IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 May 2024, accepted 8 June 2024, date of publication 18 June 2024, date of current version 27 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3416095

RESEARCH ARTICLE

Enhancing Sniffing Detection in IoT Home Wi-Fi Networks: An Ensemble Learning Approach With Network Monitoring System (NMS)

HYO JUNG JIN[1], FARSHAD RAHIMI GHASHGHAEI[1], NEBRASE ELMRABIT[2], YUSSUF AHMED[1], AND MEHDI YOUSEFI

[1]School of Computing and Digital Technology, Birmingham City University, B4 7XG Birmingham, U.K.
[2]Department of Cyber Security and Networks, Glasgow Caledonian University, G4 0BA Glasgow, U.K.

Corresponding author: Nebrase Elmrabit (nebrase.elmrabit@gcu.ac.uk)

This work was supported by Glasgow Caledonian University.

ABSTRACT Network packet sniffing is one of the techniques that is widely used in the network and cyber security fields. However, sniffing can also be used as a malicious technique that allows threat actors to intercept and capture data flow to collect various information within the victim network. Where the wireless network environment can be vulnerable to sniffing vulnerabilities attacks due to the broadcasting function of Wi-Fi network. Wi-Fi access point devices can often be compromised, and critical information is leaked through sniffing attacks. Moreover, since sniffing is usually one of passive attacks, it is very challenging to detect sniffing activity in the network completely. The primary aim of this research is to contribute to enhancing the security of Internet of Things (IoT) home Wi-Fi systems. This is achieved by applying ensemble machine learning technology with sniffing detection methods using a Network Monitoring System (NMS) to effectively identify and mitigate potential sniffing behaviour within the IoT home Wi-Fi environment. Ultimately, this research will prove whether it is possible to precisely detect abnormal sniffing in a smart home Wi-Fi environment using machine learning techniques.

INDEX TERMS Ensemble learning, network monitoring system (NMS), smart home, sniffing, Wi-Fi.

I. INTRODUCTION

The Internet of Things (IoT) has advanced in home networks, with intelligent sensors, smart devices, and web-based systems designed for health and fitness monitoring [1]. The smart home adoption in the United Kingdom is currently hovering around 62% as of 2024, and projections anticipate it to surge to 115% by the year 2028 [2]. Nevertheless, the increasing popularity of smart devices also presents potential risks to users' security and privacy [3]. An instance of this is the cyber attack on smart thermostats in Finnish apartments, planned by hackers and resulted in residents enduring extreme cold during winter [4]. Furthermore, these systems generally rely on Wi-Fi within home networks to provide services such as remote monitoring, energy efficiency management, and safety and security management [1], [5]. Home Wi-Fi networks are vulnerable to sniffing devices due to their wireless broadcasting features, as described by [6]. Particularly concerning are tiny IoT devices employed in smart home networking, where credential management becomes challenging, and information is subject to exposure through sniffing attacks. In the case of a vulnerability in smart home networking equipment, crucial information is constantly leaked through sniffing attacks. As sniffing attacks function passively and often generate minimal logs, detecting them becomes more challenging than active attacks [7]. As a result, understanding the fundamental concepts of sniffing technology is essential for developing effective detection systems.

To successfully detect sniffing attacks, it is crucial to possess a thorough comprehension of sniffing technology and the risks it entails [8]. The traditional definition of sniffing encompasses a range of activities involving the monitoring, interception, and analysis of network traffic to capture data

The associate editor coordinating the review of this manuscript and approving it for publication was Ding Xu.

packets [9], [10]. In its proper application, a legitimate packet sniffer captures and examines only the packets intended for a particular computer. However, if the network interface card (NIC) is set to promiscuous or monitoring mode, the packet sniffer acquires the ability to intercept all network packets, regardless of their intended recipient. Promiscuous mode, a configuration for a network card, enables it to send all received traffic to the central processing unit, deviating from the usual practice of forwarding packets with specific addresses. When packet sniffers are set in promiscuous mode, threat actors can capture the network traffic. Significantly, within a network, user credentials, like usernames and passwords, are frequently transmitted without encryption, making them vulnerable to analysis through intercepted packets [11]. This vulnerability positions sniffing technology as a potentially valuable tool for malicious attackers. Consequently, there has been a collective endeavour to create detection technologies for sniffing as a defence against cyber hacking, marked by the recent integration of machine learning methods in this field.

This research aims to utilise ensemble machine learning technology for the identification of sniffing behaviour within the home Wi-Fi network environment. To achieve this aim. First, the study will conduct a complete evaluation of previous sniffing detection technologies and approaches that incorporate machine learning. Additionally, a pivotal aspect involves establishing a simulated testing environment tailored for a sniffing attack in a home Wi-Fi network. This controlled setup includes essential components such as a Wi-Fi router, a Sniffing device, and a network monitoring system. Alongside configuring the testing environment, the research explores factors influenced by sniffing activities, involving selecting pertinent features and systematic data collection through well-designed experiments. An integral part of the research entails training a dedicated sniffing dataset using ensemble machine learning technology, followed by a detailed analysis of the results of this training process.

The research presents a series of essential questions to guide the investigative process. These questions encompass identifying an optimal sniffing detection method suitable for a home Wi-Fi environment, exploring the effective applications of machine learning technology to enhance sniffing detection, examining the system characteristics within a home Wi-Fi environment influenced by the operation of a sniffing program, and exploring potential limitations and ethical considerations associated with the utilization of ensemble machine learning for sniffing detection methods.

This work introduces a multifaceted approach to enhance home Wi-Fi network security, particularly against sniffing attacks, which have become a concern due to the proliferation of IoT devices. A key element is the development of an ensemble machine learning model designed to detect and counteract these attacks. The research also involves setting up a controlled testing environment, including a Wi-Fi router, sniffing device, and Network Monitoring System (NMS), to simulate and evaluate sniffing attacks effectively. Additionally, the study applies ensemble machine learning in creating a training dataset, leading to a highly accurate model for sniffing detection with practical applications. Ultimately, this research offers valuable insights and strategies to bolster the security of smart home environments, thereby making a significant contribution to the broader field of Cyber Security and IoT protection.

The paper follows a structured format, starting with an introduction that outlines aims and objectives. It proceeds with an exploration of sniffing detection principles through a literature review, followed by a detailed methodology covering the proposed detection approach and experimental processes. The paper presents experimental outcomes, analyzes results, discusses findings, and addresses limitations and ethical considerations. The paper ends with a conclusion and future research topics.

II. LITERATURE REVIEW

Although there is existing literature on sniffing technology, research on sniffing detection has been relatively inactive. Limited studies have delved into detection methods, and some tools are outdated or incompatible with current systems [7]. Detection methods are broadly classified into host-based and network-based, with network-based methods further divided into Challenge-based and Measurement-based subgroups. Nowadays, machine learning-based (ML-based) methods have been gaining prominence [7].

A. HOST-BASED SNIFFER DETECTION METHOD

The study conducted by [12] introduced an innovative host-based method to detect network sniffers on computers. This method utilizes a specialized program known as an agent, which continually sends real-time information about the computer's network status to a central server. The agent program is installed on the local host to actively monitor and identify potential sniffing threats. In practical terms, the agent program employs a straightforward comparison between the computer's Media Access Control (MAC) address and the destination address of incoming packets. If the MAC address matches the destination address of the received packet, it implies that the computer is not currently executing any sniffing programs. Conversely, if the MAC address does not align with the received packet's destination address, there is a possibility of a sniffing threat on the device.

Another crucial aspect of this method involves the monitoring of incoming network traffic. The agent program sets a predefined threshold value, and if it detects an increase in incoming traffic surpassing this threshold, it promptly notifies the central server of the suspected sniffing activity. This dynamic analysis of network traffic patterns enhances the method's capability to identify abnormal behaviour associated with potential network sniffers. The proposed approach demonstrates versatility, being effective under both distributed and centralized management systems. However, challenges emerge when applying this method to IoT environments, such as smart homes. Installing agents on

IoT devices proves challenging due to the diverse operating systems employed in these environments. Many IoT devices rely on lightweight operating systems like Contiki, TinyOS, and RIOT, which present a unique set of challenges in adapting the agent installation environment to accommodate this diversity [13].

Furthermore, the installation of agents on small IoT devices may encounter resistance or rejection. As mentioned in [7], the vulnerability of the agent program can be easily disabled or manipulated by attackers. These challenges highlight the need for continuous research and robust security measures to address the evolving threat landscape, especially in IoT environments where diverse operating systems and device constraints pose unique difficulties. Efforts to refine and adapt host-based sniffer detection methods in light of these challenges remain essential for effective network security.

B. NETWORK-BASED SNIFFER DETECTION METHODS

The network-based sniffer detection method is a technique designed to identify suspicious sniffing behaviour across an entire local network, often monitored remotely [14]. This method involves intentionally generating packets for passive sniffing detection and analyzing the responses to identify potential sniffing systems. It can be broadly categorized into two subgroups: Challenge-based and Measurement-based [7].

1) CHALLENGE-BASED APPROACH

In the Challenge-based approach, the method intentionally provokes a response from a suspected sniffing system by sending carefully crafted network packets. These packets may contain forged MAC addresses or specific DNS messages. Detection occurs when the system under scrutiny responds. The efficacy of forged MAC address packets is notable when the Network Interface Controller (NIC) is in promiscuous mode, allowing the operating system to interpret the traffic as normal and respond accordingly [15], [16]. This also involves the use of DNS messages, often referred to as decoy-based detection. This technique employs random IP addresses to generate fake traffic within the local network. Legitimate devices typically ignore this simulated traffic, while devices equipped with sniffers might initiate reverse DNS lookup requests. By creating a fake three-way handshake resembling a genuine TCP connection, the method monitors DNS lookups to identify potential sniffing behaviour [14].

However, these detection techniques may encounter challenges if sniffing devices block network traffic and fail to respond to ARP, ICMP, and DNS request packets [17]. Additionally, these methods depend on the network card operating in promiscuous mode, which differs from the monitoring mode commonly used in Wi-Fi environments. In Wi-Fi settings, monitoring mode enables the reception of data without being connected to a network, transmitting all captured information to the operating system. This mode is used for Wi-Fi sniffing without requiring an IP address, making it challenging to detect [18], [19].

2) MEASUREMENT-BASED APPROACH

This approach involves sending network traffic to a system suspected of being sniffed and observing the system's performance degradation in response. The underlying idea is that when sniffing is in operation, the specific device causing it will impose a load on the entire system, leading to performance degradation. The load-measured method is executed based on two response time measurements, determining the response time of a system under low and high network traffic conditions. The measurement method employs the Round Trip Time (RTT) check, which sends an ICMP echo request packet and waits for its response packet to determine the response time.

The z-statistics method was employed by [20] in the context of a measurement-based sniffing detection approach. Due to the nature of the sniffing system operating in promiscuous or monitoring mode, capturing and monitoring all packets, the RTT measurement value is expected to increase significantly. Based on this, the detection system sends demand request packets to the sniffing system and collects RTT measurements. Training data is then collected, including RTT average, standard deviation, and change rate. The z-statistical model is utilized to determine whether sniffing is occurring. According to [20], the basic principle of the z-statistical model is as follows:

- There are two populations of round-trip time, and the population averages are defined as each μ_1, μ_2. They cannot be known because their populations are infinite. The difference between the two population averages is defined as μ.

$$\mu_1 - \mu_2 = \mu \tag{1}$$

- In this point, the null hypothesis of $\mu = 0$ and the alternative hypothesis of $\mu > 0$ are defined. The null hypothesis is tested rather than the alternative hypothesis, and if the null hypothesis is rejected, the alternative hypothesis will be accepted. To be specific, if two random samples of size n_1 and n_2 are chosen, the averages of the two RTT samples are displayed as $\bar{X}_1$, $\bar{X}_2$ and the difference between these two average values is defined as $\bar{X}$, like the next equation.

$$\bar{X} = \bar{X}_1 - \bar{X}_2 = \sum_{i=0}^{n_1} \frac{X_{1_i}}{n_1} - \sum_{i=0}^{n_2} \frac{X_{2_i}}{n_2} \tag{2}$$

- Moreover, two expected values $\bar{X}_1$, $\bar{X}_2$ are displayed in the following equation, and they are the same as μ_1, μ_2:

$$E(\bar{X}) = E(\bar{X}_1 - \bar{X}_2) = E(\bar{X}_1) - E(\bar{X}_2) = \mu_1 - \mu_2 = \mu \tag{3}$$

- As each random sample is independent, the standard error (SE) of their sum is defined as follows:

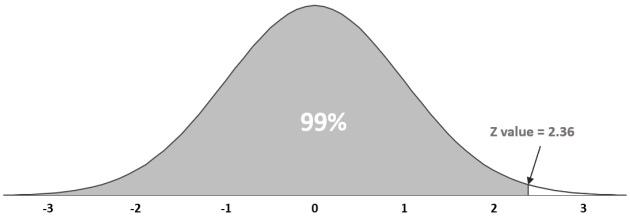


FIGURE 1. Standard normal curve.

$$SE(\bar{X}) = \sqrt{SE(\bar{X}_1)^2 + SE(\bar{X}_2)^2} \tag{4}$$

- In addition, two sample's standard deviations are:

$$S_1 = \sqrt{\frac{\sum_{i=1}^{n_1} \left(X_{1_i} - \bar{X}_1\right)^2}{n_1}} \tag{5}$$

$$S_2 = \sqrt{\frac{\sum_{i=1}^{n_2} \left(X_{2_i} - \bar{X}_2\right)^2}{n_2}} \tag{6}$$

- If the number of two samples is very large, the two sample standard deviations tend to be close to the standard deviations of the two populations:

$$SE(\bar{X}_1) \approx \frac{S_1}{\sqrt{n_1}}, \quad SE(\bar{X}_2) \approx \frac{S_2}{\sqrt{n_2}} \tag{7}$$

- Then, the standard error is reformulated in the following manner:

$$SE(\bar{X}) \approx \sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}} \tag{8}$$

- Therefore, The Z-statistic can be represented by the following equation:

$$Z = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}} \tag{9}$$

- Under the null hypothesis, the Z value should be expected as zero. However, if the expected value of Z is significantly greater than zero, it is appropriate to use a right-tail z-test and reject the null hypothesis. For example, the null hypothesis can be rejected when the z value is over 2.36 (equal to 99%), as shown in Figure 1.

C. SNIFFING DETECTION USING MACHINE LEARNING

The use of machine learning technology to improve a measurement-based method for detecting sniffing activities was demonstrated by [7]. The process involved periodically generating artificial load traffic, measuring response times through ping and curl commands on the system with sniffing, and securing this data along with sniffing system details. The XGBoost method, an ensemble machine learning technique, was then used to predict the outcomes based on the gathered dataset. The system diagram used in their experiment is detailed in the study. The MAC flooding technique served as the artificial load. This technique floods the local network with ARP messages, creating substantial traffic and proving effective for generating artificial load in confined network spaces like smart home networks. The impact of MAC flooding on system resources, especially on a device with an active sniffer, influenced response times. Machine learning techniques were then applied to determine the presence of a sniffer based on the generated data. This innovative use of machine learning demonstrated significant improvements in sniffing detection accuracy. The experimental validation showed that the AUC values exceeded 0.9999 (≈ 1), indicating exceptional model performance. The AUC values, representing the Area Under the Curve in ROC analysis, serve as a measure of the model's ability to distinguish between classes [21]. Higher AUC values indicate better predictive accuracy. Swets' interpretation of AUC values categorizes them as can be seen in the Table 1:

TABLE 1. Interpretation of AUC values.

AUC Range	Interpretation
$AUC = 0.5$	Can't distinguish better than random chance.
$0.5 < AUC \leq 0.7$	Moderate differentiation ability.
$0.7 < AUC \leq 0.9$	High precision in classification.
$0.9 < AUC < 1$	Exceptional discrimination ability.
$AUC = 1$	Flawless performance, perfect discrimination.

Despite these achievements, The study highlights a small gap in response time prediction in a wireless Wi-Fi environment. Detecting sniffing activities solely through network-based methods in a Wi-Fi setting is considered complex. Additionally, the study notes that ICMP responses operating in NIC kernel mode have minimal impact on the CPU load of the sniffing system [14]. The innovations in Wi-Fi technology can require further research to determine the feasibility of sniffing detection based solely on network response time under varying loads.

III. METHODOLOGY

This study suggests using a NMS as an effective method for detecting sniffing activities, applicable across various IoT infrastructure environments. The NMS serves as a tool to visualize the network's real-time state, determining whether connected devices display normal or abnormal behaviour [22]. Recent research has also explored lightweight network monitoring methods designed for IoT environments [23]. The rationale for adopting this approach lies in the NMS's existing connectivity and management of diverse IoT devices, simplifying the collection of machine learning data for sniffing detection. Additionally, in this research, the load-based method proposed by [7] was incorporated. Given that most sniffing operations are passive, enhancing the sniffer's response is expected to facilitate easier sniffing detection. Furthermore, this paper demonstrates the potential to detect specific sniffing programs and their operational states by applying ensemble machine-learning techniques to data extracted from the NMS.

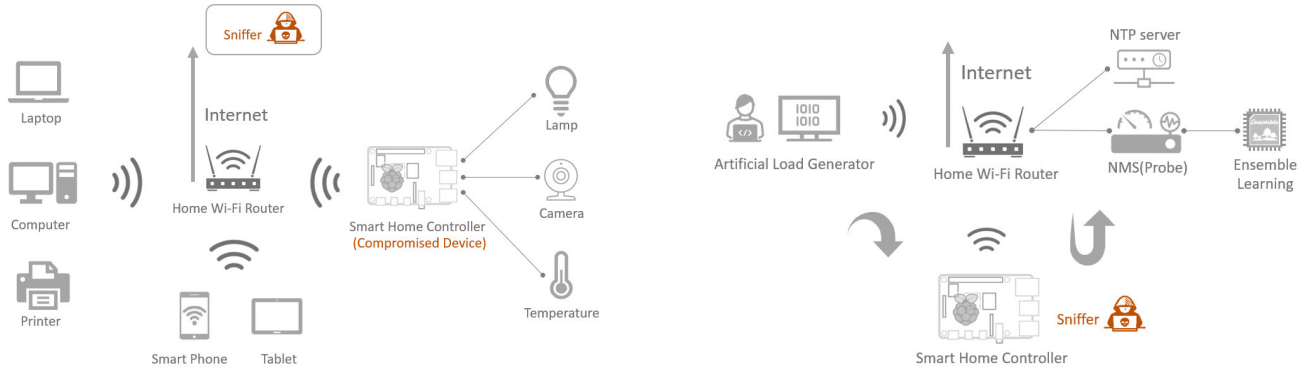


FIGURE 2. Sniffing threat model and proposed architecture for detecting sniffing.

As can be seen in Figure 2(a), a simulated environment was designed for threat modelling to make a controlled analysis of sniffing threats on home Wi-Fi networks. This environment represents a compromised device (Smart Home Controller) that actively interacts with other devices. A dedicated system, represented as the sniffer, equipped with popular packet capture tools like Tcpdump and Wireshark monitors network traffic for sniffing activity as they are recognized as the most popular open-source programs globally [10]. The system also collects data for analysis, enabling the evaluation of countermeasures against these prevalent sniffing programs. This threat modelling approach assists us in identifying potential attackers for our smart home networks and the potential impact they may have. This also helps us to detect and examine abnormal data packets and develop strategies to prevent them. In addition, this aids our ongoing attempts to mitigate these threats and ensure the security of our smart home Wi-Fi networks.

A. PROPOSED SNIFFING DETECTION

Detecting passive sniffing within a home Wi-Fi environment poses a significant challenge. This experiment adopts an artificial load-based sniffing detection method, as proposed by [7]. However, a notable modification is made to the data collection method, switching to an NMS commonly used in IoT infrastructure environments. Anomaly detection within an IoT environment through an NMS is a well-established technology [22]. According to [24], real-time identification of malicious abnormal traffic can be achieved through systems that provide various statistical analyses and graphical visualizations for the collected data. The data collection methods employed using NMS have the advantage of gathering hybrid data, combining both network-based and host-based approaches. This includes the collection of network-based features such as ICMP response time, real-time traffic volume, and router performance, along with host-based elements like CPU, memory, disk, and load performance. The justification for gathering such different data comes from the fact that, while sniffing is passive, it can have a variety of consequences on the target device. The fundamental purpose is to analyse these effects, document changes in each effect, and determine ways to identify sniffing using machine learning technologies. Furthermore, the proposed architecture for the experiment, as shown in Figure 2(b), aims to enhance sniffing detection performance by subjecting the collected data to various machine learning algorithms, aligning with the approach suggested by [7]. Specifically, measurement-based sniffing detection methods using artificial loads cause various performance changes in the target system, and the NMS collects these various performance changes as real-time data to accurately classify whether the target system is sniffing through ensemble learning.

The detailed hardware configuration used in the experiment is as follows:

1) HARDWARE CONFIGURATION

- The experiment utilized specific hardware configurations, including a Raspberry Pi Gen4 as the target device with a 1.5Ghz 64-bit Quad-core processor, 8GB of RAM, IEEE 802.11ac WiFi, and a 256GB micro-SDXC Class 10 for disk storage. Running on Raspibian 9.6d, the device employed Tcpdump and Wireshark as sniffer programs, connected to the NMS via SNMP V2C.
- An artificial load generator, implemented as a Virtual Machine, featured 2 CPUs, 8GB of RAM, IEEE 802.11ac WiFi, and a 125GB HDD. Running Kali Linux 6.1.0, the Virtual Machine utilized the macof.py MAC flooding utility and operated on VirtualBox 7.0.
- The NMS, hosted on a desktop, boasted a 3.4Ghz 64-bit Quad-core processor, 16GB of RAM, IEEE 802.11ac WiFi, and a 500GB SSD. Running on Windows 10, the NMS utilized Paessler PRTG for network monitoring.

2) SOFTWARE DESCRIPTION

- The software components included Tcpdump-4.99.3 [25] and Wireshark-4.0.6 [26] as sniffer programs, with Tshark-4.0.6 [26] as the command-line version

TABLE 2. Situations for the Sniffing detection experiment.

Symbol	Category	Description
NTS	No Sniffer (No Load)	Normal state
TVM	Tcpdump Real-time	Real-time monitoring
TWM	Tcpdump Background	Background capture
WSK	Wireshark Real-time	Real-time monitoring
TSK	Tshark Background	Background capture
NTSwL	No Sniffer (With Load)	Normal state with load
TVMwL	Tcpdump Real-time (Load)	Real-time monitoring
TWMwL	Tcpdump Background (Load)	Background capture
WSKwL	Wireshark Real-time (Load)	Real-time monitoring
TSKwL	Tshark Background (Load)	Background capture

of Wireshark. The artificial load generator employed macof.py [27] as a MAC address table overflow utility. The Network Monitoring Software, Paessler PRTG available at [28], served as a small or midsize network monitoring solution for Windows systems.

B. EXPERIMENTAL METHODOLOGY FOR DATA PREPARATION

Systems, including a target device equipped with a sniffer, artificial load generator, NMS, and ensemble learner, were deployed in a home Wi-Fi network environment. Ensuring time synchronization across all network systems through a Network Time Protocol (NTP) server was essential. This synchronization was crucial to align the timelines of the artificial load system, target device, and NMS, ensuring accurate data collection corresponding to specific scenarios. The collected data were then classified based on timestamps, a necessary step for the application of supervised machine learning to the gathered data. The artificial load generator employed the ''Macof'' tool to transmit 2,000 packets per second to the target device, regularly applying the load through the Linux command ''Crontab''. This value was determined through experimentation as a load that efficiently collects data without overloading the home Wi-Fi network. Therefore, the number of packets per second must be adjusted appropriately depending on network capacity.

Simultaneously, the target device executed each sniffing program tailored to various situations. The experiment assumed two situations: one wherein a malicious hacker utilizes a sniffing program to monitor in real-time, and the other wherein a sniffer is already installed. Sniffing data was saved to disk in the background mode. Considering the no-load test and artificial load test based on the operation of the artificial load generator, a total of 10 situations were investigated, each representing a class value for machine learning, as shown in Table 2.

NMS facilitated the real-time collection of characteristic data using a Simple Network Management Protocol (SNMP). The NMS was configured to collect data at 10-second intervals, minimizing scan time errors for characteristics such as CPU, Memory, and Disk. Training data for machine learning purposes were derived from the average data collected at 1-minute intervals.

In the experiment, the PRTG NMS, utilizing the SNMP protocol, collects diverse information about the target device's condition, including CPU, memory, disk, system load, ping time, and traffic [29]. These characteristics serve as potential features for machine learning. Notably, changes in these features reveal insights into the operating status of the sniffing program. CPU load changes with more noticeable fluctuations under loaded conditions. However, relying only on the CPU's performance for sniffing detection is difficult, especially when the program is running in write mode, which simulates regular CPU load (NTS) without sniffing. Disk access values highlight the importance of write features in detecting sniffing activities. Changes in disk write become more apparent during sniffing, but discerning these changes amidst simultaneous normal program activities poses challenges. The Linux system load reflects CPU usage variations, especially during the use of Tcpdump verbose mode or Wireshark. Lower load averages are generally favourable, and an increase signifies potential issues or an overloaded system. Memory usage remains relatively stable, yet certain sniffing programs, such as Wireshark, occupy memory, presenting a noticeable pattern. Ping response time values fluctuate notably in a Wi-Fi setting, posing challenges in reliable sniffing detection. This is even more noticeable when using artificial load generation. Multiple factors, including wireless interference, network congestion, Quality of Service (QoS), channel interference, signal strength fluctuations, and sporadic packet loss can explain the fluctuation of ping response times in Wi-Fi configurations. These factors affect the accuracy of transmission of data throughout the network. However, distinctions in response time become apparent based on load and no-load conditions. Network traffic values demonstrate changes based on load and no-load states. Identifying sniffing conditions becomes more accurate by configuring learning data through alterations in this network monitoring characteristics and applying ensemble machine learning algorithms.

C. DATASET DESCRIPTION

The dataset obtained from the experiment is designed for supervised learning, classifying instances to determine whether a sniffer is active on the target device. After 20 days of data collection, the dataset comprises 12,000 instances with 31 attributes. Each label is represented by 1,200 instances, ensuring data balance. Table 3 provides a detailed description of the dataset, featuring thirty attributes and one class. The selected features include channel data from the six scanners of the NMS.

Examining each feature shows details about the Raspberry Pi equipment used as the target device. With 4 cores, it has 5 channels, including CPU load characteristics for each processor and the average total CPU load, totalling 5 features. Disk characteristics, with a focus on the disk writing process during background operations like sniffing in write mode, result in four derived channels as features. Linux

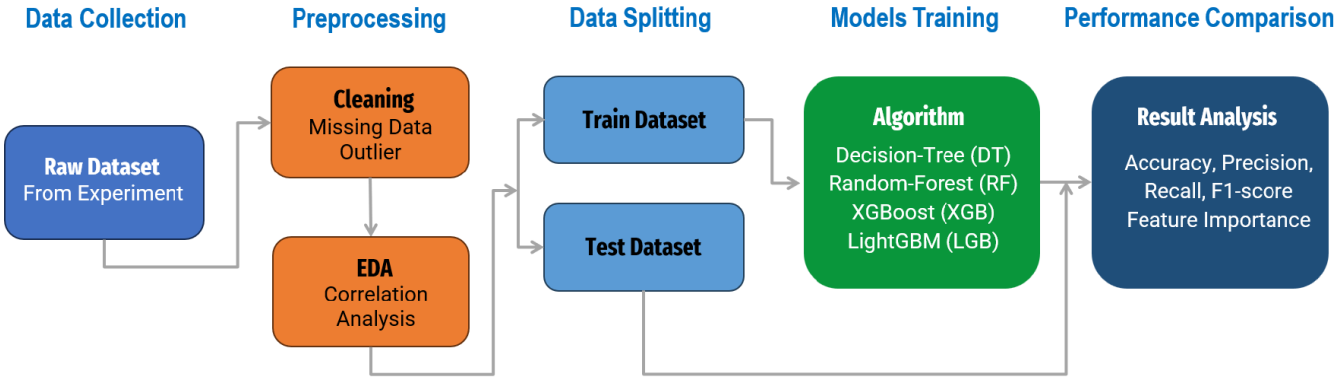


FIGURE 3. The Architecture of Ensemble Learning for Sniffing Dataset.

TABLE 3. Dataset details.

Scanner	Attribute	Description
CPU	Total, P1, P2, P3, P4	CPU load
Disk	wrVolume, wrSpeed, wraccessVolume, wraccessSpeed	Disk write volume and speed,
Load	1min, 5min, 15min	Linux system load
Memory	Avail, Avail%, Physical, Physical%, Swap, Swap%, freeTotal, freeTotal%	The percentage and volume of Available, Free physical, Free swap, and Total free memory
Ping	Average, Min, Max, PacketLoss	The Average, Minimum and Maximum Ping Time, Packet Loss
Traffic	totalVolume, totalSpeed, inVolume, inSpeed, outVolume, outSpeed	The Volume and Speed of Total, Input and Output Traffic
Class	Class	The target variable

system load attributes include average changes over 1 minute, 5 minutes, and 15 minutes, contributing three channels as features. Memory information, displayed as both percentage and volume (byte) for available memory, physical memory, swap memory, and total memory, amounts to 8 selected channels as features.

Ping values are derived from executing 5 pings, and there are 4 channels, such as the average, maximum, minimum, and packet loss in the case of no response. Traffic features consist of input, output, and total traffic volume and speed, containing six channels with distinct characteristics.

D. ADOPTING ENSEMBLE LEARNING FOR SNIFFING CLASSIFICATION

Accurate classification of sniffing in the experiment dataset is crucial, and machine learning technology plays a vital role in achieving this goal. While there are various machine learning algorithms available, this study places particular emphasis on the ensemble learning algorithm. Ensemble algorithms enhance classification performance and efficiency by combining diverse algorithms, demonstrating superior classifier performance compared to other machine learning approaches [30]. Notably, many winners of global machine learning competitions, such as Kaggle, have leveraged ensemble algorithms [31].

Ensemble algorithms offer the advantage of adaptability to changes in the data stream, a critical feature for dynamic datasets [32], [33]. Ensemble algorithms are useful because of their ability to adapt to changes in data streams, which is especially useful for dynamic datasets. This adaptability comes from their collaborative nature, as they use the combined knowledge of several models to create predictions. Ensemble algorithms may efficiently capture a broader range of patterns in data by combining predictions from multiple models, maintaining reliability even as the dataset evolves over time. Among ensemble learning methods, bagging and boosting are commonly employed to enhance accuracy. In this study, three representative ensemble learning technologies were utilized as sniffing classification models: Random Forest, based on bagging, and XGBoost and LightGBM, based on boosting. Additionally, the decision tree algorithm, though not part of the ensemble learning method, was employed as a single model for performance comparison, considering that the three ensemble algorithms are tree-based machine learning models. Moreover, ensemble learning improves performance by generating multiple trees, which complicates interpretation, whereas a decision tree uses a single tree, simplifying the interpretation based on features. The evaluation of sniffing classification accuracy and feature importance was conducted using these four algorithms. The architecture for Ensemble Learning on the collected dataset is illustrated in Figure 3.

E. INTEGRATED DATA PRE-PROCESSING AND EXPLORATORY DATA ANALYSIS (EDA)

The effectiveness of machine learning algorithms is notably influenced by data pre-processing and Exploratory Data Analysis (EDA). Pre-processing activities encompass addressing issues such as missing values, outliers, and feature correlations, which are critical for achieving optimal predictive performance. It is important to highlight that for tree-based algorithms such as random forest and XGBoost, the process of feature scaling is considered unnecessary. On the other hand, EDA involves a thorough examination of data through the use of graphical tools and descriptive

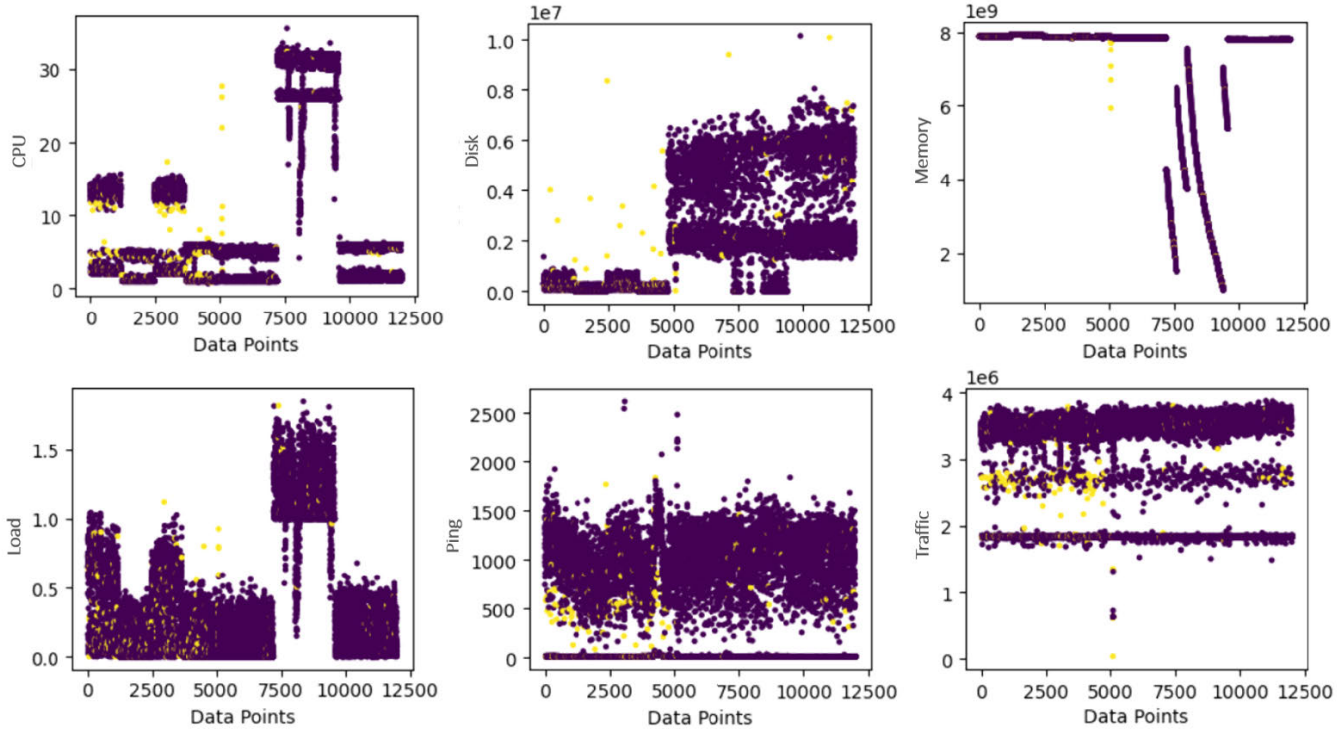


FIGURE 4. The results of the outlier detection using the LOF algorithm.

statistics, improving comprehension overall.In the final stage of pre-processing, the dataset is divided into training and testing subsets in a proportion of 80 to 20 to support ensemble learning.

1) DATA CLEANING AND INVESTIGATION

The original dataset indicated no missing values. The outliers detected by the Local Outlier Factor (LOF) algorithm were replaced to prevent accuracy loss. Features with a standard deviation of zero were eliminated. Correlation analysis helps to reduce multicollinearity. Figure 4 shows the results of the investigation of outliers. The yellow spots displayed on the screen represent outliers. For correction, outlier data was replaced with the minimum value of the inlier if it was lower than the minimum value and the maximum value of the inlier if it was higher than the maximum value. This was adjusted to find outliers in a range of 4 neighbours (k = 4) and 2% of the total data, and as a result, most outliers were taken to address.

EDA included multicollinearity detection and feature correlation analysis. A heatmap displayed closely related features that required their reduction to solve multicollinearity. Two final datasets, ''modified_18f'' and ''modified_28f,'' were generated.

2) SEPARATION OF DATA INTO TESTING AND TRAINING

The data was split into training and testing sets (80:20). The files ''train_18f,'' ''test_18f,'' ''train_28f,'' and ''test_28f'' were created, each containing 9,600 training and 2,400 test data instances.

F. ENSEMBLE LEARNING ALGORITHMS

The ensemble algorithm operates by enhancing the performance derived from a single tree. Thus, a fundamental grasp of the single tree principle, the basic structure of ensemble algorithms, and the principles of ensemble through bagging and boosting is crucial.

1) SINGLE TREE ALGORITHM: DECISION TREE

Decision Tree serves as a single tree algorithm, utilizing a tree-like structure to partition data. It starts from the root and undergoes splits until reaching a terminal leaf node, providing accurate solutions. Decision trees are used in data mining for classifying large datasets. The informativeness of attribute features is crucial, assessed through ''information gain'' and computed as the entropy of the dataset. Entropy acts as a metric for evaluating the randomness or impurity present in a dataset, with values ranging from 0 to 1. Lower entropy values indicate enhanced performance, while higher values suggest increased impurity. If the target has different attribute values denoted as c, the classification entropy of set S for c is determined by the following equation:

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i) \tag{10}$$

TABLE 4. The learning result (%) of 28 features.

Algorithm	Cross-Validation				Test			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
Decision Tree	99.80	99.81	99.80	99.80	99.67	99.67	99.67	99.67
Random Forest	99.83	99.84	99.83	99.83	99.92	99.92	99.92	99.92
XGBoost	99.75	99.75	99.75	99.75	99.83	99.83	99.83	99.83
LightGBM	99.76	99.76	99.76	99.76	99.79	99.79	99.79	99.79

TABLE 5. The learning result (%) of 18 features.

Algorithm	Cross-Validation				Test			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
Decision Tree	99.76	99.76	99.76	99.76	99.62	99.63	99.62	99.63
Random Forest	99.91	99.91	99.91	99.91	99.88	99.88	99.88	99.88
XGBoost	99.73	99.73	99.73	99.73	99.83	99.83	99.83	99.83
LightGBM	99.74	99.74	99.74	99.74	99.71	99.71	99.71	99.71

The information gain, denoted as $Gain(S, A)$, is defined by the following equation, where $V(A)$ represents the range of attribute A, and S_v is a subset of set S equal to the value of attribute v:

$$\text{Gain}(S, A) = \sum_{v \in V(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \tag{11}$$

2) RANDOM FOREST (BAGGING ALGORITHM)

Random Forest is an algorithm designed to address overfitting, a common issue with decision trees, using a bagging method. It performs effectively for both regression and classification tasks. RF leverages ensemble learning principles, employing a bootstrap method to create multiple subsets of samples. It creates decision trees for each subset and aggregates them into a random forest. The algorithm generates predictions through majority voting or averaging across decision trees. Bagging involves creating and learning individual decision trees from the dataset, contributing to the overall performance. Random Forest significantly reduces overfitting.

3) XGBOOST AND LIGHTGBM (BOOSTING ALGORITHM)

While Random Forest utilizes bagging, XGBoost (eXtreem Gradient Boosting, XGB) and LightGBM (Light Gradient Boosting Machine, LGB) enhance performance through boosting techniques. Based on Gradient Boosting (GB), XGBoost and LightGBM sequentially train numerous weak learners, each focusing on misclassified samples from the preceding learner. The objective is to minimize the cost function associated with the disparity between the actual value and its approximation, optimized using gradient descent. XGBoost combines cause-based decision trees and gradient boosting machines, improving accuracy and speed through parallel processing techniques. It efficiently handles large datasets with numerous attributes and classifications, providing an optimized solution. LightGBM, developed later than XGBoost, employs a leaf-wise algorithm, ensuring high parallel training efficiency, reduced memory consumption, and support for distributed computing.

IV. RESULTS

After preprocessing and formatting the datasets, an ensemble of classification algorithms. Initial training involved a dataset with all 28 features, followed by training with a reduced dataset containing 18 features. The 10-fold cross-validation method was applied during training, and metrics like Accuracy, F1-score, Precision, and Recall were computed. Feature importance analysis was conducted to identify the features most affected by sniffing. As can be seen in Algorithm 1, which is the pseudo-code to show how the evaluation and comparison process of the ensemble learning for detecting sniffing attack works. It starts after the data collection process by loading training and testing data from CSV files specified by train_path and test_path, with options to drop specified labels (labels_to_drop) and features (features_to_drop) that are determined in the preprocessing stage. After splitting the data into training and testing sets, it initializes Decision Tree, Random Forest, XGBoost, and LightGBM. By defining performance metrics such as accuracy, precision, recall, and F1-score, it evaluates each model through cross-validation of the training data. Then it assesses model performance on the test data and generates confusion matrices for analysis. Ultimately, it provides us with the model metrics and confusion matrices for comparison and evaluation.

A. COMPARISON OF 28 FEATURES AND 18 FEATURES LEARNING RESULTS

Two training datasets with 28 features and 18 features were created through an ensemble algorithm. The models utilized Scikit-Learn which is a Python-based machine-learning tool. All models underwent a 10-fold cross-validation process, and after training, each model was tested using a separate dataset. Tables 4 and 5 present the accuracy, precision, recall, and F-score for Cross-Validation and Test results. The obtained results reveal that both 28 features and 18 features achieved an accuracy of over 99%, with precision and recall performances surpassing 99%. This highlights the successful use of machine learning in sniffing detection. Notably, features with a correlation coefficient of 1 had

Algorithm 1 Model Evaluation and Comparison

Require: train_path (str), test_path (str), labels_to_drop (list), features_to_drop (list)
Ensure: model_metrics (dict), confusion_matrices (dict)
train_data = LoadCSV(train_path)
test_data = LoadCSV(test_path)
if labels_to_drop is not None **then**
train_data = RemoveRows(train_data, labels_to_drop)
end if
if features_to_drop is not None **then**
train_data, test_data = RemoveFeatures(train_data, test_data, features_to_drop)
end if
X_train, y_train = SplitData(train_data)
X_test, y_test = SplitData(test_data)
models = InitializeModels(DecisionTree, RandomForest, XGBoost, LightGBM)
metrics = Accuracy, Precision, Recall, F1Score
model_metrics = []
for all model in models **do**
model_metrics[model.name] = PerformCrossValidation(model, X_train, y_train)
end for
model_metrics = EvaluateOnTest(models, X_train, y_train, X_test, y_test)
confusion_matrices = GenerateConfusionMatrices(models, X_test, y_test)
return model_metrics, confusion_matrices

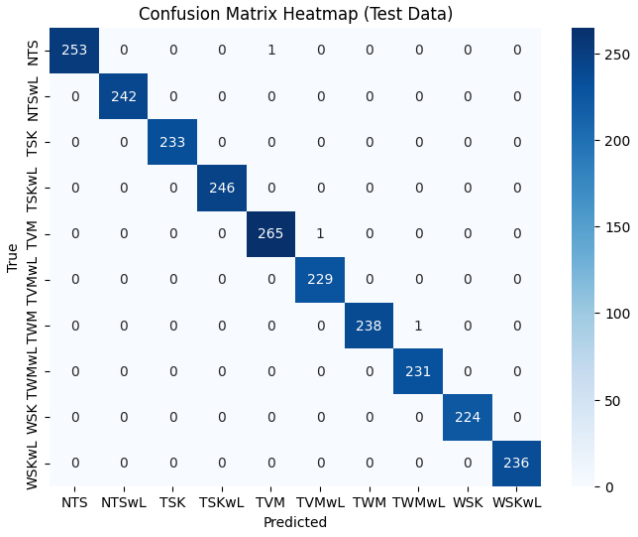


FIGURE 5. The multi-label confusion matrix heatmap using random forest.

no impact on performance. Parameters were consistently set (max_depth=7, random_state=42) to prevent overfitting. Upon examination, Random Forest exhibited superior test performance, while the Single Tree Structure showed less helpful outcomes, though without significant differences.

TABLE 6. Comparison of accuracy and training time.

Algorithm	Test Accuracy (%)		Training Time (sec)	
	18 features	7 features	18 features	7 features
Decision Tree	99.62	99.62	0.73	0.44
Random Forest	99.88	99.88	13.22	10.79
XGBoost	99.83	99.92	10.14	7.85
LightGBM	99.71	99.79	8.22	5.95

The multi-label confusion matrix heatmap in Figure 5 shows an almost perfect match between vertical predicted values and horizontal true values. This alignment indicates very accurate classification, highlighting the model's ability to predict and assign labels across several classes.

B. FEATURE IMPORTANCE

The experimental results show no significant difference in test accuracy between models with 28 features and those with a reduced set of 18 features. This suggests that removing collinear features (correlation coefficient of 1) doesn't lead to accuracy loss, indicating potential resource efficiency. Furthermore, it is essential to analyze the feature importance to identify which characteristics of the system are affected by the sniffing program. As a result, seven important features were found in memory, disk, traffic, and CPU. Among them, the operating status and type of the sniffing program have the greatest impact on memory capacity. In addition, the accuracy did not change significantly after learning after removing the remaining 11 features except for these 7 features. Comparing the test accuracy and training time of the 18 and 7 feature models, as shown in Table 6, the learning speed is improved without decreasing accuracy. In terms of speed, the decision tree is naturally the fastest because it is a single tree, while the LightGBM stands out among ensemble models.

C. THE LABEL ANALYSIS THROUGH TREE GRAPH

To analyze the predominant characteristics of each label within the 7-feature model, an examination of the decision tree model's tree graph is necessary. The ensemble model creates numerous trees to enhance learning performance, making interpretation challenging. To simplify interpretation, the maximum tree depth was adjusted by 5 without significantly reducing accuracy. Figure 6 illustrates the decision tree graph, where the blue ovals represent leaf nodes containing the highest sample counts among nodes classified by the final labels.

In the tree graph, memory usage serves as the root node for the initial decision, with Wireshark (WSK) consuming the most memory. Tshark (TSK), a related program, also exhibits notable memory usage. Furthermore, the decision on whether to use artificial load is determined by the input/output traffic volume. The difference between Tcpdump's verbose mode (TVM) and Not to sniffing (NTS) hinges on CPU usage, while the size of the disk usage determines the distinction from Tcpdump's write mode (TWM). Utilizing machine learning enables the identification of distinctive

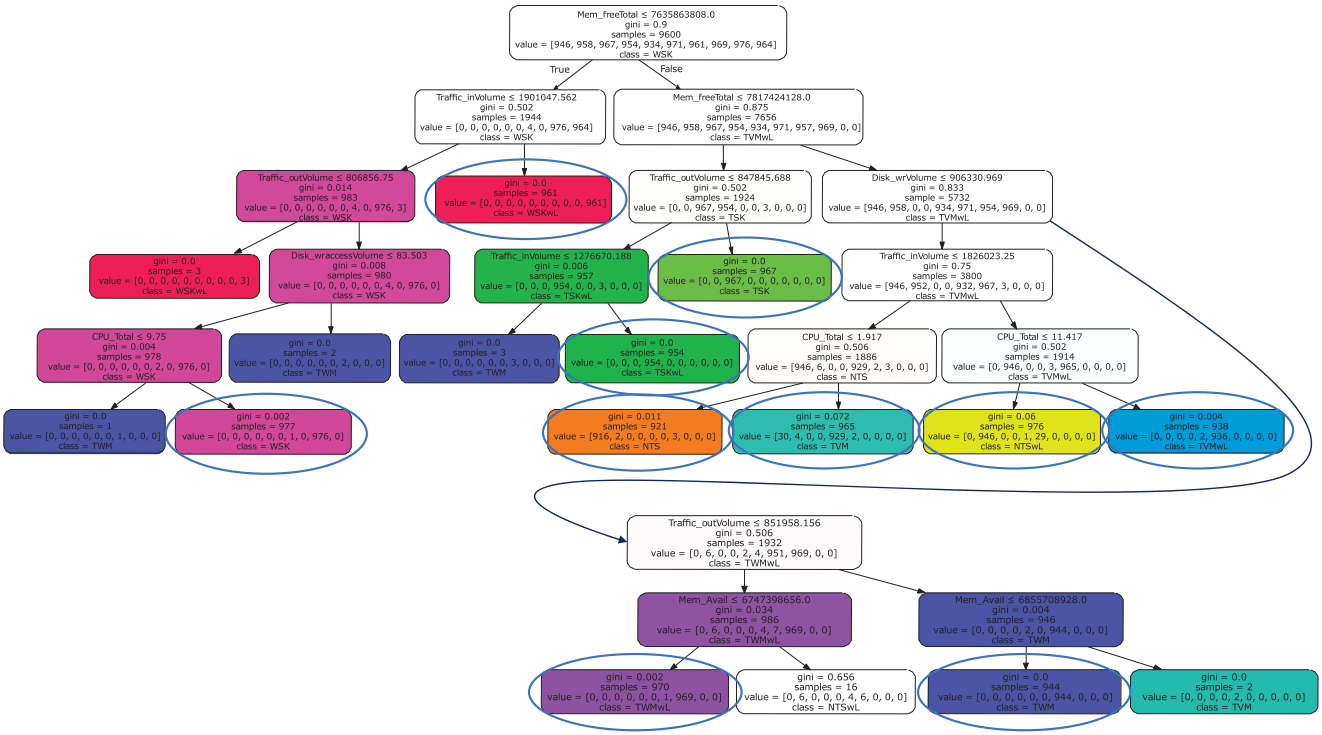


FIGURE 6. Tree graph of decision tree model.

attributes associated with each sniffing program. This process substantiates the effective detection of sniffing activities by employing a NMS coupled with ensemble machine learning technologies.

D. THE SNIFFING DETECTION PERFORMANCE OF THE PING FACTOR

An experiment was conducted, focusing purely on the ping factor, to assess the feasibility of RTT-based sniffing detection in IoT home Wi-Fi environments. The results revealed low accuracy, approximately 40%, indicating that the RTT-based method struggled to detect sniffing program characteristics. However, in binary classification between the operational states of each sniffing program (TVM, TWM, WSK, TSK) and the non-operational state (NTS), the performance averaged over 70%, with all labels displaying accuracy over 65%.

Table 7 presents the results of the binary classification experiment for each label. Notably, TCPdump demonstrated high classification accuracy, exceeding 85%, even with ping characteristics. This suggests that the RTT values of devices running the TCPdump sniffing program undergo significant changes. The accuracy of TCPdump sniffing detection decreased when using artificial loads, while the accuracy of Wireshark-based sniffing programs slightly improved under the same conditions.

Figure 7 shows the ROC curve and AUC values with and without artificial load. Using artificial load does not necessarily improve sniffing detection performance. In other words, it can be seen that the impact of artificial load varies depending on the characteristics of the sniffing program. Therefore, additional research on remote sniffing detection using artificial load and Ping in a home network Wi-Fi environment is needed.

TABLE 7. The test accuracy (%) of the binary classification experiment for each label.

Label / Algorithm	NTS ↔		NTSwL ↔	
	TVM	WSK	TVMwL	WSKwL
Decision Tree	83.27	65.27	65.61	67.57
Random Forest	83.85	66.53	66.88	68.62
XGBoost	86.35	61.72	65.82	66.32
LightGBM	86.15	67.15	64.76	67.57

E. THEORETICAL JUSTIFICATION

The suggested ensemble learning strategy is confirmed by ensemble learning theory in [34], which suggests combining multiple weak classifiers can result in a stronger classifier with a greater success rate. Our findings show that ensemble models including Random Forest surpassed single tree models, proving the benefits of this approach. Furthermore, feature selection in [35] highlights selecting features which have the largest impact on the predictions made by the model, which is similar to our focus on essential features such as memory usage and program type. Additionally, removing features with a small impact on accuracy increases the efficiency, which is aligned with anomaly detection principles, that indicate efficient resource allocation for

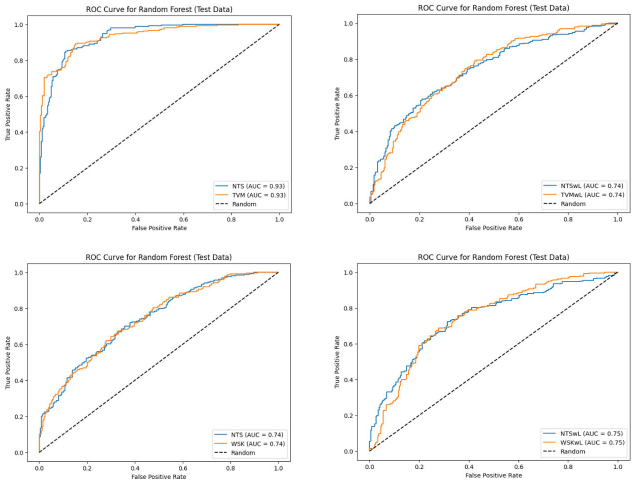


FIGURE 7. The ROC curve and AUC values with and without artificial load.

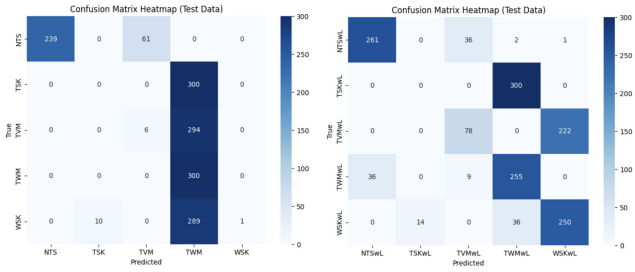


FIGURE 8. The test result for heterogeneous device data using random forest.

detecting anomalies from normal network behaviour. It is important to acknowledge that our findings suggest the need for adapting the model to specific device characteristics for optimal performance.

F. EFFECTIVENESS OF ENSEMBLE LEARNING SNIFFING DETECTION

To confirm the efficiency of the NMS and sniffing detection technology based on ensemble learning, it is crucial to conduct tests with data from diverse devices. An additional Linux-based virtual system was configured for this purpose, utilizing the following specifications for the extra test device (Virtual Machine):

- Hardware: Virtual Machine with 2 CPUs, 8GB RAM, IEEE 802.11ac WiFi, 125GB HDD
- OS: Kali Linux 6.1.0 (Debian 6.1.20)
- Virtual machine program: VirtualBox 7.0
- Link protocol to NMS: SNMP V2C

The collected extra test dataset, comprising 3,000 instances (300 per label), was tested using 7 features after removing unnecessary ones. Figure 8 illustrates the confusion matrix results for both the training dataset from the experiment and the extra test dataset from heterogeneous devices.

From Figure 8, the classification results between heterogeneous devices indicate challenges in completely distinguishing the characteristics of sniffing programs. However, the classification performance for the normal state without sniffing operations is approximately 80%. Additionally, introducing artificial loading increased the standard deviation, enhancing the classification performance of sniffing program features. This suggests that threshold characteristics for classification may vary based on device characteristics. Moreover, the use of artificial loads might slightly improve classification performance by increasing variance. Therefore, achieving more accurate classification requires adapting the learning model based on the specific device. In essence, prior training of IoT devices in the existing network environment is necessary for effective sniffing program detection using the proposed method.

V. DISCUSSION AND LIMITATIONS

This research created an ensemble machine learning model to address the challenges of detecting sniffing behaviour in home IoT Wi-Fi setups. Unlike usual methods that rely on network or host-based approaches, we took a new approach by using a specially designed NMS for small IoT devices. Our main goal is not only to find sniffing programs but also to carefully examine how these programs affect device characteristics. Additionally, we're looking into how quickly and accurately we can classify sniffing program behaviour in this complex setting. The experiment happens in a simulated IoT home network, mimicking real-world situations. In this controlled setup, we have essential components like a NMS, an artificial load generator, and simulations of ten different sniffing conditions. Using machine learning pre-processing, we distilled a comprehensive dataset of 31 features into two subsets—28 datasets and 18 datasets—for detailed analysis. We thoroughly assessed the performance of four machine learning algorithms, including a single tree and three ensemble algorithms. The results show an exceptional accuracy level, exceeding 99%, proving the effectiveness of our machine learning approach. We identified seven crucial features such as CPU, disk, memory, and traffic information that play a vital role in distinguishing the characteristics of sniffing programs. This simplified approach not only makes the detection process easier but also makes it clearer to understand the machine learning models. The interaction between these key characteristics provides insights into how sniffing programs impact devices within the IoT network. For example, the allocation of data to the hard disk is a crucial factor, significantly affecting disk write capacity. Similarly, the memory-intensive nature of Wireshark becomes evident, explaining the varied operational characteristics of these programs.

However, our study acknowledges its limitations, especially in applying findings to diverse IoT settings beyond smart home Wi-Fi networks. Questions arise about how well the proposed technology adapts to environments with complex traffic patterns, like offices. We focused on two specific sniffing programs, raising questions about generalizing observed characteristics to a broader range of

malicious hackers. The study emphasizes the need for a specific monitoring system in managed devices, clear TCP/IP communication, and the importance of prior ensemble learning to effectively determine sniffing behaviour. Ethical considerations add complexity to our research, highlighting the importance of careful tool deployment. A cautious note about the Macof tool, often associated with cyber attacks, underscores potential legal consequences and the need for careful usage. Ethical considerations also extend to the use of artificial load generators, urging a careful approach within established parameters and explicit administrator permission to avoid potential network damage. In essence, our research not only provides valuable insights into detecting sniffing behaviour in IoT environments but also navigates the ethical considerations inherent in experimental cyber security research.

VI. CONCLUSION AND FUTURE WORK

In conclusion, this research aimed to create a robust method for detecting potential sniffing attacks in the complex realm of IoT home Wi-Fi setups, utilizing ensemble machine learning. The approach involved collecting data from an NMS, using ensemble learning to analyze the impact of sniffing, and then identifying and categorizing the types of program operations involved in sniffing attacks. The simulated home Wi-Fi environment was carefully crafted, and after extensive experimentation and data preprocessing, four machine learning technologies demonstrated exceptional classification performance, exceeding 99%. The study also explored the operational characteristics of each sniffing program that aimed to identify the factors influencing the system. By simulating IoT devices' actions through sniffing, the research effectively demonstrated the efficient detection of malicious programs. This breakthrough technology not only promises quick detection of sniffing attacks in home Wi-Fi setups but also has the potential to thwart hacking and the exposure of sensitive information. Its versatility extends to use in IDS or IPS, providing a robust defence against passive attacks orchestrated by malicious actors.

In the future, the next step involves assessing the performance of this technology in more complex wireless network environments, such as corporate settings, and large scale IoT setups with high traffic. Moreover, future work will focus on exploring the practical applications of integrating this technology into the latest security mechanisms.

APPENDIX A
DATASET

Readers and researchers interested in replicating or further examining our findings can refer to this dataset by clicking here.

APPENDIX B
PICTURES

The visual representations presented here complement the main pictures in this paper. Each picture is accompanied by a caption providing context or additional information. Readers can refer to these visual aids to better understand the study's results.

APPENDIX C
PYTHON CODES FOR ENSEMBLE LEARNING

We make available the Python codes used to implement the ensemble learning approach in our analysis for reproducibility and accessibility. To help readers in learning, the code comes with descriptions, pictures, and references. Researchers interested in reproducing or improving our methods can find the detailed implementation here.

REFERENCES

[1] S. R. Pokhrel, H. L. Vu, and A. L. Cricenti, "Adaptive admission control for IoT applications in home WiFi networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 12, pp. 2731–2742, Dec. 2020, doi: 10.1109/TMC.2019.2935719.

[2] Statista. (2023). *Smart Home—United Kingdom | Statista Market Forecast*. Accessed: Feb. 7, 2024. [Online]. Available: https://www.statista.com/outlook/dmo/smart-home/united-kingdom

[3] S. Ramapatruni, S. N. Narayanan, S. Mittal, A. Joshi, and K. Joshi, "Anomaly detection models for smart home security," in *Proc. IEEE 5th Int. Conf. Big Data Secur. Cloud (BigDataSecurity) Int. Conf. High Perform. Smart Comput., (HPSC) IEEE Int. Conf. Intell. Data Secur. (IDS)*, May 2019, pp. 19–24, doi: 10.1109/BigDataSecurity-HPSC-IDS.2019.00015.

[4] L. Mathews. *Hackers Use DDoS Attack to Cut Heat to Apartments*. Forbes. Accessed: Feb. 7, 2024. [Online]. Available: https://www.forbes.com/sites/leemathews/2016/11/07/ddos-attack-leaves-finnish-apartments-without-heat/

[5] S. K. Viswanath, C. Yuen, W. Tushar, W.-T. Li, C.-K. Wen, K. Hu, C. Chen, and X. Liu, "System design of the Internet of Things for residential smart grid," *IEEE Wireless Commun.*, vol. 23, no. 5, pp. 90–98, Oct. 2016, doi: 10.1109/MWC.2016.7721747.

[6] Y. Li, J. Barthelemy, S. Sun, P. Perez, and B. Moran, "A case study of WiFi sniffing performance evaluation," *IEEE Access*, vol. 8, pp. 129224–129235, 2020, doi: 10.1109/access.2020.3008533.

[7] M. Gregorczyk, P. Zórawski, P. Nowakowski, K. Cabaj, and W. Mazurczyk, "Sniffing detection based on network traffic probing and machine learning," *IEEE Access*, vol. 8, pp. 149255–149269, 2020, doi: 10.1109/ACCESS.2020.3016076.

[8] L. Yang, A. Moubayed, and A. Shami, "MTH-IDS: A multitiered hybrid intrusion detection system for Internet of Vehicles," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 616–632, Jan. 2022, doi: 10.1109/JIOT.2021.3084796.

[9] R. Spangler, "Packet sniffing on layer 2 switched local area networks," Packetwatch Res., 2003, pp. 1–5.

[10] P. Goyal and A. Goyal, "Comparative study of two most popular packet sniffing tools-tcpdump and wireshark," in *Proc. 9th Int. Conf. Comput. Intell. Commun. Netw. (CICN)*, Sep. 2017, pp. 77–81, doi: 10.1109/CICN.2017.8319360.

[11] N. Patel, R. Patel, and D. Patel, "Packet sniffing: Network wiretapping," in *Proc. IEEE Int. Advance Comput. Conf.*, Patiala, India, Jul. 2009, pp. 6–7.

[12] A. Mishra, R. S. Chowhan, and A. Mathur, "Sniffer detection and load balancing using aglets in a cluster of heterogeneous distributed system environment," in *Proc. IEEE 7th Power India Int. Conf. (PIICON)*, Nov. 2016, pp. 1–6, doi: 10.1109/POWERI.2016.8077340.

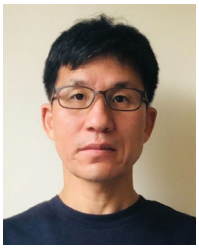
[13] N. Al-Taleb and N. Min-Allah, "A study on Internet of Things operating systems," in *Proc. IEEE Int. Conf. Electr., Comput. Commun. Technol. (ICECCT)*, Feb. 2019, pp. 1–7, doi: 10.1109/ICECCT.2019.8869062.

[14] H. A. Elhadj, H. M. Khelalfa, and H. M. Kortebi, "An experimental sniffer detector: SnifferWall," in *Proc. Atelier Securitédes Communications Internet*, 2002, p. 69.

[15] D. Sanai. *Detection of Promiscuous Nodes Using ARP Packets*. Accessed: Oct. 9, 2023. [Online]. Available: http://www.securityfriday.com

[16] D. Susid, "An evaluation of network-based sniffer detection: Sentinel," Dept. Informatics, Göteborg Univ., Göteborg, Sweden, Tech. Rep., 2004, pp. 11–22.

[17] A. N. Khan, K. Qureshi, and S. Khan, "An intelligent approach of sniffer detection," *Int. Arab J. Inf. Technol.*, vol. 9, no. 1, pp. 9–15, 2012.
[18] A. Ismukhamedova, Y. Satimova, A. Nikiforov, and N. Miloslavskaya, "Practical studying of Wi-Fi network vulnerabilities," in *Proc. 3rd Int. Conf. Digit. Inf. Process., Data Mining, Wireless Commun. (DIPDMWC)*, Jul. 2016, pp. 227–232, doi: 10.1109/DIPDMWC.2016.7529394.
[19] Z. Akram, M. A. Saeed, and M. Daud, "Real time exploitation of security mechanisms of residential WLAN access points," in *Proc. Int. Conf. Comput., Math. Eng. Technol. (iCoMET)*, Mar. 2018, pp. 1–5, doi: 10.1109/ICOMET.2018.8346378.
[20] Z. Trabelsi, H. Rahmani, K. Kaouech, and M. Frikha, "Malicious sniffing systems detection platform," in *Proc. Can. Conf. Electr. Comput. Eng., Toward Caring Humane Technol.*, 2004, pp. 201–207, doi: 10.1109/saint.2004.1266117.
[21] J. Jiang, F. Liu, W. W. Y. Ng, Q. Tang, W. Wang, and Q.-V. Pham, "Dynamic incremental ensemble fuzzy classifier for data streams in green Internet of Things," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 3, pp. 1316–1329, Sep. 2022, doi: 10.1109/TGCN.2022.3151716.
[22] D. Stiawan, Mohd. Y. Idris, R. F. Malik, S. Nurmaini, and R. Budiarto, "Anomaly detection and monitoring in Internet of Things communication," in *Proc. 8th Int. Conf. Inf. Technol. Electr. Eng. (ICITEE)*, Oct. 2016, pp. 1–4, doi: 10.1109/ICITEED.2016.7863271.
[23] W. Yahya, A. Basuki, P. E. Sakti, and F. F. Fernanda, "Lightweight monitoring system for IoT devices," in *Proc. 11th Int. Conf. Telecommun. Syst. Services Appl. (TSSA)*, Oct. 2017, pp. 1–4, doi: 10.1109/TSSA.2017.8272897.
[24] M. A. Mikki, A. A. Samra, and A. A. Bader, "Network monitoring system (NMS)," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 5, pp. 87–104, Jun. 2017.
[25] The-TCPDUMP-Group. *The-TCPDUMP-Group/TCPDUMP: The TCPDUMP Network Dissector*. GitHub. Accessed: Feb. 7, 2024. [Online]. Available: https://github.com/the-tcpdump-group/tcpdump
[26] GitLab. (2024). *Wireshark Foundation/Wireshark · GITLAB*. Accessed: Feb. 7, 2024. [Online]. Available: https://gitlab.com/wireshark/wireshark
[27] WhiteWinterWolf. *WhiteWinterWolf/macof.py:Macof.py, a MAC Address Table Overflow Utility*. GitHub. Accessed: Feb. 7, 2024. [Online]. Available: https://github.com/WhiteWinterWolf/macof.py
[28] Home PRTG 500. *Discover the 3 Paessler PRTG Monitoring Solutions*. Accessed: Feb. 7, 2024. [Online]. Available: https://www.paessler.com/prtg
[29] A. S. Shaffi and M. Al-Obaidy, "Managing network components using SNMP," *Int. J.*, vol. 2, no. 3, pp. 1493–2305, 2013.
[30] I. Syarif, E. Zaluska, A. Prugel-Bennett, and G. Wills, "Application of bagging, boosting and stacking to intrusion detection," in *Machine Learning and Data Mining in Pattern Recognition*. Springer, 2012, pp. 593–602, doi: 10.1007/978-3-642-31537-4.
[31] T. Chen and C. Guestrin, "XGBoost," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.
[32] A. Das, "Anomaly-based network intrusion detection using ensemble machine learning approach," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 2, 2022, doi: 10.14569/ijacsa.2022.0130275.
[33] J. Jiang, F. Liu, Y. Liu, Q. Tang, B. Wang, G. Zhong, and W. Wang, "A dynamic ensemble algorithm for anomaly detection in IoT imbalanced data streams," *Comput. Commun.*, vol. 194, pp. 250–257, Oct. 2022, doi: 10.1016/j.comcom.2022.07.034.
[34] R. P. Sari, F. Febriyanto, and A. C. Adi, "Analysis implementation of the ensemble algorithm in predicting customer churn in Telco data: A comparative study," *Informatica*, vol. 47, no. 7, Jul. 2023, doi: 10.31449/inf.v47i7.4797.
[35] L. Ladha and T. Deepa, "Feature selection methods and algorithms," *Int. J. Comput. Sci. Eng.*, vol. 3, no. 5, pp. 1787–1797, 2011.



HYO JUNG JIN received the master's degree in electronic engineering from the University of Seoul, South Korea. He is currently pursuing the M.Sc. degree in cyber security with Birmingham City University, where his research interests include cyber security applications of machine learning techniques. He is a cyber security and network project manager with over 17 years of work experience in the ICT field.

FARSHAD RAHIMI GHASHGHAEI received the B.Eng. degree in computer engineering from the Institute for Higher Education ACECR Khouzestan, Ahvaz, Iran, and the M.Sc. degree in cyber security from Birmingham City University. He is a motivated researcher with expertise in quantum computing. He has a strong interest in quantum cryptography. His research interests include quantum computing, cryptography, and machine learning, driven by a passion to enhance innovation and secure communication.

NEBRASE ELMRABIT received the B.Sc. degree in computer engineering from Tripoli University, Libya, in 2006, the M.Sc. degree in computer and network security from Middlesex University, in 2007, and the Ph.D. degree in cyber security from Loughborough University, in 2019. With over a decade of teaching and industry experience in U.K. and Libya, he has lectured on digital forensics, security, operating systems, and computer architecture. He also contributed to research in peer-to-peer (P2P) energy trading in the smart grid during his time at the University of Leicester, in 2018. He has been a Lecturer in cyber security and networks with Glasgow Caledonian University, since 2020. His research interests encompass insider threats, anomaly detection, artificial intelligence, penetration testing, digital forensics, and smart grid cyber security.

YUSSUF AHMED received the B.Sc. degree (Hons) in computer systems from the University of Northampton, the M.Sc. degree in information security and computer forensics from the University of East London, and the Ph.D. degree in cyber security from Birmingham City University. He is currently a Senior Lecturer in cyber security and the Director of the B.Sc. Cyber Security Program, Birmingham City University. He has more than 20 years of experience spanning both industry and academia, specializing in information security, cyber assurance, and security governance. His research interests include cyber risk, IoT security, intrusion detection, machine learning applications in cyber security, data security and privacy, and healthcare security. He is a Senior Fellow of the Higher Education Academy (SFHEA).

MEHDI YOUSEFI received the B.Sc. degree in software engineering, the M.Sc. degree in network security, and the Ph.D. degree in cyber security. He specializes in cyber security, information security, network security, computer networking, and machine learning. He is an experienced academician in cyber security and networking with seven years of experience. He has more than seven years of industry experience in networking and cyber security in the financial sector.