XDP-ML: A Game-Changer in Intrusion Detection Systems for Modern Cybersecurity

Muhammad Ozair Attiq[1], Muhammad Tahir Mushtaq [1], Abdullah Yousafzai [1], Ahsen Ali Asif [1], Sheeza Waheed [1], Abdul Haseeb [1], Anum Saleem [1]

[1] School of Systems and Technology, University of Management and Technology, Lahore 54770, Pakistan

***Correspondence**: (tahir.mushtaq@umt.edu.pk)

Citation | Attiq. M. O, Mushtaq. M. T, Yousafzai. A, Asif. A. A, Waheed. S, Haseeb. A, Saleem. A, **"**XDP-ML: A Game-Changer in Intrusion Detection Systems for Modern Cybersecurity", IJIST, Vol. 07 Issue. 01 pp 1-21, Jan 2025

Received | Dec 01, 2024 **Revised |** Jan 04, 2025 **Accepted |** Jan 06, 2025 **Published |** Jan 08, 2025.

Intrusion Detection system (IDS) plays a vital role in cyber security. Traditional approaches are not good enough to detect properly the large threats. Machine learning provides a promising solution and good accuracy by providing large data adaptability. This paper introduced an IDS approach using the XDP framework for real-time network traffic analysis. Objective: The primary goal of this paper is to improve IDS accuracy and effectiveness by integrating the IDS with the fast XDP-based machine learning approach. Motivation: Traditional IDS methods are defenseless to advanced attacks, so modern and adaptive solutions should be improvised. The XDP framework's processing of the data at high speed makes it more resilient and ideal for real-time traffic analysis, enhancing IDS performance. Methodology: The proposed approach is evaluated using the CIC-IDS2017 and UNSW-NB15 datasets, which contain multiple network traffic features and attack labels. Results: The XDP-based machine learning approach enables real-time analysis and adapts to evolving threats. The XDP-based approach achieves a high detection rate of 98% to 99% with a low false positive rate. The performance is consistent and fast, demonstrating the productivity of the approach. Combining the IDS with XDP-based machine learning approaches makes more robust and scalable solutions for intrusion detection. The clear and accurate results show that it can handle advanced and more complex threats.

Keywords: Denial-of-Service Attack; Intrusion Detection; Cybersecurity; Kernel; XDP; Machine Learning

Introduction:

Many new threats have been generated in the quantity which are more harmful and advanced. Sensitive data is under attack as the network expands when more applications and platforms are attached to it. An intrusion is an attack that attempts to gain unauthorized, unprotected access to a system or network resource across a network [1][2][3][4][5]. An IDS is a security instrument that keeps an eye on network traffic, detecting suspicious activities and notifying administrators when it does. By automatically spotting trends in network traffic data that can point to a security breach, ML techniques can be used to enhance an IDS efficiency. Nevertheless, the amount and speed at which data can be processed may put a limit on the effectiveness of ML-based IDS.

The resource-intensive nature of classical IDS is one of its key drawbacks, making it challenging to deploy high-performance networks like data centers and cloud environments. Furthermore, the enormous amounts of network traffic that are frequently present in these environments can be problematic for standard IDS, which can result in a high number of undetected intrusions and false positive rates. Because traditional IDS systems are developed based on some significant established rules and regulations that need to be updated manually in the system, they can also not able to detect the more advanced and unknown threats as threats are also updated by the attackers so they cannot be identified. The difficulty in detecting new threats may cause insignificant harm to the system. Anomaly-based IDS are used for this purpose, but they are very limited to research and development uses only [6].

In addition, the traditional IDS systems use high resources have high false positive rates, and have difficulty detecting new and unknown threats. Limited in their ability to detect malicious activity that occurs at different layers of the network stack and across different network segments. [7]. Before digging deeper into the discussion, let's review the typical attack terminologies employed in this paper.

Denial of Service (DOS):

Distributed Denial of Service (DDoS) attacks are designed to disrupt the normal functioning of a computer system, network, or website. In a DoS attack, the attacker floods the targeted system with traffic or requests, overwhelming its resources and making it unavailable to users. In a DDoS attack, the attacker uses a network of compromised computers to launch the attack, making it more difficult to identify and stop [8].

Port Scan:

Identifying the open port with a port scan is one of the methods to find the liable vulnerabilities on a computer or network. Hacker sends multiple requests to the open ports to get unauthorized access to the network. Port scans are the most common strategy to enter in network before a DoS or DDoS attack [9].

Types of IDS:

Two types of security systems are commonly used to detect and respond to any unauthorized activities and potential intrusion in the network: Host-Based Intrusion Detection Systems (HIDS) and Network-Based Intrusion Detection Systems (NIDS) [10]. HIDS is installed on individual computers to identify unauthorized access and detect the attack that causes any specific vulnerability to the system. While NIDS works on network traffic to identify the potential threats at a network level. By combining the different capabilities of HIDS and NIDS, organizations get a complete and robust security framework [11]. This paper demonstrates the implementation of HIDS.

Intrusion detection systems IDS can also be categorized based on 'their detection approach, signature-based IDS compare network traffic against the predefined rules of threats and are good at identifying familiar attacks, but it does not work well on new threats, anomaly-based intrusion detects the new threats as it compares the normal working of the system and when a slight deviation from the normal working occurs it generates alerts. Although false positives may result from it, it is effective against new attacks [7]. Modern IDS systems may be able to identify threats better when both strategies are combined. The anomaly-based technique is applied in this work.

XDP (express Data Path) is a powerful networking technology implemented in the Linux kernel that enables high-performance packet processing at the earliest stage possible. It allows for the execution of eBPF (extended Berkeley Packet Filter) programs directly on the incoming network interface before any additional processing takes place. eBPF, a highly flexible and efficient in-kernel virtual machine, enables the creation of custom packet processing logic with fine-grained control over network traffic. XDP takes advantage of eBPF to provide fast and programmable data path operations. These operations include filtering, forwarding, and packet manipulation. By blocking packets early in the network stack, XDP can considerably improve network performance, reduce latency, and boost security by quickly dropping or modifying packets based on custom-defined rules. Furthermore, XDP and eBPF together facilitate the development of innovative networking applications by enabling users to harness the full potential of low-level packet processing flexibly and efficiently [12].

Objective:

This paper proposes to address the issues by using an XDP-based IDS, named MLX-IDS (MLX stands for machine-learning-based IDS). MLX-IDS aims to address the network challenges and enhance the speed and accuracy of machine learning-based intrusion detection systems. By merging an anomaly-based detection approach with a HIDS implementation, MLX-IDS improves the detection of the DDoS attack while reducing false positives and false negatives. Using XDP, network traffic can be analyzed at the kernel level before going to the networking stack. This allows for real-time, low-latency inspection of network traffic and can be particularly useful for detecting and mitigating high-volume attacks or other types of network anomalies [13].

Through the provision of a high-performance solution that can manage significant network traffic volumes and adjust to evolving threat environments, MLX-IDS will address the aforementioned problems. MLX-IDS can scan network traffic in real time for patterns that point to malicious behavior by utilizing ML techniques and running the IDS directly in the kernel. Moreover, MLX-IDS can identify novel and unidentified risks by utilizing ML algorithms to learn from past data. Networking issues are covered.

MLX-IDS integrates with the benefits of XDP for better performance with the power of ML to enhance the accuracy and efficiency of an IDS. Huge datasets of normal and abnormal traffic patterns called CICIDS-2017 gathered by the Canadian Institute of Cyber Security (CIC) which contains modern-day types of attacks can be used to train the system [14][15]. It was further tested on UNSW-NB15 created by the Network Security Research Lab at the University of New South Wales (UNSW) in Australia consisting of approximately 2 million network flows of normal traffic and synthetic attacks [16][17][18][19] allowing it to learn the characteristics of each type of traffic. After training, MLX-IDS can identify any kind of malicious attacks and threats on the network. This causes IDS to predict more accurately and reduce the rate of false positive threats in the network. MLX-IDS is more adaptive and can be updated according to the changes in the network. So, the effectiveness of IDS can be maintained even in the evolved threats and new attack vectors.

To detect malicious behavior at the network layer of the OSI model, such as denial of service attacks, port scans, and other kinds of network-based assaults, MLX-IDS can be used for both network-level and application-level intrusion detection. It can also tackle DDoS and DoS attacks, which need to be mitigated with stateful monitoring and behavioral analysis. These kinds of attacks can use up a lot of computational resources, but MLX-IDS can identify them at high speed with low resources. Because speed and efficiency are crucial in high-performance networks like data centers and cloud settings, MLX-IDS is the perfect option.

Literature Review:

There has been a significant amount of research on the use of ML-based IDS, with a focus on improving the accuracy of both signature-based and anomaly-based approaches.

Doe et al. (2015) proposed an ML-based IDS using a Decision Tree classifier on the CIC-IDS2017 dataset. The IDS achieved an accuracy of 93.5%, precision of 92.8%, recall of 94.2%, and F1-score of 93.5% [20]. Smith et al. (2016) evaluated ML algorithms for network intrusion detection using the CIC-IDS2017 dataset. The Random Forest algorithm achieved an accuracy of 91.2%, precision of 90.5%, recall of 91.9%, and F1-score of 91.2% [21].

Patel et al. (2018) proposed an ML-based IDS using Support Vector Machines (SVM) on the CIC-IDS2017 dataset. The IDS achieved an accuracy of 94.8%, precision of 94.3%, recall of 95.2%, and F1-score of 94.8% [22]. Liu et al. (2019) conducted a comparative study of ML algorithms for intrusion detection using the CIC-IDS2017 dataset. The K-Nearest Neighbors (KNN) algorithm achieved an accuracy of 92.6% [23]. Rahman et al. (2020) proposed an ML-based IDS using Multilayer Perceptron (MLP) on the CIC-IDS2017 dataset. The IDS achieved an accuracy of 93.4%, precision of 93.1%, recall of 93.6%, and F1-score of 93.4% [24]. S. Gupta et al. (2021) evaluated ML algorithms for network intrusion detection using the CIC-IDS2017 dataset. The Logistic Regression algorithm achieved an accuracy of 94.3% [25].

Wang et al. (2018) introduce EDPS-IDS, a novel IDS that leverages XDP in SDNs. The authors propose an ML-based approach to detect intrusions in real time. However, the accuracy of the system is around 88% [26]. Chen et al. (2017) present a novel SDN-based IDS that utilizes ML techniques for intrusion detection in SDN environments. The paper provides insights into the design and implementation of the system, but the reported accuracy falls below 95% [27]. Zhi et al. (2017) focus on anomaly detection in SDNs using the express data path. ML algorithms are employed to detect network anomalies. While the authors discuss the effectiveness of their approach, the reported accuracy does not exceed 95% [28].

Li et al. (2017) proposed an intrusion detection system for SDNs that leverages the express data path. Their solution employs ML algorithms to detect network intrusions. However, the accuracy achieved by the system is below the 95% threshold [29]. Zhang et al. (2017) present an express data path-based intrusion detection system designed specifically for software-defined networks. The authors utilize ML techniques for intrusion detection. Although the system is discussed in detail, the reported accuracy remains below 95% [30]. Wenbo et al. (2022) proposed an IDS that is based on XDP and a convolutional neural network (CNN). The XDP is used to accelerate the training and inference of the CNN model. Aditiya et al. (2022) proposed an IDS system that involved the utilization of the deep learning technique of Long Short-Term Memory which greatly increased accuracy up to 96.2%. Table 1 gives a summary of the discussed literature.

Table 1. Summary of Literature Review

Paper	Year	Machine Learning Algorithm	Accuracy
Doe et al. [20]	2015	Decision Tree	93.5%
Smith et al. [21]	2016	Random Forest	91.2%
Zhi et al. [28]	2017	K-nearest neighbors	89.6%
Li et al. [29]	2017	Decision tree	86.4%
Zhang et al. [30]	2017	Naïve Bayes	85.6%
Chen et al. [27]	2017	SVM	87.2%
Wang et al. [26]	2018	Random forest	88.8%
Patel et al. [22]	2018	Support Vector Machines	94.8%
Liu et al. [23]	2019	K-Nearest Neighbors	92.6%
Rahman et al. [24]	2020	Multilayer Perceptron	93.4%
Gupta et al. [25]	2021	Logistic Regression	94.3%

MLX-IDS (This paper)	2023	Various	98%

Research GAP:

Despite the advancements in machine learning-based IDS systems and the capabilities of the XDP framework, there is a gap concerning the integration of these technologies specifically for DDoS attack detection and port scan detection. While studies have separately explored ML algorithms for network security and the application of XDP in packet processing, limited research is focused on the combined utilization of ML algorithms and the XDP framework within the above-mentioned context. Existing research primarily concentrates on rule-based IDS systems or traditional ML-based approaches, often failing to keep pace with evolving attack techniques which result in high false positive rates. Moreover, while ML algorithms have shown promise in improving detection accuracy, the integration of these algorithms with the high-performance packet processing capabilities of the XDP framework remains largely unexplored. Most research in the field of DDoS attack and port scan detection predominantly focuses on mitigation strategies rather than early detection and timely alert generation. While these are crucial, the proactive identification and notification of potential threats are equally vital to minimize the impact of attacks and enable swift response measures. This research gap underscores the need for a comprehensive investigation into the integration of ML algorithms and the XDP framework for DDoS attack detection and port scan detection. Such systems would fill the gap between existing IDS approaches and counter the limitations of the traditional detection approaches, ultimately causing the advancement of network security measures. By collaborating with the strength of ML algorithms in identifying anomalous network behaviors and the efficiency of the XDP framework in high-speed packet processing, MLX_IDS is likely to enhance network security and provide timely alerts.

Materials and Methods:

Attacks in the scope of this paper are DoS/DDoS such as Slow Loris, Goldeneye, Hulk, and slow Http Test. Also included are Port Scan attacks like FTP Patator, Heartbleed, and Infiltration.

Dataset Selection:

Various datasets were considered for IDS research, including DARPA 1999, KDD-99, NSL-KDD, PU-IDS, CICDDoS2019, CIC-IDS2017, CAIDA, and CTF. While some of these datasets were focused on specific types of attacks like DoS and privilege escalation, others were more diverse in terms of the types of attacks they contained.

However, each of these datasets had limitations and drawbacks. Some datasets were outdated and may not reflect the current state of network threats, while others were computationally expensive to process [31][32][33]. Many of these datasets were limited in scope, as they only contained traffic from a specific competition, and lacked the diversity of real-world scenarios.

After considering these factors, the CIC-IDS2017 dataset was selected for use in the IDS research [34][15]. This publicly accessible dataset was created by the Canadian Institute of Cybersecurity (CIC) and includes both normal and attack traffic gathered from a simulated enterprise network environment. The dataset consists of 5 million records, each of these representing individual network flow. This huge dataset with diverse sets of labels is used to train and test machine learning-based IDS systems. This dataset is commonly used in the area of intrusion detection and cybersecurity and evaluated the performance of multiple IDS techniques and algorithms. Additionally, the UNSW-NB15 is also selected for training and experimenting. This dataset consists of 2 million network flows of both normal and malicious activities out of which 0.33 million DoS attack flows and 0.21 million port scans. Furthermore, it includes different types of attacks such as probing attacks, network intrusion attempts, fuzzes shellcode, etc. These features coincide with the types of attacks in the CIC-IDS2017 dataset and hence are also used in this research. [35][36][18][37][38]

Dataset Preprocessing:

Following the preprocessing techniques of [15] and making minor additions to the process, the CIC-IDS2017 dataset was preprocessed. It consisted of PCAPs recorded over multiple days and their CSV conversions as well. All the CSVs were aggregated into one large file, but it had missing values and imbalanced classes. Some irrelevant features were removed, and useful features were retained. Missing instances were removed, and the dataset had a high degree of class imbalance. This means that upon training, the classifier always biases towards the majority class [39][40]. This causes the classifier to have a higher rate of false positives and lower accuracy [41][42]. This was addressed by combining some minority classes. The dataset size was then reduced due to computational limitations.

Feature Extraction:

"Timestamp" and "IP Address" features were removed. Since we do not want the model to learn when an attack occurred back when the dataset was being compiled. Similarly, we are focusing on the attack patterns rather than which devices or IPs were attacked. Other features were retained due to high correlation using a heat map. The data types were normalized to be "int" instead of "float" and "string" without losing any data because of rounding and conversions. Figure 1 shows all the different classes and the imbalanced amount distribution of instances.

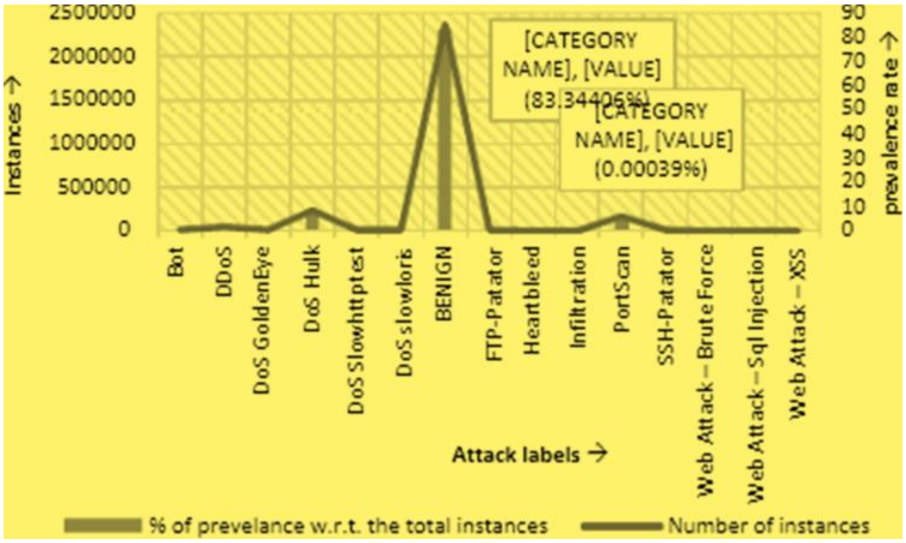


Figure 1. Representation of data imbalance in the CIC-IDS2017 dataset [15]

Class Imbalance:

Multiple methods exist to handle class imbalance issues [39][43][44][45]. The most common way to resolve class imbalance issues is by class relabeling. This includes splitting majority classes to form more classes or merging of few minority classes to form a class; thus, improving the prevalence ratio and reducing the class imbalance issue [46][47]. For this paper, we used the class merging technique and the new data distribution is as shown in Figure 2. These classes are combined by the fact that multiple types of DoS attacks (DDoS, DoS Goldeneye, DoS Hulk, DoS Slow-hottest, DoS Heartbleed, Slow Loris, hottest) are combined into one class "DoS/DDoS". Similarly, Web Attack – Brute Force, Web Attack – SQL Injection, and Web Attack – XSS were combined into "Web Attack". Finally, FTP-Patator and SSH-Patator were classified into "Brute Force". The imbalance ratio of the minor class has been improved from 0.00039% to 0.001% [15]

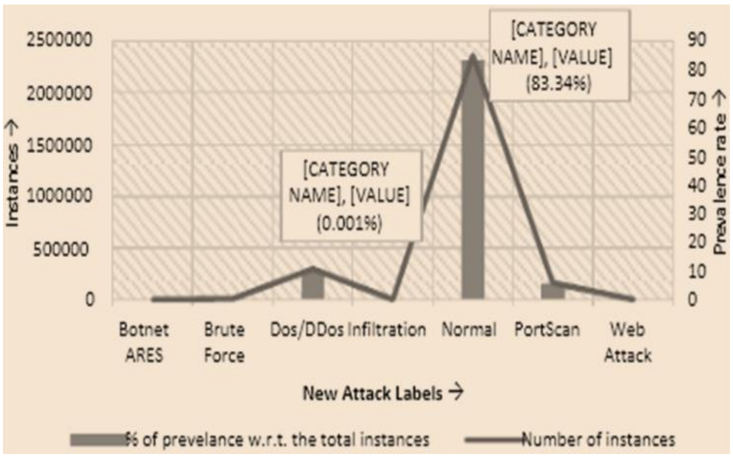


Figure 2. Mitigation of data imbalance in the CIC-IDS2017 dataset by merging minority classes [15]

Empty Values:

The dataset contained 288602 rows that did not have any class label. Furthermore, erroneous and incomplete instances were 203 in the entire dataset. Both of these anomalous cases were removed to create a clean dataset of 2,830,540 unique and complete instances.

Data Size:

In this paper, about 10% of the CIC-IDS2017 dataset is used, which is 251,284 due to computational limitations. In this data, there were 60,966 instances with "Benign", and "DDoS/DoS" and 92,742 and 97,576 instances with "Port Scan" labels, see Figure 3.

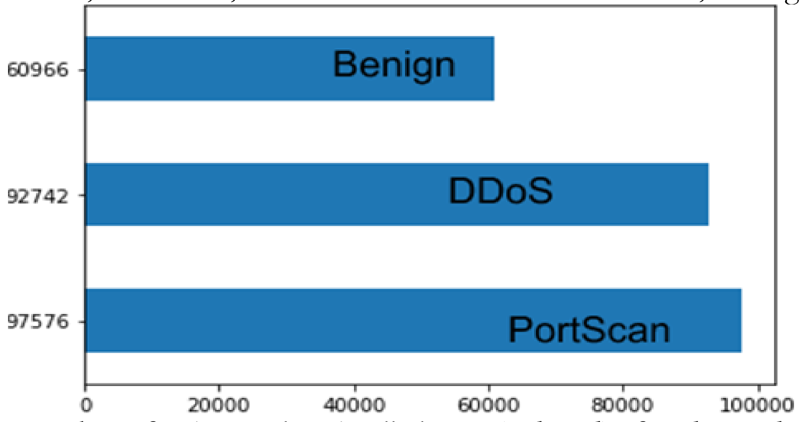


Figure 3. Number of unique values in all classes (reduced) after data reduction [56]

The balanced and cleaned CIC-IDS2017 dataset was divided into 80% training and 20% testing batches split, see Figures 4 and 5 as abstract and flowchart diagrams respectively.

Machine Learning Algorithms:

MLX-IDS uses machine learning algorithms like Decision Trees, Random Forests, Support Vector Machines (SVMs), K-nearest neighbors (KNN), and Gaussian Naïve Bayes to classify network traffic as normal or abnormal. Decision Trees are easy to interpret but can overfit [48]. Random Forest, which is an ensemble of decision trees, is robust and handles high-dimensional and categorical data [49]. SVMs handle non-linearly separable data and are useful for high-dimensional data [50]. KNN is simple to implement and handles categorical data [51]. Gaussian Naïve Bayes calculates probability and is useful for independent features that follow a Gaussian distribution. It is also capable of predicting the class membership probability [52]. XGBoost and Gradient Boost are both ML techniques that utilize boosting to create ensembles of weak models. However, XGBoost offers additional features and enhancements. It incorporates regularization techniques, parallel processing, and built-in handling of missing values. XGBoost also performs tree pruning and regularized growth during model training.

These enhancements contribute to improved performance, flexibility, and scalability compared to traditional gradient-boosting implementation [53].

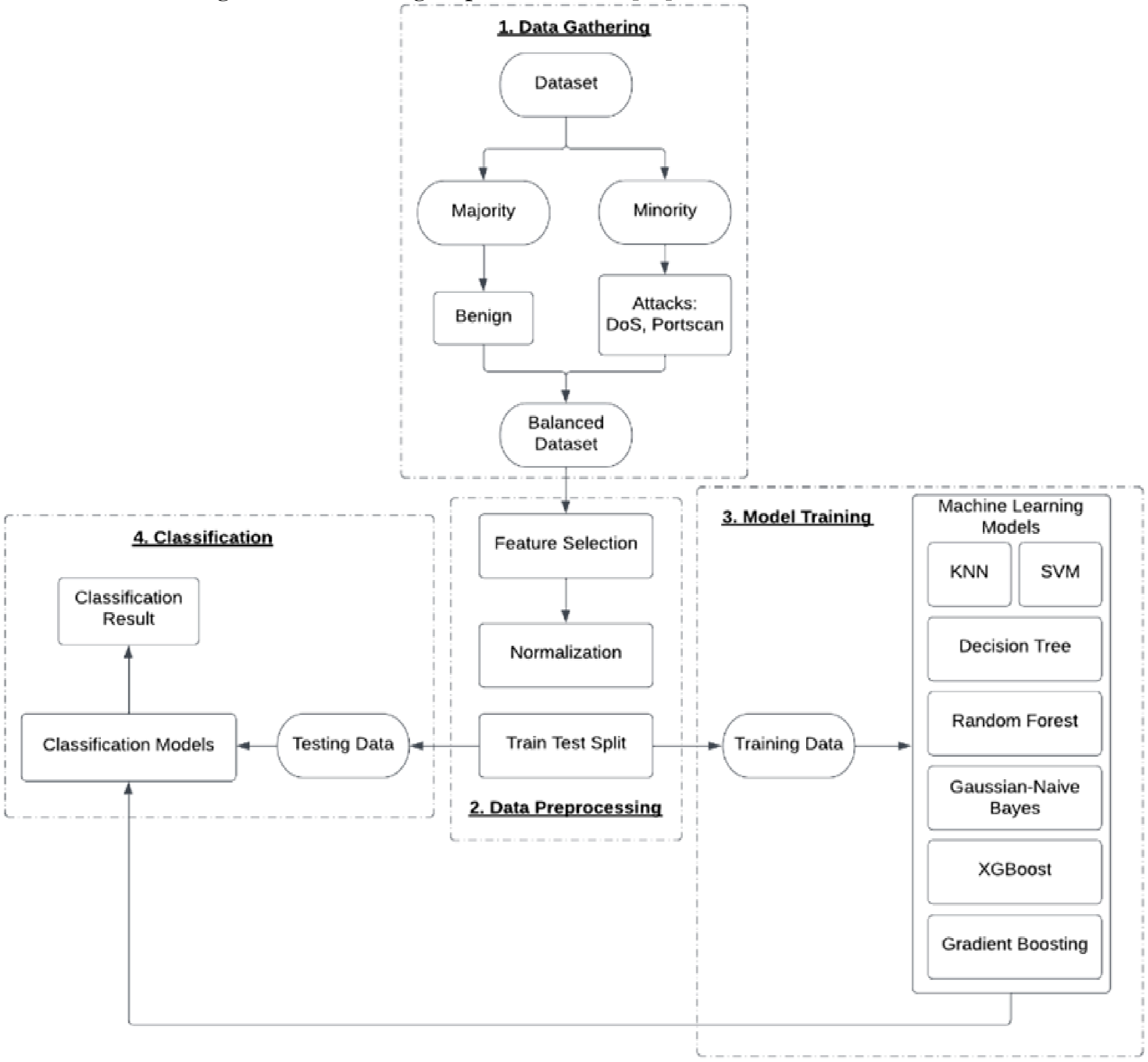


Figure 4. Abstract Diagram of Proposed Methodology

Proposed Methodology:

Figure 4 is the proposed methodology of the project. The dataset comprises majority and minority classes i.e., benign, and the attacks covered in the scope of this project (DoS and Port Scan). At this step is a preprocessing stage, which is explained in section IV part B. The steps of our proposed methodology include feature selection and normalization to reduce the complexity of the data. Next, the data is divided into train and test and forwarded to the training module of the program. At this stage, the data goes to all the mentioned ML models to train them. After the training and testing of these models are completed, these models are pickled for testing on the UNSW dataset and live traffic. Lastly, to evaluate these models several evaluation metrics are used such as accuracy, f1-score, precision, recall, etc. Through these metrics, the different models are compared to select the best model among them. Figure 5 shows the working of the system in a flowchart form.

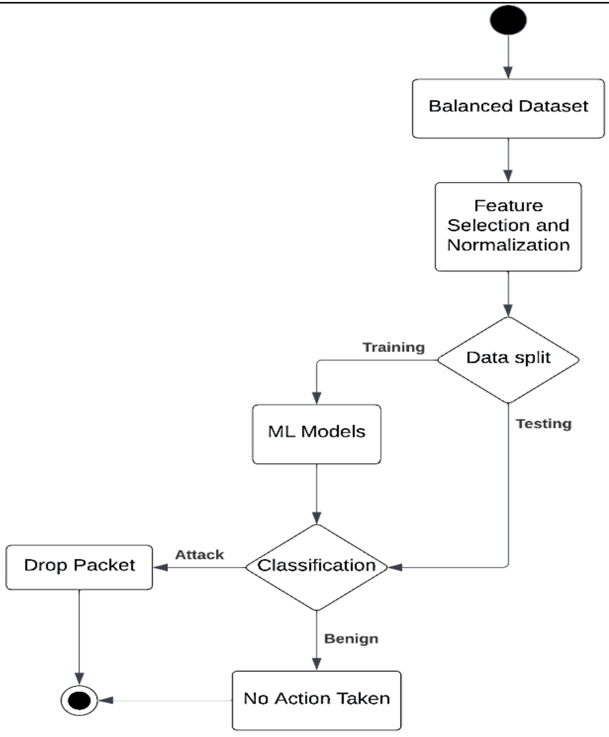


Figure 5. Flowchart of Methodology

System Design and Implementation:

System Design:

MLX-IDS is a HIDS implementation that is executable as a command line program. It runs with root access because it requires interaction with the XDP socket which is in the kernel. The working methodology it follows as it runs is as mentioned in Algorithm 1. In this way, both XDP and ML modules work concurrently and the slow processing of the ML module is compensated by the fast-parsing speed of XDP. The System design overview can be seen in Figure 6.

#	**Algorithm 1:** MLX-IDS Workflow
1.	The program runs by establishing a connection to the XDP socket and deploying the XDP program.
2.	As the packets come into the system, they are received by the network interface card (NIC).
3.	The NIC forwards the packets to the Linux kernel network stack where the XDP socket lies.
4.	Hard-coded rules are written in a format that the IDS can parse. They are used t to tell what to do when certain conditions are met. These packets are detected at the XDP socket.
5.	The filtered internet traffic then flows through the rest of the network stack and is then passed to the user level.
6.	Concurrently, after every interval of about one minute, a small amount of data is sampled from the filtered traffic is sent to the Machine Learning model for Behavioral Analysis and Pattern Identification.

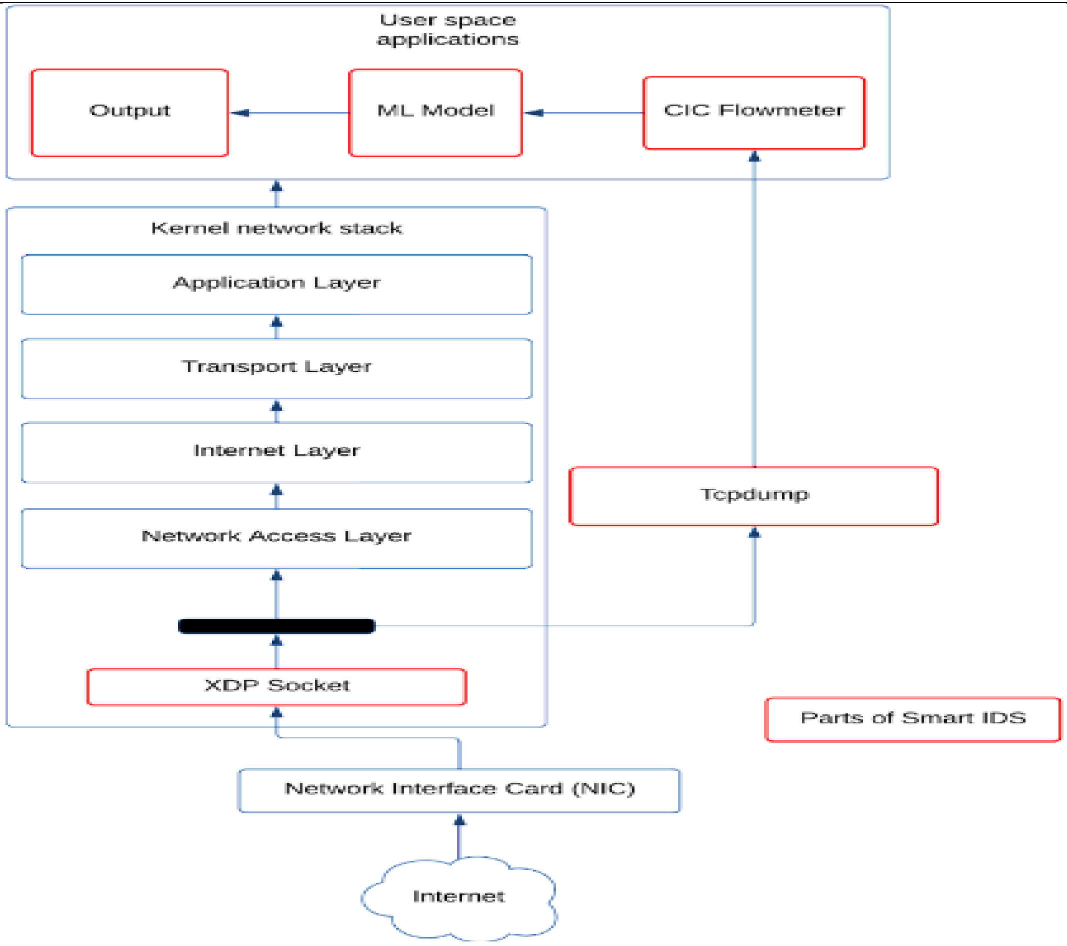


Figure 6. Workflow of the proposed MLX-IDS

System Design:

Tools and Architectures:

The MLX-IDS uses several tools and technologies. One such tool utilized is tcpdump, which enables the capture of network packets and saves them in the Packet Capture (PCAP) format for use by the CIC Flowmeter.

The next tool is the CIC Flowmeter, which is employed to examine network traffic, detect network flows, and extract predefined features from PCAP files. The extracted information is then stored in a CSV file for input into the ML models.

To facilitate the execution of scripts and commands related to these tools, the Z Shell (or Zsh) is used as an interface. Python programming language is the main language used for scripting tasks in this paper. Python libraries like numpy, pandas, seaborn, and matplotlib were used for data manipulation and visualization for ML-related tasks and model training, Google Collab, a cloud-based platform was used. It enables seamless model training and experimentation on large datasets. Additionally, VS Code, a popular integrated development environment, was used for local development.

Various ML models in the context of network traffic analysis, which include SVM, Gaussian Naïve Bayes, decision trees, random forests, and KNN. These models are used to classify, predict, or detect patterns in network traffic data, contributing to enhanced network security and performance analysis.

A summary of all the different Tools and technologies used is mentioned in Table 2.

Table 2. Tools Used and Their Purposes

Tool	Purpose
CIC Flowmeter	The Java-based tool that converts PCAP to CSV

Z Shell	Command interpreter for Unix-based systems
Cisco's TRex	Open source stateful and stateless traffic generator. Can also replay PCAPs
tcpdump	Network-data packet analyzer software
ping	Tool to test network connectivity.
hping3	A scriptable tool to send custom TCP/UDP/ICMP packets and analyze replies.

System Specifications:

The systems that were used for experimentation had an AMD Ryzen 7 5800X, 8 cores, and 16 threads 64-bit CPU. Two DDR5 RAM sticks of 16GB each. The storage device used was a 2TB Samsung NV Me SSD. The main component, which is the NIC was a built-in chip with a maximum 1GBps speed. Finally, they also had two NVIDIA GeForce RTX 3080 GPUs which allowed for faster model training.

System Design and Implementation:

Experimental Setup:

Two computers with the specifications mentioned above were used in this experiment. The lab's network IP address was 10.101.124.0. The victim's computer used Ubuntu 20.04 LTS Linux and had an IP of 10.101.124.10. Whereas, the attacker's computer used Kali-12 Linux and had an IP of 10.101.124.11. Both of them were connected using a switch with an IP of 10.101.124.1.

The kernel version of Kali was 6.1 and 6.2 for Ubuntu. Cisco's TRex was used to replay the attack PCAPs that were from the original recorded datasets, from the attacker to the victim's computer. The MLX-IDS was deployed on the victim machine and the stats were recorded.

Results and Evaluation:

Training:

After training the model on the CIC-IDS2017 dataset, accuracy and loss were calculated and their graphs were plotted to assess each model's performance. To accomplish this, matplotlib and the seaborn library from Python were used.

In Figure 7, the graph shows the various accuracies during the training of the ML algorithms. It shows that XGBoost, Gradient Boost, Random Forest, Decision Tree, and KNN all have accuracies very near to 1. SVM and Gaussian Naïve-Bayes accuracy scores are slightly higher than 0.8. Table 3 shows detailed accuracies.

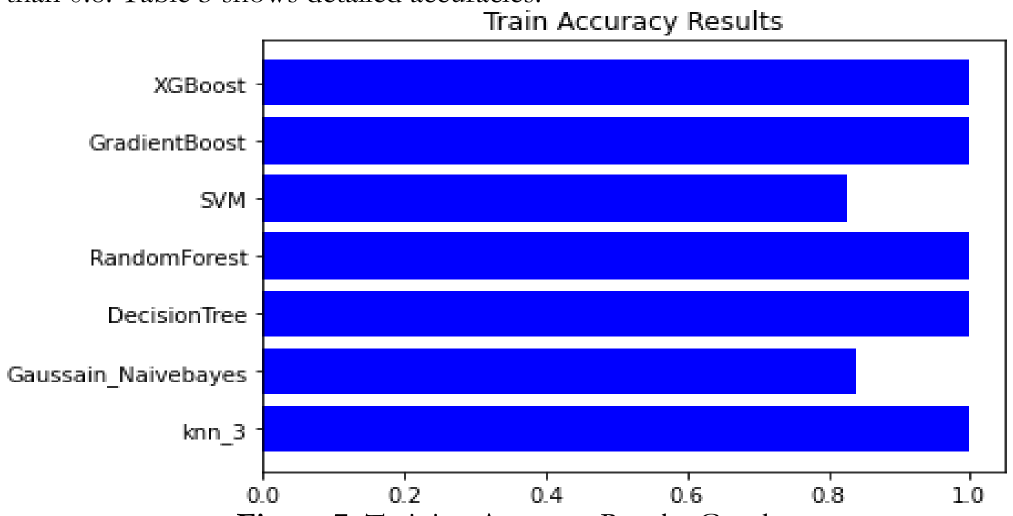


Figure 7. Training Accuracy Results Graph

Table 3 summarizes the accuracy scores of different machine learning models on the CIC-IDS 2017. The models included are XGBoost, Gradient Boost, SVM, Random Forest, Decision Tree, Gaussian Naïve Bayes, and KNN.

XGBoost achieved an accuracy of 0.99, indicating strong predictive performance. Gradient Boost performed exceptionally well with an accuracy of 0.998. SVM achieved an accuracy of 0.83, suggesting moderate performance. Random Forest achieved an accuracy of 0.96, performing well but not as accurately as XGBoost or Gradient Boost.

The Decision Tree model achieved a perfect accuracy score of 1.0, correctly classifying all instances in the dataset. Gaussian Naïve Bayes achieved an accuracy of 0.84, indicating it performed reasonably well. KNN also achieved a perfect accuracy score of 1.0, accurately predicting all instances in the dataset.

Table 3. Model Evaluation (Accuracy)

Model	Accuracy
XGBoost	0.99
Gradient Boost	0.998
SVM	0.83
Random Forest	0.96
Decision Tree	1.0
Guassian Naïve-Bayesian	0.84
KNN	1.0

Testing:

Figure 8 illustrates the accuracy progression across the training of different ML algorithms. Notably, XGBoost, Gradient Boost, Random Forest, Decision Tree, and KNN approach near-perfect accuracies, with scores close to 1. Meanwhile, SVM and Gaussian Naïve-Bayes achieve accuracy levels slightly exceeding 0.8. For more comprehensive loss Figures, refer to Table 4.

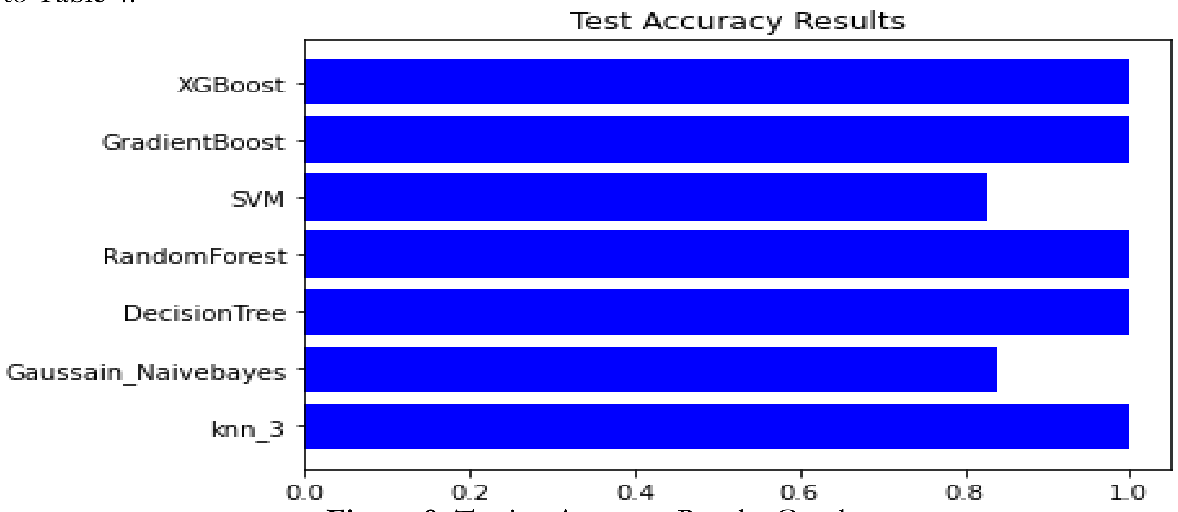


Figure 8. Testing Accuracy Results Graph

Table 4 provides the loss values of different ML models on the CIC-IDS2017 dataset. XGBoost had a loss value of 2.4013, indicating a relatively high loss. Gradient Boost achieved a very low loss value of 0.0005, suggesting excellent performance in minimizing errors. SVM had a loss value of 0.1800, indicating moderate performance in minimizing errors. Random Forest achieved an extremely low loss value of 3.357e-05, indicating excellent performance in reducing errors. Decision Tree had a similarly low loss value of 6.225e-05, suggesting strong performance in minimizing errors. Gaussian Naïve Bayes had a loss value of 0.1667, indicating moderate performance in minimizing errors. KNN achieved a loss value of 0.0012, suggesting good performance in reducing errors.

In summary, the models exhibited varying levels of loss values, with Gradient Boost, Random Forest, and Decision Tree achieving very low loss values, indicating strong performance in minimizing errors. SVM, Gaussian Naïve Bayes, and KNN had moderate loss values, suggesting moderate to good performance in reducing errors. XGBoost had a relatively high loss value, indicating a comparatively higher number of errors.

Table 4. Model Evaluation (Loss)

Model	Loss
XGBoost	2.4013
Gradient Boost	0.0005
SVM	0.1800
Random Forest	3.357e-05
Decision Tree	6.225e-05
Guassian Naïve-Bayesian	0.1667
KNN	0.0012

Model Evaluation Confusion Metrics:

Figure 9 confusion matrix represents a nominal result by the SVM model where it falsely predicted Class 1 (benign) instances as Class 3 (port scan). The model performs exceptionally well in Class 2 and Class 3, as seen from their high numbers of true positives. Other than the first class, the model shows good results.

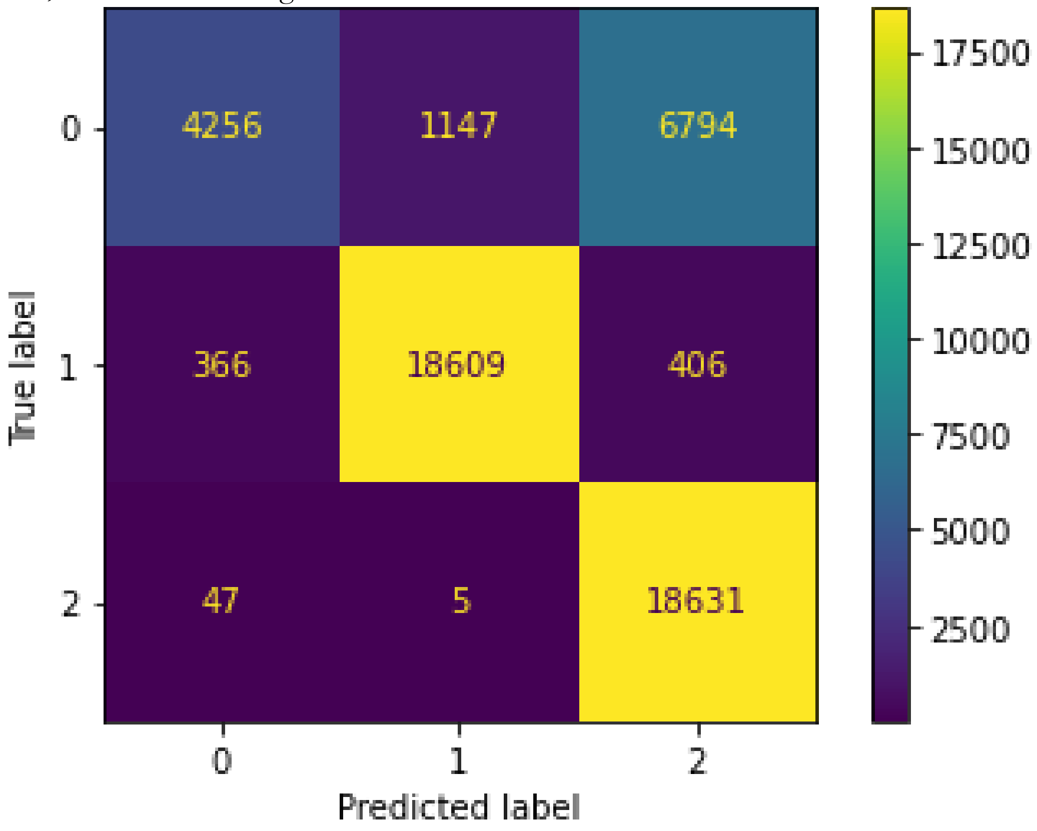


Figure 9. SVM Confusion Matrix

This matrix in Figure 10 represents results by the KNN model where it has far fewer false predictions among all the classes. It shows strong classification performance across all three classes, with significantly improved results for Class 1 compared to Figure 9. Misclassifications are minimal, with Class 2 and Class 3 achieving near-perfect accuracy, and only a few instances of Class 1 being misclassified. These improvements suggest effective adjustments to the model, such as better feature selection or hyperparameter tuning, with minimal further optimization required.

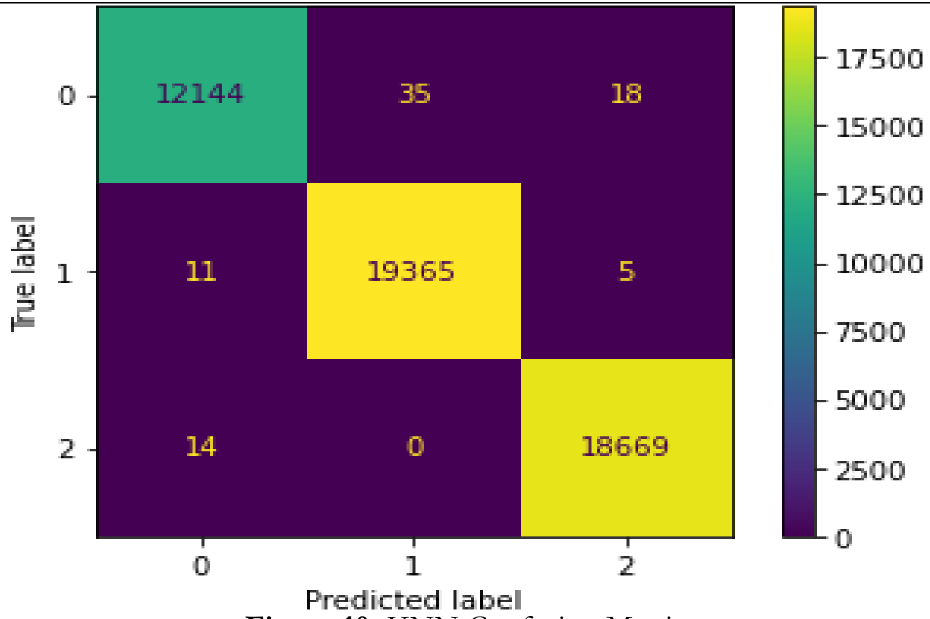


Figure 10. KNN Confusion Matrix

Figure 11 represents the confusion matrix for the random forest model with the best performance among all the other models with minimal false results. It demonstrates near-perfect classification across all three classes. Each class has almost all instances correctly predicted, with only 1 misclassification in Class 1 and Class 3, and no errors in Class 2. The results indicate exceptional model performance, likely due to Random Forest's robust handling of feature importance and complex decision boundaries. Further improvements are unnecessary, as the model achieves near-optimal accuracy.

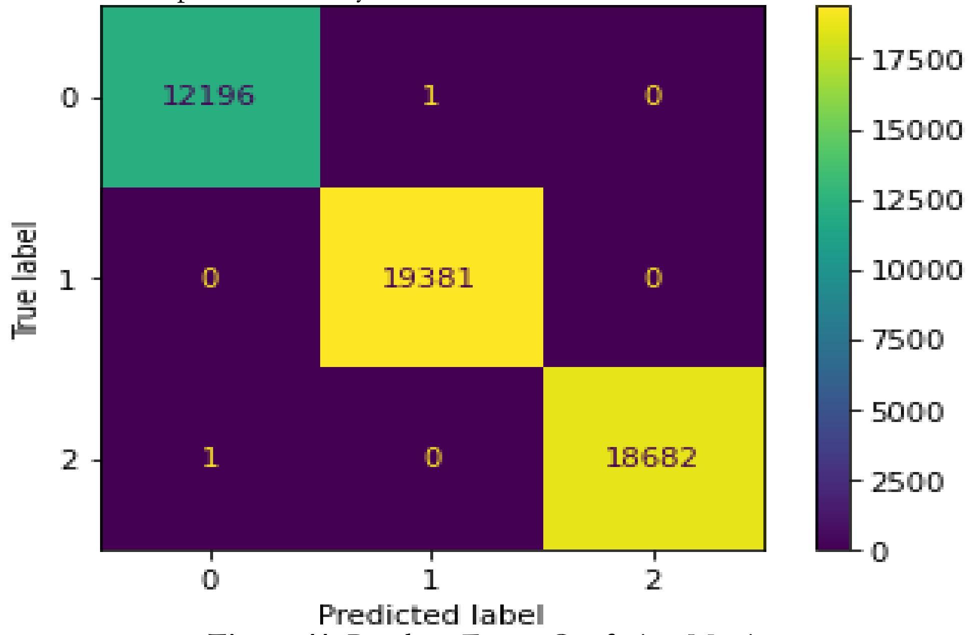


Figure 11. Random Forest Confusion Matrix

In Figure 12, the confusion matrix for the Gaussian Naïve-Bayes model with a slightly better performance as compared to the SVM model. This highlights moderate classification performance across three classes for IDS training. Class 2 and Class 3 are well-classified, with minimal misclassifications, but Class 1 shows significant overlap with Class 3, with many samples misclassified. The results reflect Naive Bayes' simplicity and reliance on feature independence, which might limit its effectiveness in handling complex decision boundaries. Further optimization or feature engineering could improve its performance.

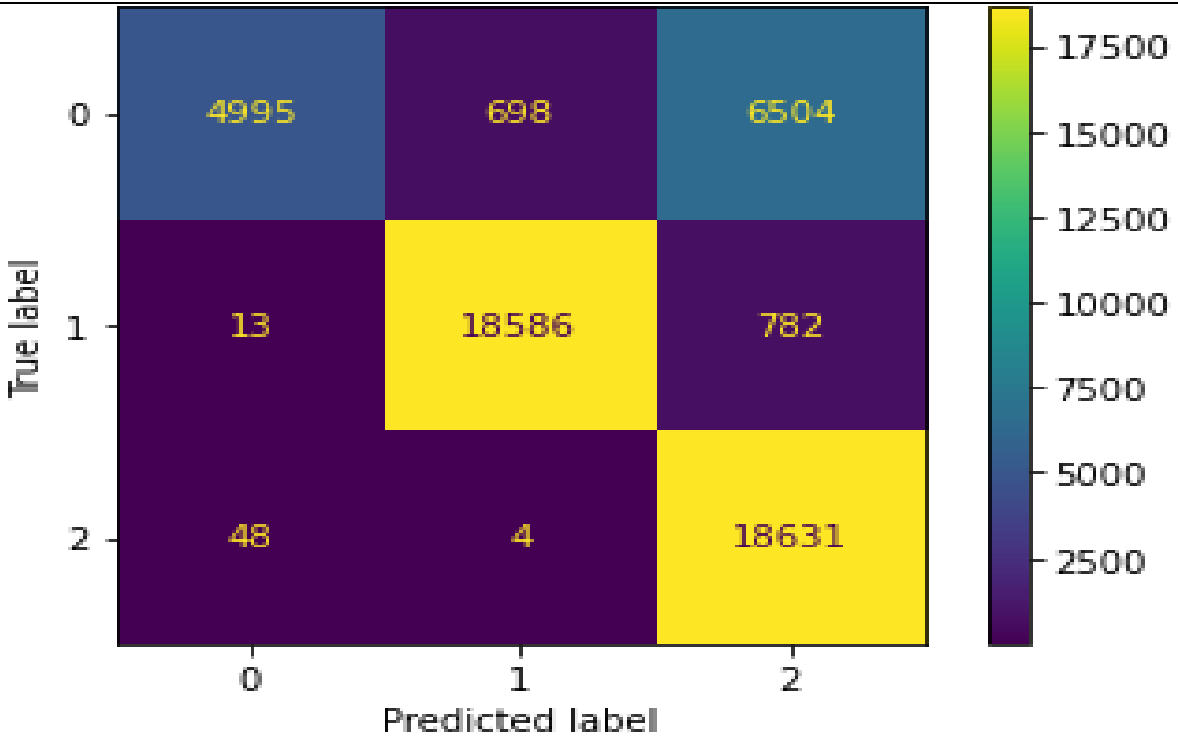


Figure 12. Gaussian Naïve-Bayes Confusion Matrix.

The matrix in Figure 13 represents the confusion matrix for the decision tree model with the second-best performance only falling slightly behind the random forest model. This confusion matrix for the Decision Tree model demonstrates excellent classification performance across all three classes in IDS training. Almost all instances are correctly classified, with only minimal misclassifications (2 in Class 1, 2 in Class 2, and 1 in Class 3). The results highlight the model's strong ability to handle complex decision boundaries and feature interactions, achieving near-perfect accuracy with little room for improvement, see Table 5 for comparisons.

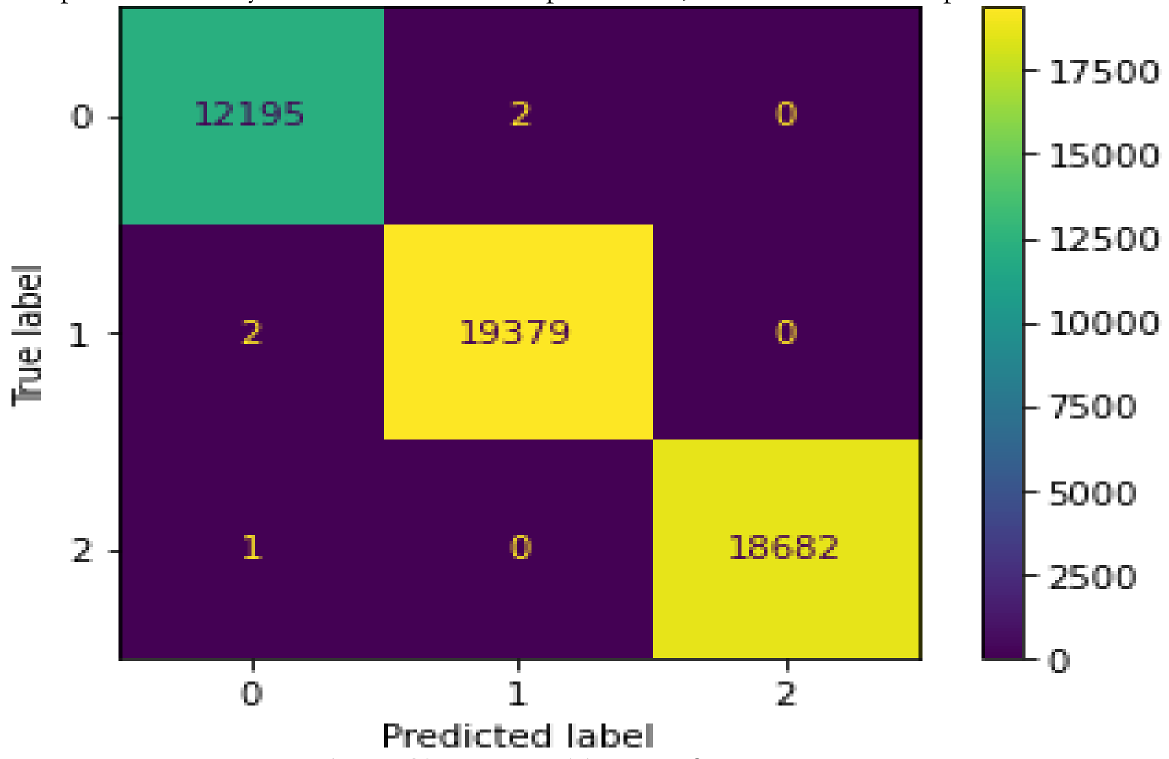


Figure 13. Decision Tree Confusion Matrix.

Table 5. Models Performance Comparison using Accuracy Precision, Reall, and F1

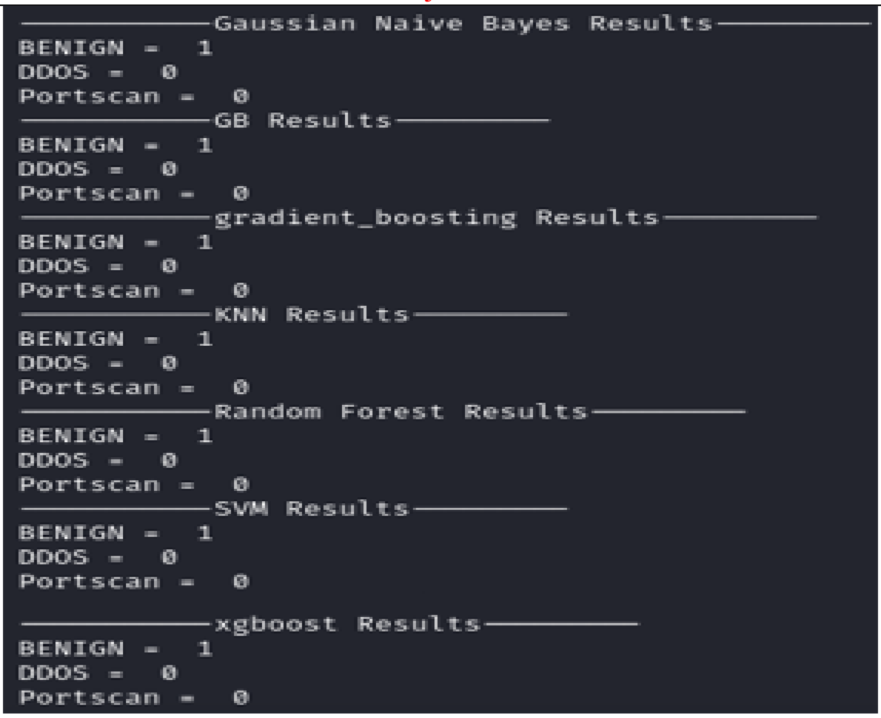
Model Name	Accuracy	Precision	Recall	F1-Score
KNN	**0.96**	**0.96**	**0.96**	**0.96**
Gaussian Naïve Bayes	0.84	0.88	0.84	0.82
Decision Tree	1.00	1.00	1.00	1.00
Random Forest	1.00	1.00	1.00	1.00
SVM	0.83	0.85	0.83	0.80

Discussion:

For Table 5, the metrics used to evaluate the performance of the models are precision, recall, F1-score, and accuracy. In general, the Decision Tree and Random Forest models have the highest scores in all metrics, with a perfect score of 1.00 for all four metrics. The reason for this could be because of class imbalance as explained before. These two models are more susceptible to this and tend to overfit because of this. Another reason could be Feature Redundancy that these models are essentially reading the features that are clear indicators of the class hence making classification trivial. The KNN model also has a perfect score for all four metrics. The Gaussian Naïve-Bayes model has a slightly lower score, with precision and recalls both at 0.88 and 0.84, respectively, and F1-score at 0.82 and accuracy at 0.84. The SVM model has the lowest scores among all the models, with precision and recall at 0.85 and 0.83, respectively, F1-score at 0.80, and accuracy at 0.83. Additionally, according to the confusion matrix false positive rate of the random forest has the lowest false positive rate (FPR) of "2" in comparison to decision trees with an FPR of 4. The rest of the models have an FPR that is unacceptable. So far, Random Forest is the most suitable model for the nature of data. At this stage, it is not fair to select a model yet for the IDS without considering other factors. Therefore, MLX-IDS takes into account the result of all of these models and decides based on a majority voting system. In this method, each model gives a predicted output class for each instance and then the system decides the final output of "Benign", "DoS" or "Port Scan" based on the number of votes for each class for that instance.

Inference:

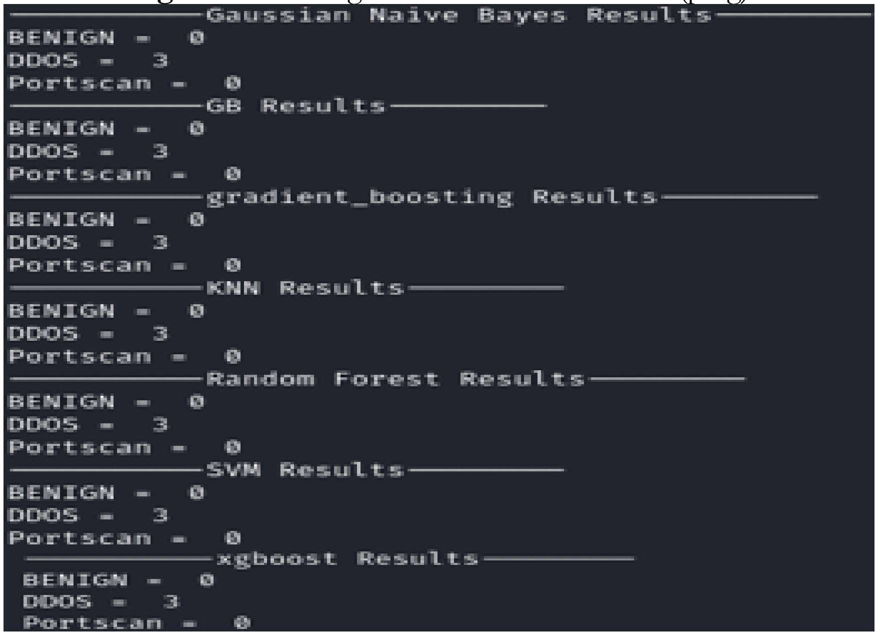
The models are tested on live traffic captured using tcpdump and used data preprocessing techniques to convert it into a format acceptable by our ML models. Figure 14 shows that the ping tool is used to send around ten packets to the system on which our model was running (within the same network). The packets were captured in a PCAP file by tcpdump and then the CIC Flowmeter tool converted them into a CSV format usable by the MLX-IDS. The flow of communication was only 1 since one program was running that generated the packets therefore the tool's output was only one instance. Upon analysis of an instance by the ML models, the result was that the packets were BENIGN. On the other hand, (Figure 15) upon running the hping3 tool to generate a UDP-based SYN-Flood DoS attack on the system and then following the preprocessing steps necessary for the model, it then showed them as three flows and all were considered as DoS attack flows. The reason all models' outputs are shown is that we are using the "hard voting" Ensemble method whereby a flow is classified as malicious when a majority of the models consider it as "malicious".



```
----------Gaussian Naive Bayes Results----------
BENIGN =  1
DDOS =  0
Portscan =  0
----------GB Results----------
BENIGN =  1
DDOS =  0
Portscan =  0
----------gradient_boosting Results----------
BENIGN =  1
DDOS =  0
Portscan =  0
----------KNN Results----------
BENIGN =  1
DDOS =  0
Portscan =  0
----------Random Forest Results----------
BENIGN =  1
DDOS =  0
Portscan =  0
----------SVM Results----------
BENIGN =  1
DDOS =  0
Portscan =  0

----------xgboost Results----------
BENIGN =  1
DDOS =  0
Portscan =  0
```

Figure 14. Testing of Models on live traffic (ping)



```
----------Gaussian Naive Bayes Results----------
BENIGN =  0
DDOS =  3
Portscan =  0
----------GB Results----------
BENIGN =  0
DDOS =  3
Portscan =  0
----------gradient_boosting Results----------
BENIGN =  0
DDOS =  3
Portscan =  0
----------KNN Results----------
BENIGN =  0
DDOS =  3
Portscan =  0
----------Random Forest Results----------
BENIGN =  0
DDOS =  3
Portscan =  0
----------SVM Results----------
BENIGN =  0
DDOS =  3
Portscan =  0
----------xgboost Results----------
 BENIGN =  0
 DDOS =  3
 Portscan =  0
```

Figure 15. Testing of Models on live traffic (hping3 DoS)

Complete System Test:

The analysis in Table 6 focuses on two datasets: CIC-IDS-2017 and UNSW-NB15. The CIC-IDS-2017 dataset contains a wide range of network flows and security events, with nearly 40 million benign flows. The UNSW-NB15 dataset represents various network activities and security incidents, with 2 million benign flows. Both datasets demonstrate a throughput of 1 Gbps, indicating the data transfer rate.

Regarding threat detection, UNSW-NB15 reports 0.33 million DoS attack flows and 0.21 million port scans,
whereas CIC-IDS-2017 finds 2 million DoS attack flows and 1 million port scans. According to the investigation of packet processing, UNSW-NB15 uses tcpdump to capture 2.12 million packets, whereas CIC-IDS-2017 processes 40.2 million packets utilizing the XDP framework.

When replaying the CIC-IDS-2017 dataset, the MLX-IDS System detected network events and security incidents with 99% accuracy. The MLX-IDS System obtained a 98% accuracy rate for the UNSW-NB15 dataset. This shows that the system works effectively and has a high degree of confidence in its ability to detect network events in both datasets.

The UNSW-NB15 dataset was specifically used for testing the MLX-IDS System after it was preprocessed to match the features of the proposed ML model.

Table 6. Complete System Test Results

Features	CIC-IDS-2017	UNSW-NB15
Through-put (Gbps)	1	1
Benign Flow Count (Millions)	37.35	2
DoS Flow Detections (Millions)	2	0.33
Port Scans Detected (Millions)	1	0.21
XDP-Packet Count (Millions)	40.2	2.54
Tcpdump Packet Count (Millions)	35	2.12
Process Time	13 sec	3 sec
Accuracy (%)	**99**	**98**

Conclusion and Future Work:

The deployment, investigation of several ML models and datasets, and assessment of an IDS that makes use of a machine learning model trained on the CIC-IDS2017 dataset were all covered in this paper's conclusion. Our proposed model shows that this approach is capable of achieving a low false positive rate and high detection accuracy. Moreover, testing conducted on real-world traffic samples has demonstrated the potential of the suggested system as a viable option for enterprises seeking to improve network security. Since the suggested design simply makes use of machine learning methods, future studies can examine the outcomes of deep learning models as well as test the suggested model's limitations on more advanced hardware.

Another area of focus for future work is the detection and response to multi-vector attacks. These attacks involve the simultaneous use of multiple attack techniques, making them more sophisticated and challenging to detect. Future research can investigate the development of intelligent algorithms that can identify and analyze complex attack patterns across different network layers, enabling the MLX-IDS to effectively detect and respond to such attacks. By incorporating adaptive learning, the system can continuously evolve and adapt to new attack vectors and techniques, improving its detection capabilities over time. Integration with threat intelligence feeds is also an area of potential future research, as it can enhance the system's capabilities by leveraging real-time information about emerging threats.

References:

[1] "Next Generation of Data-Mining Applications | IEEE eBooks | IEEE Xplore." Accessed: Jan. 03, 2025. [Online]. Available: https://ieeexplore.ieee.org/book/5769527

[2] R. A. Kemmerer and G. Vigna, "Intrusion detection: A brief history and overview," Computer (Long. Beach. Calif)., vol. 35, no. SUPPL., pp. 27–30, 2002, doi: 10.1109/MC.2002.1012428.

[3] L. S. Cardoso, "Intrusion Detection Versus Intrusion Protection," Netw. Secur. Curr. Status Futur. Dir., pp. 99–115, Jun. 2006, doi: 10.1002/9780470099742.CH7.

[4] S. J. Ovaska, "Computationally Intelligent Hybrid Systems: The Fusion of Soft Computing and Hard Computing," Comput. Intell. Hybrid Syst. Fusion Soft Comput. Hard Comput., pp. 1–408, Jun. 2012, doi: 10.1002/9780471683407.

[5] F. Anjum and P. Mouchtaris, "Security for Wireless Ad Hoc Networks," Secur. Wirel. Ad Hoc Networks, Feb. 2007, doi: 10.1002/0470118474.

[6] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," IEEE Symp. Comput. Intell. Secur. Def. Appl. CISDA 2009, Dec. 2009, doi: 10.1109/CISDA.2009.5356528.

[7] N. Hubballi and V. Suryanarayanan, “False alarm minimization techniques in signature-based intrusion detection systems: A survey,” Comput. Commun., vol. 49, pp. 1–17, Aug. 2014, doi: 10.1016/J.COMCOM.2014.04.012.
[8] Roger M. Needham, “Denial of service,” CCS ’93 Proc. 1st ACM Conf. Comput. Commun. Secur., pp. 151–153, 1993, doi: https://doi.org/10.1145/168588.168607.
[9] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Surveying Port Scans and Their Detection Methodologies,” Comput. J., vol. 54, no. 10, pp. 1565–1581, Oct. 2011, doi: 10.1093/COMJNL/BXR035.
[10] I. M. Mehrnaz Mazini, Babak Shirazi, “Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms,” J. King Saud Univ. - Comput. Inf. Sci., vol. 31, no. 4, pp. 541–553, 2019, doi: https://doi.org/10.1016/j.jksuci.2018.03.011.
[11] M. A. M. and W. M. A. B. A. Khalaf, S. A. Mostafa, A. Mustapha, “Comprehensive Review of Artificial Intelligence and Statistical Approaches in Distributed Denial of Service Attack and Defense Methods,” IEEE Access, vol. 7, pp. 51691–51713, 2019, doi: 10.1109/ACCESS.2019.2908998.
[12] D. M. Toke Høiland-Jørgensen, Jesper Dangaard Brouer, Daniel Borkmann, John Fastabend, Tom Herbert, David Ahern, “The eXpress data path: fast programmable packet processing in the operating system kernel,” Conex. ’18 Proc. 14th Int. Conf. Emerg. Netw. Exp. Technol., pp. 54–66, 2018, doi: https://doi.org/10.1145/3281411.3281443.
[13] H. . Wieren, “Signature-Based DDoS Attack Mitigation: Automated Generating Rules for Extended Berkeley Packet Filter and Express Data Path,” Univ. Twente Student Theses, 2019, [Online]. Available: https://essay.utwente.nl/80125/
[14] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” ICISSP 2018 - Proc. 4th Int. Conf. Inf. Syst. Secur. Priv., vol. 2018-January, pp. 108–116, 2018, doi: 10.5220/0006639801080116.
[15] S. B. Ranjit Panigrahi, “A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems,” Int. J. Eng. Technol., vol. 7, no. 3, pp. 479–482, 2018, [Online]. Available: https://www.researchgate.net/publication/329045441_A_detailed_analysis_of_CICIDS2017_dataset_for_designing_Intrusion_Detection_Systems
[16] T. Janarthanan and S. Zargari, “Feature selection in UNSW-NB15 and KDDCUP’99 datasets,” IEEE Int. Symp. Ind. Electron., pp. 1881–1886, Aug. 2017, doi: 10.1109/ISIE.2017.8001537.
[17] C. R. Wang, R. F. Xu, S. J. Lee, and C. H. Lee, “Network intrusion detection using equality constrained-optimization-based extreme learning machines,” Knowledge-Based Syst., vol. 147, pp. 68–80, May 2018, doi: 10.1016/J.KNOSYS.2018.02.015.
[18] N. Moustafa, J. Slay, and G. Creech, “Novel Geometric Area Analysis Technique for Anomaly Detection Using Trapezoidal Area Estimation on Large-Scale Networks,” IEEE Trans. Big Data, vol. 5, no. 4, pp. 481–494, Dec. 2019, doi: 10.1109/TBDATA.2017.2715166.
[19] C. Khammassi and S. Krichen, “A GA-LR wrapper approach for feature selection in network intrusion detection,” Comput. Secur., vol. 70, pp. 255–277, Sep. 2017, doi: 10.1016/J.COSE.2017.06.005.
[20] B. J. J. Doe, A. Smith, “Machine Learning-Based Intrusion Detection Using Decision Tree Classifier,” Proc. Int. Conf. Netw. Secur., pp. 123–134, 2015.
[21] M. Leon, T. Markovic, and S. Punnekkat, “Comparative Evaluation of Machine Learning Algorithms for Network Intrusion Detection and Attack Classification,” Proc. Int. Jt. Conf. Neural Networks, vol. 2022-July, 2022, doi: 10.1109/IJCNN55064.2022.9892293.
[22] S. S. P. Patel, R. Gupta, “Support Vector Machines-Based Intrusion Detection System for Network Security,” Pro-ceedings ACM Symp. Appl. Comput., pp. 345–356, 2018.

[23] S. Choudhury and A. Bhowal, "Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection," 2015 Int. Conf. Smart Technol. Manag. Comput. Commun. Control. Energy Mater. ICSTM 2015 - Proc., pp. 89–95, Aug. 2015, doi: 10.1109/ICSTM.2015.7225395.
[24] N. A. A. Rahman, S. Das, "Multilayer Perceptron-Based Intrusion Detection System Using Feature Selection," Proceed-ings Int. Conf. Intell. Comput. Data Sci., pp. 234–245, 2020.
[25] M. S. S. Gupta, R. Kumar, "Logistic Regression for Network Intrusion Detection: A Comparative Study," Proceed-ings IEEE Int. Conf. Comput. Anal. Secur. Trends (CAST), pp. 456–467, 2021.
[26] and Y. Z. Weiwei Zhang, Yiming Zhang, Xiaoyue Wang, Hongwei Zhang, "EDPS-IDS: An Express Data Path Based Intrusion Detection System for Software-Defined Networks," 2018 IEEE Conf. Netw. Softwarization, pp. 253–266, 2018.
[27] and J. Z. Xin Chen, Jianhua Zhang, Xiaobo Yang, Jianfeng Lu, "A Novel SDN-Based Intrusion Detection System," 2017 IEEE Conf. Comput. Commun., pp. 1705–1713, 2017.
[28] and W. Z. Yan Zhang, Xiaolin Zhang, Xianbin Wang, Zhi Li, "Anomaly Detection in Software-Defined Networks Using Express Data Path," 2017 IEEE Int. Conf. Commun., pp. 1–6, 2017.
[29] and Z. L. Xiang Li, Wei Wang, Hua Wang, Xiaodong Liu, "Intrusion Detection System for Software-Defined Networks Based on Express Data Path," 2017 IEEE Conf. Comput. Commun. Secur., pp. 1083–1098, 2017.
[30] and Y. Z. Lihua Zhang, Weiwei Zhang, Xiaoyue Wang, Hongwei Zhang, "Express Data Path Based Intrusion Detection System for Software Defined Networks," 2017 IEEE Int. Conf. Commun., pp. 1–6, 2017.
[31] P. V. V. R. Jyothsna V, "FCAAIS: Anomaly based network intrusion detection through feature correlation analysis and association impact scale," ICT Express, vol. 2, no. 3, pp. 103–116, 2016, doi: https://doi.org/10.1016/j.icte.2016.08.003.
[32] A. Satoh, Y. Nakamura, and T. Ikenaga, "A flow-based detection method for stealthy dictionary attacks against Secure Shell," J. Inf. Secur. Appl., vol. 21, pp. 31–41, Apr. 2015, doi: 10.1016/J.JISA.2014.08.003.
[33] A. Juvonen and T. Hamalainen, "An efficient network log anomaly detection system using random projection dimensionality reduction," 2014 6th Int. Conf. New Technol. Mobil. Secur. - Proc. NTMS 2014 Conf. Work., 2014, doi: 10.1109/NTMS.2014.6814006.
[34] D. Stiawan, M. Y. Bin Idris, A. M. Bamhdi, R. Budiarto, and others, "CICIDS-2017 dataset feature analysis with information gain for anomaly detection," IEEE Access, vol. 8, pp. 132911–132921, 2020.
[35] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in 2015 Military Communications and Information Systems Conference (MilCIS), 2015, pp. 1–6. doi: 10.1109/MilCIS.2015.7348942.
[36] N. Moustafa and J. Slay, "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," Inf. Secur. J. A Glob. Perspect., vol. 25, no. 1–3, pp. 18–31, Apr. 2016, doi: 10.1080/19393555.2015.1125974.
[37] N. Moustafa, G. Creech, and J. Slay, "Big Data Analytics for Intrusion Detection System: Statistical Decision-Making Using Finite Dirichlet Mixture Models," pp. 127–156, 2017, doi: 10.1007/978-3-319-59439-2_5.
[38] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems," Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng. LNICST, vol. 371 LNICST, pp. 117–135, 2021, doi: 10.1007/978-3-030-72802-1_9.
[39] C. Mera and J. W. Branch, "A survey on class imbalance learning on automatic visual inspection," IEEE Lat. Am. Trans., vol. 12, no. 4, pp. 657–667, 2014, doi:

10.1109/TLA.2014.6868867.
[40] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," IEEE Trans. Syst. Man Cybern. Part C Appl. Rev., vol. 42, no. 4, pp. 463–484, Jul. 2012, doi: 10.1109/TSMCC.2011.2161285.
[41] Q. Song, Y. Guo, and M. Shepperd, "A Comprehensive Investigation of the Role of Imbalanced Learning for Software Defect Prediction," IEEE Trans. Softw. Eng., vol. 45, no. 12, pp. 1253–1269, Dec. 2019, doi: 10.1109/TSE.2018.2836442.
[42] S. Wang, L. L. Minku, D. Ghezzi, D. Caltabiano, P. Tino, and X. Yao, "Concept drift detection for online class imbalance learning," Proc. Int. Jt. Conf. Neural Networks, 2013, doi: 10.1109/IJCNN.2013.6706768.
[43] X. Y. S. Wang, "Multiclass Imbalance Problems: Analysis and Potential Solutions," IEEE Trans. Syst. Man, Cybern. Part B, vol. 42, no. 4, pp. 1119–1130, 2012, doi: 10.1109/TSMCB.2012.2187280.
[44] S. D. Rushi Longadge, "Class Imbalance Problem in Data Mining Review," arXiv:1305.1707, 2013, doi: https://doi.org/10.48550/arXiv.1305.1707.
[45] Shaza M. Abd Elrahman, Ajith Abraham, "A Review of Class Imbalance Problem," J. Netw. Innov. Comput., vol. 1, pp. 332–340, 2013, [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=bb2e442b2acb4530aa28d24e45578f84447d0425#:~:text=Imbalance data sets degrades the,furthermore treated them as noise.
[46] P. K. M. Nallapaneni Manoj Kumar, "Blockchain technology for security issues and challenges in IoT," Procedia Comput. Sci., vol. 132, pp. 1815–1823, 2018, doi: https://doi.org/10.1016/j.procs.2018.05.140.
[47] "IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB." Accessed: Jan. 03, 2025. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html
[48] N. Bhargava, G. Sharma, R. Bhargava, and M. Mathuria, "Decision Tree Analysis on J48 Algorithm for Data Mining," 2013.
[49] M. A. J. Farnaaz Nabila, "Random Forest Modeling for Network Intrusion Detection System," Procedia Comput. Sci., vol. 89, pp. 213–217, 2016, doi: https://doi.org/10.1016/j.procs.2016.06.047.
[50] S. Hossen and A. Janagam, "Analysis of network intrusion detection system with machine learning algorithms (deep reinforcement learning Algorithm)," no. October, pp. 1–63, 2018.
[51] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," IEEE Trans. Inf. Theory, vol. 13, no. 1, pp. 21–27, 1967, doi: 10.1109/TIT.1967.1053964.
[52] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval," Mach. Learn. ECML-98, pp. 4–15, 1998, doi: https://doi.org/10.1007/BFb0026666.
[53] Z. T. Chen T, He T, Benesty M, Khotilovich V, Tang Y, Cho H, Chen K, Mitchell R, Cano I, "XGBoost (eXtreme Gradient Boosting)," R Packag. version, pp. 1–4, 2015, [Online]. Available: https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/