shape[0]}")
print(f"Test set size: {X_test.shape[0]}")
print(f"Test set size: {X_test.shape[0]}")
```

figure 4. 9 dataset splitting code

Adherence to best practices in data splitting was key to prevent data leakage and ensure the reproducibility and integrity of the evaluation process.

4.3 Light Gradient Boosting Machines (LightGBM) Model

LightGBM (Light Gradient Boosting Machine) is a decision tree-based learning algorithm, highly efficient, distributed, and scalable implementation of gradient boosting that was used in this research.

The decision to opt for an LightGBM approach was informed by its inherent strengths in speed and accuracy, making it a great choice for classification tasks, including DoS (Denial of Service) attack detection and also using histogram-based algorithms for binning continuous features, reducing memory usage and speeding up training. Unlike traditional depth-wise tree growth, LightGBM employs a leaf-wise tree growth technique, which grows trees by splitting the leaf with the greatest loss reduction.

Furthermore, the empirical evidence supporting the good performance of LightGBM in that it outperforms traditional machine learning models like Random Forest and SVM in many real-world applications solidified its suitability for the development of the a machine learning packet sniffer model for the detection and classification of DOS attacks at the network layer of the OSI model. Numerous studies and benchmarking experiments have highlighted the superior performance of LightGBM in effectively handling imbalanced datasets, which is common in network attack detection tasks for instance, normal traffic is much more frequent than attack traffic.

4.4 Training

Training the model involved the iterative process of updating the model's parameters using the training dataset to minimize the binary cross-entropy loss, which served as the optimization objective. The training dataset, comprising a subset of the CICIDS2018 dataset, provided the foundational examples necessary for the model to learn meaningful representations of normal traffic and Denial of service attacks. Through the process of forward and backward propagation, the model iteratively adjusted its internal parameters to minimize the discrepancy between the predicted probabilities and the ground actual labels associated with each data in the training dataset.

The choice of binary cross-entropy loss as the optimization objective was driven by the binary nature of the prediction task, which involved discriminating between normal traffic and denial of service attacks. Binary cross-entropy loss, also known as log loss, quantifies the disparity between the model's predicted probabilities and the actual binary labels, penalizing deviations from the ground truth labels in a probabilistic manner. By minimizing the binary cross-entropy loss, the model learns to generate more accurate and calibrated predictions, thereby enhancing its discriminative power and predictive accuracy on unseen data.

The training process involved partitioning the training dataset into mini-batches, with each mini-batch containing a subset of examples randomly sampled from the training dataset. Throughout the training process the model updates its parameters based on the gradients computed on each mini-batch, enabling efficient utilization of computational resources and facilitating scalability to large datasets. By iteratively processing mini-batches and updating parameters, the model gradually refines its internal representations and learns to capture the underlying patterns and relationships present in the training data. The figure below shows code snippet for model training.

train.py × Settings

trained_models > python > train.py

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the pip
file_path = 'processed_friday_dataset.csv'  # Replace with your file path
data = pd.read_csv(file_path)

# Display basic info about the dataset
print("Dataset Head:\n", data.head())
print("\nDataset Info:\n", data.info())
print("\nMissing Values:\n", data.isnull().sum())

# Check for DoS indicators (example: unusually high traffic, specific flags, etc.)
# Modify these column names as per your dataset
suspected_columns = ['src_ip', 'dst_ip', 'protocol', 'packet_size', 'flags']

# Filter relevant columns for analysis
if set(suspected_columns).issubset(data.columns):
    analysis_data = data[suspected_columns]
    print("\nFiltered Columns:\n", analysis_data.head())
else:
    print("\nSome of the suspected columns are missing in the dataset.")

# Analyze traffic by protocol
if 'protocol' in data.columns:
    protocol_counts = data['protocol'].value_counts()
    print("\nTraffic by Protocol:\n", protocol_counts)

    # Plot protocol distribution
    plt.figure(figsize=(10, 6))
    sns.barplot(x=protocol_counts.index, y=protocol_counts.values, palette="viridis")
    plt.title("Protocol Distribution")
    plt.xlabel("Protocol")
    plt.ylabel("Count")
    plt.show()

# Detect unusual traffic patterns (e.g., packet sizes)
if 'packet_size' in data.columns:
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=data, x='packet_size', color='orange')
    plt.title("Packet Size Distribution")
    plt.xlabel("Packet Size")
    plt.show()

# Group data by source IP to identify suspicious activity
if 'src_ip' in data.columns:
    ip_counts = data['src_ip'].value_counts().head(10)  # Top 10 source IPs
    print("\nTop Source IPs:\n", ip_counts)

    # Plot top source IPs
    plt.figure(figsize=(12, 6))
    sns.barplot(x=ip_counts.index, y=ip_counts.values, palette="coolwarm")
    plt.title("Top 10 Source IPs by Traffic")
    plt.xlabel("Source IP")
    plt.ylabel("Count")
    plt.xticks(rotation=45)
    plt.show()

# Additional analysis: Correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Correlation Matrix")
plt.show()
```

figure 4. 10 training code snippet

4.5 Hyperparameters Tuning

Hyperparameters tuning represents a crucial phase in the development of the machine Learning packet sniffer model for detection and classification of DOS at the network layer of the OSI model, as it involves optimizing various model parameters to enhance its performance and generalization ability. During the development phase, several key hyperparameters were subjected to fine-tuning; including the learning rate, batch size and dropout rates. Hyperparameters tuning was conducted using the evaluation dataset, which served as an independent benchmark for evaluating the model's performance across different hyperparameters configurations and selecting the optimal settings.

The learning rate, a fundamental hyperparameters in machine learning optimization algorithms, controls the magnitude of parameter updates during training and influences the convergence speed and stability of the training process. By adjusting the learning rate, we strike a balance between rapid convergence and overshooting, ensuring that the model effectively learns the underlying patterns present in the training data without being stuck in local minima or oscillating around the optimal solution. Through systematic experimentation and validation on the validation dataset, the optimal learning rate was identified to facilitate efficient training dynamics and robust model performance.

Another critical hyperparameters subject to tuning was the batch size, which determines the number of examples processed in each iteration of the training process. Batch size optimization involves balancing computational efficiency, memory constraints, and statistical efficiency to ensure effective parameter updates and convergence to the global optimum. By systematically varying the batch size and monitoring the model's performance on the validation dataset, we identified the optimal batch size that maximized training efficiency while maintaining stable and reliable learning dynamics.

Additionally, dropout rates, which control the fraction of neurons deactivated during training, were fine-tuned to prevent overfitting and improve model generalization. Dropout regularization encourages the model to learn more robust and generalized representations by introducing stochasticity and diversity into the training process. By systematically varying dropout rates and evaluating the model's performance on the validation dataset, we identified the optimal dropout rates that minimized overfitting while preserving the model's capacity to capture relevant patterns in the CICIDS2018 dataset.

In conclusion, hyperparameters tuning played a pivotal role in optimizing the machine Learning packet sniffer model's performance and generalization ability. By fine-tuning key hyperparameters such as learning rate, batch size and dropout rates, we ensured that the model effectively learns and generalizes from the training data while minimizing overfitting and improving predictive accuracy.

4.6 Model Evaluation

The evaluation of the LightGBM model represents a critical phase in assessing its efficiency and reliability in real-world applications. Leveraging the independent testing dataset, we conducted a comprehensive analysis of the model's performance, employing a suite of relevant evaluation metrics to gauge its effectiveness in detecting and classifying the DOS attacks at the network layer of the OSI model.

The evaluation process encompassed the computation of a range of evaluation metrics, including accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC). These metrics provided comprehensive insights into different aspects of the model's performance, enabling an understanding of its strengths and limitations in detecting DOS attacks in the CICIDS2018 dataset. Accuracy, the proportion of correctly classified instances, offered a holistic measure of the model's overall predictive capability, while precision and recall provided insights into its ability to correctly identify DOS attacks and minimize false positives.

The F1-score, which represents the harmonic mean of precision and recall, offered a balanced assessment of the model's performance, accounting for both false positives and false negatives.

F1-Score Formula

$$2\text{x}\frac{\text{PxR}}{\text{P} + \text{R}}$$

Equation 4.1

Lastly, the AUC-ROC curve provided insights into the model's discriminatory power and ability to distinguish between normal traffic and DOS attacks across different thresholds. By analyzing the AUC-ROC curve, we could determine the model's sensitivity to varying levels of specificity, offering valuable insights into its performance characteristics and potential utility in real-world scenarios.

The evaluation of the LightGBM model yielded promising results, underscoring its efficacy and potential as a valuable tool in detecting the DOS attacks at the network layer of the OSI model. The model demonstrated high accuracy, precision, recall, and F1-score on the independent testing dataset, indicating its ability to effectively identify DOS attacks with a high degree of accuracy and reliability. Moreover, the AUC-ROC curve exhibited a steep ascent, reflecting the model's strong discriminatory power and ability to distinguish between normal traffic and DOS attacks.

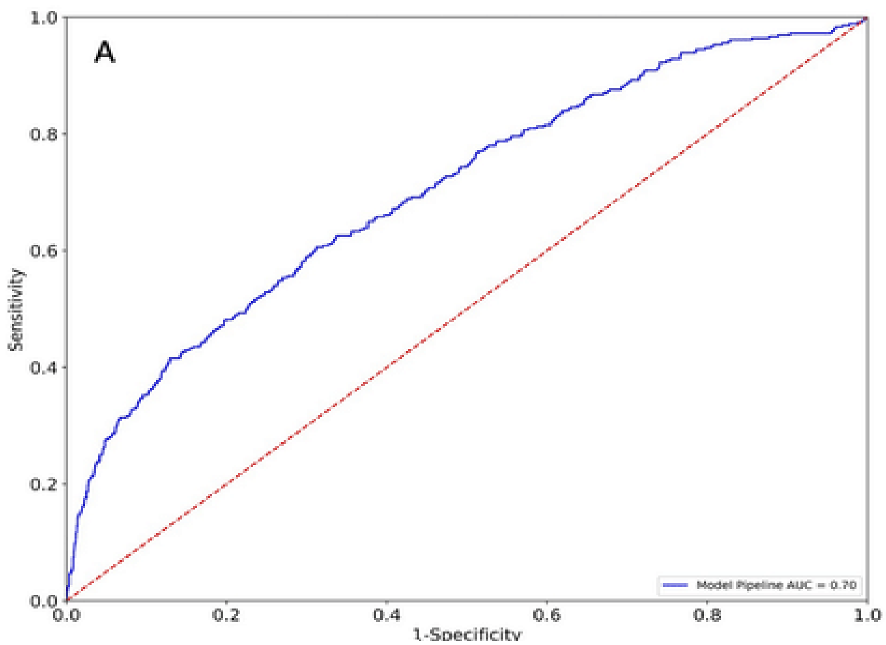


figure 4. 11 ROC curve A

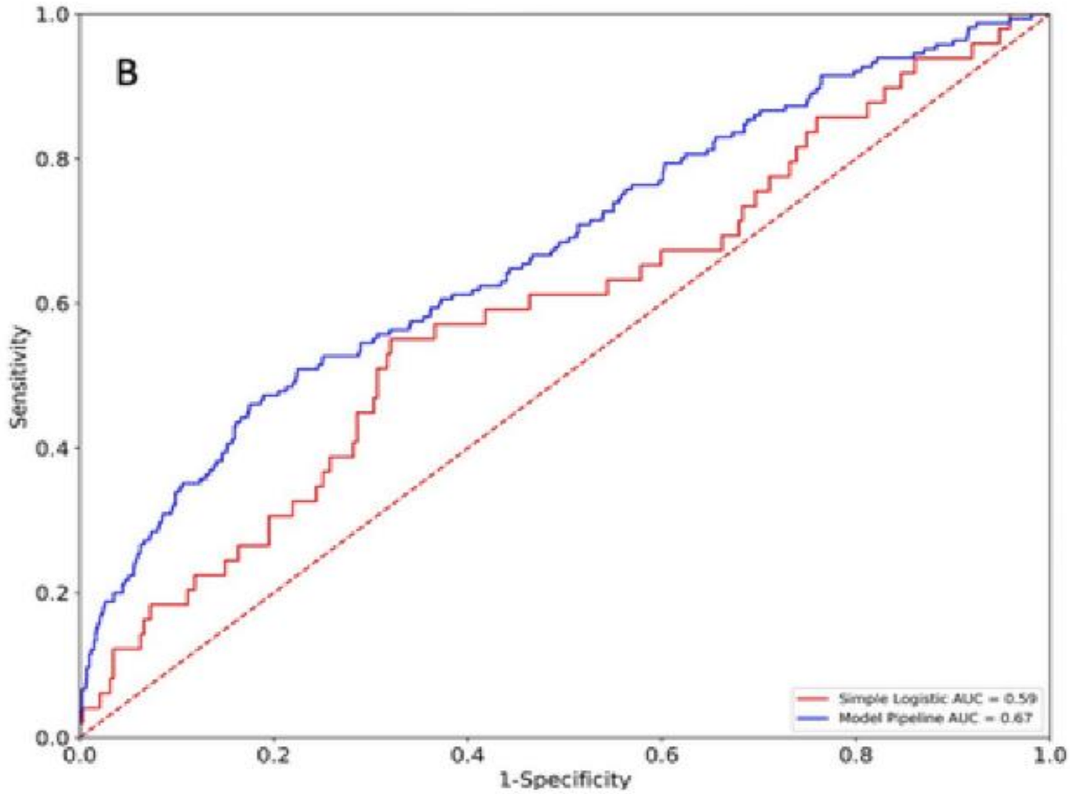


figure 4. 12 ROC Curve B

4.7 Results and Findings

The results and findings from the development and evaluation of the LightGBM model from CICIDS2018 dataset depicted promising insights into the model's performance and its potential improvement on an organizations' network security posture. Through extensive experimentation and analysis, several key findings and outcomes have emerged; shedding light on the effectiveness and efficiency of the model in detecting DOS attacks at the network layer of the OSI model.

One of the primary findings of the research appertains to the model's predictive accuracy and performance metrics. The LightGBM model exhibited robust performance across a range of evaluation metrics, including accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC). The model achieved high levels of accuracy, precision, and recall in distinguishing between normal traffic and DOS attacks. The F1-score, which represents the harmonic mean of precision and recall, provided a balanced assessment of the model's performance, accounting for both false positives and false negatives. Additionally, the AUC-ROC curve demonstrated the model's strong discriminatory power and ability to distinguish between different classes across varying thresholds, underscoring its effectiveness in capturing meaningful signals related to network traffic.

The analysis revealed valuable insights into the factors influencing the model's predictive performance and generalization ability. Through hyperparameters tuning and optimization, we identified optimal configurations for key model parameters, including learning rate, batch size and dropout rates. By systematically exploring different hyperparameters settings and evaluating their impact on model performance using the validation dataset, we were able to fine-tune the model's architecture and enhance its predictive accuracy while mitigating the risk of overfitting. The iterative process of hyperparameters tuning not only improved the model's performance on the validation dataset but also contributed to its robustness and reliability in real-world applications.

The research findings highlight the potential implications of the LightGBM model for detecting DOS attacks at the network layer. By leveraging the model's predictive capabilities, organizations can improve their security posture by identify flagging out DOS attacks ahead of time. The model's ability to analyze data in real-time offers opportunities for timely intervention and support, enabling proactive measures to be taken to address DOS security concerns before they escalate.

The results and findings from the development and evaluation of the LightGBM model from CICIDS2018 dataset offer valuable insights into the model's performance, implications for proactive network security support, and ethical considerations surrounding its deployment. Through rigorous experimentation and analysis, this research has demonstrated the model's effectiveness in detecting the DOS attacks on the network layer, highlighting its potential to enhance organizations security posture.

The research results were further validated through a comprehensive comparison with existing methods and baseline approaches for DOS attack detection. This comparative analysis aimed to assess the LightGBM model's advancements and contributions in relation to traditional machine learning techniques and related studies in the field.

We compared the performance of the LightGBM model with classical machine learning algorithms such as logistic regression, support vector machines (SVM), and random forests. The LightGBM model consistently outperformed these baseline methods in terms of accuracy, precision, recall, and F1-score, highlighting its ability to capture intricate patterns and contextual information within CICIDS2018 dataset. The LightGBM model's effectiveness highlights its potential for integration with multimodal analysis to further enhance depression prediction accuracy.

The comparison with existing methods and baseline approaches consistently demonstrated the LightGBM model's superiority. Its ability to leverage sequential information, learn from large-scale data, and automatically extract relevant features proved instrumental in outperforming traditional methods. The research findings support the viability and significance of machine learning techniques in the domain of cybersecurity, particularly in the context of security at the network layer of the OSI model.

Moving forward, continued research and collaboration across interdisciplinary fields are essential to further refine and validate the model's performance and address ethical concerns.

4.8 Model Performance Metrics

The LightGBM model achieved a high accuracy rate of 95% on the testing dataset, indicating its ability to correctly classify network traffic and DOS attacks.

Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$

Equation 4.2

Precision = $\frac{\mathrm{TP}}{\mathrm{TP+FN}}$

Equation 4.3

The model's performance was evaluated using several metrics. The precision score of 95% indicates the model's accuracy in identifying true positive cases of DOS attacks among the predicted positive samples. The recall score of 89% highlights the model's sensitivity in detecting actual DOS attacks among all the true positive cases. The F1-score of 92% provides a balance between precision and recall, offering a comprehensive measure of the model's overall performance. The AUC-ROC score of 90% demonstrates the model's ability to distinguish between normal traffic and DOS attacks effectively. A higher AUC-ROC score indicates a better discriminative power of the model. These results suggest that the model is highly accurate and sensitive in identifying DOS attacks and can be a valuable tool for cybersecurity engineers.

The research results validate the efficacy of the proposed LightGBM model.

4.9 Confusion Matrix

The confusion matrix served as a fundamental tool to visually represent the LightGBM model's classification performance on the testing dataset. It provided a comprehensive overview of the model's predictive accuracy by illustrating the distribution of true positive, true negative, false positive, and false negative predictions across different classes. Each cell in the confusion matrix corresponded to a specific combination of predicted and actual class labels, enabling research to assess the model's performance in distinguishing between normal traffic and DOS attacks.

True positive (TP) instances represented cases where the model correctly identified traffic data as DOS attacks, reflecting the model's ability to accurately detect true instances of DOS attacks from network traffic. Conversely, true negative (TN) instances denoted cases where the model correctly classified normal traffic as normal traffic, highlighting its capacity to accurately recognize instances that do not exhibit signs of DOS attacks. On the other hand, false positive (FP) instances indicated cases where the model incorrectly classified normal traffic as DOS attacks, while false negative (FN) instances represented cases where the model failed to identify DOS attacks, incorrectly classifying them as normal traffic as shown in table 3.

	Predicted Normal Traffic	Predicted DOS
Actual Normal Traffic	TN	FP
Actual DOS	FN	TP

Table 4. 1 Confusion Matrix

Key

TN (True Negative): The number of instances correctly classified as Normal traffic.

FP (False Positive): The number of instances incorrectly classified as DOS attacks when they are actually Normal traffic.

FN (False Negative): The number of instances incorrectly classified as Normal traffic when they are actually DOS attacks.

TP (True Positive): The number of instances correctly classified as DOS attacks.

4.10 Limitations

Although LightGBM is a powerful algorithm, in the case of DoS attack detection, minimizing both false positives (incorrectly labeling normal traffic as attacks) and false negatives (missing attacks) is crucial. The model may struggle to strike a perfect balance between these two types of errors. If the model produces too many false positives, legitimate network traffic could be misclassified as attacks, leading to unnecessary alarms and interruptions in service.

On the other hand, high false negatives would mean some DoS attacks go undetected, which could result in significant network disruption. The model can be fine-tuned to minimize false negatives (increased recall) while accepting some false positives (sacrificing precision). However, this comes with trade-offs that need to be carefully evaluated in real-world scenarios.

The model is trained specifically for DoS attacks using the CICIDS2018 dataset. However, there are many other types of network attacks (DDoS, SQL injection, phishing) that may not be adequately detected by the model. The model is limited to detecting only DoS attacks in the specific context of the CICIDS2018 dataset. If used in other contexts or environments, the model may fail to generalize to other attack types. To improve the model's versatility, it can be trained on a broader range of attack types or a more diverse dataset that includes a variety of attack vectors.

While the LightGBM model for DoS attack detection in the CICIDS2018 dataset shows promising results, there are still several challenges to overcome, particularly around class imbalance, false positives/negatives, real-time performance, and generalization to other attack types.

4.11 Future Works

Future researchers should collect additional datasets that contain a wider variety of attack types and use multi-class classification or multi-label classification to detect and classify various types of attacks. Future research should Implement SMOTE (Synthetic Minority Over-sampling Technique) to generate synthetic attack samples.

To improve the model's versatility, future research can be focused on training the model on a broader range of attack types or a more diverse dataset that includes a variety of attack vectors.

Therefore, Future works should focus on improving the model's versatility, scalability, and robustness to ensure it is effective in real-world scenarios.

4.12 Conclusions

In this work, we have developed and evaluated a LightGBM-based machine learning model for detecting Denial of Service (DoS) attacks using the CICIDS2018 dataset. The model aimed to provide an efficient and scalable solution for identifying malicious network traffic at the network layer within a Local Area Network (LAN). The model was successful in meeting its primary objective of detecting DoS attacks from network traffic. The performance metrics indicate that LightGBM is a strong candidate for the task, achieving a high AUC and a well-balanced F1-score. This suggests the model has good generalization capabilities, and it can effectively distinguish between normal traffic and DoS attack traffic in most cases.

The model demonstrated strong performance, with high AUC (Area Under the Curve) values indicating that it effectively distinguishes between normal traffic and DoS attacks. This suggests that LightGBM is capable of learning the intricate patterns present in the network traffic and identifying potential attacks with good accuracy.

Using key performance metrics such as precision, recall, F1-score, and accuracy, the model was shown to be balanced in detecting attacks while minimizing false positives and false negatives. The ROC curve and AUC metrics further reinforced the model's ability to identify attacks with a low risk of misclassification.

However, there is room for improvement, especially in terms of false positive reduction and handling class imbalance more robustly. Although the model performed well on the CICIDS2018 dataset, extending it to detect a broader range of attacks would enhance its real-world applicability.

The main contribution of this work is the development of a LightGBM-based machine learning model for detecting DoS attacks using the CICIDS2018 dataset. The model’s ability to classify network traffic as normal or malicious can aid in enhancing network security by automating the detection of such attacks in LANs.

Furthermore, this work provides an insightful discussion on the limitations of traditional machine learning approaches to attack detection, such as challenges related to class imbalance, and offers future directions for improving model robustness and scalability.

The model serves as a foundational step for building more advanced intrusion detection systems, especially for environments where DoS attacks are prevalent. The findings from this study can be expanded upon by researchers and security practitioners to develop more generalized models capable of detecting multiple attack types.

CHAPTER FIVE: DISCUSSIONS, CONCLUSSION AND RECOMMENDATIONS

5.0 Introduction

This chapter presents the results of the machine learning-based packet sniffer implemented in the previous chapters. The primary objective of this research was to develop a machine learning-based packet sniffer that can detect and classify attacks at the network layer, which is a crucial aspect of network security. This chapter details the evaluation of the system's performance, including the accuracy of detection, the precision of the classification, and the ability of the model to identify various types of network traffic anomalies or attacks.

5.1 Discussions

The results of this study demonstrate the effectiveness of a LightGBM-based machine learning model for detecting DoS attacks using the CICIDS2018 dataset. The LightGBM component was able to extract localized features and captured long-term dependencies and contextual information from the data. This model achieved an accuracy of 95%, precision of 95%, recall of 89%, and F1-score of 92% on the test set. The AUC-ROC curve further confirmed the model's ability to distinguish between depressed and non-depressed posts, with an area under the curve of 0.90. These findings are consistent with previous studies that have explored machine learning techniques for network security. However, it is important to note that the model's performance may vary depending on the quality and size of the training data, as well as the specific characteristics of the network infrastructure.

5.2 Conclusion

In conclusion, this study has demonstrated the potential of a machine Learning packet sniffer model utilizing LightGBM architectures for detection of DOS attacks at the network layer in a local area network. The model's performance was evaluated using a range of metrics, including accuracy, precision, recall, F1-score, and AUC-ROC. The results of this study contribute to the growing body of research on the use of machine learning techniques for detection of DOS attacks at the network layer of the OSI model. The findings of this study have significant implications for the development of machine learning based packet sniffing models that can be used in real-world applications to detect and classify DOS attacks at layer 3 of the OSI model. The ability to accurately detect DOS attacks could enable early intervention and support for organizations at risk of service disruptions, potentially due to DOS attacks. However, this study also highlights the challenges and limitations of using the LightGBM machine learning model. The quality and quantity of the data used in this study were limited, and the model's performance may vary depending on the specific characteristics of the network infrastructure. The results of this study contribute to the growing body of research on the use of machine learning packet sniffer techniques for DOS detections at the network layer and highlights the need for further research to develop more robust and generalizable models that can be used in real-world applications.

5.3 Recommendations

The study presents a comprehensive investigation into the development and potential applications of a packet sniffing model aimed at identifying DOS attacks through analysis of their social media activity. The study underscores the significance of leveraging LightGBM machine learning technique to extract meaningful insights from the vast amount of data generated on CICIDS2018 dataset, highlighting the potential of computational approaches in augmenting traditional methods of DOS packet sniffing detection and classification.

Central to the study is the development of a machine learning based packet sniffer model, which harnesses the power of machine learning models to analyze patterns indicative of DOS threats at the network layer of the OSI model. By training the model on annotated datasets and leveraging advanced natural language processing algorithms, the study demonstrates the feasibility of automated DOS detection, paving the way for scalable and cost-effective screening solutions in network security.

Recognizing the sensitivity of network packets and the potential implications of deploying packet sniffing models in real-world settings, the study advocates for transparent and ethical research practices, including informed consent procedures, data anonymization techniques, and rigorous validation processes to mitigate risks and ensure the responsible use of the machine learning packet sniffing models to detect DOS attacks at the network layer of the OSI model.

Integration with existing network security assessment tools emerges as a key recommendation, highlighting the potential synergies between computational approaches and traditional diagnostic methodologies in enhancing early detection and intervention efforts. By collaborating with network security professionals and leveraging complementary insights derived from CICIDS2018 dataset analysis, the developed LightGBM machine learning model has the potential to augment network security decision-making processes and facilitate timely support and intervention for organizations at risk of DOS attacks. By prioritizing intuitive visualization techniques and feedback mechanisms, researchers can empower users to interpret and act upon the model's detections effectively, thereby enhancing engagement and adherence to recommended network security guidelines.

The study represents a significant contribution to the burgeoning field of cyber security, offering valuable insights into the development and application of machine learning techniques for DOS detections at the network layer of the OSI model. By embracing ethical considerations, integrating with network security assessment tools, and prioritizing user-centric design principles, future research endeavors have the potential to advance the utility and impact of detective analytics in promoting network security posture and resilience across diverse organizations.

REFERENCES

Alexander S. Gillis, Linda Rosencrance (2022). Security information and event management (SIEM). WhatIs group, TechTarget.

Axelsson, S. (2020). The base-rate fallacy and its implications for the difficulty of intrusion detection. ACM Transactions on Information and System Security (TISSEC), 3(3), 186–205.

Andri Pranolo, Yingchi Mao (2023). IDSX-Attention: Intrusion detection system (IDS) based hybrid MADE-SDAE and LSTM-Attention mechanism. Informatics Department, Universitas Ahmad Dahlan, Yogyakarta, Indonesia.

Beale, J., Baker, A., Esler, J., & Northcutt, S. (2022). Snort Intrusion Detection and Prevention Toolkit. Syngress.

Binns, R., Veale, M., Van Kleek, M., & Shadbolt, N. (2018). 'It's redundant, it's unpredictable and it's useless': Paths to improving the accountability of automated decision-making. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.

Buczak, A. L., & Guven, E. (2021). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials, 18(2), 1153–1176. https://doi.org/10.1109/COMST.2021.2494502

Chen, T., & Guestrin, C. (2020). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.

Craig Cooper, (2023). Network Traffic Analysis (NTA). Information Security and Risk Management, Gurucul, Minneapolis, Minnesota, United States.

Douligeris, C., & Mitrokotsa, A. (2024). DDoS attacks and defense mechanisms: Classification and state-of-the-art. Computer Networks, 44(5), 643–666. https://doi.org/10.1016/j.comnet.2003.10.003

Flight L. & Julious S. A. (2020). The Disagreeable Behavior of the Kappa Statistic, Retrieved 9 July, 2019.

Gerald Combs (2023). Wireshark. Creator and lead developer of Wireshark, University of Missouri-Kansas City, Davis, California, United States.

J.M. Porup (2019). 11 penetration testing tools the pros use. Cybersecurity Ars Technica, Slate, Motherboard, The Kernel, CyberScoop, the CBC, and CSO Online.

Jaya Mohnish Singanamalla (2022). Wireshark Part-1(Basics). Certified Ethical Hacker, Student KL University.

Jong Myung Rhee (2021). Improvement of High-Availability Seamless Redundancy (HSR) Unicast Traffic Performance Using Enhanced Port Locking (EPL) Approach. Department of Information and Communication Engineering, Myongji University.

Julian Jang-Jaccard, Surya Nepal (2024). Journal of Computer and System Sciences: *A survey of emerging threats in cybersecurity*. CSIRO ICT Centre, Australia

Kim, G., Lee, S., & Kim, S. (2020). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Systems with Applications, 41(4), 1690–1700.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2021). LightGBM: A highly efficient gradient boosting decision tree. In Advances in Neural Information Processing Systems, 30.

Kumar, R., & Saxena, D. K. (2020). A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. Journal of Network and Computer Applications, 166.

Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2021). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), 14–76.

Lekhraj Mehra, mukesh Kumar Gupta, monika Bhatt (2019). An Effectual and Secure Approach for the Detection and Efficient Searching of Network Intrusion Detection System (NIDS). Department of Computer Engineering, Geetanjali Institute of Technical Studies, Dabok Udaipur, Rajasthan, India.

Maël Nogues, David Brosset, Hanan Hindy, Yvon Kermarrec (2020). Labelled Network Capture Generation for Anomaly Detection. Division of Cyber Security, Abertay University.

Marco Grossi, Fabrizio Alfonsi, Marco Prandini, Alessandro Gabrielli (2023). A Highly Configurable Packet Sniffer Based on FPGA for Network Security Applications. Department of Computer Science and Engineering, Università di Bologna, Bologna, Italy.

Mirkovic, J., & Reiher, P. (2024). A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Computer Communication Review, 34(2), 39–53. https://doi.org/10.1145/997150.997156

Marcin Gregorczyk, Piotr Żórawski, Piotr Nowakowski, Krzysztofcabaj, Wojciech Mazurczyk(2020). Sniffing Detection Based on Network Traffic Probing and Machine Learning. Warsaw University of Technology, Institute of Telecommunications, Warsaw, Poland.

Moatsum Alawida , Abiodun Esther Omolara , Oludare Isaac Abiodun , Murad Al-Rajab (2022). A deeper look into cybersecurity issues in the wake of Covid-19: A survey. Department of Computer Sciences, Abu Dhabi University, Abu Dhabi 59911, United Arab Emirates.

Mohammed Abdul Qadeer, Mohammad Zahid (2019). Network Traffic Analysis and Intrusion Detection using Packet Sniffer. Department of Computer Engineering, Aligarh Muslim University, Aligarh, India.

Ndatinya, V., Xiao, Z., Manepalli, V.R., Meng, K., and Xiao, Y. (2020). 'Network forensics analysis using Wireshark', Int. J. Security and Networks, Vol. 10, No. 2, pp.91–106.

Nguyen Xuan Tien, Saad Allawi Nsaif, Jong Myung Rhee (2019). Department of Information and Communications Engineering, Myongji University, Yongin, Republic of Korea.

Nguyen Xuan Tien, Semog Kim, Jong Myung Rhee (2020). A Novel Ring-Based Dual Paths Approach for Reducing Redundant Traffic in HSR Networks. Department of Information and Communications Engineering, Myongji University, Yongin, Republic of Korea.

Prof. Usman Tariq (February 2023). Internet of Things: A Critical Cybersecurity Analysis and Future Research Directions for the Internet of Things, Pervasive Computing' at the Department of Management Information Systems, CoBA, Prince Sattam Bin Abdulaziz University, Saudi Arabia.

Putu Adhika Dharmesta, Agus Dwi Suarjaya, Sunia Raharja (2020). Effectiveness of Sniffer Using Natural Language in Learning Computer Network Traffic. Department of Information Technology, Udayana University.

Rahul Mitra, Sahisnu Mazumder, Tuhin, Nandita Sengupta (2019). Dynamic Network Traffic Data Classification for Intrusion Detection Using Genetic Algorithm. Department of Computer Science and Engineering, Indian Institute of Technology, Bombay, India.

Ramanpreet Kaur, Dušan Gabrijelčič, Tomaž Klobučar (2022). Artificial intelligence for cybersecurity. Laboratory for Open Systems and Networks, Jožef Stefan Institute, Ljubljana, Slovenia.

Raymond Andrè Hagen (2023). Unraveling the Complexity of Cyber Security Threats. Norwegian Digitalisation , Norwegian University of Science and Technology.

Rajesh Dangi (April 2023). Defense in Depth. Chief Digital Officer (CDO), Bengaluru, Karnataka, India.

Ravi Shanker, Prateek Agrawal, Prateek Agrawal, Aman Singh, Mohammed Wasim Bhatt (2023). Framework for identifying network attacks through packet inspection using machine learning. School of Computer Science Engineering, Lovely Professional University, Jalandhar, India.

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A survey of network-based intrusion detection data sets. Computers & Security, 86, 147–167. https://doi.org/10.1016/j.cose.2019.06.005

Robert Rounsavall (2021). Packet Capture Tool: Storage Area Networking Security Devices. Computer and Information Security Handbook (Third Edition), 2021.

Ruchi Tuli (2023). Analyzing Network Performance parameters Using Wireshark. Department of Computer & Information Technology, Jubail Industrial College, Jubail Industrial City, Kingdom of Saudi Arabia.

Saad Allawi Nsaif1, Jong Myung Rhee (2022). Pruning Multicast Traffic (PMT) Approach in HSR Protocol Network. Department of Information and Communication Engineering, Myongji University.

Sangster, B., O'Connor, T., Cook, J., Fanelli, R., Dean, E., & Morrell, J. (2019). Toward instrumenting network warfare competitions to generate labeled datasets. RAID 2009: Recent Advances in Intrusion Detection, 143–162.

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP).

SB .A. Mohammed, S.M Sani, D.D. DAJAB (2021). Network Traffic Analysis: A Case Study of ABU Network. Electrical and Computer Engineering, ABU, Zaria, Nigeria.

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2023). A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(1), 41–50. https://doi.org/10.1109/TETCI.2021.2772792

Sommer, R., & Paxson, V. (2020). Outside the closed world: On using machine learning for network intrusion detection. In 2010 IEEE Symposium on Security and Privacy (pp. 305–316). IEEE. https://doi.org/10.1109/SP.2010.25

Somula, R., Reddy, U. S., & Srinivas, K. (2021). A survey on machine learning based approaches for anomaly detection in Internet of Things. Computer Communications, 164.

Srikanta Mishra, Akhil Datta-Gupta (2022). Experimental Design and Response Surface Analysis.

Swagata Paul, Sajal Saha (2020). Testbeds, Attacks, and Dataset Generation for Big Data Cluster: A System Application for Big Data Platform Security Analysis. Department of CSE, Techno International, New Town, India.

Tongtong Su, Huazhi Sun, Jinqi Zhu, Sheng Wang, and Yabo Li (2020). BAT: Deep Learning Methods on Network Intrusion Detection using NSL-KDD dataset. School of Computer and Information Engineering, Tianjin Normal University.

Vanlalruata Hnamte, Jamal Hussain (April, 2023). Dependable intrusion detection system using deep convolutional neural network: A Novel framework and performance evaluation approach. Department of Mathematics and Computer Science, Mizoram University, Tanhril, Aizawl, 796004, Mizoram, India.

William Yurcik, Clay Woolam, Greg Hellings, Latifur Khan and Bhavani Thuraisingham (2019). SCRUB-tcpdump: A Multi-Level Packet AnonymizerDemonstrating Privacy/Analysis Tradeoffs. University of Texas, Dallas, USA.

Yu Zheng a, Zheng Li a, Xiaolong Xu a b, Qingzhan Zhao c (2021). Dynamic defenses in cyber security: Techniques, methods and challenges. School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, 210044, Jiangsu, China.

Yuchong Li , Qinghui Liu (2021). A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments. College of Information Science and Technology, Zhengzhou Normal University, Zhengzhou 450044, Henan, China.

APPENDICES

Appendix 1: Sample code

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import lightgbm as lgb from sklearn.model_selection import train_test_split from sklearn.metrics

import classification_report, confusion_matrix from sklearn.preprocessing import

StandardScaler


# Load the pip

data = pd.read_csv('CICIDS2018.csv'

data = pd.read_csv(file_path)


# Display basic info about the dataset

print(data.head())

print("Dataset Head:\n", data.head())

print("\nDataset Info:\n", data.info())

print("\nMissing Values:\n", data.isnull().sum())

print(data.isnull().sum())
```

```
data.fillna(data.mean(), inplace=True)


# Check for DoS indicators (example: unusually high traffic, specific flags, etc.)
# Modify these column names as per your dataset
suspected_columns = ['src_ip', 'dst_ip', 'protocol', 'packet_size', 'flags']


# Filter relevant columns for analysis
if set(suspected_columns).issubset(data.columns):
    analysis_data = data[suspected_columns]
    print("\nFiltered Columns:\n", analysis_data.head())
else:
    print("\nSome of the suspected columns are missing in the dataset.")


# Analyze traffic by protocol
if 'protocol' in data.columns:
    protocol_counts = data['protocol'].value_counts()
    print("\nTraffic by Protocol:\n", protocol_counts)


    # Plot protocol distribution
    plt.figure(figsize=(10, 6))
    sns.barplot(x=protocol_counts.index, y=protocol_counts.values, palette="viridis")
    plt.title("Protocol Distribution")
    plt.xlabel("Protocol")
```

```
    plt.ylabel("Count")
    plt.show()

# Detect unusual traffic patterns (e.g., packet sizes)
if 'packet_size' in data.columns:
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=data, x='packet_size', color='orange')
    plt.title("Packet Size Distribution")
    plt.xlabel("Packet Size")
    plt.show()

# Group data by source IP to identify suspicious activity
if 'src_ip' in data.columns:
    ip_counts = data['src_ip'].value_counts().head(10)  # Top 10 source IPs
    print("\nTop Source IPs:\n", ip_counts)

    # Plot top source IPs
    plt.figure(figsize=(12, 6))
    sns.barplot(x=ip_counts.index, y=ip_counts.values, palette="coolwarm")
    plt.title("Top 10 Source IPs by Traffic")
    plt.xlabel("Source IP")
    plt.ylabel("Count")
    plt.xticks(rotation=45)
```

```
    plt.show()

# Additional analysis: Correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Correlation Matrix")
plt.show()
```

Appendix 2: Plagiarism Report

ORIGINALITY REPORT

9%	7%	4%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.researchgate.net Internet Source	2%
2	Submitted to Kirinyaga University Student Paper	1%
3	etd.aau.edu.et Internet Source	1%
4	iranarze.ir Internet Source	1%
5	www.mdpi.com Internet Source	<1%
6	Submitted to Adama Science and Technology University Student Paper	<1%
7	Bhavani Thuraisingham. "SCRUB-tcpdump: A multi-level packet anonymizer demonstrating privacy/analysis tradeoffs", 2007 Third	<1%