electronics

MDPI

Article

Rule-Based System with Machine Learning Support for Detecting Anomalies in 5G WLANs

Krzysztof Uszko, Maciej Kasprzyk, Marek Natkaniec and Piotr Chołda *

Institute of Telecommunications, AGH University of Krakow, 30-059 Kraków, Poland; kuszko@student.agh.edu.pl (K.U.); mkas@student.agh.edu.pl (M.K.); natkanie@agh.edu.pl (M.N.)
* Correspondence: piotr.cholda@agh.edu.pl

Abstract: The purpose of this paper is to design and implement a complete system for monitoring and detecting attacks and anomalies in 5G wireless local area networks. Regrettably, the development of most open source systems has been stopped, making them unable to detect emerging forms of threats. The system provides a modular framework to create and add new detection rules as new attacks emerge. The system is based on packet analysis modules and rules and incorporates machine learning models to enhance its efficiency. The use of rule-based detection establishes a strong basis for the identification of recognized threats, whereas the additional implementation of machine learning models enables the detection of new and emerging attacks at an early stage. Therefore, the ultimate aim is to create a tool that constantly evolves by integrating novel attack detection techniques. The efficiency of the system is proven experimentally with accuracy levels up to 98.57% and precision as well as recall scores as high as 92%.

Keywords: 5G Wi-Fi security; MAC layer threats; network traffic analysis; threat detection; machine learning

Citation: Uszko, K.; Kasprzyk, M.; Natkaniec, M.; Chołda, P. Rule-Based System with Machine Learning Support for Detecting Anomalies in 5G WLANs. *Electronics* **2023**, *12*, 2355. https://doi.org/10.3390/electronics12112355

Academic Editor: Christos J. Bouras

Received: 27 April 2023
Revised: 19 May 2023
Accepted: 21 May 2023
Published: 23 May 2023

1. Introduction

Wireless networking is a fundamental aspect of modern society, fostering connectivity and communication that spur economic development and social interaction. The demand for mobile devices and the widespread use of the Internet are among the factors driving the growth of wireless networks. Additionally, advancements such as 5G technology have greatly enhanced the throughput and capacity of these networks, allowing for more data and users to be accommodated. A recent study by IoT Analytics Research 2022 [1] showed significant growth in wireless networks. Overall, 5G connectivity is the fastest growing, with a predicted growth of 159%, followed by Low-Power Wide Area Networks (LPWANs) designed for low power consumption. Wireless Local Area Networks (WLANs) are an important component of 5G networks. In general, 5G cellular networks are designed to deliver high-speed wireless connectivity over a wide area, while 5G WLANs are designed to provide local wireless connectivity within a limited area. One of the key features of 5G networks is the ability to offload data traffic from cellular networks to Wi-Fi networks, which can help to reduce network congestion and improve overall performance. WLANs can also be considered part of 5G networks because they fulfill the spectral efficiency requirements that are a key component of 5G technology. They can play a key role in 5G networks by providing a reliable and secure Wi-Fi connection. In fact, 5G networks are expected to incorporate Wi-Fi 6 technology, which is the latest and most advanced version of the Wi-Fi standard. This means that 5G networks will be able to support both cellular and Wi-Fi connections and will be able to seamlessly switch between the two depending on the network conditions and the user's location. Overall, 5G WLANs and cellular networks are complementary technologies that are expected to work together to provide seamless and reliable wireless connectivity in both local and wide area networks.

Wireless networks, like any other technology, are susceptible to numerous threats that can jeopardize security and integrity. Despite the convenience of wireless networks, they remain vulnerable to security threats, largely due to the radio waves that transmit the data, which offer opportunities for unauthorized network access. These threats can come in various forms, such as hacker operations that disrupt network availability or unauthorized access to the system. Another weakness may be the architecture of wireless communication itself. Unlike wired networks that use physical cables to transmit data, wireless networks transmit data through radio waves, making them more vulnerable to physical attacks, such as interference, jamming, or eavesdropping. WLANs are also vulnerable to various types of MAC layer threats, such as flooding, injection, and impersonation attacks. All these attacks can disturb data transmission. Therefore, securing 5G wireless networks is a crucial concern for individuals, businesses, and organizations to protect against potential threats and preserve the reliability and integrity of the network.

Wireless Intrusion Detection Systems (WIDS) are designed to detect and prevent unauthorized access to WLANs. They typically operate by monitoring network traffic for suspicious activity and alerting administrators to potential security threats. With the emergence of 5G and 6G technologies, the role of WIDS in network security has become increasingly important [2,3]. To this end, 5G networks offer significant improvements in data rates, and latency, making them a promising technology for many industries. The high bandwidth and low latency of 5G networks require the deployment of more advanced WIDS systems. This enables WIDS to analyze network traffic in real time and provide faster responses to potential threats. WLANs, on the other hand, are becoming increasingly common in both residential and commercial settings. However, they present significant security challenges due to their wireless nature. By integrating WIDS with 5G WLANs technology, network administrators can significantly improve network security, preventing unauthorized access and protecting sensitive data. WIDS can monitor network traffic and detect anomalies that could indicate a security breach. With the use of advanced WIDS, network administrators can more effectively defend against cyber-attacks. This can help organizations to protect sensitive data and maintain business continuity.

In this paper, we propose an enhanced WIDS that uses rule-based approaches supported by machine learning techniques to detect different networking threats. This system was inspired by the OpenWIPS-ng [4] project and papers like [5], but upon further investigation, it became apparent that they lacked sufficient functionalities we decided to fill these in. IDS systems are engineered to detect and notify administrators of any potential security risks to a wireless network. They can be used to identify unauthorized access points (APs), recognize attempts to breach the network, and monitor traffic for malicious behavior. Despite the numerous scientific publications on this technology, there are only a limited number of tools that concentrate on wireless data link layer security, and most of them have become unsupported. WIDS can be an efficient way to identify potential threats in wireless networks, but it is crucial to regularly review and update them to ensure their effectiveness against newly discovered threats. Therefore, an intelligent WIDS platform was proposed and implemented to manage the performance correctly. The platform is beneficial because while one attack may be easily detected through a recognizable signature, such as an unusual frame, others may require more data, making it ideal for machine learning.

To fully implement the system, the following challenges were addressed:

1. Develop a module that can effectively monitor and analyze network frames. The module is our own achievement presented originally in this study.
2. Enhance the processing capability of the module to support efficient data analysis. The quality of the module is high since we are able to operate with more than 90% efficiency. In particular, we achieved precision as well as recall scores as high as 92% and accuracy levels up to 98%.
3. Research and implement detection for common Wi-Fi attacks.

4. Create an effective machine learning module and overcome the challenge of collecting and preparing relevant data for optimal performance. This aspect is a useful add-on to the whole system.

The main research contributions of this paper are manifold:

- We conducted many research experiments on existing systems to detect new attacks;
- We proposed and implemented the attack detection methods implemented as rules;
- We investigated the effectiveness of our methods in various real-world environments and optimized them to avoid false positives to ensure high efficiency;
- We proposed a new method for feature selection based on the three criteria: universality, independence, and connection to the attack;
- We evaluated the performance of five machine learning models and compared results for precision, recall, and accuracy metrics;
- We proposed a unique approach that evaluated algorithms based on accuracy from training and testing data and on specific datasets;
- We determined the most appropriate algorithm based on two datasets with Beacon Flood and deauthentication attacks;
- We verified that the proposed system is efficient, assuring very good performance metric results.

The structure of this paper is as follows. Section 2.1 provides a brief overview of works related to intrusion detection in wireless networks, including studies on selected attacks. Section 2.2 discusses various attacker techniques used in WLANs, categorizes them, and provides a detailed description of the relevant attacks selected and the tools used to carry out these attacks. Section 3 presents the features of a proposed WIDS and the key elements that have been considered in its development. It briefly describes rule-based detection methods and their implementations, referencing different approaches proposed in the cited literature. Then, it focuses on the use of machine learning for attack classification, including the description of various methods, the selected dataset, and its preparation for supervised machine learning. In Section 4, we present the results of the tests conducted on models trained using various methods. This includes details on the hardware and software environment and the libraries utilized for building intrusion detection tools. Additionally, the section highlights the metrics emphasized and the comparison between testing the models on training data and real-world data. The entire contribution is critically discussed in Section 5. Finally, Section 6 summarizes the work with the conclusions and describes future work.

2. Background

2.1. Related Work

Intrusion Detection Systems (IDS) are widely used in the field of attack monitoring, analysis, and detection and play a crucial role in the security infrastructure of wireless networks. Moreover, the recently popular application of machine learning in anomaly detection can greatly enhance their performance, especially in detecting wide-area network spoofing attacks [6]. However, basing IDS only on machine-learning methods appeared to be insufficient [7]. This is especially the case when they can detect only one type of attack.

IDS collects network traffic data and alerts the system of any possible unauthorized intrusion attempts. These systems constantly monitor network flow and compare it to predefined signatures or threat detection functions. Authors of [8] differentiate between the three main types of detection systems: host-based, network-based, and hybrid. Depending on IDS placement in the system, the system collects network traffic data and generates alerts of any possible unauthorized intrusion attempts. The WPA2 encryption standard is currently the most widely used security for wireless networks. Despite being available since 2006, new vulnerabilities associated with this standard continue to be discovered. The most recent popular discoveries related to WPA2 include KRACK, discovered by Mathy Vanhoef in 2017 [9] and the Kr00k vulnerability based on KRACK, discovered by researchers from ESET in 2019 [10]. Both the attack and exploitation of the vulnerability involve the attacker

gaining a man-in-the-middle position, intercepting and/or altering a frame with a changed cryptographic key. Despite efforts by suppliers to enhance security, many devices remain vulnerable to this type of attack due to users/companies and institutions being hesitant to replace their hardware. These findings reveal significant gaps in the field of Wi-Fi IDS systems, which our system addresses, since other IDS systems do not prioritize the detection of such vulnerabilities.

Industry trends show that organizations of all sizes, from small businesses to large corporations, are gradually transitioning from wired to wireless infrastructure to enhance mobility and improve the quality of service delivery [11]. There are numerous papers that outline various methods for detecting anomalies in wireless networks, some of which utilize the machine learning algorithms [12,13] we base our research on in this work. Interestingly, some research studies also involve the development of techniques that allows for cleaning anomalies in traffic flow data [14]. The paper [15] presents an IDS comprised of several modules: a data capture module, a packet analysis module, a filtering module, and a suspicious packet detection module. The authors propose several methods for detecting suspicious frames, the first being network anomaly detection. This is a technique that identifies network events that deviate from a known, well-analyzed behavior. If a set of behavioral parameters diverges from the established pattern and exceeds certain threshold values, it is classified as an anomaly-based intrusion detection. Another proposed method is abuse detection, which is critical as the system may produce a high number of false alarms based solely on anomalies. This type of detection utilizes rules that can be easily added, removed, or modified by the user, thereby enhancing the system's flexibility and scalability. The last module, the statistical module, proves useful for analysts who need to collect data, such as the number of control frames, management frames, data frames, control and broadcast frames, packet call speed, number of intercepted packets, most common attack types, and their frequency of occurrence [6,15]. However, they do not address machine-learning methods in their work. It is worth noting that statistical analysis of data is an important research tool also in other research works, e.g., evaluation of energy sources [16]. The authors of [17] introduced a semi-supervised neural network solution known as a 'ladder network'. The two-layer model (with 80 neurons in each layer) showed better performance than traditional machine-learning methods. The results demonstrated the network's advantage in detecting flooding attacks while still maintaining high accuracy in classifying normal network traffic. However, the ladder network had lower efficacy in identifying 'injection' and 'impersonation' attacks, with an average decrease of 10% compared to other neural networks. In paper [5], the authors present a solution that combines rule-based and machine-learning methods for detecting security threats in wireless networks. The proposed architecture leverages edge computing to achieve a balance between high security and low latency. The system was designed to detect four types of WLAN attacks: WEP secret cracking, WPA/WPA2 dictionary attack, DoS, and KRACK. Using 630,000 frames containing attacks, the authors found that the accuracy of both the rule-based and machine-learning methods was over 90% for each type of attack. However, despite its promising results, the solution is not publicly available. Moreover, there are still some more common types of attacks that need to be addressed. In [18], a two-stage ML-based Wi-Fi Network Intrusion Detection System (WNIDS) is proposed to enhance detection accuracy. An ML model was developed for each stage to classify the network records into normal or one of the specific attack classes. The proposed system achieves an accuracy of 99.42% for multi-class classification with a reduced set of features. Unfortunately, in the proposed WNIDS, many flooding attack records in the test dataset have been identified as normal records. This may affect the availability of the network in the case of a flooding attack. The authors of [6] successfully designed the Deep-Feature Extraction and Selection (D-FES) algorithm based on Artificial Neural Network (ANN) classification, which achieved an impressive success rate of 99.91% in detecting spoofing attacks. This is a significant improvement compared to previous methods, which had an efficiency of only 22% on the AWID dataset [6]. In work [19], a deep learning algorithm

was proposed to effectively detect anomalies and classify attacks. The authors used a Stacked Auto-Encoder (SAE), which is a neural network composed of multiple layers of auto-encoders, with the output of each layer serving as the input to the next layer. The proposed frameworks consisted of two or three hidden layers, each learning different features from the raw input. The first layer learned first-order features, the second learned pattern-matching features from the first-order features, and the third learned features by matching patterns from second-order features. Although some results are very good, no architecture has achieved an accuracy of at least 60% for different configurations of neural networks.

Currently, commonly used datasets for research in wireless network security are the NSL-KDD dataset, the UNSW-NB15 dataset, the ADFA-LD dataset, the KDDCUP99 dataset, and the AWID dataset. However, NSL-KDD and KDDCUP99, which were created over a decade ago, are no longer used in recent research [20]. The popularity of the AWID dataset, specifically the AWID3 [21], has increased in recent years. The AWID dataset categorizes attacks into three types: impersonation, flooding, injection, and normal traffic. A total of six classification methods were used in the study: CNN, Naïve Bayes, Random Forest, Random Tree, J48, Laddernetowork+NSVM. The results showed that the Random Forest and J48 methods had the best performance for normal traffic, while the CNN method showed its advantages for all other attacks [20]. Despite the excellent results, no information was provided on the specific distribution of the classified data, and the features used were not specified.

The analyzed papers indicate that there are numerous methods for creating a system to detect attacks in IEEE 802.11 networks. Unfortunately, most open-source systems are no longer being developed, and many of the latest types of attacks, such as Key reinstallation attacks, are not detected by them. The use of machine learning in this field is highly promising, as shown in the cited works. However, these scientific papers rarely demonstrate the practical application of such models in real-world environments. Hence, the goal of this work is to merge signature-based classification methods with machine learning to make the best use of the resources of the log-analyzing machine. The developed tool should continually be improved by incorporating new attack detection techniques.

2.2. Attacks on 5G Wi-Fi Infrastructures

To construct WIDS, the most prevalent and hazardous attacks were selected; we focused on detecting them. The effectiveness, impact, detection, and mitigation methods of the listed attacks were thoroughly analyzed and evaluated. Based on the methods of attack execution, their detection was developed and implemented. A general classification of the attacks is presented below, with the description of the most critical ones highlighted. The most important and popular attacks can be summarized in Table 1, which presents the four most common types: Denial-of-Service (DoS), Man-in-The-Middle (MiTM), Reconnaissance and Information Gathering, and Keystream and Key Cracking. Some of the attacks' acronyms are explained in more detail later on the next page.

Table 1. Attacks categorization by their purpose.

DoS	MiTM	Reconnaissance and Information Gathering	Keystream and Key Cracking
Jamming	Rogue AP [1]	War driving	IV attack [2]
deauthentication	Evil Twin	Packet sniffing	WEP attacks
RTS/CTS flood [3]	Karma-Manna	Near field communication	WPA attacks
Beacon Flood	Replay attacks		WPS attacks
Reassociation	KRACK [4]		

[1] Access Point. [2] Initialization Vector attack. [3] Request to Send/Clear to Send flood. [4] Key Reinstallation Attack.

IEEE 802.11 networks are less secure than wired networks and are vulnerable to attacks by adversaries who aim to access data transmitted over the network. For instance, an open Wi-Fi network is a network without a password, which means that the communication between the Access Point (AP) and clients is not encrypted and can be easily monitored by anyone, a behavior commonly known as a Man-in-the-Middle (MiTM) attack. Some APs support MAC Address filtering, which allows only legitimate MAC addresses to join the network. However, adversaries with packet-injection-enabled network cards can craft packets with fake MAC addresses and bypass this security measure. There are also password-protected networks, known as Pre-Shared Key (PSK) networks, which are protected by the Wi-Fi Protected Access (WPA) protocol. If an attacker gains access to a PSK network, they can monitor the packets and attempt to crack the password, although this is a resource-intensive and time-consuming process. Tools such as Aircrack-ng [22] are available to help crack Wi-Fi passwords. Both open Wi-Fi and password-protected networks are vulnerable to certain attacks, which will be described below.

2.2.1. Denial-of-Service Attacks

- **Jamming** is a technique that creates interference in a local wireless network, resulting in a Denial-of-Service (DoS) attack. In this scenario, users and devices cannot transmit data because the access points are jammed. This mode of operation is obviously illegal and involves the creation of interference within radio channels to prevent communication across the local area network (LAN).
- **Deauthentication**: Denial of Service (DoS) is one of the most prevalent types of attacks. Management and control frames are transmitted in plain text, making them vulnerable to spoofing. Upon receiving a deauthentication frame, the client is no longer authenticated and associated. Since the deauthentication frame is a management type of frame, it is sent in plain text, and the AP cannot verify its authenticity. This vulnerability allows an attacker to repeatedly send deauthentication frames to disconnect the victim [23]. In [24], the authors listed three ways in which an attacker can launch a deauthentication attack: spoofed AP to client frame, where the attacker forges a frame that appears to be originating from the AP to the client; spoofed client to AP frame, which is similar to the previous approach but with reversed source and destination MAC addresses; and broadcast spoofed frame, where the attacker sets the source MAC address as the AP MAC address and the destination as broadcast (`FF:FF:FF:FF:FF:FF`). By using the last method, the attacker can disconnect all clients associated with the Access Point.
- **RTS/CTS flood:** flooding attacks are also a type of Denial of Service (DoS) attack that involve overwhelming a network with redundant and unnecessary data frames [25]. Management and control frames are used for communication and management between wireless devices. However, the former are not encrypted [26]. On the other hand, control frames may be encrypted. The method involves utilizing the Request-to-Send (RTS)/Clear-to-Send (CTS) mechanism and entails dispatching numerous CTS frames while setting the duration to its maximum value. The outcome is that the STA is made to await a non-existent transmission, thereby obstructing other clients' access to the medium [27].
- **Beacon Flood:** the technique entails transmitting several beacon frames containing distinct SSIDs, which can result in confusion for end-users who may find it challenging to connect to the intended network [28].
- **Reassociation:** is a type of wireless security exploit where an attacker intercepts and modifies the communication between a client device and an Access Point (AP). In this attack, the attacker tricks the client device into disassociating from its legitimate AP and then reassociating with the attacker's rogue AP instead. Once the client device is connected to the attacker's AP, the attacker can eavesdrop on the wireless traffic, steal sensitive information, or launch other types of attacks. This type of attack can be

carried out by exploiting weaknesses in the authentication and encryption protocols used in Wi-Fi networks.

2.2.2. Man-in-the-Middle Attacks

- A **rogue access point** is an unauthorized access point that is set up within a network, not known to the network administrator or any other user. This type of access point can potentially compromise user privacy and security as it operates outside of the network's established security protocols.
- The **Evil Twin** is a type of attack where the attacker creates a rogue access point that appears as a legitimate one. This access point typically has a stronger signal than other available APs in the network, making it more appealing to users who may not suspect its malicious intent. It is an inexpensive way to carry out an attack, as the only requirement is a basic access point device. This type of attack is often carried out on open Wi-Fi networks, as the attacker does not need to provide any passwords or credentials. Evil Twin is a copy of a legitimate AP used by a hacker for intercepting wireless communication. By combining this with other techniques, such as DNS spoofing, the attacker can gain sensitive information, such as passwords or credit card information, from unsuspecting users. Evil Twin APs are often positioned closer to the victims or have stronger signals than legitimate APs. According to [29], most wireless device operating systems will automatically connect to the AP with the strongest Received Signal Strength Indication (RSSI) when given a choice between multiple APs with the same `SSID`. To perform this attack, the attacker only needs to determine the SSID of the network to be replicated. Then, using tools such as Airbase-ng (part of the Aircrack toolset [22]) and a wireless adapter that supports master mode, the attacker can start an Evil Twin AP. The level of legitimacy of the malicious network can be increased depending on the hacker's knowledge of the victim's AP and their social engineering skills.
- The **Karma** attack is a type of Evil Twin attack aimed at stealing victims' confidential information. It uses the Probe Request/Response subtype of Management frames in IEEE 802.11. Devices have a list of trusted Wi-Fi network names (`SSIDs`), known as the Preferred Network List (`PNL`), and the Karma attack leverages this. When a mobile device sends a probe request, the attacker intercepts it and sends a crafted probe response impersonating the requested `SSID` to trick the device into connecting to the rogue AP. However, current Android and iOS devices no longer send direct probe requests with the `SSID` name to the `PNL`. Instead, they only send broadcast probe requests, making it difficult for the Karma attack to be successful. The security standard has been upgraded, causing most devices to compare the received `SSID` from a probe response with their up-to-date `PNL` list. A new type of attack called **Manna**, has emerged as a new approach to conducting the Karma attack by bypassing these security upgrades. Manna responds to broadcast probe requests from a device with directed probe responses for the network. If a device is seen sending direct probe responses for the same network, Manna sends a probe response to it. Additionally, Manna sends probe responses to all clients across the network in a loud mode, considering all mobile devices to have the same `PNL`.
- A **replay attack** is a type of man-in-the-middle attack where an adversary intercepts and retransmits data. The attacker intercepts an authentication key and can then impersonate the person whose key was intercepted.
- The **Key Reinstallation Attack (KRACK)**: Today, most Wi-Fi networks are secured using encryption mechanisms such as Wi-Fi Protected Access (WPA/WPA2/WPA3). When a client wants to connect to a Wi-Fi network, it authenticates and associates with the Access Point (AP) through a process called the "4-way handshake". As the name implies, this involves exchanging four messages between the supplicant (client) and the authenticator (AP). The authentication is based on the Pairwise Master Key (PMK), and a session key Pairwise Transient Key (PTK) is created. In [9], it was demonstrated

that the 4-way handshake is susceptible to KRACK. To comprehend this attack, it is necessary to examine the 4-way handshake process more closely. The client installs the session key after receiving Message 3. However, due to potential frame loss during transmission, if the AP does not receive a response after sending Message 3, it will retransmit the message with a reset nonce. An attacker can exploit this by collecting and replaying the retransmissions, which can result in nonce resets. By utilizing this technique, the attacker can decrypt or forge packets transmitted during the session.

2.2.3. Reconnaissance and Information Gathering Attacks

- **War driving** refers to the act of searching for publicly accessible Wi-Fi networks. Attackers often use a moving vehicle and a mobile device equipped with war driving software, such as Aircrack, along with GPS. This type of attack aims to locate unsecured Wi-Fi networks and steal personal user information. However, with the advent of more secure WEP, WPA, WPA2, and WPA3 wireless networks, war driving has become less common in recent years.
- **Packet sniffing** in wireless networks is a common threat due to the lack of encryption for a large part of transmitted information. It is easy to capture and view all the data over the wireless network, but it is crucial to remain undetected. The persons performing packet sniffing must ensure that their device is not transmitting any information to the network.
- The **Near Field Communication (NFC) attack:** the NFC attack is a type of wireless security exploit where an attacker uses an NFC-enabled device to gain unauthorized access to a Wi-Fi network. This type of attack works by tricking the victim into touching their NFC-enabled device to an attacker's device that is configured as an NFC reader. Once the connection is established, the attacker can use the victim's device to gain access to the Wi-Fi network. NFC attacks can be prevented by disabling NFC functionality on devices when it is not needed and by implementing secure Wi-Fi authentication protocols such as WPA2-Enterprise.

2.2.4. Keystream and Key Cracking Attacks

- The **Initialization Vector (IV) attack** takes advantage of vulnerabilities in the WEP protocol, a protocol that was once commonly exploited but is no longer supported on wireless devices due to its unreliable nature. The 24-bit IV vector makes it susceptible to attacks and easy to recover. The attack involves capturing packets with the IV in plain text, and in a busy network, the IV will soon repeat. After collecting a large number of packets, a statistical attack can be performed to determine the key, allowing the attacker to read the encrypted information.
- **WEP attacks**: the Wired Equivalent Privacy (WEP) protocol, which is no longer supported on wireless devices due to its proven insecurity, was once a popular target of attacks. The attack on WEP involves capturing packets with the Initialization Vector (IV) in plain text, which soon starts repeating in a busy network. After a sufficient number of packets have been collected, a statistical attack can be conducted to determine the encryption key. Once the key is obtained, the encrypted information can be accessed.
- **WPA attacks**: the Wi-Fi Protected Access (WPA) protocol is similar to WEP, but with a key that is unique or temporary for each message. This means that collecting a large number of packets will not help in cracking the protocol. Only the four handshake packets, sent each time a new device tries to connect to the wireless network, contain information that can be used to determine the key. To crack the password, the handshake must be captured, and a word list containing passwords must be used.
- **WPS attacks**: the Wi-Fi Protected Setup (WPS) protocol was designed to simplify connecting devices to a Wi-Fi network. To connect, a device must enter an 8-digit PIN code. However, this method is easily exploitable as there are only 11,000 possible

codes. The last digit of the PIN code is a checksum calculated from the first seven digits, which limits the number of combinations to 10,000. The first 4 digits are verified by the base station, while the next 3 digits provide another 1000 possible combinations. In total, there are 11,000 possible PIN codes.

3. Design and Functioning of the Proposed WIDS

Here, we present the idea of the Wireless Intrusion Detection System (WIDS) promoted in this paper.

3.1. System Design

When creating tools for network monitoring and anomaly detection, it is important to consider the level of operation. In wired environments, intrusion detection and prevention systems are typically placed at devices where data from various sources converge, such as firewalls, routers, or access points. However, this is not the case in wireless networks, as data transmitted through the air can be intercepted. This characteristic of wireless networks makes them ready for wider use of WIDS. As illustrated in Figure 1, such a tool can monitor traffic and detect anomalies across multiple WLANs without disrupting their infrastructure.

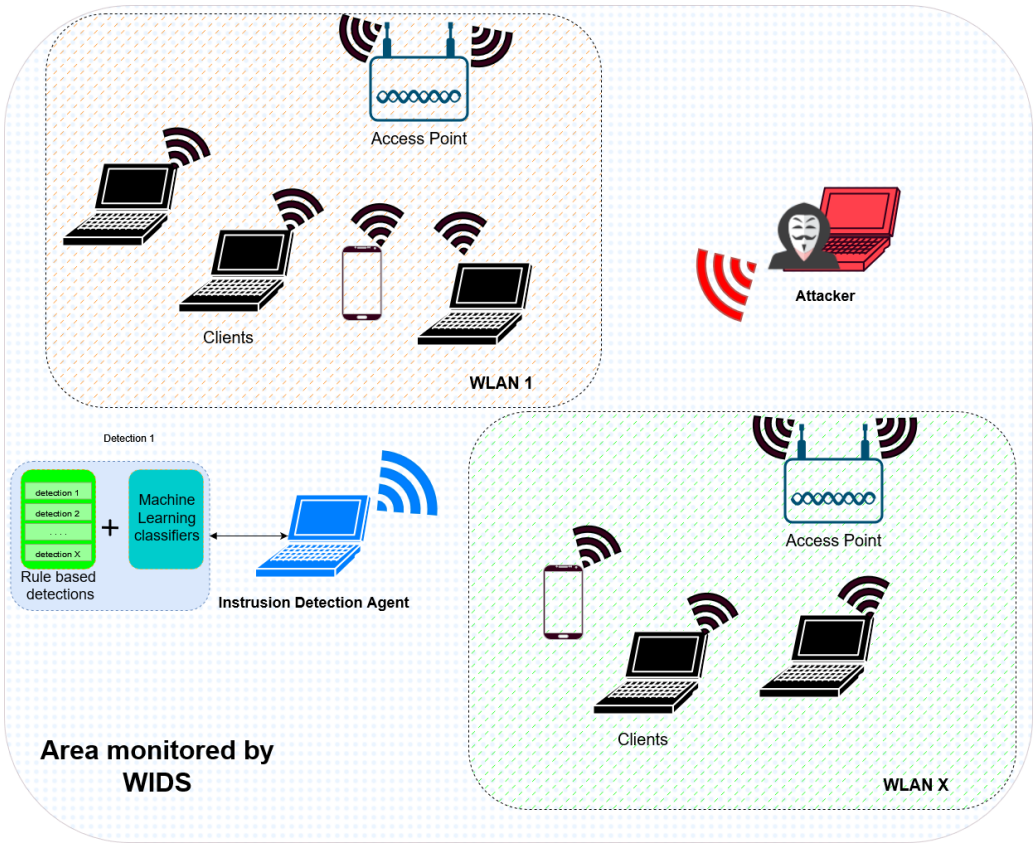


Figure 1. Example wireless environment monitored by WIDS.

As outlined in Section 2.2, networks are susceptible to a variety of attacks today. Some attacks share similarities, such as those in the Denial-of-Service group, while others are more complex, for example, by exploiting specific and non-obvious protocol weaknesses. Without a thorough understanding of each type of attack, it can be challenging to determine the most effective means of detection in real-world networks. This study focuses on creating a hybrid solution that combines rule-based and machine-learning detection. This approach increases the chances of successfully identifying anomalies.

Network protection systems must be easily scalable from the security viewpoint, as new vulnerabilities are continuously being discovered with advancing technology. With this in mind, the implementation of detections began by creating a class that holds all common information, such as the detection result, array of suspicious frames, detection lifecycle, and a wrapper that starts detection on another thread. This solution allows for the development of new rules in a modular manner simply by writing new detection logic based on the provided base class. The developed tool has implemented a multi-threading feature—each detection in the 'live' mode works on a separate thread, which allows for parallel analysis of frames. In addition, the buffer (a common table storing 'frames') stores only the necessary information (Dot11, EAPOL fields—according to the operation of the `scapy` library). The use of multi-threading also enhances performance by allowing different methods to run concurrently without blocking the queue. The classification using machine learning is based on pre-trained models, as described in Section 3.2.2 below.

To minimize false positives and conserve computational resources, a network architecture data enrichment mechanism, referred to as 'Access Point Info', was implemented in WIDS. When creating the WIDS project, there was a problem regarding false positives—for example, unusual network configurations, often used by ISP for providing 'Free Wi-Fi' services. In real environments, some of these cases are well known; however, no work has addressed this problem in data collection or network monitoring. In order to ensure the best transparency of the system, it should be somehow described using 'everyday' behavior—which the mechanism described above is responsible for. This module is responsible for storing information about the network status, such as the correlation between `SSID`, `BSSID`, average antenna signal, and additional flags sent in Beacon frames by access points.

3.2. Detection of Attacks

Rule-based detection and machine learning detection are two complementary approaches to detecting wireless network attacks. Rule-based detection relies on predetermined rules and patterns to identify specific types of attacks, while machine learning uses statistical algorithms and models to learn patterns and detect anomalies.

By combining these two detection methods, wireless network security can be strengthened, as rule-based detection provides a solid foundation for identifying known threats, while machine learning detection can provide early warning for new and emerging attacks. Ultimately, the importance of rule-based detection lies in its role in supporting machine learning detection and improving the overall security of wireless networks.

3.2.1. Rule-Based Detection

Rule-based detection is a fundamental approach to network security that plays a critical role in defending against known threats. Creating a framework for identifying already analyzed attacks allows for quick detection and response to well-known attacks, reducing the risk of damage to the network. Here, we present the attack detection methods implemented as rules. All codes are presented in Appendix A in the form of listings. To assist with understanding the code examples provided, frequently used expressions are described in Table A1 presented in Appendix B.

The presented codes present our original contribution to enable the implementation of the detection based on general attack logic, block schemes, and pseudo codes collected from various sources. The rules have been appropriately adjusted by conducting various tests of attacks within the testing environment, as well as utilizing publicly available data sets. When implementing the detection, the presumption was to react early; hence the thresholds were set to be highly restrictive.

Deauthentication Attack

This type of attack is simple to execute; however, manual detection can be difficult to implement. In several previous works that describe algorithms for detecting deauthentication attacks [30,31], authors use fixed thresholds for the number of deauthentication

frames as an indicator of an ongoing attack. Drawing from existing solutions and through observations, the authors of this work developed a novel detection rule, which is based on the number of deauthentication frames and the number of frames transmitted by the client, as presented in Listing A1. The first step in detecting this type of attack is to create a list of unique users in a specified time interval where the detection system is running. Next, two counters are established: one for the number of frames sent by the client and another for the number of deauthentication frames sent to the client. Then, the two counters are compared, and if the number of deauthentication frames is greater than the number of frames sent by the client plus a predefined threshold to eliminate false positives, an alarm can be triggered. The threshold is necessary to account for small data sets where the communication between the AP and the client may be minimal.

RTS/CTS Flood

This detection has a threshold that is triggered when the number of frames divided by the time between two consecutive frames is greater than the threshold value. The detection has been tested for both normal and malicious traffic and has been found to accurately report anomalies. The first step is to determine if the frame is of type CTS or RTS. If so, the CTS or RTS counter value is incremented, and the time value of the frame is recorded. The threshold value has been set to 5. The procedures are presented in Listing A2. This detection utilizes a threshold that is triggered when the ratio of the number of frames to the time between two consecutive frames exceeds the specified threshold value. The detection has been tested and found to be effective in reporting anomalies, both for normal and malicious traffic.

Beacon Flood

Experiments and frame density measurements conducted in [32] are summarized in Table 2. It shows the full set of Beacon frames that were collected in various locations. Since the values are averaged, it is necessary to set a value that will not generate excessive noise, as this would result in a high number of false positives and reduce the value of the detections. However, the value should not be set too high, as this would result in missed detections. This has been compared with results from our research environment to show what value it generates. An additional test performed by the authors led to the threshold being set at 55%. Thanks to this, we can safely set the threshold at a level that will not generate false positives, and we will also be sure that in larger networks, local will still be effective.

Table 2. Beacons frames density.

	Small Office	**Cafeteria**	**Shopping Complex**	**Own Environment**
Attack	66.7%	65.2%	61.3%	75.5%
No attack	9%	5.6%	7.7%	43%

The procedure for detecting Beacon Flood is presented in Appendix A as Listing A3.

Other Class 3 Frame Attacks

During tests conducted with publicly available tools for wireless network penetration testing, it was found that in most cases where deauthentication attacks are used, a specific reason code can be seen in the spoofed frames. The reason code for deauthentication is 7, which indicates that a class 3 frame was received from a non-associated station (refer to Listing A4). Although this method of detection may result in some false-positive responses, as the same frame may be seen when a client attempts to send data before association, tests showed that this simple detection method could detect many attack techniques in their early stages.

Rogue Access Point

There are several methods for detecting fake access points (APs), with the most popular and commercially used methods based on detecting signal strength. However, the method described in this section, as presented by the authors in [33], is also widely used and has been implemented in the WID tool (according to Listing A5). To detect a fake AP, management frames with a subtype of 8 (Beacon) must be captured. The timestamp field value in these frames is particularly important for this detection, as fake APs can send fake beacons with randomly generated timestamps. This method collects Beacon frames and the timestamp values into a dictionary, where each `BSSID` is a key, and the timestamps are the values. After collecting this information, the timestamps are checked to see if they are in ascending order. If there is any anomaly, the counter value is incremented, and if the value exceeds the threshold, a detection is triggered. The threshold value can be set to 5 in low-traffic environments, while the authors of this paper recommend increasing the threshold value in more heavily trafficked networks, such as corporate networks, to reduce the number of false positives.

Evil Twin

As mentioned before, the objective of this attack is to impersonate a legitimate access point. The 'Access Point Info' module (described further in Section 4.1) is utilized for detection. Two different methods were developed to detect evil access points. The first approach compares the signal strength of access points with the same `SSID`, and the information about the current environment (obtained from the 'Access Point Info' module) is used to determine which one is the legitimate access point. The second approach triggers an alert if the MAC address in the beacon frame does not match the MAC address/es for the same `SSID` in the information provided by the 'Access Point Info' module. They are presented in Listing A6. The algorithm processes a list of frames, and for each beacon frame, it checks if an entry with the same `SSID` as extracted from the frame already exists in the 'Access Point Info' module, which represents the normal environment. If a match is found, the antenna signal is compared for the same `SSID` and `BSSID`. If the signal exceeds the threshold (20 dBm in the provided example), an alarm is raised. If the `BSSID` is different, and it is new to the tool, there is a high probability that the beacon frame originates from an Evil Twin.

Karma–Manna Attack

This detection rule is based on a mechanism referred to as DARMA, as proposed in paper [32]. The detection implementation described in the paper consists of two stages:

- **Detection**: in the first step, the frames need to be divided into encrypted and non-encrypted ones. To achieve this, the capability field, which has privacy properties, is checked. This information is contained within Beacon frames, which are identified by a frame subtype equal to 8. All non-encrypted frames are considered potentially malicious, and their `BSSID`s are recorded in the dictionary 'no_enc_beacons'. Similarly, all probe response frames are stored in the 'probe_responselist', which is created in a similar manner, but instead of beacon frames, this time frames with a subtype equal to 5 are considered. Listing A8 presents the related operation. Next, it is necessary to check if there are any `BSSID`s with multiple `SSID`s. To do this, both lists 1. 'no_enc_beacons' and 2. 'probe_responselist' must be checked. If a `BSSID` appears in both lists, the frame is added to BKL2. These dictionaries contain `BSSID` and `SSID` values from frames that are suspected to be malicious, and on this basis, it is possible to suspect that a Karma–Manna attack is being carried out.
- **Verification**: In the event that the live detection mode is utilized, there may be a confirmation step. This step involves setting up a honeypot and sending forged probe requests. If a fake access point responds to our request, it confirms that an active Karma–Manna attack has been detected. The related code is shown in Listing A7.

KRACK Attack

According to the [9], the key elements of executing this attack are the third message of EAPOL retransmission and an attacker establishing a 'man in the middle' position by creating a fake AP. This section will only focus on detecting the retransmission of message 3, as the detection of fake APs will be discussed in a separate point. The detection is performed as presented in Listing A9. The primary aim of this detection is to identify pairs of APs and clients that undergo a 4-way handshake and then wait for the retransmission of message 3. According to the 802.11 standard [34] regarding EAPOL-KEY frames, the type of message being sent during the handshake can be determined by examining 4 bytes of the raw frame. When message 3 is detected, it should be verified if it has already been sent in the same connection establishment attempt.

3.2.2. Machine Learning-Based Detection

Over the past few decades, advancements in computing power and storage capabilities have driven numerous innovations in the field of artificial intelligence, including chess engines, self-driving cars, and many other tools that enhance people's daily lives. Machine learning (ML) has also found its place in networks. A comprehensive overview of the different methods, categories, challenges, and use cases of ML in anomaly detection in rapidly developing networks can be found in [35]. When examining IEEE 802.11 network data flow, it can be fragmented into sets of frames, with each frame represented as a structure of data with similar features. As a result, every IEEE 802.11 traffic can be represented as a set of data, which can be used to train machine learning models.

Dataset Description

Our objective was to develop a system that is suitable for current operating environments. To achieve this, the authors needed to find a dataset that was created using recent technology and still relevant attacks on wireless devices. The most appropriate dataset for this purpose was the University of the Aegean's AWID3 dataset created in 2021 [21]. The data was collected in a physical lab that consisted of 16 different physical devices and virtual machines running multiple operating systems. The dataset includes 19.3 GB of data, which can be accessed by requesting it. The data consists of multiple `.csv` files, which are extracted frame data from `.pcap` files captured by Wireshark. The files are organized into folders that represent the attacks used to create the dataset. Each folder contains multiple `.csv` files, each of which contains 50 000 records categorized as 'Normal' or '$< Attack\_type >$'.

Feature Selection

The selected dataset contains entries with 254 features, including 253 generic features and a label. However, not all features are useful for model training. Some features, such as source and destination addresses and `SSIDs`, are highly dependent on the environment, while others do not provide useful information for identifying normal and attack traffic. Based on the research reported in [13,36], sixteen features were selected that can distinguish between normal and abnormal traffic.

When selecting features, three criteria were considered. Firstly, the features had to be directly related to the properties of the 802.11 protocol. Secondly, the features should be independent of each other, for example, time-related fields were not considered. Lastly, the features should be directly related to the attack, rather than the network structure (such as MAC addresses or `SSIDs`), and should theoretically indicate properties of the attack.

The selection of features in the dataset was based on the following criteria: universality, independence, and connection to the attack. Universal fields directly relate to the properties of IEEE 802.11 and do not reveal the network structure. Independent fields do not depend on other frame fields, such as time-related fields. The remaining features are flags that, when combined with other parameters, can aid in the classification of various types of attacks. The use of such a reduced dataset helps to improve the training time of machine

learning models. The selected features are described in Table 3, with the first seven being universal fields for IEEE 802.11, and the rest serving as flags for attack classification.

Table 3. Features selected for learning.

No.	Feature	Description
1	`frame.len`	Total frame length
2	`radiotap.length`	Length of the radiotap data
3	`radiotap.dbm_antsignal`	Antenna radio frequency signal power (in dBm)
4	`radiotap.channel.freq`	Channel frequency in MHz
5	`wlan.fc.type`	Frame type
6	`wlan.fc.subtype`	Frame subtype
7	`wlan.duration`	Total duration of the aggregate in microseconds—might be used as there might be difference in distance between attacker or legitimate station and AP
8	`radiotap.present.tsft`	Presence of Time Synchronization Function Timer in radiotap, combined with `frame.len` might be used for detecting impersonation
9	`radiotap.channel.flags.cck`	Complementary Code Keying flag—possible multiple channel flooding or impersonation recognition
10	`radiotap.channel.flags.ofdm`	Orthogonal Frequency-Division Multiplexing flag, indicator of attacks that use multiple channels
11	`wlan.fc.ds`	Distribution System flag value—might indicate impersonation attack
12	`wlan.fc.frag`	Previous frame is being sent in more fragments
13	`wlan.fc.retry`	Retry frame, may indicate flooding attacks
14	`wlan.fc.pwrmgt`	Power management—in some cases might be used for identification of attacker impersonating idle station
15	`wlan.fc.moredata`	More Data buffered—some rogue APs might have this flag enabled
16	`wlan.fc.protected`	MSDU payload encryption—useful when detecting impersonation and flooding type of attacks

Dataset Normalization

The analysis of the AWID3 dataset revealed a significant imbalance between normal and attack traffic data. Such an imbalance would result in unreliable outcomes if left unaddressed. For instance, if the training data ratio was 20:1 (normal to attack traffic), the model would be overly trained to favor normal traffic. To mitigate this, frames were selected from the dataset `.pcaps` in a roughly equal ratio. For example, if there were 19,738 Re(Assoc) attack frames, an equivalent number of Normal traffic frames was selected from the same files. The number of frames for each type of attack varies, so it was necessary to consider that those models might be biased towards attacks with more individual frame appearances in the dataset.

Before selecting records for the final `.csv` file, any records where more than 30% of the selected columns were empty were dropped. To prepare the data for training, missing or '?' entries were changed to the 'NaN' data type. Data normalization was then applied to several fields, including `frame.len`, `radiotap.length`, `radiotap.dbm_antsignal`, `wlan.duration`, and `radiotap.channel.freq`, using min–max normalization.

The `radiotap.present.tsft` and `wlan.fc.ds` fields were transformed into categorical data types, and all columns except for categorical ones were converted to floating-point numbers.

Custom Datasets

Custom datasets for machine learning can be prepared by utilizing testing environments (more specifically described in Section 4.1.1). This involves preparing attack scenarios, tools used to perform attacks, setting up collectors, and collecting and processing

the data. Two attacks—deauthentication and Beacon Flood were executed; in all, 4708 and 6513 frames have been collected, respectively. Frame gathering was performed using the developed WIDS system, then the data underwent processing. They were processed using the tshark tool to obtain a `.csv` file—16 fields were selected as in the case of the AWID3 dataset. Using a personal environment for dataset preparation makes it possible to test models on an additional data source not directly related to the source on which the models were trained. By adopting custom datasets testing methods, it becomes possible to demonstrate that the models have not been overfitted.

Selected Algorithms

The state-of-the-art in machine learning for anomaly detection in wireless networks overviewed in Section 2.1 presents multiple approaches with high accuracy rates, often exceeding 90%. However, it is challenging to determine the best method for our needs. As a result, a decision was made to conduct tests to determine the best algorithm. Unlike previous studies, the authors employed a unique approach that evaluated algorithms based on accuracy from training and testing data on specific datasets. This approach can help identify potential limitations of the algorithms that may not be apparent from standard testing datasets. Custom datasets were used to test and validate the machine learning models by providing a more realistic evaluation of the models' performance. The algorithm's ability to handle different types of data and scenarios was proven by using those tailored datasets.

- The **Decision Tree** is a commonly used algorithm in machine learning and is known for its ease of implementation and understanding. As its name suggests, the algorithm builds a tree-like structure to categorize data. The tree has a root node that represents the most common features, while the leaf nodes represent the class labels. The process of training the Decision Tree, also known as 'growing a tree,' involves deciding which features and conditions to use in order to make accurate class predictions. In this study, Gini impurity was used as a measure of the quality of the split, and the minimum number of samples required at a leaf node was set to 2.
- **Gaussian Naïve Bayes** classifiers are widely used in machine learning. As the name suggests, this family of algorithms is based on Bayes' theorem, which describes the probability of an event occurring based on related conditions. The main advantage of Naïve Bayes methods is their high scalability. The Gaussian Naïve Bayes classifier assumes that the data for each class is evenly distributed.
- **Random Forest** is a widely used classification algorithm in machine learning. It is used for solving both classification and regression problems and is a type of supervised learning. Unlike the Decision Tree algorithm, which models the entire dataset, Random Forest is constructed from multiple Decision Trees, each built on randomly selected parameters. This approach provides a higher chance of avoiding overfitting compared to using just a single Decision Tree.
- **MultiLayer Perceptron (MLP)**: This is a classification algorithm based on an artificial neural network (ANN). It has a layer-like structure, where a set of outputs is generated based on a provided set of inputs, and there can be multiple hidden layers with a chosen number of neurons. The training process of MLP involves forwarding the data from the input layer to the output layer, then calculating the error between the prediction and the actual class, propagating the error back, and adjusting the weights for different features. This process is repeated multiple times, referred to as epochs. In this work, the activation function used is 'ReLU', and the maximum number of iterations is set to 200.

Although other ANN algorithms (e.g., convolutional neural networks, CNN) have been reported to achieve impressive results, they were not employed in this research. The main reason for their exclusion is the utilization of only one dataset (AWID3) during the learning stage. CNNs typically require large amounts of training data to learn relevant features accurately. Although AWID3 is a relatively large dataset, the distribution of

individual attacks is not uniform. Although neural networks can be very efficient in classification tasks, simpler models, including Gaussian Naïve Bayes classifier and Decision Trees, can provide a better performance, which was the main focus of the WIDS project.

The scalability of the WIDS system is also a concern. Selected models are often faster to train and run than neural networks. This can be an advantage when dealing with large datasets, as it can reduce training time and computational resources.

Last but not least: an additional advantage of the models used is their interpretability; with elements such as 'feature importance', it is easier to understand, how classification decisions are made, correct errors, and understand the final classification results.

4. Results

In this section, the testing environment, including the main hardware components, is presented. The proposed WIDS project and software used to execute the attacks are also briefly described. Next, the performance of analyzed machine learning models is shown.

A very important result is related to the accuracy of the rule-based threat detection modules. Namely, we have achieved 100% effectiveness of their operation in the test environment. Each of the attacks was performed 30 times, and each time the attack was correctly detected. During 24 h of network operation, no false alarms related to detecting any attacks were observed either.

4.1. Methods to Test Attacks

4.1.1. Testing Environment

In order to make the measurements as reliable as possible, lots of testing needs to be performed. To avoid being restricted solely to information associated with commonly accessible datasets, testing can take place in a real environment. By creating such a system, it was possible to execute as well as intercept and analyze any attacks, additionally knowing all network parameters, which enables even better detection tuning.

Setting up a testing environment can be time-consuming and difficult to reproduce. To enhance time management efficiency, two similar environments were created to enable asynchronous development and testing of detection modules. The development stack is depicted in Figure 2.

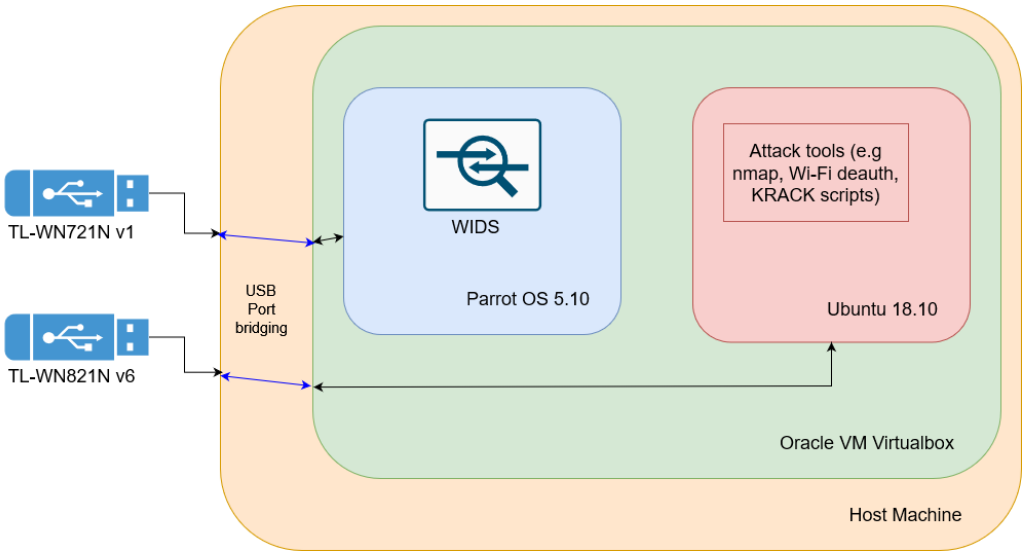


Figure 2. Development setup schema.

The main hardware components include wireless cards that are used for monitoring and conducting attacks. After researching various chipset and driver options, the decision was made to use two models of cards manufactured by TP-Link: the TL-WN821N version 6

and the TL-WN721N version 1. Both adapters connect via USB and operate on the 2.4 GHz band. The TL-WN721N is used for frame collection as it supports Monitor Mode, which allows monitoring of all wireless traffic. It uses the Atheros AR9002U chipset [37] and can be fully utilized with the rtl8192eu [38] driver. The second adapter is used primarily for testing by running attacks from different tools. The TL-WN821N v6 uses the Realtek 8192EU chipset [39] that is capable of packet injection, which was used to perform attacks.

To minimize the number of physical devices, the environment was set up by using virtual machines. The virtualization manager software, Oracle VM VirtualBox Manager, was used on a host machine running Windows 10. Two virtual machines were configured, with the first running Parrot OS version 5.10, used for developing and running the WID system and for data collection and manipulation tasks (e.g., running Wireshark or converting `.pcap` files to `.csv`). A USB bridging mechanism was used to connect the wireless USB adapters to the virtual machines, allowing the VMs to see the Wi-Fi cards as if they were physically connected. As previously mentioned, the wireless cards were connected to the respective virtual machines: the TL-WN821N to the Ubuntu machine and the TL-WN721N to the Parrot OS.

4.1.2. Software Development for WIDS

The source code for the WIDS project can be found in [40]. The intrusion detection system for wireless networks must be fast and scalable to handle large amounts of data analysis and future detections. The system was developed in Python 3, as it offers a variety of libraries and frameworks for computer networking and artificial intelligence. The main focus of the system is frame collection and processing, and the Scapy library [41] is used to optimize performance. The system is designed as a standalone application and features a graphical user interface created with the Kivy library [42]. Kivy is an open-source, cross-platform framework that leverages Cython and OpenGL hardware acceleration for optimal performance.

4.1.3. Software to Execute Attacks

To execute the Karma–Manna attack, the following tools are required: Aircrack-ng [22] and hostapd-mana [43]. It is also important to verify if the network card supports 'master mode', as this enables the use of the card as an access point (AP).

The Wi-Fi testing tool MDK4 [44] is utilized to carry out the attack. This tool supports ten different attack modes, which can be used to simulate and demonstrate various types of attacks.

4.2. Detection of Attacks with Machine Learning Models

In machine learning, various metrics are used to evaluate the performance of models. Common metrics include accuracy, precision score, recall, F1-score, mean squared error (*MSE*), and many others. The most widely used metric is accuracy, which is defined as the ratio of correct predictions made by the model to the total number of predictions made. Accuracy can be calculated using the formula:

$$\frac{TP + TN}{TP + TN + FP + FN'} \tag{1}$$

where *TP* stands for True Positive, *TN* for True Negative, *FP* for False Positive, and *FN* for False Negative.

The initial evaluation compared the overall accuracy of different models. The results are presented in Figure 3. As depicted, the best performing models are Decision Tree with 98.57% accuracy and Random Forest with a depth of 4, achieving 97.46% accuracy. Other models achieved lower accuracy scores, with Gaussian Naïve Bayes scoring the lowest at 61.9%.

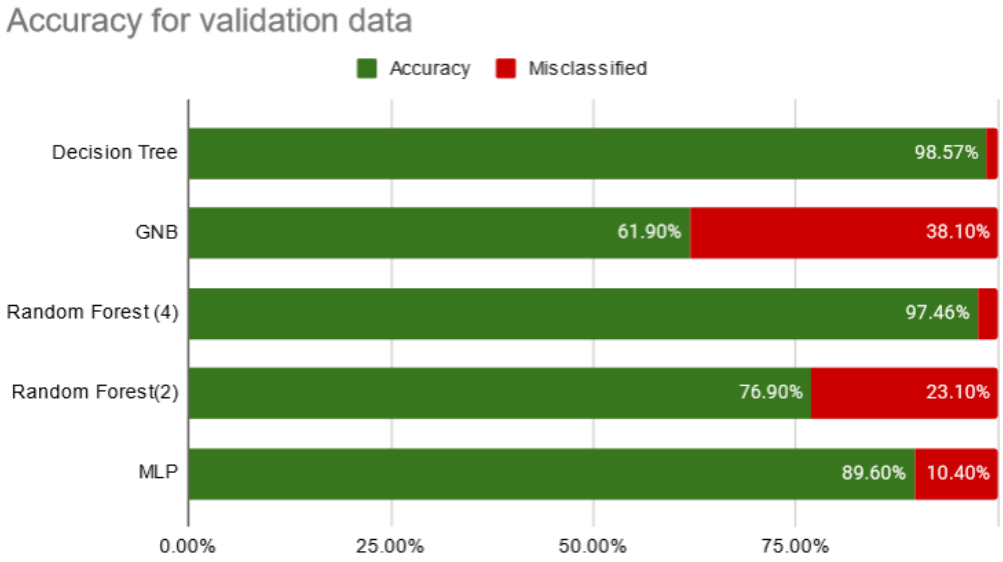


Figure 3. Different models accuracy on validation data.

To gain deeper insights into the results, confusion matrices were created for each model. This representation of data enables us to determine how each model performs for different types of attacks. In the confusion matrix, the rows represent the actual classes, and the columns are the predictions made by the model. Ideally, the fields resulting from the intersection between the true classes and predicted classes should be maximized, with the other fields equal to zero. Figure 4 displays the confusion matrices (showing values of *TP*, *TN*, *FP*, *FN*) for each of the trained models.

Another parameter used to evaluate the performance of machine learning models is the **Precision Score**. Precision refers to the proportion of positive predictions made by the model that are actually true. It can be calculated using the following formula:

$$\frac{TP}{TP + FP}. \tag{2}$$

Precision is often referred to as a measure of quality, but there is another metric that pertains to the quantity—**Recall**. It is calculated as the number of true positive predictions made by the model divided by the total number of actual positive examples in the dataset. The formula for this metric is:

$$\frac{TP}{TP + FN}. \tag{3}$$

By combining precision and recall, a comprehensive understanding of the model's performance on a classification task can be obtained.

The results for each model and frame classification label are presented in Tables 4 and 5, respectively.

Table 4. Precision score by model and attack types.

	Decision Tree	GNB	Random Forest (2)	Random Forest (4)	MLP
Reassoc	99.81%	88.08%	0%	100%	98.48%
Deauth	97.36%	38.36%	0%	92.61%	81.05%
Disas	92.67%	42.72%	0%	79.79%	49.73%
Evil Twin	99.93%	57.97%	99.99%	98.83%	97.10%
Kr00k	98.02%	68.62%	61.71%	97.36%	85.77%
KRACK	100%	95.26%	100%	100%	96.31%
Normal	99.96%	76.66%	95.56%	99.17%	96.31%
Rogue AP	100%	35.09%	0%	0%	86.23%

As indicated in the tables, the values of interest are zero in the same fields in all the tables. The formulas for the previously mentioned metrics reveal that this is due to the absence of true positive classified frames, which are all zeros for the shown attacks and models (as can be seen from the confusion matrices). Another notable observation is the significant differences between the different metrics in the same fields. For instance, the precision score for the Gaussian Naïve Bayes model for the deauthentication attack is 38.36%, while its recall score for the same model and attack is 85.54%. This means that out of all the frames labeled as deauthentication attacks, 85.54% were correctly classified, but only 38.36% of all frames classified as 'Deauth' by the model actually belonged to the attack frames.

Table 5. Recall score by model and attack types.

	Decision Tree	GNB	Random Forest (2)	Random Forest (4)	MLP
Reassoc	99.81%	91.35%	0%	45.28%	91.17%
Deauth	96.99%	85.54%	0%	99.60%	99.89%
Disas	93.61%	58.13%	0%	89.89%	67.41%
Evil Twin	99.92%	90.60%	95.94%	99.55%	96.26%
Kr00k	97.71%	72.50%	100%	92.51%	73.82%
KRACK	100%	34.60%	85.58%	97.59%	96.23%
Normal	99.96%	43.90%	99.87%	99.86%	96.35%
Rogue AP	99.73%	100%	0%	0%	99.47%

Although high overall accuracy scores are impressive, it is important to consider the effectiveness of each model, specifically in detecting different attacks when selecting the best one. To facilitate this analysis process, accuracy has been categorized by attack and model, and the results are displayed in Table 6.

Table 6. Accuracy by model and attack types.

	Decision Tree	GNB	Random Forest (2)	Random Forest (4)	MLP
Reassoc	99.76%	91.36%	0%	90.93%	93.85%
Deauth	97.02%	85.53%	0%	99.87%	85.28%
Disas	93.65%	57.56%	0%	79.15%	85.03%
Evil Twin	99.92%	89.58%	95.92%	99.51%	95.69%
Kr00k	97.72%	72.51%	100%	99.79%	65.46%
KRACK	100%	33.26%	84.18%	99.99%	94.74%
Normal	99.97%	45.91%	97.60%	99.86%	97.81%
Rogue AP	99.74%	100%	0%	0%	94.18%

Weighted averages have been calculated using the following formula:

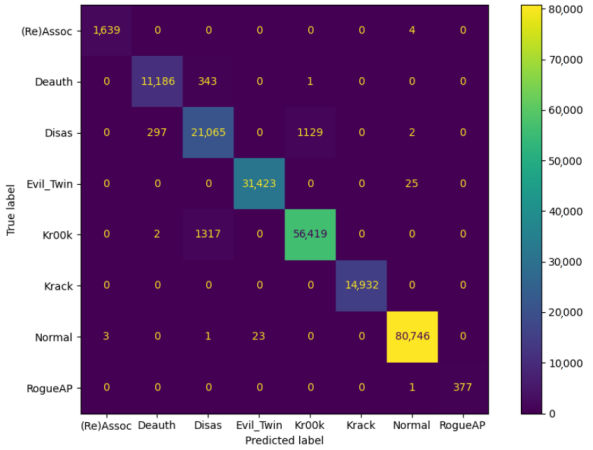
$$\sum_{i=1}^{8} \frac{frames[i]}{total\_frames} \times accuracy[i] \tag{4}$$

where i corresponds to one of the eight labels.

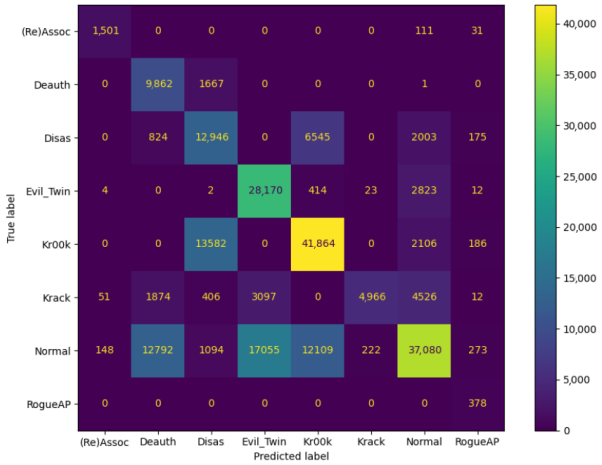
Values of the weighted accuracy for different models are presented in Table 7.

Table 7. Weighted accuracy by model.

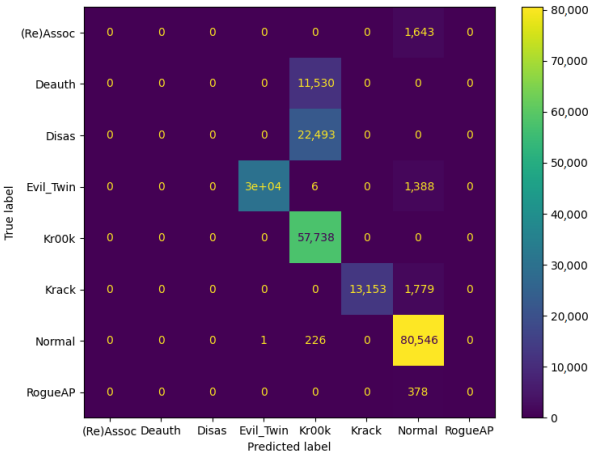
Model	Weighted Accuracy
Decision Tree	99.57%
GNB	61.88%
Random Forest (2)	97.44%
Random Forest (4)	81.07%
MLP	86.87%



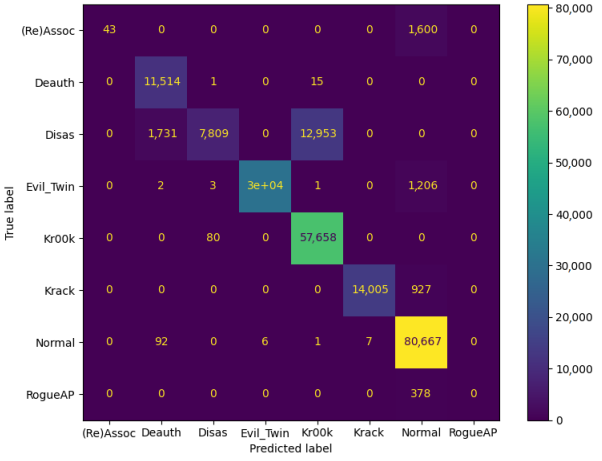
(**a**) Decision Tree



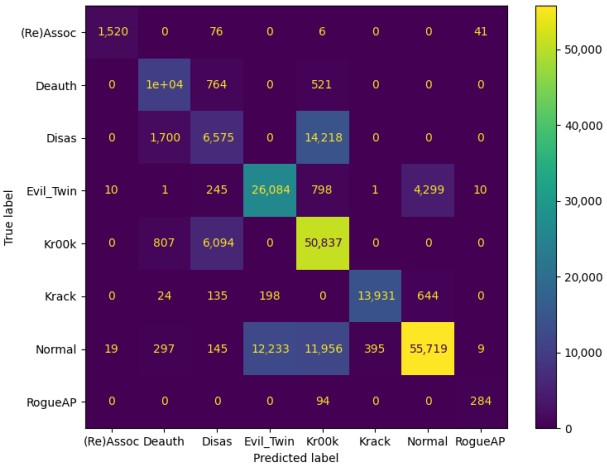
(**b**) Gaussian Naïve Bayes classifier



(**c**) Random Forest (depth = 2)



(**d**) Random Forest (depth = 4)



(**e**) MultiLayer Perceptron

Figure 4. Confusion matrices with results for various attack detectors.

5. Discussion

One significant issue that is commonly encountered in many publications is model overfitting. This occurs when the algorithm has too many parameters relative to the size of the dataset, leading to very high accuracy on the training data but poor performance on

completely new data. This problem was taken into consideration during the preparation of the models for this work.

The authors in [13] reported impressive results for various classification methods, with an accuracy of 99.88% for the Decision Tree, 99.76% for the Random Forest, and 99.73% for the MLP (among other models, though they are not relevant to this paper and all had an average accuracy of over 99%). However, the above work did not include testing on additional datasets, making it challenging to determine if the models are not overfitting. On the other hand, in [20], the authors reported the following results: for the Naïve Bayes model, there was 95.82% accuracy in classifying 'Normal' frames, 0% for the 'Impersonation' category, 72.97% for flooding attacks, and 1.69% accurately classified 'Injection' frames. For the Random Forest model (with no specified depth given), the results were 99.99%, 6.43%, 49.65%, and 85.19%. Although it is not possible to compare these results directly with those achieved in our work (since different datasets were used for training and validation, and there is no information on the other parameters used for training), it is worth noting that, despite an overall average accuracy, the Naïve Bayes method performs better than the Random Forest.

To determine the most appropriate algorithm for real-world applications, we conducted testing in a controlled environment using two datasets with Beacon Flood and deauthentication attacks. Each model was evaluated, and the results were compared and presented in Table 8.

Table 8. Results on custom prepared datasets. Detection ratios are given for two attack scenarios. (**a**) Beacon Flood (Reassoc). (**b**) Deauth.

(**a**)

Method	$\frac{\text{Reassoc + Normal}}{\text{All}}$	*FP*	Notes
Decision Tree	96.79%	3.21%	
GNB	96.42%	3.58%	
Random Forest (2)	0%	100%	0 Reassoc detected/rest classified as Normal/KRACK/Kr00k
Random Forest (4)	0%	100%	0 Reassoc detected/rest classified as Normal/KRACK
MLP	0%	100%	0 Reassoc detected/rest classified as KRACK/Normal

(**b**)

Method	$\frac{\text{Deauth + Normal}}{\text{All}}$	*FP*	Notes
Decision Tree	97.03%	2.97%	
GNB	94.60%	5.40%	
Random Forest (2)	0%	100%	No single Deauth classification/almost all data classified as Normal
Random Forest (4)	0.06%	99.94%	Almost all data classified as KRACK
MLP	2.51%	97.49%	

As indicated in the table, only the Decision Tree and Gaussian Naïve Bayes models produced positive results on the test data prepared by the authors of this work. For the Beacon Flood attack, the main distinguished classification was 'Reassoc', as both attacks have similar characteristics (more details can be found in Section 3.2.1). The results for the Random Forest with various parameters and the MultiLayer Perceptron were worse than random classification. An interesting pattern was observed for these models—most data were classified as the KRACK attack, even though the two attacks have different specifications. For the deauthentication attack, the situation was similar, but a tendency to classify as the KRACK attack was only observed for the Random Forest with a depth of 4, and no significant patterns were noticed for the other poorly performing models.

Mitigation strategies for identified anomalies involve implementing various security measures to prevent or minimize the impact of these attacks. In the case of wireless networks, intercepting and blocking traffic is practically impossible; only the appropriate architecture and configuration of the system could increase its security. For DoS attack types, traffic shaping and rate limiting are suggested. To address MiTM attacks, it is advisable

to verify the vulnerability status of the firmware installed on the devices being utilized. Moreover, employing robust encryption algorithms plays a crucial role in safeguarding the confidentiality of data. In isolated environments, access control measures assume significance by permitting network connectivity only to authorized devices. It is important to note that these are general mitigation strategies, and the specific actions required may vary depending on the network infrastructure, devices, and technologies being used.

6. Conclusions

This paper encompasses the following topics: the idea of detecting threats in 5G Wi-Fi networks, the available intrusion detection solutions, the most prevalent attacks, and methods for detecting these attacks. Additionally, a review of the application of machine learning in identifying anomalies in wireless networks, as well as the practical utilization of these techniques in WIDS, was conducted. Finally, the gathered information was utilized to develop comprehensive software for detecting both attacks and anomalies in wireless networks. The unique feature of our system is that it combines packet analysis modules, rules, and machine learning models to optimize its performance. The rule-based detection provides a solid foundation for identifying known threats, while the incorporation of machine learning detection allows for the early detection of new and emerging attacks.

We also conducted extensive research experiments on existing systems to uncover novel attacks. As a result, we developed and implemented a set of attack detection methods in the form of rules. These methods were thoroughly tested in diverse environments to gauge their effectiveness and tuned. In addition, we put forth a novel approach to feature selection, which adhered to three fundamental criteria. Firstly, we ensured that the selected features were directly associated with the properties of the 802.11 protocol. Secondly, we placed emphasis on selecting independent features, disregarding those with time-related aspects. Lastly, we focused on features that directly pertained to the attack itself rather than being linked to the network structure, such as MAC addresses or SSIDs. This deliberate approach aimed to identify features that theoretically conveyed meaningful properties of the attack under scrutiny. By applying these criteria, we ensured that our feature selection process remained targeted, relevant, and aligned with the goals of our research. We meticulously evaluated the performance of five different machine learning models, comparing typical performance metrics. To further distinguish our work, we introduced a unique methodology for assessing algorithms based on accuracy metrics derived from both training and testing data, specifically tailored to certain datasets.

A testing and development environment was set up with the appropriate wireless network cards for capturing and transmitting packets. The environment was established for two objectives. Next, various attacks were categorized, and based on that, detection modules were designed. We outlined the rule-based detection modules, while the second section detailed the data preprocessing carried out for utilizing machine learning models. Finally, the results were collected, and the performance of the models was compared across various datasets. Issues with classifying frames from the environment created by the authors and the one used in AWID3 were also identified for different methods.

After testing, the two best machine-learning methods were determined to be the Decision Tree and the Gaussian Naïve Bayes classifier. Care was taken to avoid overfitting the models, so the aim was not to achieve the highest possible metric scores. The Decision Tree model achieved 98.57% accuracy on validation data from AWID3 and 96.79% and 97.03% accuracy on custom datasets containing Beacon Flood and deauthentication attacks, respectively. The Gaussian Naïve Bayes method produced the following results: 61.90% accuracy on validation data, 96.42% accuracy for the Beacon Flood attack, and 94.60% accuracy for the deauthentication attack.

The objective of an effective intrusion detection system is to detect all potential attacks while ensuring that attacks are not overlooked. In machine learning, these properties are represented by true positives and false negatives, respectively. In this paper, precision and recall scores of over 92% were achieved for each attack type. The methods were optimized

for real-world environments, so it is challenging to compare these results with those of other studies. Many publications on intrusion detection in 5G 802.11 networks present impressive results on testing data; however, these models are often not tested on data from environments that differ from those used for training.

All of the planned objectives have been successfully accomplished. However, some limitations of the research should also be mentioned. So far, we have been unable to test the functionality of our system in detecting attacks performed at the physical layer. We plan to analyze the behavior of the system under various attacks of this type, e.g., when the carrier sensing mechanism is disabled or when there are misbehaving users. However, there is still room for improvement and further development of the proposed tool. In the future, the authors plan to add more detection methods and update the system in case of any new attacks. As more data are collected, the plan is to improve the machine learning algorithms to make them more versatile. We would also like to apply reinforcement learning algorithms in our system to deal with the changing and unpredictable behavior of the attackers. Another objective is to create a service with multiple endpoints, enabling users to install WIDS agents in their local networks and monitor wireless traffic, even remotely.

Author Contributions: Conceptualization, M.N.; methodology, K.U., M.K., M.N. and P.C.; software, K.U. and M.K.; validation, K.U. and M.K.; formal analysis, K.U., M.K., M.N. and P.C.; investigation, K.U. and M.K.; resources, K.U. and M.K.; data curation, K.U. and M.K.; writing—original draft preparation, K.U., M.K., M.N. and P.C.; writing—review and editing, K.U., M.K., M.N. and P.C.; visualization, K.U., M.K., M.N. and P.C.; supervision, M.N.; funding acquisition, P.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data and software supporting the reported results can be found at https://github.com/krzyusz/WIDS (accessed on 20 May 2023).

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

Abbreviations

The following abbreviations are used in this manuscript:

5G	Fifth Generations systems
ANN	Artificial Neural Network
AP	Access Point
BSSID	Basic Service Set Identifier
CNN	Convolutional Neural Networks
CTS	Clear-to-Send
DNS	Domain Name System
DoS	Denial of Service
D-FES	Deep-Feature Extraction and Selection
GNB	Gaussian Naïve Bayes (classifier)
IDS	Intrusion Detection System
IoT	Internet of Things
IV	Initialization Vector
KRACK	Key Reinstallation Attack
LAN	Local Area Network
LPWAN	Low-Power Wide Area Network
MAC	Medium Access Control
MiTM	Man-in-the-Middle
ML	Machine Learning
MLP	MultiLayer Perceptron
NFC	Near Field Communication
PMK	Pairwise Master Key
PNL	Preferred Network List

PSK	Pre-Shared Key
PTK	Pairwise Transient Key
RSSI	Received Signal Strength Indication
RTS	Request-to-Send
SAE	Stacked Auto-Encoder
SSID	Service Set IDentifier
WEP	Wired Equivalent Privacy
WIDS	Wireless Intrusion and Detection System
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network
WNIDS	Wi-Fi Network Intrusion Detection System
WPA	Wi-Fi Protected Access
WPS	Wi-Fi Protected Setup

Appendix A. Listings of Codes for Rule-Based Detection

Listing A1. Deauthentication attack detection code.

```
for frame in self.frame_array:                                  #
    if frame.type == 0 and frame.subtype == 12:                 # add each unauthenticated client to array
        if frame.addr1 not in self.unique_deauthed_clients: #
            self.unique_deauthed_clients.append(frame.addr1)#
for client_address in self.unique_deauthed_clients:
    for i in range(0,ceil(len(self.frame_array)/chunk_size)):#
        deauth_frames_counter = 0               #
        frames_sent_by_client_counter = 0       #
        start = i*chunk_size                            #divide frames into chunks, compare deauth frames to others in one chunk
        end = 0                                 #
        if i*chunk_size+chunk_size > len(self.frame_array):#
            end = len(self.frame_array)         #
        else:
            end = i*chunk_size+chunk_size
        for frame in self.frame_array[i*100:end]:
            try:
                if frame.type == 0 and frame.subtype == 12 and frame.addr1 == client_address:
                    deauth_frames_counter += 1
                elif frame.addr2 == client_address:
                    frames_sent_by_client_counter += 1
            except Exception as e:
                pass #some control frames do not have the sender
        if deauth_frames_counter > frames_sent_by_client_counter and deauth_frames_counter>2 and frames_sent_by_client_counter>2:
        #values below 2 may be false positives
            for frame in self.frame_array:
                if frame.type == 0 and frame.subtype == 12 and frame.addr1 == client_address:
                    self.detection_result = True
                    self.suspected_frames_array.append(frame)
```

Listing A2. RTS/CTS flood detection code.

```
for frame in self.packet_array:
    if frame.type == 1 and frame.subtype == 12:
        ctsCNT += 1
        stamp = frame.time
        x1 = stamp
        time_diff = x1 - x0
        x0 = x1
        if ctsCNT/time_diff > THRESH:
            print("Detected CTS Flood attack. \n")
            print(ctsCNT/time_diff)
            self.detection_result = True
            self.suspected_packets_array.append(frame)
        else:
            print("Not detected")
            print(ctsCNT/time_diff)
```

Listing A3. Beacon flood detection code.

```
for frame in self.packet_array:
            if frame.type == 0 and frame.subtype == 8:
                ssid = frame.info.decode(''utf-8'')
                bssid = frame.addr2
                stamp = frame[Dot11Beacon].timestamp #str(p.getlayer(Dot11).timestamp)
                if bssid not in ssidDict:
                    ssidDict[bssid] = []
                    ssidCnt[bssid]=0
                elif (int(stamp) < int(ssidDict[bssid][len(ssidDict[bssid])-1])):
                    ssidCnt[bssid]=ssidCnt[bssid]+1
                    if (ssidCnt[bssid] > THRESH):
                        print(''[*] - Detected fakeAP for: ''+ssid)
                        self.detection_result = True
                        self.suspected_packets_array.append(frame)
```

Listing A4. Other class 3 frame attacks detection code.

```
for frame in self.frame_array:
            if frame.type == 0 and frame.subtype == 12: # Deauth frame
                if frame[Dot11Deauth].reason == 7: # class 3 frame received from nonassociated STA
                    self.detection_result = True
                    self.suspected_frames_array.append(frame)
```

Listing A5. Rogue Access Point attack detection code.

```
for frame in self.packet_array:
    if frame.type == 0 and frame.subtype == 8:
        ssid = frame.info.decode(''utf-8'')
        bssid = frame.addr2
        stamp = frame[Dot11Beacon].timestamp
        if bssid not in ssidDict:
            ssidDict[bssid] = []
            ssidDict[bssid].append(stamp)
            ssidCnt[bssid]=0
        else:
            ssidDict[bssid].append(stamp)
        if (int(stamp) < int(ssidDict[bssid][len(ssidDict[bssid])-1])):
            ssidCnt[bssid]=ssidCnt[bssid]+1
            print(''This is working'', ssidCnt)
            if (ssidCnt[bssid] > THRESH):
                print(''[*] - Detected fakeAP is: ''+bssid)
```

Listing A6. Evil Twin detection code.

```
for frame in self.frame_array:
    if frame.type == 0 and frame.subtype == 8:
        if frame.info.decode(''utf-8'') in self.ap_info.ap_names:
            if frame.addr2 in self.ap_info.ap_info_list[frame.info.decode(''utf-8'')]:
            #same mac address, different signal strength +/- 20 dBm
                original_signal_strength = int(self.ap_info.ap_info_list[frame.info.decode(''utf-8'')][frame.addr2][0])
                if not original_signal_strength - 20 <= int(frame[RadioTap].dBm_AntSignal) <= original_signal_strength + 20:
                    print(''Possible evil twin attack for SSID: ''+ frame.info.decode(''utf-8'') + '', MAC: '' + str(frame.addr2))
                    print(''original info: '' + str(self.ap_info.ap_info_list[frame.info.decode(''utf-8'')][frame.addr2]))
                    print(''possible evil twin: '' + '','',join([str(frame[RadioTap].dBm_AntSignal),
                    str(frame[RadioTap].ChannelFrequency), str(frame[RadioTap].ChannelFlags)]))
                    self.detection_result = True
                    self.suspected_frames_array.append(frame)
            else:
            #different mac address
                print(''Possible evil twin attack for SSID: ''+ frame.info.decode(''utf-8''))
                print(''original info: '' + str(self.ap_info.ap_info_list[frame.info.decode(''utf-8'')]))
                print(''possible evil twin: '' + '','',join([str(frame.addr2),str(frame[RadioTap].dBm_AntSignal),
                str(frame[RadioTap].ChannelFrequency), str(frame[RadioTap].ChannelFlags)]))
                self.detection_result = True
                self.suspected_frames_array.append(frame)
```

Listing A7. Karma–Manna Attack verification code.

```
def detection_BKL4(self):
    ssid = ''random1234''
    packet = RadioTap()/Dot11(type=0,subtype=4,addr1=''ff:ff:ff:ff:ff:ff'', addr2=''00:11:22:33:44:55'',
    addr3=''ff:ff:ff:ff:ff:ff'')/Dot11Elt(ID=''SSID'', info='''')
    response = sr(packet)
    if response.type == 0 and response.type == 5:
        if response.ssid == ssid:
            BKL4.append(frame(BSSID))
```

Listing A8. Karma–Manna Attack detection code.

```
for frame in self.packet_array:
    if frame.type == 0 and frame.subtype == 8:
        if frame[Dot11Beacon].cap.privacy == True:
            if frame.info.decode(''utf-8'') not in self.enc_beacons:
                self.enc_beacons[frame.info.decode(''utf-8'')]=[]
                self.enc_beacons[frame.info.decode(''utf-8'')].append(frame.addr2)
            else:
                if frame.addr2 not in self.enc_beacons[frame.info.decode(''utf-8'')]:
                    self.enc_beacons[frame.info.decode(''utf-8'')].append(frame.addr2)
        elif frame[Dot11Beacon].cap.privacy == False:
            if frame.info.decode(''utf-8'') not in self.no_enc_beacons:
                self.no_enc_beacons[frame.info.decode(''utf-8'')]=[]
                self.no_enc_beacons[frame.info.decode(''utf-8'')].append(frame.addr2)
            else:
                if frame.addr2 not in self.no_enc_beacons[frame.info.decode(''utf-8'')]:
                    self.no_enc_beacons[frame.info.decode(''utf-8'')].append(frame.addr2)
```

Listing A9. KRACK attack detection code.

```
for frame in self.frame_array:
    if frame.type == 2 and frame.subtype == 8:# QoS Data, contains EAPOL
        try:
            if str(frame[EAPOL].type) == ''3'':# EAPOL key
                if binascii.hexlify(bytes(frame[Raw]))[2:6] == b'13ca':# message 3, Key ACK set, Key MIC set,
                Encrypted Key Data set)
                    if str(frame.addr2) + ''_'' + str(frame.addr1) not in self.message_3_sent_pairs:
                        self.message_3_sent_pairs.append(str(frame.addr2) + ''_'' + str(frame.addr1))
                    else:
                        self.suspected_frames_array.append(frame)
                        self.detection_result = True
                elif binascii.hexlify(bytes(frame[Raw]))[2:6] == b'010a':# message 2,
                4 way handshake initiated again, clear table data
                    if str(frame.addr2) + ''_'' + str(frame.addr1) in self.message_3_sent_pairs:
                        self.message_3_sent_pairs.remove(str(frame.addr2) + ''_'' + str(frame.addr1))
```

Appendix B. Table with Common Code Expressions

Table A1. Common code expressions.

Expression	Description
`self.frame_array`	Array of frames provided for object detection
`self.suspected_frames_array`	Array with frames which may indicate attack
`self.detection_result`	Flag that indicates if an attack was detected
`self.ap_info`	Object of 'Access Point Info' module described in Section 3.1

References

1. Hasan, M. State of IOT 2022: Number of Connected IOT Devices Growing 18% to 14.4 Billion Globally. 2022. Available online: https://iot-analytics.com/number-connected-iot-devices/ (accessed on 23 April 2023).
2. Sousa, B.; Magaia, N.; Silva, S. An Intelligent Intrusion Detection System for 5G-Enabled Internet of Vehicles. *Electronics* **2023**, *12*, 1757. [CrossRef]
3. Oleiwi, H.W.; Mhawi, D.N.; Al-Raweshidy, H. A Meta-Model to Predict and Detect Malicious Activities in 6G-Structured Wireless Communication Networks. *Electronics* **2023**, *12*, 643. [CrossRef]
4. Aspyct.org. OpenWIPS-ng. Available online: https://openwips-ng.org (accessed on 23 April 2023).

5. Liu, R.; Wang, W.; Wang, J.; Ou, Z.; Qiu, H.; Wang, B.; Liu, Q. iWEP: An Intelligent WLAN Early Warning Platform Using Edge Computing. In Proceedings of the 2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN), Shenzhen, China, 11–13 December 2019; pp. 384–389. [CrossRef]
6. Aminanto, M.E.; Choi, R.; Tanuwidjaja, H.C.; Yoo, P.D.; Kim, K. Deep Abstraction and Weighted Feature Selection for Wi-Fi Impersonation Detection. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 621–636. [CrossRef]
7. Cetin, B.; Lazar, A.; Kim, J.; Sim, A.; Wu, K. Federated Wireless Network Intrusion Detection. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 6004–6006. [CrossRef]
8. Yu, D. Research on Anomaly Intrusion Detection Technology in Wireless Network. In Proceedings of the 2018 International Conference on Virtual Reality and Intelligent Systems (ICVRIS), Hunan, China, 10–11 August 2018; pp. 540–543. [CrossRef]
9. Vanhoef, M.; Piessens, F. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS'17, Dallas, TX, USA, 30 October–3 November 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 1313–1328. [CrossRef]
10. Miloš, C.; Štefan, S.; Róbert, L. KR00K-CVE-2019-15126 Serious Vulnerability Deep Inside Your Wi-Fi Encryption. 2020. Available online: https://www.eset.com/int/kr00k/ (accessed on 23 April 2023).
11. Brandon Butler, R.M. lIDC MarketScape: Worldwide Enterprise WLAN 2019 Vendor Assessment Brandon Butler, Rohit Mehra. Technical Report, IDC. 2019. Available online: https://www.idc.com/promo/idcmarketscape (accessed on 20 May 2023).
12. Kolias, C.; Kambourakis, G.; Stavrou, A.; Gritzalis, S. Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset. *IEEE Commun. Surv. Tutori.* **2016**, *18*, 184–208. [CrossRef]
13. Chatzoglou, E.; Kambourakis, G.; Kolias, C.; Smiliotopoulos, C. Pick Quality Over Quantity: Expert Feature Selection and Data Preprocessing for 802.11 Intrusion Detection Systems. *IEEE Access* **2022**, *10*, 64761–64784. [CrossRef]
14. Chen, X.; Wu, S.; Shi, C.; Huang, Y.; Yang, Y.; Ke, R.; Zhao, J. Sensing Data Supported Traffic Flow Prediction via Denoising Schemes and ANN: A Comparison. *IEEE Sens. J.* **2020**, *20*, 14317–14328. [CrossRef]
15. Jian, W.; Zhi-Feng, F.; Yong, C. Design and Implementation of Lightweight Wireless Lan Intrusion Detection System. In Proceedings of the 2012 Fourth International Conference on Multimedia Information Networking and Security, Nanjing, China, 2–4 November 2012; pp. 75–78. [CrossRef]
16. Lin, S.S.; Shen, S.L.; Zhou, A. Energy Sources Evaluation Based on Multi-Criteria Decision Support Approach in China. *Sustain. Horizons* **2022**, *2*, 100017. [CrossRef]
17. Ran, J.; Ji, Y.; Tang, B. A Semi-Supervised Learning Approach to IEEE 802.11 Network Anomaly Detection. In Proceedings of the 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), Kuala Lumpur, Malaysia, 28 April–1 May 2019; pp. 1–5. [CrossRef]
18. 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]
19. 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. [CrossRef]
20. Duan, Q.; Wei, X.; Fan, J.; Yu, L.; Hu, Y. CNN-based Intrusion Classification for IEEE 802.11 Wireless Networks. In Proceedings of the 2020 IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 11–14 December 2020; pp. 830–833. [CrossRef]
21. Chatzoglou, E.; Kambourakis, G.; Kolias, C. Empirical Evaluation of Attacks against IEEE 802.11 Enterprise Networks: The AWID3 Dataset. *IEEE Access* **2021**, *9*, 34188–34205. [CrossRef]
22. Aspyct.org. Aircrack-ng. Available online: https://www.aircrack-ng.org/ (accessed on 23 April 2023).
23. Schepers, D.; Ranganathan, A.; Vanhoef, M. *On the Robustness of Wi-Fi Deauthentication Countermeasures*; WiSec'22; Association for Computing Machinery: New York, NY, USA, 2022; pp. 245–256. [CrossRef]
24. Agarwal, M.; Biswas, S.; Nandi, S. Detection of De-Authentication DoS Attacks in Wi-Fi Networks: A Machine Learning Approach. In Proceedings of the 2015 IEEE International Conference on Systems, Man, and Cybernetics, Hong Kong, China, 9–12 October 2015; pp. 246–251. [CrossRef]
25. Gast, M. *802.11 Wireless Networks the Definitive Guide*, 2nd ed.; O'Reilly Media: Sebastopol, CA, USA, 2005.
26. Ram, J.R.; Sak, B. *Mastering Kali Linux Wireless Pentesting*; Packt Publishing: Birmingham, UK, 2016.
27. Sawwashere, S.S.; Nimbhorkar, S.U. Survey of RTS-CTS Attacks in Wireless Network. In Proceedings of the 2014 Fourth International Conference on Communication Systems and Network Technologies, Bhopal, India, 7–9 April 2014; pp. 752–755. [CrossRef]
28. Martínez, A.; Zurutuza, U.; Uribeetxeberria, R.; Fernández, M.; Lizarraga, J.; Serna, A.; Vélez, I. Beacon Frame Spoofing Attack Detection in IEEE 802.11 Networks. In Proceedings of the 2008 Third International Conference on Availability, Reliability and Security, Barcelona, Spain, 4–7 March 2008; pp. 520–525. [CrossRef]
29. Yang, C.; Song, Y.; Gu, G. Active User-Side Evil Twin Access Point Detection Using Statistical Techniques. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 1638–1651. [CrossRef]
30. Agarwal, M.; Biswas, S.; Nandi, S. Detection of De-authentication Denial of Service attack in 802.11 networks. In Proceedings of the 2013 Annual IEEE India Conference (INDICON), Mumbai, India, 13–15 December 2013; pp. 1–6. [CrossRef]

31. Baharudin, N.; Ali, F.H.M.; Darus, M.Y.; Awang, N. Wireless Intruder Detection System (WIDS) in Detecting De-Authentication and Disassociation Attacks in IEEE 802.11. In Proceedings of the 2015 5th International Conference on IT Convergence and Security (ICITCS), Kuala Lumpur, Malaysia, 24–27 August 2015; pp. 1–5. [CrossRef]
32. Baharudin, N. DARMA: Defeating and Reconnaissance Manna-karma Attacks in 802.11 with Multiple Detections and Prevention. *Int. J. Adv. Trends Comput. Sci. Eng.* **2020**, *9*, 92–100. [CrossRef]
33. OConnor, T. Detecting and Responding to Data Link Layer Attacks. In Proceedings of the 2015 5th International Conference on IT Convergence and Security (ICITCS), Kuala Lumpur, Malaysia, 24–27 August 2015.
34. *IEEE Std 802.11i-2004*; 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 6: Medium Access Control (MAC) Security Enhancements. 2004. Available online: https://standards.ieee.org/ieee/802.11i/3127/ (accessed on 20 May 2023).
35. Wang, S.; Balarezo, J.F.; Kandeepan, S.; Al-Hourani, A.; Chavez, K.G.; Rubinstein, B. Machine Learning in Network Anomaly Detection: A Survey. *IEEE Access* **2021**, *9*, 152379–152396. [CrossRef]
36. Alipour, H.; Al-Nashif, Y.B.; Satam, P.; Hariri, S. Wireless Anomaly Detection Based on IEEE 802.11 Behavior Analysis. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 2158–2170. [CrossRef]
37. WikiDev. TP-LINK_TL-WN721N Documentation. Available online: https://deviwiki.com/wiki/TP-LINK_TL-WN721N (accessed on 23 April 2023).
38. Mange. rtl8192eu Linux Drivers. Available online: https://github.com/Mange/rtl8192eu-linux-driver (accessed on 23 April 2023).
39. WikiDev. TP-LINK_TL-WN821N_v6 Documentation. Available online: https://deviwiki.com/wiki/TP-LINK_TL-WN821N_v6 (accessed on 23 April 2023).
40. WIDS Project Source Code. Available online: https://github.com/krzyusz/WIDS (accessed on 20 May 2023).
41. Biondi, P.; The Scapy Community. Scapy Project v2.5.0. Available online: https://scapy.net/ (accessed on 23 April 2023).
42. Kivy: The Open Source Python App Development Framework. Available online: https://kivy.org/ (accessed on 23 April 2023).
43. White, D. Karma Manna Attacks. 2018. Available online: https://github.com/sensepost/hostapd-mana/wiki/KARMA---MANA-Attack-Theory (accessed on 23 April 2023).
44. E7mer. "ASPj", P.L. MDK4. Available online: https://salsa.debian.org/pkg-security-team/mdk4 (accessed on 23 April 2023).