JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 41, 217-235 (2025)
DOI: 10.6688/JISE.202501_41(1).0012

Detecting and Preventing Wireless Deauthentication Flood Attacks over 802.11 Networks

CHENG-HAN SHIE[1], PIN-YEN CHIU[1], PO-HSUAN HANG[1], CONG-HAN LAI[1], SHENG-FENG LU[1] AND CHUN-I FAN[1,2,3,+]
[1]*Department of Computer Science and Engineering*
[2]*Information Security Research Center*
[3]*Intelligent Electronic Commerce Research Center*
National Sun Yat-sen University
Kaohsiung, 804 Taiwan
E-mail: hanhan3927@g-mail.nsysu.edu.tw; b083040029@mail.nsysu.edu.tw; b083040019@mail.nsysu.edu.tw; andy0414@g-mail.nsysu.edu.tw; m103140005@mail.nsysu.edu.tw; cifan@mail.cse.nsysu.edu.tw[+]

In recent years, with the widespread adoption of wireless network, devices can access network connections anytime and anywhere. However, vulnerabilities in the widely used IEEE 802.11 series protocols render Wi-Fi communications susceptible to deauthentication flood attacks. Consequently, this study proposes a model for detecting wireless network deauthentication flood attacks and establishes a reporting system based on this model. The system can notify users of their device's status through LINE or Slack messages, enabling them to take appropriate measures against subsequent attacks when an attack is detected, thereby achieving a defensive purpose. Moreover, this research also designs a website for users to view the attack history, which can dynamically update information about the attack process.

The deauthentication flood attack detection and defense model designed in this study boasts advantages such as not requiring hardware modifications to devices, high accuracy, and real-time responsiveness. When an attack occurs, the system sends notifications to alert users to take countermeasures, mitigating the possibility of further attacks by malicious actors. It is expected to be applicable to most devices currently using older 802.11 protocols, providing users with a more secure Wi-Fi communication method.

Keywords: deauthentication flood attacks, wireless networks, IEEE 802.11, random forests, XGBoost, KNN

1. INTRODUCTION

IEEE 802.11, commonly known as Wi-Fi, is one of the most widely applied communication wireless network protocols, characterized by its fast transmission speed, long transmission distance, stable signal, high compatibility with devices, and extensive coverage. However, with various protocol versions of Wi-Fi, cases continue to emerge, and such attacks may lead to possibilities like malicious loss, alteration of messages, or even service disruption. Current research indicates that, although Wi-Fi is susceptible to multiple types of attacks, Denial of Service (DoS) attacks are among the lowest cost and easiest to execute. DoS attacks can prevent devices from connecting and communicating, rendering them

Received November 30, 2023; revised January 1 & February 21, 2024; accepted March 5, 2024.
Communicated by Ming-Hung Wang.
[+] Corresponding author.

unable to provide services. For household appliances, this may simply mean the inability to access current information, but for certain devices, the loss of communication can lead to dangerous situations, as exemplified by the current popularity and low acquisition cost of drones.

Despite the convenience drones bring, their security cannot be overlooked. Wireless transmission implies vulnerability to hacker attacks. Using insecure encryption methods can lead to easy theft of passwords and personal data or result in drones being hacked and controlled by others. Consider a drone, hacked hundreds of meters above the ground, losing control and crashing, potentially causing damage to the drone and casualties on the ground. Alternatively, hackers might gain control of a drone to carry out terror attacks, raising concerns about privacy and national security. Existing research indicates that an attack on a single device can lead to numerous security concerns, and the consequences are unimaginable if such attacks occur in scenarios with multiple drones, like at performance venues or national events.

Currently, there are various methods of cyber-attacks, even tools and kits specifically designed for them, and their implementation instructions are readily accessible to the public. Anyone can execute a significant cyber-attack by following these steps. Given the fixed nature of communication protocols, this research aims to detect Wi-Fi packets in the surrounding environment without altering existing protocols, alerting to specific network attacks.

Among the various methods of wireless network attacks, the most straightforward and brute-force approach is the deauthentication flood attack, a type of DoS attack. It is characterized by its ease of implementation, low cost, and challenging defense. This method is a common attack strategy within the Wi-Fi protocol, forcibly disconnecting devices from Access Points (APs) against users' will, thus effectively disrupting the service. Additionally, attackers may use the deauthentication flood attack to capture handshake packets generated during the Four-Way Handshake process, thereby cracking the Wi-Fi password.

However, the defense mechanisms against deauthentication flood attacks in current literature often involve protocol modifications. This approach has the disadvantage of high costs in retrofitting existing devices, requiring updates to all deployed hardware. Additionally, compatibility issues between new and old protocols must be considered. Merely detecting deauthentication flood attacks without implementing defensive measures gives attackers the opportunity to launch more severe attacks. Therefore, this research employs machine learning techniques to develop a highly accurate attack detection model and a defense system that can be applied to Wi-Fi devices. This system alerts users to change passwords to prevent cracking by attackers and records attack history in a database. A web interface is also developed for users to view related attack information, thereby bridging the security gap during the transition period of updating new and old Wi-Fi protocols.

2. BACKGROUND AND RELATED WORKS

This section primarily providing an in-depth analysis of the strengths and weaknesses of existing defensive approaches, which are taken into consideration in the development of our defense model.

2.2 Cancellation of Flood Attack Defense Technology for Verification

This section categorizes the related technologies mentioned in the literature into three types: modification of communication protocol defense, non-machine learning detection, and machine learning detection. It introduces their methods and discusses the advantages and disadvantages of each technology to implement a more comprehensive defense system.

2.2.1 Modification of communication protocol defense to counteract deauthentication flood attacks

(A) Letter-Enveloped Protocol

Before establishing a connection, the AP and Client each randomly generate two very large prime numbers and multiply them to get a large number, called the Envelope. When establishing a connection, the AP and Client exchange Envelopes. To disconnect, the party wishing to disconnect sends a Deauthentication Frame along with the prime number previously generated, called the Letter. The receiving end calculates whether the Envelope is divisible by the Letter to determine if the Deauthentication Frame is legitimate [1]. The Letter-Enveloped Protocol uses the difficulty of factoring large numbers to ensure the security of the protocol. Although this method effectively defends against deauthentication flood attacks, it requires changing the hardware architecture of AP and Client to accommodate the new communication method, and the generation and calculation of prime numbers also impose additional burden on the equipment.

(B) Using Hash Functions

Arora's research [2] proposes that during connection establishment, both the AP and Client generate a number called the Universally Unique Identifier (UUID), which is then input into the SHA-512 hash function for calculation. The hash value is exchanged and stored during authentication frame exchange. When sending a Deauthentication Frame, it should include the previously generated UUID. The receiving device recalculates the hash value, and only hash values that have been recorded are considered legitimate deauthentication frames. However, this method still requires updating the hardware of AP and Client, incurring additional costs.

(C) Extending the Duration of Effect for management frames

After receiving a Management Frame, the device waits 5 to 10 seconds before executing it. The purpose is to give the device enough time to observe the frame information received after the Deauthentication Frame. If it is a data frame, it is very likely that the device is under a deauthentication flood attack, as in most cases, sending a deauthentication request implies disconnection, and there should be no subsequent data transmission. This method only requires minor changes to the existing device hardware to defend against deauthentication flood attacks [3]. However, the downside is that when the Client roams, the Association Frame, also a Management Frame, is also delayed, preventing the device from switching its connected AP in time, leading to data transmission interruptions [2].

(D) Using Encryptable management frames in the 802.11 Series Standards

As previously mentioned, deauthentication flood attacks originate from the lack of encryption and authentication in management frames. Therefore, in 2009, the IEEE Standards Committee implemented the 802.11w standard to defend against denial-of-service

attacks, including deauthentication flood attacks.

Modifying the communication protocol can significantly defend against deauthentication flood attacks. However, its biggest drawback is the incompatibility with existing devices. Weighing the options, accurately detecting deauthentication flood attacks and sending notifications to alert users is an effective defense strategy during the transition period between the old and new protocols.

2.2.2 Non-machine learning methods for detecting deauthentication flood attacks

(A) Detection using Sequence Numbers

In the Security Wireless System developed by Anjum *et al.* [4], the Sequence Number within a frame is used to detect deauthentication flood attacks. The Sequence Number automatically increases with each frame transmission. By setting up a frame monitor to compare the Sequence Numbers of received frames, the defense system can detect forged frames [5]. This method has many optimization strategies, such as using the difference in Sequence Numbers combined with the Transmission Rate to make judgments, reducing the rate of false positives when frame loss causes interruptions in Sequence Numbers [6]. Using Sequence Numbers for detection does not require modification of existing hardware. The rapid change in Sequence Numbers under a high volume of frame transmission increases the difficulty for attackers to analyze and forge frames. However, if attackers successfully capture frames and learn the current Sequence Number, they can calculate and forge frames with the correct Sequence Number to continue the attack by sending deauthentication frames [5].

(B) Setting a Threshold for Deauthentication Frame Count

The Intrusion Prevention System designed by Agwaral *et al.* [7] detects deauthentication flood attacks by manually setting a threshold. When the accumulated count of received deauthentication frames exceeds this threshold, the AP ignores subsequent deauthentication frames. In their research, the size of the threshold directly affects the Detection Rate and Accuracy Rate. A higher threshold prevents misjudging deauthentication frames sent during normal disconnections, increasing accuracy but reduces the detection rate as the device can only respond after the attack has been ongoing for a while. Conversely, a lower threshold detects attacks faster but is more prone to false positives. Additionally, a static threshold is ineffective in varying environmental conditions.

(C) Detection using Timestamps

In literature, Rajinder Singh *et al.* [8] detect attacks using the reason code and MAC Timestamp of deauthentication frames. Most deauthentication flood attacks have a reason code of 7, differing from normal situations. Detecting solely based on this feature increases the False Positive Rate (FPR). However, the MAC Timestamps of consecutively sent deauthentication frames during an attack are similar and show little variation, so including this feature in detection can reduce the FPR. Although this detection method is simple to set up, the authors did not provide details on its accuracy rate.

These methods can detect deauthentication flood attacks without hardware adjustments. However, they use fixed rules for detection and lack sufficient parameters to judge attacks, leading to poor detection results when attackers adjust their methods or when environmental conditions change. Therefore, this research will adopt a machine learning ap-

proach for detection, aiming to provide a more flexible detection method.

2.2.3 Machine learning methods for detecting deauthentication flood attacks

Agwaral *et al.* [7] designed an intrusion detection system using five different machine learning models, including Naïve Bayes, to detect denial-of-service attacks including deauthentication flood attacks, comparing the detection rate and accuracy of each model. Their training dataset was self-generated, using nine Wi-Fi-enabled devices as normal users and one device as the attacker. The attacker sporadically sent various denial-of-service attacks to a varying number of devices. Using a packet monitor to collect these frames, they extracted six features including the time difference between frame transmissions to train the models. According to their research, the J48 model, based on decision trees, performed better, achieving an accuracy rate of 96%. However, other models also had over 80% accuracy.

Ghanem *et al.* [9] focused on constructing an intrusion detection system using the Support Vector Machine (SVM) model, analyzing the performance of One-class, Two-class, linear, and nonlinear SVMs, noting the advantage of SVM in requiring only a small dataset for training and being suitable for large data assessments. In their paper, their dataset was also self-generated, with the attacker using the Aircrack-ng suite to forge deauthentication frames and extracting five features including signal strength for analysis. Their results showed that the linear Two-class SVM performed better, achieving nearly 100% detection rate.

However, using this model required preprocessing and labeling of the data beforehand. Machine learning detection of deauthentication flood attacks has the advantage of high accuracy. The choice of dataset significantly affects the training outcome and metrics. This research, referencing the above two papers, will use devices to simulate users and attackers, collecting frames to generate a dataset that more closely resembles the surrounding environment. Although machine learning can accurately detect deauthentication flood attacks, the papers did not discuss how to defend against them post-detection. Therefore, this research will implement a reporting system on Raspberry Pi that integrates the detection model and sends notifications to alert users when an attack is detected.

3. METHODOLOGY

This section primarily discusses the proposed system. The system's flow, as shown in Fig. 1, includes generating the Deauthentication Flood Attack Dataset, data preprocessing, and model training. Finally, an alert system will be established to notify users when an attack occurs, indicating that their device is currently undergoing a Deauthentication Flood Attack.

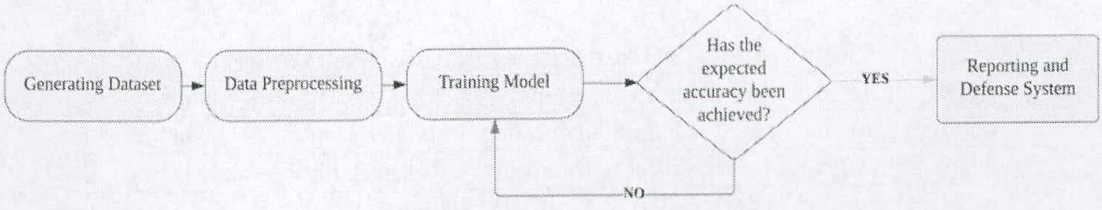


Fig. 1. The flow of the proposed method.

3.1 Generating the Deauthentication Flood Attack Dataset

To simulate the real-world scenario of deauthentication flood attacks, this research implementation will use various Wi-Fi-enabled devices (such as computers, tablets, mobile phones, IoT devices, *etc.*) to mimic the victims in the attack. A Kali Linux virtual machine will be used as the device for simulating the attack, supplemented by another computer as the device for collecting packet data, as shown in Fig. 2. These victims (User) will connect to the same Wi-Fi AP and repeatedly perform a series of actions such as browsing the internet, uploading data, downloading data, disconnecting from and reconnecting to the Wi-Fi AP. The attacker will launch deauthentication flood attacks at irregular intervals and durations. The monitoring device (Monitor) responsible for collecting packets will detect and record the packet data received by the victims, including transmission time, packet type, packet content, packet length, source address, destination address, *etc.* This simulation will be conducted multiple times to collect a sufficient volume of data, enabling the subsequent training step to produce a machine model that meets the required effectiveness. The next phase involves feature extraction; in addition to the basic information collected, this phase will extract finer features from these data, such as the frequency of each packet transmission, received signal strength, noise level, *etc.*, to identify which device transmitted the packets or to determine whether the packet transmission behavior is normal to assess whether it is a deauthentication flood attack.

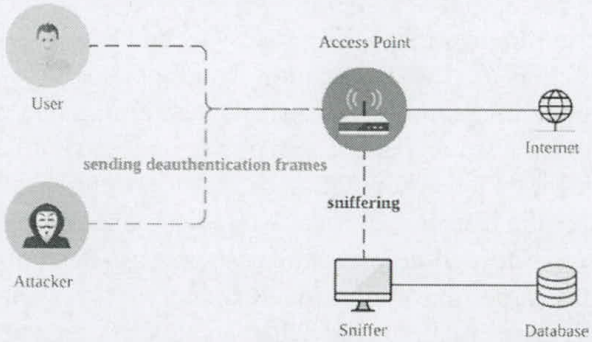


Fig. 2. The scenario for generating a deauthentication flood attack dataset.

3.2 Data Preproccessing

(A) Initial Screening

Before preprocessing, it is necessary to use TShark to convert the PcapNg (PCAP Next Generation Dump File Format) file generated by the sniffer into a csv file, retaining the frame information fields that will be used later. This allows for the execution of subsequent preprocessing steps. The features selected after initial screening are listed in the Table 1.

Table 1. The selected features after initial screening.

wlan_radio.preamble	frame.time_epoch	wlan.fc.subtype	wlan.fc.type
radiotap.dbm_antsignal	wlan_radio.duration	wlan.duration	wlan.ra
wlan_radio.signal_dbm	radiotap.quality	wlan.fc.retry	wlan.ta
wlan.fixed.reason_code			

(B) Padding the Missing Values

In the 802.11 protocol, if the amount of data transmitted in one instance is too large, it will be split into several smaller frames for transmission. However, only the header of the first frame will fully contain the values for all fields. Certain fields related to signal strength, for instance, will not appear in subsequent frames. Since these frames with missing values and the first frame represent the same data, the field information from the first frame is used to impute the missing values.

(C) Labeling

The range of reason code values for deauthentication packets sent by the attacker is from 0xFD (253) to 0xFF (255). Therefore, by simply observing whether the values in this field fall within the aforementioned range, the packets can be marked as attack packets.

(D) Additional Features

In addition to the features originally present in the packets, after careful observation of the characteristics of the deauthentication flood attack, this research has additionally incorporated the following features:

1. Whether the packet is a deauthentication packet: Since the purpose of this model is to detect deauthentication flood attacks, packets that could potentially be part of an attack can be quickly filtered out by examining the type and subtype of the frames.
2. Whether the destination address is a broadcast address: Attackers can input a broadcast address in the destination address field of the frame to carry out a deauthentication flood attack on all devices connected to an AP.
3. Whether the source address is a broadcast address: Under normal circumstances, packets are not sent with a broadcast address as the source. However, the Mdk4 toolkit sets the source address of attack packets to a broadcast address, thus this information is included as a feature.
4. Time difference since receiving the last deauthentication packet: Attackers typically send a large number of attack packets in a short period. This research considers that the shorter the interval between receiving two deauthentication packets, the higher the likelihood of an attack.
5. Number of deauthentication packets received in the last second: The previous item posits that receiving a large number of deauthentication packets within a certain timeframe is highly indicative of an attack.
6. Number of data frames received in the last second: When a user is actively using network resources, the AP sends a large number of data frames to the device. Users usually do not disconnect from the AP voluntarily under these circumstances. If data frames are detected along with deauthentication packets, it is highly likely to be an attack.

(E) Removal of Unnecessary Features

After incorporating the self-generated features, fields with redundant information can be removed, ultimately retaining the features listed in Table 2.

Table 2. The selected features after data preprocessing.

wlan_radio.signal_dbm	wlan_radio.duration	wlan.fc.retry	previous_deauth_packet_diff_time
radiotap.dbm_antsignal	wlan_radio.preamble	wlan.duration	radiotap.quality
is_deauthentication	ta_is_broadcast	ra_is_broadcast	data_packet_count_in_1_secs_before
deauth_packet_count_in_1_secs_before			

3.3 Training the Detecting Model

In the stage of training the detecting model phase, various machine learning models such as KNN (K-Nearest Neighbors), Random Forest, Decision Tree, and XGBoost will employ to detect deauthentication flood attacks. The detection and accuracy rates of each model will be compared, and the one with the best performance data will be selected as the model for the defense system.

As a suitable public dataset for training a model to detect deauthentication flood attacks was not found, this research will use the aforementioned self-generated dataset. The data will undergo preprocessing tailored to the nature of its features. Then, each machine learning model will be trained using this processed data. Subsequently, based on the accuracy rates of the detection models, features that may cause errors in the detection models will be adjusted. After repeated testing and fine-tuning, a final training dataset will be obtained. A detection model with an accuracy rate of over 80% will be trained using this dataset.

3.4 Establish a Reporting and Defense System

Based on the highly accurate detection model trained as mentioned above, this step will implement a detector that can be used on Wi-Fi devices. Not only will it detect deauthentication flood attacks by analyzing packets, but it will also notify users when an attack occurs. Additionally, a webpage will be set up for users to view information related to the attack. The completion of this system is divided into the following two points.

(A) Deployment of the Machine Learning Model to Raspberry Pi

In this step, a Raspberry Pi chip with a wireless network card is used as the packet detector, and the detection model developed in the previous step is deployed on the chip. When an attack is detected, notifications are sent to users through IFTTT (IF This Then That), including alerts on LINE and Slack, advising users to change their passwords. Additionally, this research records the history of attacks in a NoSQL database, facilitating easier access and maintenance of data.

(B) Website Construction

The construction of the website can be divided into server setup and web front-end development. The backend server needs to be able to access database data accurately and stably, while the front-end webpage should be designed to be simple, clear, and easy to operate, allowing users to easily view attack-related information at a glance.

4. EVALUATION

In the section, we will describe how to generate the attack dataset used for training and testing, and through data, demonstrate the accuracy, detection rate of the attack detection model of this system. These metrics will be used to assess the effectiveness of the system's detection capabilities.

4.1 Experimental Setup

The training and testing dataset was created by simulating real scenarios of wireless network deauthentication flood attacks, utilizing ten physical devices of various models. It involved simulating the usage of Wi-Fi devices in a home environment for dataset generation. Out of these, seven were user devices, including three smartphones, two laptops, one tablet, and a IoT device. The remaining devices comprised an AP, a notebook for the attacker, and a desktop for the sniffer. packet-collecting detector. The detailed information about the devices is presented in the Table 3.

Table 3. Experimental device information.

	Device Type	Brand and Model	Operating System	Network Interface Card
AP	Wireless Router	ASUS RT-AC53	N/A	Broadcom BCM5358U
User Device 1	Smart Phone	OPPO Reno5 Z	Android 12	N/A
User Device 2	Smart Phone	Samsung S10e	Android 11	N/A
User Device 3	Smart Phone	iPhone Xs	iOS 14.0.1	N/A
User Device 4	Notebook	Asus	Windows 10	N/A
User Device 5	Notebook	MacBook Pro 2019	MAC OS Big Sur	N/A
User Device 6	Tablet	iPad Air	iOS 14.0.1	N/A
User Device 7	Internet of Things Device	Raspberry Pi 3 Model B 2GB	Raspberry Pi OS	N/A
Attacker	Notebook	Acer Swift 515-5IT	Kali Linux	D-Link DWA-125_GT
Sniffer	Desktop	N/A	Kali Linux	TP-Link Archer T2U Plus

4.2 Generating the Deauthentication Flood Attack Dataset

The attack program is written in Python 3 and executed on a VMware virtual machine running Kali Linux version 5.14.0. To enhance the program's adaptability, enabling its application in various environments and devices, this research utilizes a virtual machine to implement the attack program. Instead of setting specific parameters in the program, it reads files generated by the initial script after execution to complete necessary parameter configurations. These include the name of the targeted AP device, the list of victim devices, and the current channel of the AP.

Subsequently, the program sends Deauthentication Frames randomly multiple times within a set period to execute the attack. To enrich the dataset and facilitate the production of a more comprehensive model in the subsequent training phase, this research opts to use Scapy, Aireplay-ng and Mdk4 to launch the deauthentication flood attack on the victim devices.

Scapy is a Python package that allows the creation and sending of custom packets. Its greatest advantage is the high flexibility in modifying multiple fields within a packet. It has been observed that the Sequence Number field in packets increases with each successful transmission. Therefore, to enhance the stealth of the attack, all attack packets sent via Scapy randomly generate an initial sequence number value, which incrementally increases with each transmission.

Another advantage of Scapy is the ability to select packet recipients arbitrarily. In this program, three different scenarios of deauthentication flood attacks are executed using Scapy. These include attacks from the AP to a specific client, the AP broadcasting to all clients, and a client sending to the AP.

Aireplay-ng is a tool built into Kali Linux that specializes in forging frames, including various WPA/WPA2 attack methods such as the deauthentication flood attack. It requires only a few commands to launch an attack, making it a convenient and accessible tool for wireless network attacks, favored by beginners and advanced hackers alike.

In this program, since the main process needs to repeatedly initiate attacks through a loop without external interruption, we use the fork function to create a subprocess for Aireplay-ng's command input. The main process continues after the attack is completed. Aireplay-ng can execute two types of attacks: forging deauthentication frames from the AP to a single client or broadcasting them. It is important to note that the Sequence Number in Deauthentication Frames sent via Aireplay-ng is a fixed value and does not increase with the number of frames sent.

Mdk4 is another tool that facilitates launching deauthentication flood attacks, distinct from Aireplay-ng in that it is not pre-installed in Kali Linux 5.14.0. Consequently, this research involves the additional installation of the Mdk4 package from the official team's repository on GitHub.

Similar to Aireplay-ng, this research also requires the use of the Fork function to create a separate subprocess for handling command input in Mdk4. This approach allows for efficient execution of Mdk4's functionalities in parallel with the main program, maintaining the overall flow and effectiveness of the attack strategy.

During a one-hour program execution period, the attacker randomly sleeps for 2 to 23 seconds. Upon waking, the attacker randomly selects one of the aforementioned attack methods and targets, which include any device under the AP and broadcasting packets to attack all devices. The frequency of packet transmission is randomly chosen between 0.01 to 1 second, and the attack is paused for 1 to 15 seconds after sending attack packets, repeating this process until the time limit is reached. The rationale behind the choices of attack duration, interval, and sleep time is to ensure that Deauthentication Frames comprise 2% to 3% of the total packets in the dataset, approximating the ratio of attack frames in another public wireless network dataset, AWID3. Algorithm 1 represents the pseudocode of the attacker program's algorithm.

Algorithm 1 Attacker algorithm

$time \leftarrow 0$
$tool\_list \leftarrow \{Aireplay\text{-}ng,\ Scapy,\ Mdk4\}$
$victim\_list \leftarrow \{all\ device,\ specific\ device\}$
while $time \leq 3600$ **do**
  $sleep(random(2, 23))$
  $tool \leftarrow random\_select(tool\_list)$
  $victim \leftarrow random\_select(victim\_list)$
  $attack\_time \leftarrow random(1, 15)$
  $packet\_send\_interval \leftarrow random(0.01, 1)$
  $send\_packet$(tool, victim, attack_time, packet_send_interval)
  $update$(time)
end while

The biggest challenge in generating the dataset was how to label the forged Deauthentication Frames. Initially, we observed that the length of forged frames was shorter than normal packets, attempting to use packet length differences as a basis for labeling. However, it was later discovered that the difference in packet length was mainly due to varying Radiotap lengths added by different network card drivers, not related to the forgery of packets. Radiotap is additional information attached by network card drivers in monitor mode when capturing packets, including signal strength, noise level, current channel, transmission rate, *etc.*, displayed in packet sniffing tools like Wireshark.

Subsequently, we attempted to use Scapy's ability to modify packet fields, altering certain bits in the Radiotap field as a label. However, this method faced issues due to different network cards interpreting Radiotap differently, hindering accurate identification. Additionally, arbitrarily altering frame fields led Wireshark to identify them as forged packets. While devices could still receive these packets, we deemed any frame identified as forged by Wireshark to have significant flaws, thus rejecting this method.

Finally, we decided to use the reason code, a field originally intended as a feature for model recognition, as a marker for identifying forged packets during data collection. The reason code for forged packets was changed to a value that conforms to frame format but is not actually used, specifically between 253 and 255. Scapy used 253, Aireplay-ng 254, and Mdk4 255. Among the three attack tools, Scapy and Aireplay-ng could change the reason code in their function calls. However, Mdk4 did not support this feature, so we modified Mdk4's C language code to make the reason code of the frames it sent match our requirements.

4.3 Training Model

In this research, the main task of the model is to distinguish between normal and attack packets within deauthentication packets. Since only deauthentication packets can potentially be attack packets and they constitute less than 3% of the overall traffic, using accuracy as the sole metric for model evaluation is imprecise due to the extreme disparity

in packet type proportions. Therefore, this research uses metrics less affected by data type proportions, such as Recall, Precision, F1 score, and ROC score, to assess model performance.

This step involves using three different machine learning models, with one of five datasets selected for training. To avoid overfitting, the datasets are split into 60% training, 20% validation, and 20% testing sets. Each model has its characteristics due to different implementation methods, but they all share the common goal of achieving high scores in Recall, Precision, F1 score, and ROC score.

4.4 Model Optimization

(A) Model Selection

As shown in Table 4, the XGBoost model outperformed in all four metrics and given its interpretability in decision-making processes, was selected for further optimization as the primary model for the feedback system. In addition, Table 5 provides the feature weights of the XGBoost model used in this research.

Table 4. The recall, precision, F1 score and ROC score for the KNN model, random forests model and XGBoost model.

Classifier	Recall	Precision	F1 Score	ROC Score
KNN	0.993	0.9689	0.9808	0.9963
Random Forests	**0.9992**	0.9616	0.9801	**0.9994**
XGBoost	0.9954	**0.9985**	**0.9969**	0.9977

Table 5. The feature weights of the XGBoost model.

Feature	radiotap.quality	wlan_radio.signal_dbm	wlan_radio.duration	wlan.duration	wlan_radio.preamble
Weight	0.00161	0.00000	0.00000	0.01843	0.00000
Feature	wlan.fc.retry	is_deauthentication	ta_is_broadcast	ra_is_broadcast	previous_deauth_packet_diff_time
Weight	0.00051	0.94331	0.00433	0.00038	0.00034
Feature	deauth_packet_count_in_1_secs_before	data_packet_count_in_1_secs_before	radiotap.dbm_antsignal		
Weight	0.00217	0.00079	0.02812		

(B) Re-screening Features

After merging data from five different period of training dataset and training the model, a decrease in accuracy was observed. This suggested that the initially chosen features lacked universality. Therefore, a reevaluation of the features was conducted. Based on feature weights and corresponding accuracy, the feature set was iteratively adjusted and refined. The final set of features chosen for model training is listed in Table 6.

Table 6. List of the final selected features and their descriptions.

Feature	Description	Data Type
ta_is_broadcast	We observed that the 'mdk4' package sets the Transmitter Address as the Broadcast Address, a scenario rarely seen in normal packets.	Boolean
ra_is_broadcast	One method of deauthentication attack involves forging an AP to send deauthentication notifications to all Users.	Boolean
wlan.fc.retry	It was noted that attack packets might be retransmitted if they don't receive an ACK (Acknowledgment).	Boolean
previous_deauth_packet_diff_time	Attackers send many deauthentication frames in a short period to disrupt the victim's service, leading to a decrease in this value during an attack.	Floating Point
deauth_packet_count_in_1_secs_before	Similar to the above, this number increases during an attack.	Integer
data_packet_count_in_1_secs_before	Deauthentication flood attacks disrupt the victim's ability to send and receive data smoothly, causing this number to decrease during an attack.	Integer
rssi_diff	A larger difference suggests a greater distance of the transmission source, which may indicate an attack.	Integer

(C) Model Hyperparameter Setting

The setting of hyperparameters is crucial for machine learning models. It not only determines the model's convergence speed and fitting but can also significantly impact the model's accuracy. In this research, hyperparameters for the XGBoost model were set based on model accuracy and dataset characteristics. The settings for the XGBoost model's hyperparameters are listed in Table 7.

Table 7. The hyperparameter settings for the XGBoost model.

Hyperparameters	Description	Value
n_estimators	This represents the number of Decision Trees used in the model. It affects the complexity of the model. If the value is too low, the model may underfit, while a higher value can lead to overfitting. The default value is 100.	140
max_depth	This parameter controls the depth of the Decision Trees. As this research involves a large number of features, this hyperparameter is constrained to prevent the model from overfitting. The default value is 6.	4
class_weight	Due to the significant disparity in the number of attack and normal packets in the dataset, this parameter adjusts the class weights inversely proportional to the frequency of the classes in the data. This setting aims to balance the dataset by giving more weight to less frequent classes.	"Balanced"
Else hyperparameters	This refers to the default value for a given parameter or setting in the XGBoost.	Default

4.5 Validation and Analysis

Due to the significant disparity between the number of validation frames and normal data packets in the dataset, traditional machine learning metrics such as accuracy and recall rate are prone to biases and may not accurately reflect the quality of the model. To optimize the model, this study categorizes the classification results more effectively into TPR (True Positive Rate), TNR (True Negative Rate), FPR (False Positive Rate), and FNR (False Negative Rate). A well-trained model should have a higher TPR and TNR, while minimizing FPR and FNR, to reduce the likelihood of misclassification by the model.

Furthermore, to ensure a more comprehensive validation of the model, this study, in addition to using our own generated dataset to evaluate the model's effectiveness, also employs publicly available datasets for validation. The following sections will describe the evaluation results from each dataset and conduct an analysis and discussion.

(A) Validation with the Deauthentication Flood Attack Dataset

Initially, this research set out to use "individual frames" as the unit of judgment, analyzing accuracy after training the model. However, during training, many unexpected deauthentication frames were found in the dataset. These were neither forged deauthentication frames resulting from attacks nor manually disconnected frames simulated to mimic user network entry and exit. These unexpected deauthentication frames appeared after devices were disconnected by attacks, closely resembling attack frames, causing the model to frequently misclassify them as attacks. Despite maintaining a high accuracy rate, the model also had a high false positive rate. If these unexpected deauthentication frames were considered normal, out of 666,524 normal frames in the test set, 1,262 were deauthentication frames, with 602 of them misclassified as attacks, resulting in an FPR of about 48%.

Analysis using Wireshark, as shown in Fig. 3, revealed that these frames had a reason code of 7, indicating that the AP received a Class 3 frame (including data frames) from an unassociated STA. It is hypothesized that these frames were generated because the devices were transmitting data before being attacked and, due to propagation delay, these data frames reached the receiver after the device was disconnected by the attack. Since the receiver also received a Deauthentication Frame from the same device, it considered the device's behavior illegitimate and sent another Deauthentication Frame to notify it of disconnection.

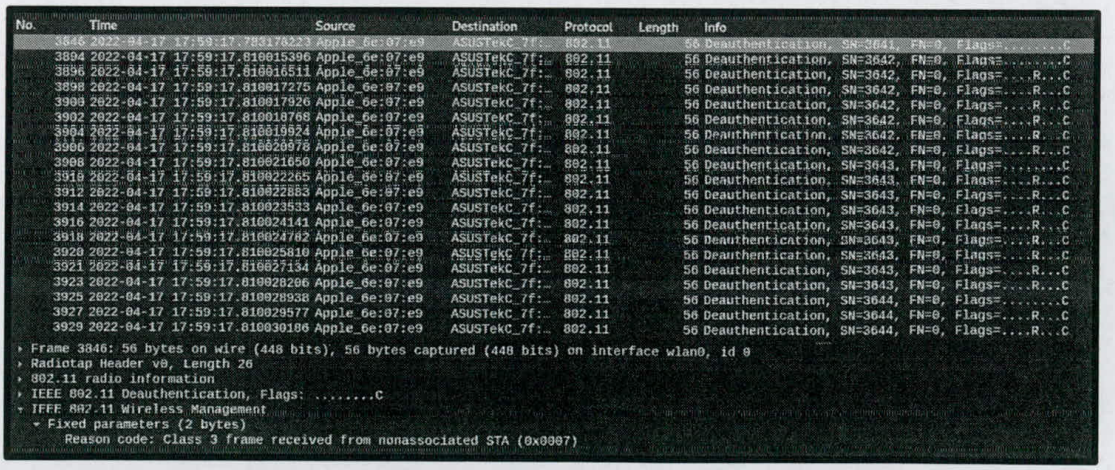


Fig. 3. The AP received a class 3 frame from an unassociated STA.

Given the prevalence of these unexpected deauthentication frames in the dataset, either discarding them or marking them as attack packets significantly affected model training. As these frames almost coincided with the attack period, this research eventually decided to use "time periods" as the unit for determining whether an attack was occurring, distinguishing between attack periods and normal periods. Attack periods are intervals when deauthentication flood attacks were launched during dataset generation; all other times are normal periods. A time period is approximately 10 to 30 seconds. If the model predicts any packet in a period as an attack packet, that period is predicted as an attack period; otherwise, it is considered a normal period. In the test set, out of 193 attack periods, only 6 were not detected, with an FNR of about 3%. Out of 194 normal periods, 12 were misclassified as attacks, with an FPR of about 6%.

(B) Validation with Public Dataset

Considering the training dataset was self-generated, this research decided to use the public AWID2 dataset [10] to verify the model's adaptability in different attack environments. Since AWID2 aims to distinguish various types of network attacks and contains many packets from other kinds of network attacks, unrelated packets were filtered out, leaving only Deauthentication and normal traffic packets to avoid affecting the model's training outcomes.

Since the exact start and end times of attacks in the original dataset were unknown, "individual frames" were used as the judgment unit.

In the test set, it is evident that out of 301,021 normal frames, only 12 were misclassified as attacks. With TN and TP rates close to 99%, it can be inferred that the model is not only adaptable to different environments but also maintains a high level of accuracy.

5. REPORTING AND DEFNSE SYSTEM

This section will also implement a reporting and detection system through the actual deployment of the model and use attack simulation tests to evaluate the system's effective detection and its ability to immediately notify users.

The reporting and defense system utilizes the Scapy package to monitor real packet traffic. After preprocessing, features are fed into the trained XGBoost model for classification. If a packet is identified as part of an attack, a notification is immediately sent to alert the user, and the details of the attack are recorded in the database after the current attack period ends. To simultaneously monitor packet traffic and access network services, this research employs two network cards, with one set in monitor mode.

Furthermore, since sending notifications and uploading data to the database can be time-consuming, there's a risk that the main program might not detect packets during this process, potentially missing an attack. To mitigate this, this research employs an asynchronous approach, running these two services on separate threads. This ensures that the system is always capable of detecting attacks without interruption.

When the model detects an attack, the system uses IFTTT to send notifications to LINE, Slack, and mobile phones. The messages, as shown in Figs. 4 and 5, include the MAC Address of the victim device and the time of the attack. They alert the user that they are under attack and advise them to change their Wi-Fi password promptly. This precaution

is necessary to prevent the handshake packets sent by the device when reconnecting from being captured and cracked by the attacker, potentially leading to a secondary phase of the attack. Sending these notifications enables users to respond immediately to the deauthentication flood attack, thereby maximizing the reduction of harm caused by the attack.

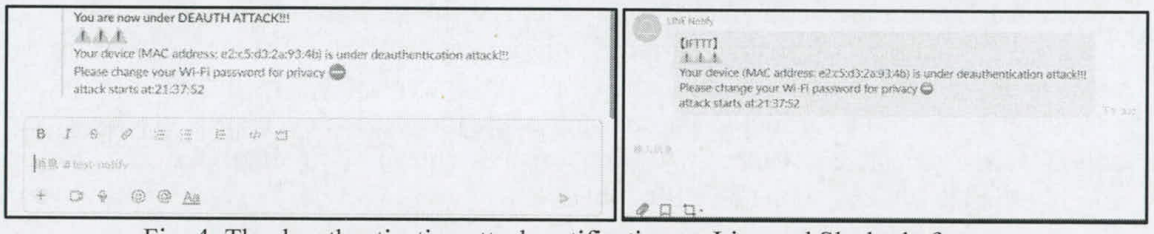


Fig. 4. The deauthentication attack notification on Line and Slack platform.

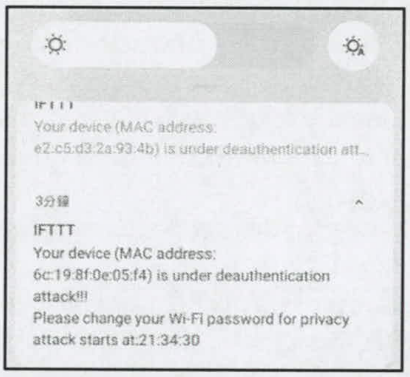


Fig. 5. The deauthentication attack notification on an Android smartphone

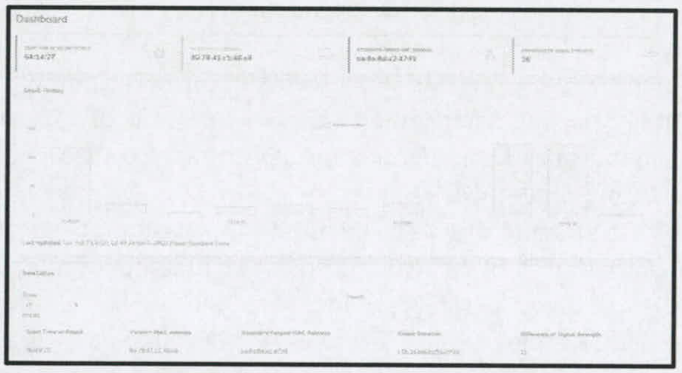


Fig. 6. The website is to provide users to view the history of deauthentication flood attacks

In addition, the system come with a website that provides users with the ability to view the history of attacks, such as the start time of the most recent attack, the MAC address of the victim, the spoofed MAC address of the attacker, the difference in signal strength, and the duration of the attack. The website shown in Fig. 6 uses Python and Flask to build the backend server, and HTML, CSS, Bootstrap, and JavaScript to create the front-end user interface. A key feature of this website is its ability to dynamically update information about the attack history. The front-end webpage will use Ajax to periodically request new attack history information from the backend server in the background. Upon receiving a request, the backend server checks the database for any changes; if there are updates, it will respond to the front-end with the requested information, thereby dynamically updating the user interface.

6. CONCLUSIONS

This research has created a dataset suitable for deauthentication flood attacks in wireless networks, comprised of various attack sources and victim device types, thoroughly considering the diversity of the attack environment. After preprocessing and feature engineering, several explanatory features were selected. This research proposes a deauthenti-

cation flood attack detection framework based on XGBoost. Experiments have confirmed that this framework achieves a 99% TNR and a 99% TPR in tests using the AWID2 public dataset. Finally, the notification and reporting system established based on this framework can use IFTTT for real-time notification of attack information to users, advising them to change passwords in response to potential subsequent attacks. This research also presented a user-friendly web interface for users to view the history of attacks.

Future research directions include expanding the model's capability to detect various types of wireless network attacks, providing comprehensive protection for users. The expansion of our current system's adaptability to different environments includes several key strategies. Firstly, collecting more diverse datasets or simulating attacks on more common and generic Wi-Fi modules can help in understanding a broader range of attack patterns. Secondly, employing unsupervised learning or anomaly detection methods can enhance the robustness of our model, as well as its ability to detect unknown attacks.

Additionally, reducing the system's FPR and FNR is another important direction. FPR and FNR affect the rate of false alarms and missed detections, respectively. These inaccuracies could lead to either unwarranted alerts or overlooked threats. Therefore, future efforts will focus on how to improve in these areas to minimize the occurrence of false reports and missed detections.

Considering that the detector itself may be attacked and unable to connect to the internet, future iterations of this research will test other wireless transmission mediums like 4G and bluetooth to ensure reliable delivery of warning notifications to users. Currently, this research can only detect attacks and prompt users to act to stop damage. In the future, there's potential to integrate software defined network technologies to intercept packets identified as attacks by the model, thereby preventing damage before it occurs.

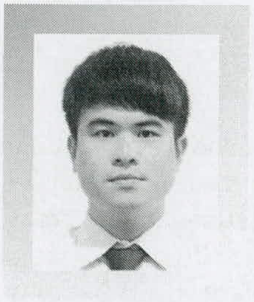
ACKNOWLEDGEMENTS

This work was supported in part by the National Science and Technology Council of Taiwan under Grant 112-2634-F-110-001-MBK, in part by the Information Security Research Center at National Sun Yat-sen University, and in part by the Intelligent Electronic Commerce Research Center from the Featured Areas Research Center Program through the Framework of the Higher Education Sprout Project by the Ministry of Education in Taiwan.

REFERENCES

1. T. D. Nguyen, D. H. Nguyen, B. N. Tran, H. Vu, and N. Mittal, "A lightweight solution for defending against deauthentication/disassociation attacks on 802.11 networks," in *Proceedings of IEEE 17th International Conference on Computer Communications and Networks*, 2008, pp. 1-6.
2. A. Arora, "Preventing wireless deauthentication attacks over 802.11 networks," *arXiv Preprint*, 2018, arXiv:1901.07301.
3. J. Bellardo and S. Savage, "802.11 Denial-of-service attacks: Real vulnerabilities and practical solutions," in *Proceedings of USENIX Security Symposium*, 2003, p. 2.

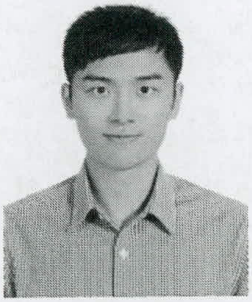
4. F. Anjum, S. Das, P. Gopalakrishnan, L. Kant, and B. Kim, "Security in an insecure WLAN network," in *Proceedings of International Conference on Wireless Networks, Communications and Mobile Computing*, Vol. 1, 2005, pp. 292-297.
5. M. Agarwal, S. Biswas, and S. Nandi, "Detection of de-authentication DoS attacks in Wi-Fi networks: A machine learning approach," in *Proceedings of IEEE International Conference on Systems*, 2015, pp. 246-251.
6. Y. Chen and J. Yang, "Defending against identity-based attacks in wireless networks," *Handbook on Securing Cyber-Physical Critical Infrastructure*, Morgan Kaufmann, Chapter 8, 2012, pp. 191-222.
7. M. Agarwal, S. Biswas, and S. Nandi, "Detection of de-authentication denial of service attack in 802.11 networks," in *Proceedings of Annual IEEE India Conference*, 2013, pp. 1-6.
8. R. Singh and S. Kumar, "A light weight solution for detecting de-authentication attack," *International Journal of Network Security & Its Applications*, Vol. 11, 2019, pp. 15-26.
9. K. Ghanem, F. J. Aparicio-Navarro, K. G. Kyriakopoulos, S. Lambotharan, and J. A. Chambers, "Support vector MAChine for network intrusion and cyber-attack detection," in *Proceedings of IEEE Sensor Signal Processing for Defense Conference*, 2017, pp. 1-5.
10. C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Communications Surveys & Tutorials*, Vol. 18, 2016, pp. 184-208.



Cheng-Han Shie received the MS degree in Computer Science and Engineering at National Sun Yat-sen University, Kaohsiung, Taiwan, in 2022. He is currently working toward a doctorate in Computer Science and Engineering with Information Security at National Sun Yat-sen University, Kaohsiung, Taiwan. His research interests include network security, software-defined network security, AI security, information security, cryptographic protocols, and applied cryptography.

Pin-Yen Chiu received the BS degree in Computer Science and Engineering at National Sun Yat-sen University, Kaohsiung, Taiwan, in 2023. His research interests include wireless networks and information security.

Po-Hsuan Hang received the BS degree in Computer Science and Engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2023. His research interests include wireless networks and information security.



Cong-Han Lai received the BS degree in Computer Science and Engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2023. His research interests include wireless networks and information security.

Sheng-Feng Lu received the MS degree in Information Security with National Sun Yat-sen University. His research interests include mobile security and AI-assisted malware detection applications.

Chun-I Fan received the MS degree in Computer Science and Information Engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1993, and the Ph.D. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, in 1998. From 1999 to 2003, he was an Associate Researcher and a Project Leader with Telecommunication Laboratories, Chunghwa Telecom Company, Ltd., Taoyuan, Taiwan. In 2003, he joined the faculty of the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan. He has been a Full Professor since 2010 and a Distinguished Professor since 2019. His current research interests include applied cryptology, information security, network security, and AI security.