*MUST Journal of Research and Development (MJRD) Volume 6 Issue 2, June 2025*
*e ISSN 2683-6467 & p ISSN 2683-6475*

# Improving Intrusion Detection Accuracy in Campus Networks: A Dataset Driven by Real-Time Traffic and Honeypot Simulations

[1]Daud M. Sindika*, [2]Mrindoko R. Nicholaus and [3]Nabahani B. Hamadi
[1]Directorate of Information and communication Technology
[2]Department of Computer Science and Engineering
[3]Department of Information System Technology
[1,2,3]Mbeya University of Science and Technology, P.O Box 131 Mbeya, Tanzania

DOI: https://doi.org/10.62277/mjrd2025v6i20011

## ARTICLE INFORMATION

### Article History

*Received:* 20th January 2025
*Revised:* 10th May 2025
*Accepted:* 12th June 2025
*Published:* 30th June 2025

### Keywords

Intrusion Detection System
Dataset
Network Traffic
Honeypot
Campus Network

## ABSTRACT

This article describes the creation of a domain-specific Intrusion Detection System (IDS) dataset customised for campus networks to overcome the constraints of out-of-date public datasets such as KDD'99 and NSL-KDD. The dataset depicts the various user behaviours, traffic patterns, and device interactions that are unique to educational contexts because it captures network traffic straight from a university. Real-time logs from firewalls, routers, and switches are used as data sources, as is the simulated attack traffic collected by honeypots, which are false open network ports meant to entice malicious behaviour. This technique ensures a balanced mix of normal and attacking actions. Machine learning models trained on this dataset have a 99% detection rate, exceeding models trained on public datasets (95%), while also lowering false positives. The dataset is continually updated to reflect changes in user behaviour, software, and threats, maintaining its long-term usefulness. This work establishes a realistic, adaptive, and effective framework for developing scalable IDS models designed for campus network protection.

*Corresponding author's e-mail address: dsindika@yahoo.com (Sindika, D.M)

## 1.0 Introduction

Network-based attacks are becoming more common, necessitating the development of accurate detection and mitigation techniques. Intrusion Detection Systems (IDS) are crucial in detecting and responding to such threats, transcending standard defences such as firewalls, passwords, and antivirus software by monitoring network activity in real time (Aljanabi *et al.,* 2021; Al-Qatf *et al.,* 2018). IDSs safeguard the confidentiality, integrity, and availability of digital assets and are especially crucial in dynamic situations such as educational and cloud-based networks.

As attacks become more advanced, machine learning (ML) has become a key approach in developing IDS because it can identify complicated, evolving threats, lower false alarms, and adapt its responses. However, the success of machine learning-based intrusion detection systems is strongly dependent on the availability of high-quality, relevant datasets for model training.

Publicly available datasets, like KDD'99 and NSL-KDD, are frequently utilised, but they are typically out of date and lack the diversity and specificity required for modern detection tasks (Khraisat *et al.,* 2019; Ghurab *et al.,* 2021; Komisarek *et al*., 2021). These statistics usually lack real-time traffic characteristics, have inadequate tagging, and do not reflect current attack vectors. As a result, ML models trained on them may perform poorly in real-world settings. Tailored datasets are critical for improving detection accuracy in specific network scenarios (Devi & Kannan, 2021).

This study tackles this problem by creating a domain-specific IDS dataset using traffic collected at a Tanzanian university campus. The dataset includes real-time logs from firewalls, routers, and switches, as well as honeypot-generated traffic that replicates attack behaviour. This technique captures different user behaviours, device interactions, and localised dangers that differ dramatically from commercial or enterprise contexts.

Despite the availability of general-purpose datasets, there is currently no domain-specific dataset that addresses traffic characteristics of campus networks in underdeveloped countries such as Tanzania. This study bridges that gap, providing a foundation for developing more accurate, adaptive IDS models customised to educational institutions' specific security requirements.

## 2.0 Materials and Methods

### *2.1 Study Area Description*

This research was done at Mbeya University of Science and Technology (MUST) in Tanzania's Mbeya Region. MUST was chosen because of its unique network features, which differ from those found in corporate or commercial situations.

These include various user behaviours, device kinds, and communication protocols, all of which result in unique traffic patterns. Key features like a large and changing group of users, VLAN-based segmentation, guest access for visitors, and access to important information like student records and research data all lead to realistic and varied network activity.

These characteristics make MUST a suitable platform for creating a domain-specific IDS dataset.

### *2.2 Study Design and Data Collection*

The network traffic collected over six days within the MUST computer network was analysed using an experimental research design in this study.

Data gathering is the initial stage in creating an IDS dataset. Using sensors and monitoring technologies, higher learning institutions (HLIs) collect network logs, packet captures, and other pertinent data. Examples of logs include firewall logs, honeypot logs, network flow statistics, DNS logs, and authentication logs. These sources provide a comprehensive picture of both legitimate and harmful network activities.

Data is typically acquired using packet- or flow-based approaches. Packet-level collection means setting up network devices with mirrored ports to capture all the data being sent, while flow-based approaches gather information about the connections instead (Ring et al., 2019). This investigation's flow-based dataset was collected

over six days on a campus network and consisted solely of DNS connections.

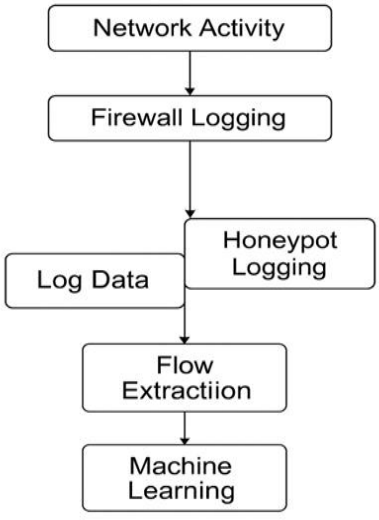
There were 610,600 unidirectional flows observed. Out of these, 94,248 were found to be harmful by comparing them with firewall logs, IDS alerts, and honeypot activity, which network security experts carefully reviewed to ensure they were correct using a method that combines information from different sources, often used in research on detecting intrusions. The remaining flows were deemed normal user activity.

Because of the sensitivity of the data and the availability of personal and institutional information, the dataset known as MD23 is not publicly available. However, ethical concerns were resolved by ensuring data anonymity (for example, masking IP addresses) and collecting only metadata when possible. Furthermore, the study adhered to institutional ethical requirements and received permission from the university's research ethics council prior to data collection.

Figure 1 depicts the stages of the process, beginning with network activity and progressing to firewall and honeypot logging, flow extraction, and feature development. This visual representation shows how raw data was turned into a structured dataset for IDS model training.

Figure 1

*Illustration of the Data Flow*




Our generated dataset was named MD23. Four categories are used to categorise attacks in this dataset:

a) DoS: A denial of service (DoS) attack stops authorised users from using network and system resources. Online banking and email might be affected (Yu & Bian, 2020). DoS assaults include the SYN flood assault, Smurf attack, teardrop, land, Neptune, and mailbomb (Mishra et al., 2019).
b) Remote to Local (R2L): R2L attacks entail an attacker trying to access the target workstation without authorisation (Yu & Bian, 2020). Examples of R2L attacks are named sendmail, worm, xnoop, ftp_write, imap, multihop, phf, spy, warezclient, xclock, snmpgetattack, snmpguess, and snmpguess (Mishra et al., 2019).
c) User to Root (U2R): This assault aims to give the perpetrator local access privileges on the victim's machine (Yu & Bian, 2020). Examples of U2R attacks are buffer_overflow, loadmodule, perl, rootkit, httptunnel, ps, xterm, and sqlback (Mishra et al., 2019).
d) Probe: In Probe, the attacker focuses on the host and seeks to learn more about it (Yu & Bian, 2020). Ipsweep, Resetscan, ACK scan, UDP scan, and FIN scan are examples of probe attacks (Mishra et al., 2019).

During data collection, the institution firewall called Sophos XG 330, which is running on the computer network of the university, was used to capture the network traffic and system logs through the observation method, as shown in Tab. 1.

The Honeypot system software was employed in the study to record the attack patterns and behaviours of the attacker in finding ways to penetrate the systems.

Table 1
*Collected Data before Preprocessing*

| Log subtype | Firewall rule | NAT rule | Firewall rule name | NAT rule name | Rule type | Log occurrence | Live PCAP | Src IP | Src port | Dst IP | Dst port | In interface | Out interface | protocol | Bytes sent | Bytes received | Connection duration | Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Denied | N/A | 0 | | | 0 | 1 | Open PCAP | 172.16.0.216 | 62211 | 139.177.237.224 | 443 | | | TCP | 0 | 0 | 0 | Malicious |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.100 | 58742 | 157.240.195.63 | 443 | Port1 | Port5 | TCP | 1002 | 4730 | 92 | Normal |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.16.130 | 41368 | 173.222.106.185 | 443 | Port1 | Port5 | TCP | 80813 | 26685 | 615 | Normal |
| Allowed | 4 | 8 | Allow DNS | fw#4_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.0.26 | 23452 | 41.59.226.59 | 53 | Port1 | Port5 | UDP | 66 | 367 | 30 | Normal |
| Denied | N/A | 0 | | | 0 | 1 | Open PCAP | 172.16.0.216 | 62211 | 139.177.237.224 | 443 | | | TCP | 0 | 0 | 0 | Malicious |
| Denied | N/A | 0 | | | 0 | 1 | Open PCAP | 172.16.16.140 | 56482 | 102.132.120.164 | 443 | Port1 | | TCP | 0 | 0 | 0 | Malicious |
| Denied | N/A | 0 | | | 0 | 1 | Open PCAP | 172.16.16.140 | 56482 | 102.132.120.164 | 443 | Port1 | | TCP | 0 | 0 | 0 | Malicious |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.114 | 55545 | 172.217.170.170 | 443 | Port1 | Port5 | TCP | 989 | 5932 | 20 | Normal |
| Allowed | 4 | 8 | Allow DNS | fw#4_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.0.202 | 54507 | 41.59.226.59 | 53 | Port1 | Port5 | UDP | 84 | 274 | 31 | Normal |
| Allowed | 5 | 12 | DMZ_TO_WAN | fw#5_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.7.14 | 41186 | 196.13.105.13 | 443 | Port3 | Port5 | TCP | 1877 | 6659 | 10 | Normal |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.126 | 61154 | 157.240.196.35 | 443 | Port1 | Port5 | TCP | 2634 | 4444 | 236 | Normal |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.0.26 | 39285 | 35.190.0.66 | 443 | Port1 | Port5 | UDP | 4283 | 6789 | 62 | Normal |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.0.86 | 45132 | 157.240.196.15 | 443 | Port1 | Port5 | TCP | 1017 | 6951 | 48 | Normal |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.100 | 58720 | 157.240.195.63 | 443 | Port1 | Port5 | TCP | 1136 | 1697 | 94 | Normal |
| Allowed | 4 | 8 | Allow DNS | fw#4_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.110 | 25575 | 41.59.226.59 | 53 | Port1 | Port5 | UDP | 73 | 376 | 30 | Normal |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.193 | 48176 | 216.58.223.106 | 443 | Port1 | Port5 | TCP | 2159 | 6868 | 43 | Normal |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.16.124 | 55337 | 104.26.12.49 | 443 | Port1 | Port5 | TCP | 2078 | 1265 | 10 | Normal |
| Allowed | 4 | 8 | Allow DNS | fw#4_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.110 | 2577 | 41.59.226.59 | 53 | Port1 | Port5 | UDP | 61 | 349 | 30 | Normal |
| Allowed | 4 | 8 | Allow DNS | fw#4_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.110 | 1230 | 41.59.226.59 | 53 | Port1 | Port5 | UDP | 61 | 349 | 30 | Normal |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.0.86 | 45130 | 157.240.196.15 | 443 | Port1 | Port5 | TCP | 4898 | 16830 | 48 | Normal |
| Allowed | 4 | 8 | Allow DNS | fw#4_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.114 | 58703 | 41.59.226.59 | 53 | Port1 | Port5 | UDP | 77 | 380 | 30 | Normal |
| Denied | N/A | 0 | | | 0 | 1 | Open PCAP | 172.16.17.146 | 41886 | 104.21.233.217 | 443 | | | TCP | 0 | 0 | 0 | Malicious |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.135 | 3639 | 171.96.220.168 | 13954 | Port1 | Port5 | TCP | 52 | 40 | 0 | Normal |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.155 | 59153 | 102.132.120.162 | 443 | Port1 | Port5 | TCP | 1870 | 3450 | 13 | Normal |
| Allowed | 4 | 8 | Allow DNS | fw#4_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.7.14 | 51763 | 41.59.226.59 | 53 | Port3 | Port5 | UDP | 80 | 140 | 30 | Normal |
| Allowed | 4 | 8 | Allow DNS | fw#4_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.108 | 57123 | 8.8.8.8 | 53 | Port1 | Port5 | UDP | 69 | 109 | 31 | Normal |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.115 | 60192 | 35.240.191.84 | 2048 | Port1 | Port5 | TCP | 60 | 300 | 77 | Normal |
| Allowed | 4 | 8 | Allow DNS | fw#4_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.110 | 9635 | 41.59.226.59 | 53 | Port1 | Port5 | UDP | 61 | 349 | 30 | Normal |
| Denied | N/A | 0 | | | 0 | 1 | Open PCAP | 102.132.120.164 | 443 | 41.59.86.254 | 34132 | | | TCP | 0 | 0 | 0 | Malicious |
| Denied | N/A | 0 | | | 0 | 1 | Open PCAP | 172.16.16.140 | 56482 | 102.132.120.164 | 443 | Port1 | | TCP | 0 | 0 | 0 | Malicious |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.111 | 34132 | 102.132.120.164 | 443 | Port1 | Port5 | TCP | 1479 | 22900 | 27 | Normal |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.120 | 51560 | 20.199.120.85 | 443 | Port1 | Port5 | TCP | 52 | 52 | 60 | Normal |
| Allowed | 1 | 11 | LAN_Allow-internet | fw#1_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.100 | 58746 | 157.240.195.63 | 443 | Port1 | Port5 | TCP | 1234 | 5078 | 92 | Normal |
| Allowed | 4 | 8 | Allow DNS | fw#4_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.17.108 | 63226 | 41.59.226.59 | 53 | Port1 | Port5 | UDP | 65 | 465 | 31 | Normal |
| Allowed | 5 | 12 | DMZ_TO_WAN | fw#5_migrated_NAT_Rule | 1 | 1 | Open PCAP | 172.16.7.14 | 41174 | 196.13.105.13 | 443 | Port3 | Port5 | TCP | 2437 | 6011 | 11 | Normal |
| Denied | N/A | 0 | | | 0 | 1 | Open PCAP | 172.16.17.187 | 49608 | 157.240.196.61 | 443 | | | TCP | 0 | 0 | 0 | Malicious |

The statistics of the data gathered for a week to build the dataset are displayed in Table 2.

Table 2
*Collected Data*

| Date | Number of Flows | Number of Attacks | Description |
|---|---|---|---|
| 15/05/2023 | 155,495 | 26,790 | Normal, DoS, and Probe |
| 16/05/2023 | 145,201 | 22,452 | Normal, Dos, and probe |
| 17/05/2023 | 29,101 | 22,747 | Normal, U2R, and R2L |
| 18/05/2023 | 63,901 | 20,998 | Normal, U2R, and R2L |
| 19/05/2023 | 98,801 | 51,454 | Normal, Dos, Probe, U2R, and R2L |
| 03/06/2023 | 118,101 | 41,801 | Normal, Dos, Probe, U2R, and R2L |
| Total | 610,600 | 186242 | Normal, Dos, Probe, U2R, and R2L |

### *2.3 Data Analysis*

Python (Python Software Foundation, Wilmington, DE, USA) was used to clean the data, create visualisations, apply feature selection methods, and build the machine learning models in the Anaconda environment (Anaconda Software Distribution, Austin, TX, USA). Many Python packages were used. Pandas was used to clean and organise the data; Matplotlib was used to create the visualisations; Seaborn was used to create a heat map showing the correlation between features; Sci-Kit Learn was used for all machine learning and feature selection operations, and Numpy was used for general mathematical operations.

#### *2.3.1 Data Preprocessing*

After being collected, the data was preprocessed to verify its quality and uniformity. Data preprocessing is a crucial step in preparing the data for IDS training. It involves cleaning and transforming the data into an appropriate format.

Given that the information was obtained in a controlled environment, special care was taken to capture essential identity indications from network traffic. Payload size, protocol flags (such as SYN, ACK, and FIN), source and destination IP addresses and ports, and packet size distribution were all recovered during preprocessing. These characteristics are crucial in detecting communication habits and anomalies. Abnormal payload sizes or TCP flag combinations, for example, are frequently indicative of scanning or attack attempts. All characteristics were cleaned, anonymised to preserve sensitive data, and standardised to maintain consistency in future machine learning activities.

Typical preprocessing procedures of this dataset included:

i. Dealing with missing values, choosing a method for handling missing data, such as imputation or removing rows or columns with empty values.
ii. Categorical variables were required to be converted into numerical representations using methods like one-hot encoding or label encoding for data that contains any of these variables.
iii. Scaling/normalisation: Another procedure was to scale or normalise the numerical features to make sure they are on a similar scale, depending on the specific methods or models you intend to apply.
iv. Define the target variable (e.g., malicious or benign label): Establish the IDS's target variable. This might be a categorisation that is either true or false.
v. Data Split: Split the data into training and testing sets using a 70/30 split to evaluate the performance of your IDS.

Table 3 presents the dataset's statistics following data preprocessing to develop a model.

Table 3
*After Data Pre-Processing*

| Number of Flows | Number of Attacks | Description | |
|---|---|---|---|
| 559,735 | 92,000 | Dos | 23,250 |
| | | Probe | 27,250 |
| | | U2R | 21,250 |
| | | R2L | 20,250 |
| | | Normal | 407,168 |
| | | Abnormal/Unclassified | 60,567 |

### *2.3.2 Feature Selection*

Feature selection is critical in improving the performance of machine learning models for intrusion detection by removing irrelevant or duplicate features. In this study, a decision tree-based classifier and Recursive Feature Elimination (RFE) were used on the IDS dataset to determine the most important features that improve detection accuracy while reducing computing cost.

To improve feature relevance, the select the top K highest-scoring features from the dataset (SelectKBest) technique from the Scikit-Learn module were selected. SelectKBest assesses features using univariate statistical tests and chooses the top-k features with the highest scores. Initially, the dataset had 35 features, but after feature selection, 15 top-ranked features were maintained for model training.

These chosen features include source/destination IP addresses, ports, protocols, packet sizes, and timestamps. This reduction not only enhanced detection performance but also reduced model overfitting and shortened training time.

Table 4
*Feature Importance Scores for Selected Features*

| No. | Feature Name | Description | Type | Importance Score |
|---|---|---|---|---|
| 1 | Time | Time of the connection | Continuous | 0.72 |
| 2 | Log comp | 1 if valid component; 0 otherwise | Discrete | 0.45 |
| 3 | Log subtype | 1 if successfully logged in; 0 otherwise | Discrete | 0.48 |
| 4 | Firewall rule | Which firewall rule allowed the connection | Continuous | 0.53 |
| 5 | NAT rule | Which NAT rule allowed the connection | Continuous | 0.40 |
| 6 | Firewall rule name | 1 if LAN_Allow-internet; 0 otherwise | Discrete | 0.55 |
| 7 | NAT rule name | 1 if fw#1_migrated_NAT_Rule; 0 otherwise | Discrete | 0.42 |
| 8 | Rule type | 1 if rule type 1; 0 otherwise | Discrete | 0.39 |
| 9 | Log occurrence | Number of log entries | Continuous | 0.50 |
| 10 | Live PCAP | Live packet capture presence | Continuous | 0.57 |
| 11 | Src IP | Source IP address | Continuous | 0.62 |
| 12 | Src port | Source port | Continuous | 0.68 |
| 13 | Dst IP | Destination IP address | Continuous | 0.63 |
| 14 | Dst port | Destination port | Continuous | 0.70 |
| 15 | In interface | Incoming interface | Continuous | 0.48 |
| 16 | Out interface | Outgoing interface | Continuous | 0.46 |
| 17 | Protocol | Protocol type (TCP, UDP, etc.) | Discrete | 0.77 |
| 18 | Bytes sent | Bytes sent from source to destination | Continuous | 0.85 |
| 19 | Bytes received | Bytes received from the destination to the source | Continuous | 0.83 |
| 20 | Connection duration | Duration of the connection (in seconds) | Continuous | 0.80 |
| 21 | Status | Connection status (normal or error) | Discrete | 0.58 |

This technique aligns with the findings of Nkiama et al. (2016), who discovered that precise feature selection significantly enhances IDS accuracy. Furthermore, investigations by Desyani et al. (2020), Upadhyay et al. (2021), and Otchere et al. (2022) demonstrate the efficiency of SelectKBest in reducing dimensionality while maintaining model performance.

### *2.3.3 Labeling*

Each network activity must be tagged as normal or malicious to construct a supervised learning dataset. Data labels can be appropriately labelled by network administrators, cybersecurity specialists, and threat intelligence. The labelled data was used to train the model. Based on our data set, we labelled normal as 1 and malicious as 2, 3, 4, 5, and 6 for Probe, R2L, U2R, DoS, and Abnormal/Unclassified, respectively.

Table 5

*After Preprocessing*

| Time | Log_comp | Log_comp | Log_subty | Log_subty | Firewall_r | NAT_rule | Firewall_r | Firewall_r | NAT_rule | NAT_rule | NAT_rule | Rule_type | log_occur | Live_PCAP | SrcIP | Srcport | DstIP | Dstport |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 45062 | 0 | 1 | 0 | 1 | -1 | 0 | -1 | -1 | -1 | -1 | -1 | 0 | 1 | 1 | -1 | 443 | 1 | 36144 |
| 45081 | 1 | 0 | 1 | 0 | 1 | 11 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 55213 | 1 | 443 |
| 45062 | 1 | 0 | 1 | 0 | 4 | 8 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 45891 | 1 | 53 |
| 45064 | 0 | 1 | 0 | 1 | -1 | 0 | -1 | -1 | -1 | -1 | -1 | 0 | 1 | 1 | 1 | 42230 | -1 | 443 |
| 45066 | 1 | 0 | 1 | 0 | 4 | 8 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 57741 | 1 | 53 |
| 45081 | 0 | 1 | 0 | 1 | -1 | 0 | -1 | -1 | -1 | -1 | -1 | 0 | 1 | 1 | 1 | 64406 | -1 | 443 |
| 45062 | 1 | 0 | 1 | 0 | 1 | 11 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 51922 | -1 | 80 |
| 45063 | 1 | 0 | 1 | 0 | 4 | 8 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 42540 | 1 | 53 |
| 45064 | 1 | 0 | 1 | 0 | 4 | 8 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 49380 | 1 | 53 |
| 45062 | 1 | 0 | 1 | 0 | 1 | 11 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 49501 | -1 | 443 |
| 45063 | 0 | 1 | 0 | 1 | -1 | 0 | -1 | -1 | -1 | -1 | -1 | 0 | 1 | 1 | 1 | 443 | 1 | 63148 |
| 45064 | 1 | 0 | 1 | 0 | 4 | 8 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 48547 | 1 | 53 |
| 45080 | 1 | 0 | 1 | 0 | 0 | 4 | 1 | 1 | 8 | 8 | 8 | 31 | 1 | 55 | 1 | 1 | -1 | 53 |
| 45080 | 1 | 0 | 1 | 0 | 1 | 11 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 55754 | 1 | 443 |
| 45081 | 1 | 0 | 1 | 0 | 1 | 11 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 52942 | -1 | 443 |
| 465 | 1 | 1 | 0 | 0 | 0 | 11 | 0 | 0 | 1 | 1 | 1 | 0 | 104 | 1 | -1 | 12420 | 1 | 161 |
| 45080 | 1 | 0 | 1 | 0 | 4 | 8 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 57153 | 1 | 53 |
| 45062 | 1 | 0 | 1 | 0 | 1 | 11 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 52812 | -1 | 80 |
| 45081 | 1 | 0 | 1 | 0 | 1 | 11 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1032 | -1 | 443 |
| 45063 | 1 | 0 | 1 | 0 | 4 | 8 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 3296 | 1 | 53 |
| 45062 | 1 | 0 | 1 | 0 | 4 | 8 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 41047 | 1 | 53 |
| 45062 | 1 | 0 | 1 | 0 | 1 | 11 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 38136 | -1 | 443 |
| 45063 | 1 | 0 | 1 | 0 | 1 | 11 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 61850 | -1 | 33155 |
| 45080 | 0 | 1 | 0 | 1 | 0 | 0 | -1 | -1 | -1 | -1 | -1 | 0 | 0 | 0 | 1 | 1 | -1 | 443 |

*3.0 Dataset Analysis and Applications*

*3.1 Dataset Distribution*

An initial analysis of the dataset shows a balanced distribution of benign and malicious traffic, making it suitable for training IDS models. The diversity of attack types captured by the honeypot contributes to the dataset's robustness.

*3.2 Splitting the Dataset*

Before model training, the dataset should be subdivided into training, validation, and testing subsets. The training set will be used to train the model, the validation set will be used to fine-tune hyperparameters, and the testing set will be used to assess the final model's performance.

The study employed a percentage-based technique in Python with the scikit-learn module to divide the dataset into training, testing, and validation subsets for evaluating the performance of the IDS model. Here is how it works:

Let:

The total number of samples (rows) in the dataset is 559,735.

Split into training (70%) and remaining data (30%).

*n_train* be the proportion of the dataset that was dedicated to train the model 70%

*n_train* = 559,735 * 70% = 391,814

Split remaining data (30%) into validation (50% of remaining) and test (50% of remaining).

Remaining data =Total number – training number

Remaining = 559,735 – 391,814

Remaining data =167,921

*n_test* be the proportion of the dataset that was dedicated to test the model 50% of the remaining

*n_test* = 167,921 * 50%= 83,961

The remained data will be for validation

*n_val* is the 50% of remained dataset that was dedicated to validation.

*n_val* = 167,921 – 83,961 = 83,960

It should be noted that n_train + n_test + n_val should equal one, i.e., 100% of the dataset. After determining the number of samples for each subset, the dataset was split using the scikit-learn library's train_test_split function twice: The pandas' to_csv method was employed to save the datasets created using scikit-learn's train_test_split function to CSV files. At the end, we have six CSV files with the respective data subsets (X_train.csv, X_test.csv, X_val.csv, y_train.csv, y_test.csv, y_val.csv).

Table 6

*Summary of the MD23 Dataset*

| Attack Type | MD23 | MD23-TRAIN-1 | MD23-VAL | MD23-TEST |
|---|---|---|---|---|
| Dos, Probe, U2R, R2L | 559,735 | 391,814 | 83,960 | 83,961 |

## 4.0 Results

*4.1 Model Selection and Training*

A PC with an Intel(R) Core (TM) 7 150U CPU running at 1.80 GHz and 16 GB of RAM was used to carry out the research. The MD23 dataset's CSV files were used to test binary classification and multiclassification techniques.

This study examined several machine learning models, including Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), XGBoost, and LightGBM. These models were chosen based on their proven effectiveness in prior IDS research and their ability to handle structured tabular data with complicated, non-linear patterns.

*4.1.1 Justification for Model Choice*

- Both XGBoost and LightGBM were utilised because, although they are gradient boosting frameworks, they have different strengths: XGBoost is well-known for its regularisation skills, which assist in reducing overfitting. It is resistant to noisy data and works well with medium-sized datasets.
- LightGBM, on the other hand, is intended for fast training and low memory utilisation, making it ideal for large-scale datasets. It employs a histogram-based method with leaf-wise tree growth, resulting in faster convergence and increased accuracy in many circumstances.
  Using both models enables a comparative investigation, exploiting their complementary capabilities to identify the best model for real-time IDS applications in resource-constrained contexts such as Tanzanian HLIs.

Using both models enables for a comparative investigation, exploiting their complementary capabilities to identify the best model for real-time IDS applications in resource-constrained contexts such as Tanzanian HLIs.

*4.1.2 Hyperparameters Used*

The following hyperparameters were set after tuning via grid search and cross-validation:

- XGBoost: n_estimators = 100, learning rate: 0.1, maximum depth: 6, subsample: 0.8, column sample by tree: 0.8, reg_alpha: 0.1, reg_lambda: 1.0.
- LightGBM: Number of estimators: 100, learning rate is 0.1, the number of leaves is 31, and the maximum depth is -1 (no restriction), features fraction: 0.8, bagging fraction: 0.8, bagging frequency: 5

These parameters were chosen to strike a compromise between model complexity, detection accuracy, and training speed, particularly considering the demand for scalable solutions in dynamic campus network contexts.

*4.2 Model Performance Evaluation*

We evaluated various machine learning techniques for Intrusion Detection Systems (IDS) using three datasets: MD23, UNSW_NB15, and KDD Cup99. To guarantee a fair comparison, the same number of data samples (559,735 instances) was randomly chosen from the UNSW-NB15 and KDD Cup 1999 datasets to match the size of the MD23 dataset during model evaluation.

The algorithms compared were Decision Tree, Gradient Boosting, XGBoost, and Random Forest. Accuracy, precision, recall, and F1-score were the performance metrics utilised to compare. Demonstration of the proposed model's results is shown in Table 5 accuracy (A), precision (P), recall (R), and F1-score (F1).

Table 7

*Proposed Model's Results Using the MD23 Dataset*

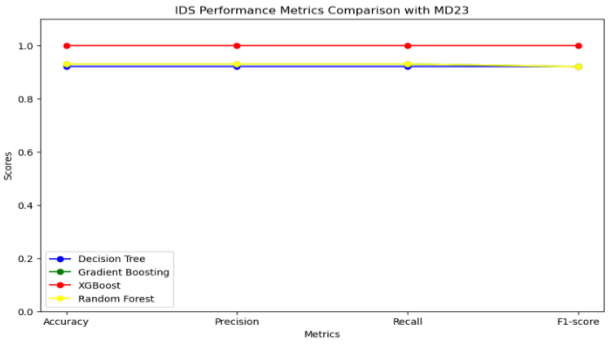
| Model | A% | P% | R% | F1% |
|---|---|---|---|---|
| DT | 0.92 | 0.92 | 0.92 | 0.92 |
| RF | 0.93 | 0.93 | 0.93 | 0.92 |
| XGBOOST | 0.99 | 0.99 | 0.99 | 0.98 |
| GB | 0.93 | 0.93 | 0.93 | 0.93 |

*4.2.1 The Overall Performance of MD23*

Table 7 compares the performance of multiple machine learning classifiers on the MD23 test dataset. XGBoost beat other models, earning the best accuracy (99.0%), precision (99.0%), and recall (99.0%), as well as a slightly lower but still respectable F1-score (98.0%). These findings illustrate its excellent capacity to detect both regular and malicious traffic, indicating its applicability for real-time intrusion detection in campus networks.

Figure 2

*Overall IDS Performance Metrics with MD23*

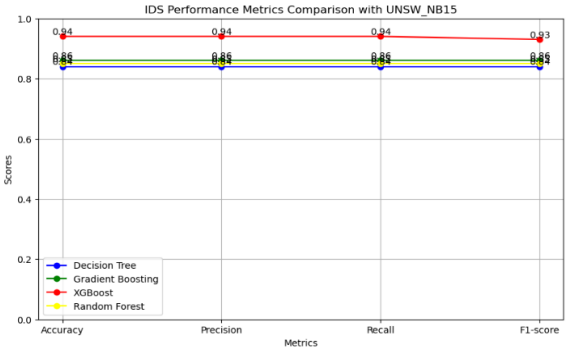


*4.2.2 The Overall Performance of UNSW_NB15*

Figure 5 shows the performance characteristics of various Intrusion Detection Systems (IDS) models tested on the UNSW_NB15 dataset. Accuracy, precision, recall, and F1-score are among the metrics used. These metrics provide a full evaluation of each model's ability to detect and categorise network intrusions.

Figure 3

*Overall IDS Performance Metrics with UNSW_NB15*

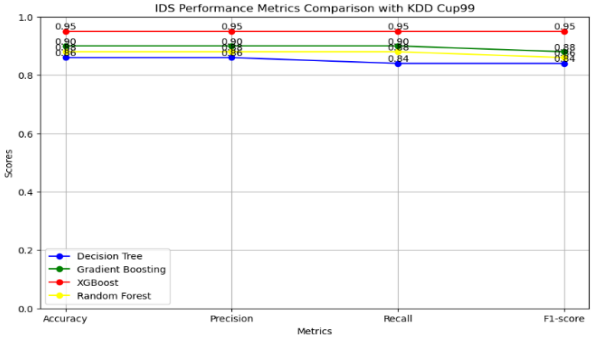



*4.2.3 The Overall Performance of KDD Cup99*

The KDD Cup99 dataset was used to evaluate different Intrusion Detection Systems (IDS) models, and the results are displayed in Figure 6. Among the measurements employed are accuracy, precision, recall, and F1-score. These metrics offer a comprehensive assessment of each model's classifier and intrusion detection capabilities.

Figure 4

*Overall IDS Performance Metrics with KDD Cup99*




Although the research was primarily concerned with offline evaluation, the method was developed with real-time deployment in mind. The feature extraction and classification stages are computationally efficient, and the models were designed to have low inference delay. Future work will involve deploying the system in a live environment to assess latency and throughput under realistic traffic loads, ensuring responsiveness for time-sensitive applications.

## 5.0 Discussion

The MD23 dataset is used to check how well IDS performs, and it shows that combined methods, especially XGBoost and Random Forest, often perform better than simpler models like Decision Tree and Gradient Boosting in all important measures. According to Table 5, XGBoost obtained 99.0% accuracy, 99.0% precision, 99.0% recall, and a 98.0% F1-score on the MD23 test set. These statistics demonstrate the model's high capacity to detect malicious traffic with few false alarms, which is critical for practical network security deployment.

This remarkable performance confirms the dependability of ensemble approaches for difficult IDS problems. XGBoost and Random Forest, unlike simpler classifiers, can better capture non-linear relationships and interactions in data, which is critical for accurately modelling traffic behaviour in Tanzanian higher learning institutions (HLIs), which frequently have unique infrastructure and user behaviour when compared to commercial networks.

This study fills a huge research gap by focusing on domain-specific datasets from developing-country university environments. The MD23 dataset, which was acquired from the MUST campus network in Tanzania, provides contextually relevant insights into local network setups, protocols, and user activity. As a result, the findings are particularly relevant to Tanzanian HLIs and similarly structured academic institutions in the Global South.

Gradient boosting performed well, especially on the KDD Cup99 and MD23 datasets, but it was significantly less robust than XGBoost. Random Forest scored well but fell below the best models, presumably due to its reliance on random feature selection, which can neglect critical clues. Despite being understandable and simple to execute, the decision tree repeatedly produced the lowest

scores, exposing its limits in dealing with complicated intrusion detection tasks.

The system design is scalable and resource-efficient. As the dataset volume rose, the model maintained good accuracy while incurring minimal computing cost. This is mostly owing to efficient feature selection and preprocessing techniques, which enable deployment in resource-constrained situations such as academic institutions with restricted IT budgets.

Despite these advantages, the study is not without limits. First, the MD23 dataset only covers DNS flows, which may not reflect the entire range of attack behaviours found in broader network traffic. Second, despite having 559,735 classified flows, the dataset was not made available owing to privacy and ethical concerns. Future iterations should include anonymisation techniques or seek ethical approval to allow open access for replication. Finally, the results are based solely on classification accuracy; using additional variables such as real-time detection delay may provide a more thorough performance assessment.

## 6.0 Conclusion

This study illustrates that creating a domain-specific dataset improves the performance of Intrusion Detection Systems (IDS) when compared to using only public datasets. Our empirical study of three datasets—MD23 (custom), UNSW_NB15, and KDD Cup99—showed that machine learning models, particularly XGBoost, produced superior accuracy, precision, recall, and F1-score on the MD23 dataset, with near-perfect performance (1.00 across all measures).

The enhanced results illustrate the necessity of customising datasets to the relevant network environment, since MD23 better represented Tanzanian college networks' unique traffic patterns and security features. Public datasets, on the other hand, lacked contextual significance, resulting in significantly inferior performance.

Aside from accuracy, the suggested approach has several practical advantages, including reduced false positive rates, scalability to varied network conditions, and efficient resource utilisation. These properties make it ideal for real-time deployment in higher learning institutions (HLIs), particularly in underdeveloped countries.

This paper presents a verified methodology for creating and using domain-specific datasets in IDS development, as well as the MD23 benchmark for evaluating machine learning models in academic network contexts.

## 7.0 Recommendations

Based on the results of this study, we make the following recommendations:

i. Domain-Specific Data Collection
   Domain-specific datasets should be collected and curated by organisations as a top priority for IDS deployment. This ensures that the IDS is educated on data that accurately reflects the organisation's network's traffic and threat landscape.
ii. Algorithm Selection
   Choosing the proper algorithm is as critical as having domain-specific data. XGBoost outperformed expectations in this investigation and should be regarded as a top contender for IDS implementations.
iii. Continuous Data Updates
   Updating domain-specific datasets regularly is crucial for adapting to shifting threats. Continuous data collection and periodic IDS retraining will help to maintain high detection accuracy.
iv. Hybrid Approaches
   Exploring hybrid techniques that incorporate the strengths of various algorithms could help IDS perform even better. For example, combining XGBoost with other ensemble algorithms may result in even better outcomes.
v. Using Custom Datasets in Other Domains
   This study's methodology and findings can be applied to other fields where IDSs are crucial. Custom dataset collection should be a routine technique for improving IDS performance across multiple sectors.

The study concludes that a domain-specific dataset improves IDS performance significantly when compared to public datasets. Adopting the

recommended procedures can assist organisations in developing more robust and effective IDS systems adapted to their specific network environments. The use of XGBoost and Random Forest models is recommended for IDS applications with MD23 due to their superior performance in all evaluated metrics. The decision tree, while still effective, may not be as robust or reliable as the ensemble methods for this specific use case.

## 8.0 Funding

This work was supported by Mbeya University of Science and Technology.

## 9.0 Acknowledgement

We would like to acknowledge the MUST management for providing the infrastructure necessary for data collection and for supporting this research.

## 10.0 Conflict of Interest

The authors declare no conflict of interest.

## 11.0 References

Aljanabi, M., Ismail, M. A., & Ali, A. H. (2021a). Intrusion detection systems, issues, challenges, and needs. *International Journal of Computational Intelligence Systems, 14* (1), 560-571. https://doi.org/10.2991/IJCIS.D.210105.001

Al-Qatf, M., Lasheng, Y., Al-Habib, M., & Al-Sabahi, K. (2018). Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. *IEEE Access, 6*, 52843-52856. https://doi.org/10.1109/ACCESS.2018.2869577

Desyani, T., Saifudin, A., & Yulianti, Y. (2020). Feature selection based on Naive Bayes for Caesarean section prediction. *IOP Conference Series: Materials Science and Engineering, 879* (1), 012091. https://doi.org/10.1088/1757-899X/879/1/012091

Devi, P. P., & Kannan, S. (2021). Performance analysis of machine learning models for threats and attacks in network security traffic model. *International Journal of Engineering Research & Technology, 48* (12). *(Note: Journal title and other publication info should be confirmed.)*

Ghurab, G. G., Alshami, F., Alshamy, R., & Othman, S. (2021). A detailed analysis of benchmark datasets for network intrusion detection system. *Asian Journal of Research in Computer Science, 7* (4), 39-53. https://doi.org/10.9734/AJRCOS/2021/v7i430185

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity, 2*(1), 1-22. https://doi.org/10.1186/s42400-019-0038-7

Komisarek, M., Pawlicki, M., & Kozik, R. (2021). How to effectively collect and process network data for intrusion detection? *Entropy, 23* (11), 1532. https://www.mdpi.com/1099-4300/23/11/1532

Lalduhsaka, R., Khan, A. K., & Roy, A. K. (2021). Issues and challenges in building a model for intrusion detection systems. In *2021 5th International Conference on Information Systems and Computer Networks (ISCON)*. https://doi.org/10.1109/ISCON52037.2021.9702322

Mishra, P., Varadharajan, V., Tupakula, U., & Pilli, E. S. (2019). A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE Communications Surveys & Tutorials, 21*(1), 686-728. https://doi.org/10.1109/COMST.2018.2847722

Ngueajio, M. K., Washington, G., Rawat, D. B., & Ngueabou, Y. (2023). Intrusion detection systems using support vector machines on the KDDCUP'99 and NSL-KDD datasets: A comprehensive survey. In *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics* (Vol. 543 LNNS, pp. 609-629). https://doi.org/10.1007/978-3-031-16078-3_42

Nkiama, H., Zainudeen, S., Said, M., & Saidu, M. (2016). A subset feature elimination mechanism for intrusion detection system. *International Journal of Advanced*

*Computer Science and Applications, 7*(4), 239-244. https://doi.org/10.14569/IJACSA.2016.070419

Otchere, D. A., Ganat, T. O. A., Ojero, J. O., Tackie-Otoo, B. N., & Taki, M. Y. (2022). Application of gradient boosting regression model for the evaluation of feature selection techniques in improving reservoir characterisation predictions. *Journal of Petroleum Science and Engineering, 208*, 109244. https://doi.org/10.1016/j.petrol.2021.109244

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

Upadhyay, D., Manero, J., Zaman, M., & Sampalli, S. (2021). Gradient boosting feature selection with machine learning classifiers for intrusion detection on power grids. *IEEE Transactions on Network and Service Management, 18*(1), 1104-1116. https://doi.org/10.1109/TNSM.2020.3032618

Yu, Y., & Bian, N. (2020). An intrusion detection method using few-shot learning. *IEEE Access, 8*, 49730-49740. https://doi.org/10.1109/ACCESS.2020.2980136