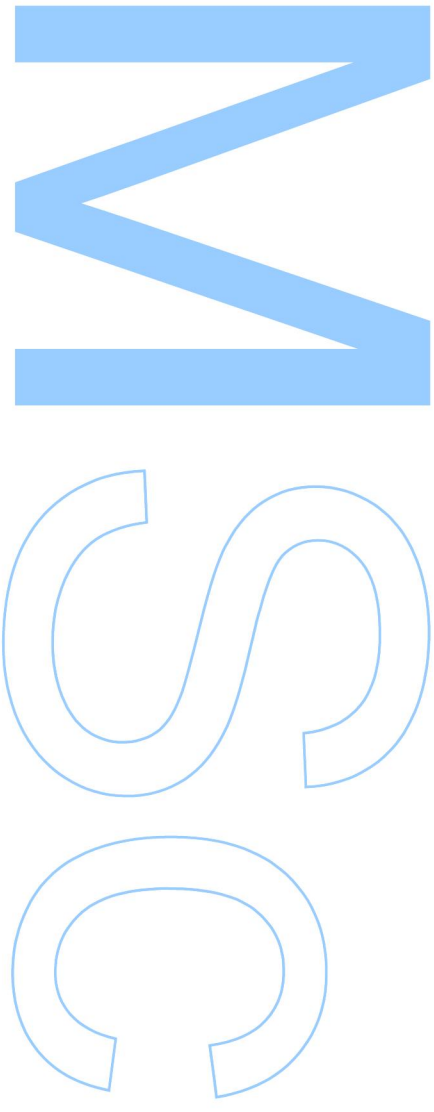
A flexible framework for Rogue Access Point detection

Ricardo Jorge Esteves Gonçalves
Mestrado Integrado em Engenharia de Redes
e Sistemas Informáticos
Departamento de Ciências de Computadores
2016/2017

Orientador
Pedro Miguel Alves Brandão, Professor Assistente,
Faculdade de Ciências da Universidade do Porto

Coorientador
Manuel Eduardo Correia, Professor Assistente,
Faculdade de Ciências da Universidade do Porto



U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, / /

Abstract

Nowadays wireless communications are ubiquitous and getting over requested. It has become a demand in modern days life; people have to be connected always and everywhere. To meet these needs there is an increasing number of WiFi Access Points (APs), located everywhere: schools, coffee shops, shopping malls, airports, trains, buses, and all the free ones from the Internet Service Providers (ISPs) distributed by users' home networks.

Among all these APs how can a user be sure that is connecting to a trusted source?

In a small-medium sized company, without spending much money, how do they guarantee their wireless security?

In order to address these questions there is the need to detect Rogue Access Points (RAPs). There are solutions described in the literature and enterprise solutions. Relative to the latter, it has become obvious that they are not accessible to everyone (high prices), and the first ones do not address all the types of RAPs.

Apart from these solutions we did a thorough study on the most commonly used and recent types of WiFi attacks. With this knowledge we were able to develop a solution to detect RAPs which covers the most commonly known attacks.

The proposed solution, is a modular framework composed of Scanners, Detectors and Actuators, which are responsible for: scanning for available APs, apply a set of heuristics to detect RAPs and alert the application user. And finally, apply a countermeasure mechanism.

Resumo

Hoje em dia, as comunicações sem fios são ubíquas e cada vez mais requisitadas. Tornando-se uma exigência para a maioria das pessoas, causando uma necessidade de estarem ligadas, sempre e em qualquer lugar. Para responder a esta necessidade existe um número crescente de Pontos de acesso (PA) WiFi, localizados em todos os lugares: escolas, cafés, *shoppings*, aeroportos, comboios, autocarros e os PA gratuitos dos Internet Service Providers (ISPs) distribuídos através das redes domésticas dos utilizadores.

Entre todos estes PA como é que um utilizador pode ter certeza de que se está a ligar a uma rede confiável?

Numa pequena ou média empresa, sem gastar muito dinheiro, como se pode garantir a segurança da rede sem fios?

De forma a responder a estas questões existe a necessidade de se detectar PA *rogue*. Para tal, existem soluções descritas na literatura e soluções empresariais. Estas últimas, não são acessíveis para todos (preços altos), e as primeiras não abordam todos os tipos de PA *rogue*.

Para além das soluções mencionadas, foi feito um estudo completo aos tipos de ataques mais recentes e frequentes a redes sem fios. Desta forma, foi possível desenvolver uma solução para detectar PA *rogue*, que cobre os ataques mais conhecidos.

A solução proposta é uma ferramenta modular composta por *Scanners*, *Detectors* e *Actuators*, que são responsáveis por: escutar o ar de forma a encontrar PA disponíveis, aplicar um conjunto de heurísticas para detectar PA *rogue* e alertar os utilizadores. E finalmente aplicar uma contramedida.

Contents

Abstract i

Resumo iii

Contents vi

List of Tables vii

List of Figures ix

Listings xi

Acronyms xiii

1 Introduction 1

2 Background 3

2.1 Wireless Network . . . 3

2.1.1 Types of wireless networks . . . 3

2.1.2 Wireless Network Architecture . . . 5

2.1.2.1 Wireless Network Components . . . 5

2.2 The 802.11 Standard . . . 7

2.2.1 Frame types . . . 7

2.2.2 802.11 Security . . . 10

2.2.3 802.11 Channels . . . 12

2.3 Taxonomy of Rogue Access Points (RAPs) 12
2.4 WiFi Attacks 14

3 Related Work **19**

3.1 Available 802.11 security to protect from RAPs 20
3.2 Available RAP Detection approaches 21

4 Architecture **29**

4.1 Profiles 30
4.2 Modules 30
4.2.1 Scanners 31
4.2.2 Detectors 31
4.2.3 Actuators 33

5 Development **37**

5.1 Overview 37
5.2 Scanners 40
5.3 Detectors 44
5.4 Actuators 45

6 Results **49**

6.1 Proof of Concept 49

7 Conclusions **53**

7.1 Future Work 54

Bibliography **55**

List of Tables

2.1 802.11 frame subtypes 8

3.1 Evil Twin Coexistence techniques 22

3.2 Coexistence and Replacement Evil Twin techniques 23

3.3 Detection techniques 27

3.4 Strengths and weaknesses of existing techniques 27

6.1 Test cases 52

List of Figures

2.1 Connection for open authentication 9

2.2 Evil Twin RAP type 13

2.3 Passive Scan 15

2.4 Karma attack in Active Scan 15

2.5 RAP with radius server attack 16

4.1 Framework architecture 29

4.2 Framework Modules 31

5.1 Application structure 38

5.2 iwlist scan output 42

5.3 Parsed iwlist scan output 42

5.4 Wireshark scan 43

6.1 Detection 1 50

6.2 Detection 2 50

6.3 Detection 3 51

6.4 Detection 4 51

6.5 Detection 5 52

Listings

4.1 Whitelist heuristic 32
4.2 Encryption heuristic 32
4.3 RSSI heuristic 32
4.4 Call Active Detectors module 32
4.5 Blacklist heuristic 33
4.6 Active Detectors heuristics 34
4.7 Association process 34
4.8 Detection of Karma attacks 35
5.1 Scanning APs algorithm 40
5.2 Scanning APs tuned algorithm 40
5.3 Filter AP algorithm 40
5.4 Scapy scan snippet 43
5.5 Passive RAP algorithm detection 44
5.6 Random SSID Prob requests 46
5.7 Generate random SSIDs 46
5.8 Deauthentication request 46

Acronyms

AP Access Point

PA Pontos de acesso

RAP Rogue Access Point

MITM Man-in-the-Middle

LAN Local Area Network

WLAN Wireless Local Area Network

WPAN Wireless Personal Area Network

WANET Wireless Ad Hoc Network

MANET Mobile Ad Hoc Network

WMAN Wireless Metropolitan Area Network

ISP Internet Service Provider

IANA Internet Assigned Numbers Authority

WWAN Wireless Wide Area Network

WAN Wide Area Network

WMN Wireless Mesh Network

IEEE Institute of Electrical and Electronics Engineers

OSI Open Systems Interconnection

OFDM Orthogonal frequency-division multiplexing

NIC Network Interface Card

ATIM Ad Hoc Traffic Indication Map

FCS Frame Check Sequence

MAC Media Access Control

CRC Cyclic Redundancy Checking

BS Base Station

PNL Preferred Network List

SSID Service Set Identifier

BSS Basic Service Set

BSSID Basic Service Set Identifier

RSSI Received Signal Strength Indication

RSS Received Signal Strength

TSF Timing Synchronization Function

DNS Domain Name Server

DHCP Dynamic Host Configuration Protocol

IP Internet Protocol

TCP Transmission Control Protocol

HTTP Hypertext Transfer Protocol

RTT Round Trip Time

LRTT Local Round Trip Time

WPA WiFi Protected Access

WPA2 WiFi Protected Access II

TIM Traffic Indication Map

WEP Wired Equivalent Privacy

RC4 Rivest Cipher 4

IV Initialization Vector

XOR Exclusive OR

TKIP Temporal Key Integrity Protocol

AES Advanced Encryption Standard

CCM Counter with CBC-MAC

OCB Offset Codebook Mode

CCMP Counter-Mode/Cipher Block Chaining Message Authentication Code Protocol

WRAP Wireless Robust Authenticated Protocol

RADIUS Remote Authentication Dial In User Service

WPA-PSK WPA Pre-shared key

PSK Pre-shared key

PAP Password Authentication Protocol

EAP Extensible Authentication Protocol

EAP-TLS EAP Transport Layer Security

EAPOL Extensible Authentication Protocol over LAN

EAP-SWAT Simple Wireless Authentication Technique

CA Certificate Authority

SSL Secure Sockets Layer

MSK Master Session Key

PMK Pairwise Master Key

PTK Pairwise Transient Key

GTK Group Transient Key

MIC Message Integrity Code

KCK Key Confirmation Key

KEK Key Encryption Key

GEK Group Encryption Key

GIK Group Integrity Key

IDS Intrusion Detection System

VPN Virtual Private Network

SSH Secure Shell

SVM Support Vector Machine

DWSA Distributed Wireless Security Auditor

API Application Programming Interface

Chapter 1

Introduction

The dimension of the Internet these days is something unmatched with a past not so far away where the tasks were mainly focused in information search. In the days we live in, the Internet tasks evolved to banking, shopping, messaging, video calls, gaming, social networks and geolocation. In short a lot of essential requirements for the day-to-day of several persons. In addition to the evolution in terms of use and services, the Internet has also transformed itself in the eyes of its users. With the arrival of devices with wireless capabilities we had a transition to WiFi, which today takes the Internet to almost everywhere. We can say that the Internet, is nowadays the world in its virtual format.

These changes have associated new requirements for its users, most online services require permissions and access to personal data that then move to and live in the virtual environment.

The Wireless Local Area Networks (WLANs) are wireless computer networks that connect users' devices using wireless transmissions within a limited area, such as home, university or office. Allowing its users to access the network as long as they are within the covered range of the WLAN. Associated with the WLANs, we have Access Points (APs) which are responsible for coordinating the wireless users and connecting them to the wired part of the network and thus to the Internet. APs are usually wireless routers, and transmit and receive radio frequencies with which devices with wireless capabilities communicate. These APs are everywhere, from schools to hospitals, bus, airports, coffee shops and shopping malls, in order to provide Internet connections.

In a generic scenario of Internet usage several APs are used, this happens because the majority of the users' devices such as laptops, smartphones and tablets are meant to ensure mobility for its users and thus the possibility of being "always" connected to the Internet. When a user wants to connect to the Internet, mainly two alternatives are available: using mobile cellular networks or through WiFi networks. The first option carries a higher cost and usually has a traffic limit for the users although it guarantees a greater autonomy (i.e., there is no need to look for a WiFi hotspot). The second option in some cases is free, which makes it a better option for many users. But when a user requests a wireless Internet connection, how can they be sure that they are

connecting to a trusted source?

In order to understand and answer this question, the work of this dissertation addresses the problem of false APs that were not installed by an authorized network administrator, i.e., Rogue Access Points (RAPs). The word "rogue" clarifies the intention of this type of APs that is a malicious one.

When an attacker sets up a RAP he will monitor all the traffic that goes through it and will be able to perform several type of attacks, being the most common Man-in-the-Middle (MITM) [44] attacks, resulting in traffic monitoring, DNS Hijacking, SSL MITM and Application attacks. Hence, when a benign client connects to that AP, this client which is now a victim, will be exposed to these security issues. Lets say that that person wants to check his electronic bank account. Since the person is on the RAP network, the attacker could forge the original electronic bank page and steal that users' credentials. Therefore, we should be able to identify whether an AP is a trusted one or a rogue.

It is important to notice the fact that we are concerned about detection, and we are aware of prevention mechanisms, such as the 802.1X [1] which is one of the most widely known and used in the enterprise world.

With this in mind we focused on the possible solutions to this challenge, meaning detection methods. For such, we studied several works from the state of the art literature, using them for our proposed model.

In the first part of this dissertation it is exposed the existing types of RAPs and the security mechanisms available to prevent them, this is, the state of the art proposed works. This information will be the basis for the development of this project: the RAP detection tool. Further on, we describe the architecture model of our proposal and explain how it is designed. Ending with a proof of concept derived from the developed tool.

One of the motivating factors for the development of this tool and dissertation was the ease with which these RAPs are created and the contrasting difficulty in identifying them, driven by the lack of a standard method that allows their identification and blockage.

Chapter 2

Background

In this chapter we are going to analyze the main theory behind the subject of this project. We start by describing and explaining the types of wireless networks, and then architectures and their components. After this, we introduce the 802.11 standard and explain its security. Finally we present the taxonomy of RAPs according to the literature and explain the most common WiFi attacks.

2.1 Wireless Network

A wireless network consists of a number of nodes which communicate with each other over a wireless channel [23]. There are some wireless networks which contain a wired backbone with only the last hop being wireless, it can be cellular voice, data networks or mobile IP. In other types of wireless networks all of its links are wireless. An example of these networks is Mobile Ad Hoc Networks (MANETs). As more examples of wireless networks we can have Wireless Local Area Networks (WLANs), wireless sensor networks, satellite communication networks or terrestrial microwave networks. The implementation of the radio communications used in wireless networks takes place at the physical layer of the OSI model.

In the following sections we will explain the different types of wireless networks in order to understand the ones related to the subject of this project.

2.1.1 Types of wireless networks

- Wireless Personal Area Network (WPAN)

  The WPANs, as it names suggests, relates to networks of a personal scope that interconnect devices within small areas. As an example we have Bluetooth radio and infrared light technologies which can provide the personal area network that gives the connectivity between the desired devices; this could be between a mouse device and a computer or a pair of headsets and the computer.

For the scope of this project networks of this kind are not relevant.

- Wireless Local Area Network (WLAN)

  In a WLAN typically we have a user that connects to a Local Area Network (LAN) through a wireless radio connection which is used to transmit and receive the data over the air. Usually these are short distances networks and the connection is done through an Access Point (AP) which will provide the internet access. The possibility of a user remaining connected to the network while moving around within the area is due to spread-spectrum and OFDM technologies. Regarding the WLANs technologies there are a family of specifications, 802.11 standard developed by the Institute of Electrical and Electronics Engineers (IEEE). The base standard, the 802.11, provides the basis for wireless network products using the WiFi certification.

  The WLAN and the 802.11 standard will be of much importance in this dissertation since the main activity, Rogue Access Points (RAPs) deployment and its detection, will use the messaging, state machines, and other details of the standard.

- Wireless Ad Hoc Network (WANET)

  This type of network is also known as Wireless Mesh Network (WMN) or MANET. Here, a group of radio nodes are organized in a mesh topology and build the network. The messages are forwarded and routed through the nodes. The wireless network is decentralized and does not rely on a pre existing infrastructure, it builds itself by its member nodes.

  Since there is no infrastructure such as routers or access points this type of networks will not be relevant for this thesis.

- Wireless Metropolitan Area Network (WMAN)

  WMAN connects several wireless LANs. WMANs interconnects users in a geographic area or region, usually within a city or town and serves as an Internet Service Provider (ISP) for larger LANs.

- Wireless Wide Area Network (WWAN)

  WWAN covers a large geographical area. For example, a country or a continent. A Wide Area Network (WAN) interconnects several LANs or WMANs and provides solutions to companies or organizations working from distant geographical locations. Thus, users and devices from different locations are able to communicate with users and devices from other locations.

  The Internet is the largest WAN in the world.

2.1.2 Wireless Network Architecture

As in wired networks the wireless networks also utilize components; but its components have different scopes, here it is necessary to convert the information signals into a form suitable for transmission through the air medium. Apart from that, the different components of the WLAN should be designed and arranged in order to create the network architecture.

The main reference for this section is the: Wireless System Architecture: How Wireless Works, from Cisco Press articles [1].

2.1.2.1 Wireless Network Components

In the life-cycle of a wireless network there are a series of components involved. These components support the communications using radio or light waves that propagate through the air.

- Users

  A user is anything that interacts with the wireless network. The most common type of user is a person and it is a key factor in the network since the user will provide data that will flow through the network and will receive the benefits of using the wireless network. As an example, someone accessing the Internet from a public wireless LAN at a coffee shop or shopping mall is a user. Also, network administrators, are another example of users. These two examples play an important role in the scope of this project since alerting them of possible RAPs is the main motivation behind this project.

  and therefore they play an important role in the scope of this dissertation since alerting the users of possible RAPs is the motivation

  We can differentiate three types of users: mobile, portable and stationary. The mobile type, are the kind of users which tend to move around a facility or city. And thus, the assurance of mobility is a key benefit from wireless networks. An example for this type of user is the case where a person is chatting from a smartphone while on a bus or walking through a shopping mall. Here the smartphone must establish frequent connections to the APs in different networks, i.e., the AP in the bus and APs in the shopping mall.

  The portable type of users stay at a particular location while using the wireless network for a specific period of time. Here we can use the example of using a laptop to access the wireless network during a meeting conference and at the end the user will turn off the laptop and no longer be connected. Regarding the stationary users, it is considered the cases where a person uses the wireless network from a fixed place for longer periods. For example, a user in its home network, or an employee in the work station.

  Here the wireless network does not need to support continual movement. For the last type, stationary users, it is a type that operates from one place for an indefinite period of time. An example is a user working from a wireless computer in the work office.

[1] http://www.ciscopress.com/articles/article.asp?p=344242

- Computer Devices

  The computer devices, also referred as clients, operate on a wireless network. There are computer devices specifically for users, that can be for example: smartphones, laptops, desktop pcs, printers or desk phones, and end systems, such as: servers, databases and websites. All the computer devices are crucial parts for this project as they are all involved in the process of interaction between the users and the services, which run on the devices.

- Network Interface Cards (NICs)

  The NIC stands for network interface card and its function is to provide the interface between the computer devices and the wireless network infrastructure. There are NICs that are inside the computer device and others that are external network adapters.

  There are wireless network standards that define how the NICs operates. The IEEE 802.11 standard has the information for wireless LAN NIC implementation. Therefore the NIC will only be able to interface with wireless network infrastructures complying with that standard.

  An important point to our work are the antennas of the wireless NICs. An antenna converts electrical signals to radio or light waves for propagation over the air medium [37] for transmission. It realizes the reverse process for reception. The antennas can have many structures, as they can be external, internal, permanent, or detachable. For this project we will mainly focus on WiFi antennas, once they will be used to perform WiFi attacks, RAP deployment and detection.

- Air Medium

  It is in the air medium where the wireless communications signals propagate, these signals are the "engine" for the wireless networking. The air is the conduit by which the information flows among the computer devices and the wireless infrastructure. In this medium we have the same problem as in a simple conversation which is, as we get further apart from someone we are talking to we will eventually stop listening to that person's voice. As radio signals lose power as distance increases, the separation between transmitter and receiver is paramount for viable communication. We should also consider the presence of obstacles in the medium which may lessen or scatter the strength and range of the signals. For WLAN type of networks things like walls, furniture or electronic devices are some examples of obstacles that may interfere with the signal. In WMANs or WWANs weather conditions as rain, snow or smog will influence the propagation of wireless communications signals as other obstacles like trees or buildings.

- Wireless Network Infrastructure

  The wireless infrastructure is where users and end systems communicate. These infrastructures are composed by hardware and software resources that make possible the network connectivity, communication, operations and management.

  The network infrastructure is often composed by base stations, access controllers, application connectivity software, and a distributed system. To this project it is important to understand

the Base Station (BS) component, which interfaces the wireless communications signals traveling through the air medium and bridges them to a wired network. A base station enables the network users to access the network services, such as web browsing, e-mail access, online gaming, video messaging and others. Depending on the purpose, a base station can have different names. For example, an AP represents a generic base station for a wireless LAN. And consequently, a generic example of a network infrastructure could be a set of APs within a WLAN which would give mobility to its users in that facility. Then, the NIC in the users device would connect him to the nearest AP giving him access to the network infrastructure. As the user moved through the facility he gets automatically reconnected to the nearest AP. Another example of base stations can be gateways and routers. The gateway can work as access controller and application connectivity. And a router would manage a facility giving network capability to several devices simultaneously.

2.2 The 802.11 Standard

In this dissertation when we mention wireless we are addressing the 802.11 Standard [2], which is a set of rules on how to implement and use wireless networks. Previously, we overviewed some types of wireless networks from where we can understand that there are two basic modes of operation: the infrastructure and ad hoc. As explained, in networks using infrastructure mode, the wireless clients, for example a smartphone, has to connect to an AP to join the network. Thus, we are going to focus on the infrastructure mode of 802.11 as we are addressing issues between users and APs, in particular, *rogue* APs.

2.2.1 Frame types

In 802.11 networks we have three packet frame types: management, control and data. Each of them has subtypes, listed in Table 2.1.

The management frame types are mainly used for network management and admission control, for example they allow WLAN devices to initiate and maintain communications. The control types are usually used for access control, i.e. allow some stations to access the medium while denying access to other. And the data frame types are used for data transmission [9].

Our main focus is in the management frame (and its subframes), since it is there where all the APs main interactions happens. For our work we need to clarify how APs are deployed and how they advertise themselves. As well as what they advertise and the type of capabilities they have.

Beacon frames information

The APs sends out periodically broadcast frames called Beacon frames to announce its

[2] https://standards.ieee.org/about/get/802/802.11.html

Management	**Control**	**Data**
Association request	PS-Poll (Power Save Poll)	Data
Association response	RTS (Request To Send)	Data + CF-ACK
Reassociation request	CTS (Clear To Send)	Data + CF-Poll
Reassociation response	ACK (Acknowledgement)	Data + CF-ACK + CF-Poll
Probe request	CF End (Contention Free End)	Null Function (no data)
Probe response	CF End + CF-ACK	CF-ACK (no data)
Beacon		CF-Poll (no data)
ATIM		
Disassociation		
Deauthentication		

Table 2.1: 802.11 frame subtypes

presence, i.e., a wireless LAN. They contain all the information about that respective network. A beacon frame, like other frame types, has a structure which consists of a header, frame body and a Frame Check Sequence (FCS). In the header it has the source and destination MAC addresses and information regarding the communication process. The FCS is based on the Cyclic Redundancy Checking (CRC) algorithm and it is applied to bytes from the header and body. The relevant part for our understanding resides in the frame body which contains the following information [18]:

- Beacon interval - amount of time between beacon transmissions. A station needs to know the beacon interval (before entering power save mode) in order to wake up to receive the beacon.
- Timestamp - when a station receives a beacon frame, it uses the timestamp value to update its local clock. This enables the synchronization between all stations that are associated with the same AP. This information will be useful in the process of identifying RAPs, since software based access points have accuracy flaws in the timestamps [32].
- Service Set Identifier (SSID) - identifies a specific WLAN. Prior to associating with a WLAN, a station must know the SSID to connect to. The NIC's firmware will be set for this SSID for connection.
- Supported rates - information that describes the rates that some WLAN supports. With this information, the stations can use performance metrics to choose which AP to associate with.
- Parameter Sets - information about the specific signaling methods. For example, a beacon would include, in the right parameter set, the channel number that an AP is using. Furthermore, a beacon belonging to frequency hopping network would indicate the hopping pattern and live time.

- Capability Information - requirements for stations that want to connect to the WLAN that the beacon represents. For example, this information may indicate that all stations must use WiFi Protected Access (WPA) so they can associate on the network.
- Traffic Indication Map (TIM) - are sent periodically by an AP (within a beacon) to identify which stations using power saving mode have data frames waiting for them in the APs buffer. The TIM identifies a station by the association ID that the AP assigned it during the association process.

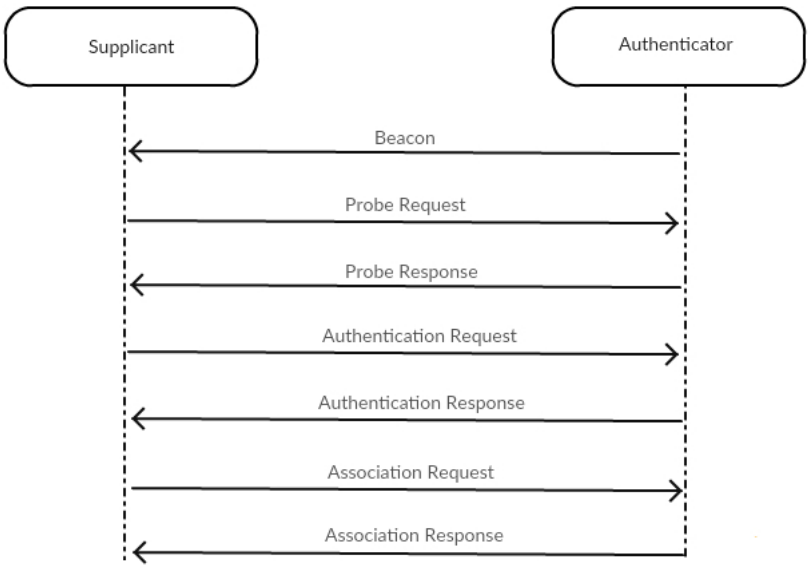


Figure 2.1: Connection for open authentication

In a typical scenario the AP transmits beacon frames periodically in order to announce its presence and send information. The information contained in a frame will be of much interest for the development of our tool, for instance it contains information like: Beacon interval, Timestamp, SSID and Capability Information. The clients (radio NICs) continually scan all 802.11 radio channels and listen to beacons in order to choose which AP is the best to associate with [19]. On the other hand the clients can actively send probe requests, which has the SSID that it wants to connect to, in order to identify APs within range. In response to these probe requests, the AP sends a probe response frame, much like the beacon frame, that contains information like supported rates and capabilities of the network. After this discovering phase, the client should be authenticated by the AP, and associated, in order to get more privileges, where authentication and association messages are exchanged. In the authentication process the identity of the client is sent to the AP through an authentication request frame, then the AP either accepts or rejects its identity (with an authentication response). The default system is the open authentication, shown in figure 2.1, where there is no identity checking, i.e., the AP only sends the authentication response with acceptance or rejection. After this the client will send an association request in order to enable the AP to allocate resources and establish the synchronization with the user. The AP send the association response with the information of

its acceptance or rejection [34]. Hereupon, the user and the AP can exchange data. In secure connections it is sent an Authentication challenge to the supplicant, and the specific details are explained in the next subsection.

2.2.2 802.11 Security

In order to establish a secure communication the connection model needs additional steps. In particular, it is necessary to perform a four-way handshake [26], where the wireless client (supplicant) and the AP (authenticator) derive the encryption keys to be used to encrypt the wireless data frames which will guarantees the communication security.

Encryption

- Wired Equivalent Privacy (WEP)

  In the WEP, as the name suggests, the main goal was to provide a confidentiality level close to that from wired networks. Let us divide the WEP algorithm in two sections: Authorization and Traffic Encryption. Starting from the Authorization part the WEP, it has two methods for the authentication: open system and shared key. In the open system the client does not need to provide any credentials in order to connect to the AP. The authentication gets completed after the exchange of two messages. In these scenarios the network can be protected with white lists of specific MAC addresses. The shared key authentication uses the four-way handshake prior to the client joining the network. In particular, (i) the client sends an Authentication Request message with the MAC addresses of the both parties (client and AP); (ii) then the AP responds with a challenge message which contains a 128 bits random number; (iii) the client sends the response message which contains the random number encrypted with the WEP shared key. Then the AP decrypts the previous messages using its shared key. Then if both numbers match, the AP acknowledges that the client has the shared key. In the final step (iv), the AP sends an Authentication Response message with the outcome of the authentication process. As we can conclude from this reasoning the authentication process is unidirectional where the AP authenticates the client [31].

  Regarding traffic encryption, WEP makes use of the Rivest Cipher 4 (RC4) algorithm for confidentiality and CRC-32 for message integrity. The confidentiality process uses a static key (called root key), and it can be of two types (regarding key size): WEP-40 and WEP-104.

  The WEP-40 key is the basis for the generation of a session key and supports key sizes of 40 bits. Here only the data frames are protected, management and control are unprotected. The process of encrypting a packet involves the following steps: First, is generated a 24-bit long Initialization Vector (IV). Then, a root key is appended to the IV defining the "per packet key", which will seed the RC4 algorithm to produce a key sequence, known as keystream. Finally, this keystream is XORed with the concatenation of the packet plaintext

and its CRC-32 value which will create the packet ciphertext. The WEP-104 uses the same process with a key size of 104 bits.

- WiFi Protected Access (WPA)

  The WPA is a security technology introduced in the 802.1X to fix the weaknesses of the original security mechanism. The lack of native wireless security was the main factor to the development of 802.11i. Therefore, WPA was a transitional point while WiFi Protected Access II (WPA2) was being developed. In WPA stronger encryption mechanisms were introduced: Temporal Key Integrity Protocol (TKIP) and Advanced Encryption Standard (AES). Furthermore, WPA addressed critical security issues, like mutual authentication through the use of 802.1X and the Extensible Authentication Protocol (EAP), better IV lengths, stronger integrity check mechanism and secure re-keying function. WPA may make use of central authentication servers like Remote Authentication Dial In User Service (RADIUS) for user authentication, access control and management. This practice is mainly used in enterprise environments. For home users a variation of WPA was developed, named WPA Pre-shared key (WPA-PSK). This is a simplified version of WPA which uses a passphrase as a pre-shared secret key for the network users.

- WiFi Protected Access II (WPA2)

  The commonly known WPA2, developed in IEEE 802.11, was an amendment to the original IEEE 802.11 standard in order to increase the security of the protocol. The current version of the standard, 802.11ac [2] still uses the 802.11i as the primary security protocol.

  Regarding WPA2 we will focus on the Key Construction process and Traffic Confidentiality and Integrity.

  In the Key Construction process all the keys are derived from a single key (the key in the highest layer of the hierarchy). This key will have two types resulting from the authentication method used. The two methods that can be used in WPA2 for the authentication process may be based on a pre-shared key or on the 802.1X framework. If the first is used, the top key will be called Pre-shared key (PSK), if the second is chosen, the top key will be called Master Session Key (MSK).

  Then, the top level key is used to generate the primary keying material, which is the Pairwise Master Key (PMK). The next level of the keying hierarchy is composed by the Pairwise Transient Key (PTK) and the Group Transient Key (GTK), these keys are specific to client-AP pair and are produced during the authentication process.

  Finally the PTK key is split in five sub keys: one temporal encryption key, two temporal Message Integrity Code (MIC) keys, EAPOL-Key Key Confirmation Key (KCK), EAPOL-Key Key Encryption Key (KEK). Within these keys, the temporal key is the one used to encrypt or decrypt unicast traffic.

  The GTK is split in two keys: Group Encryption Key (GEK) used to encrypt or decrypt multicast traffic, and Group Integrity Key (GIK) used for verifying the MIC of multicast or broadcast traffic.

Regarding the traffic confidentiality and integrity of WPA2, it supports three alternative protocols for protecting the network traffic. These methods are: TKIP which is based in RC4 algorithm, Counter-Mode/Cipher Block Chaining Message Authentication Code Protocol (CCMP) based on the AES algorithm in CCM mode, and Wireless Robust Authenticated Protocol (WRAP) which is based on AES OCB.

2.2.3 802.11 Channels

All WiFi networks transmits and receives data on a certain frequency, or channel. In the current 802.11 standard there are five distinct frequency ranges: 2.4 GHz, 3.6 GHz, 4.9 GHz, 5 GHz, and 5.9 GHz bands [2]. Each of these ranges is divided into channels, where the number of allowable channels is defined by country as well as the number of allowed users and maximum power levels.

As the WiFi data is digital, different devices are able to communicate successfully in the same channel. When more than one device is using the same channel, although the data does not interfere with each other, the channel gets contended. When devices are in neighbor channels the transmissions will start to interfere with each other causing other issues than contention in the case of single channel. This happens because the transmissions are not restricted to one frequency and thus the communication is spread across five channels in total. For example if a device (router) was on channel 6 the communication will range channel 4 to channel 8, then the neighbor channels (for example 5 or 7) will suffer from interference.

2.3 Taxonomy of RAPs

We will consider the taxonomy of RAPs based in the current literature. Thus, RAPs are classified into four categories: Evil-Twin, Improperly Configured, Unauthorized, and Compromised. In [5] two more types of RAPs are also classified: RAP-based deauthentication/disassociation attacks and forging of the first message in a four-way handshake type.

- Evil Twin

  This type of RAP is sometimes referred as Spoofed AP and its name represents, in a figure of speech, the type of the attack. Evil Twin RAPs rely on a software-based AP which is often installed/configured on a portable device. This portable device must have a wireless card capable of working in monitor mode. It is also quite common the use of external wireless antennas. With these conditions the attacker can make use of tools like Airbase-ng [3] in order to set up the RAP.

  In the IEEE 802.11 standard there are only two parameters to identify APs to users. They are: the SSID and the BSSID (Media Access Control (MAC) address of the AP). The main problem resides in the fact that these identifiers can be spoofed, and therefore the

[3] Airbase-ng: https://www.aircrack-ng.org/doku.php?id=airbase-ng

benign AP will be cloned (spoofed) in an Evil Twin AP and both will be indistinguishable from the point of view of the wireless users.

The Evil Twin RAP type can exists in two forms: Coexistence and Replacement. In the first one the legitimate AP and the Evil Twin coexist in the same location. The process is like the one previously described, with the extra detail of the attacker increasing the RAP signal strength to force the users to connect to it. And as the IEEE 802.11 standard states that WLAN clients must connect to APs that have the strongest signal, the attacker will benefits from it. In the second type, Replacement, the Evil Twin replaces the legitimate AP by shutting it down. Here the RAP needs to have an Internet connection while in the first case it could relay the packets to the legitimate AP.

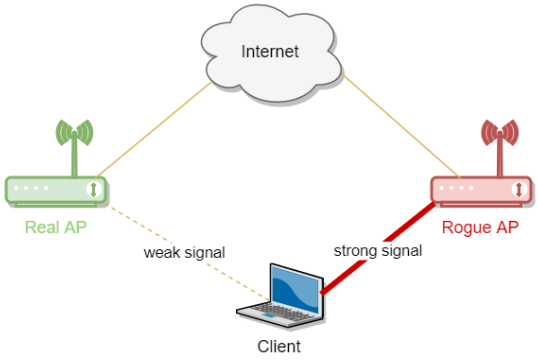


Figure 2.2: Evil Twin RAP type

Evil Twin RAPs are mainly used to listen to users traffic (as they browse the Internet), and carry out attacks such as interception, replaying, and traffic manipulation, to these users devices [47].

The configuration of an Evil Twin AP might require some extra steps to avoid Intrusion Detection Systems (IDSs), in these cases an attacker masquerades the AP MAC address and SSID, configures a Domain Name Server (DNS) server to connect to the Internet and a Dynamic Host Configuration Protocol (DHCP) server to assign IP addresses.

Nowadays, Evil Twin APs can be deployed from smartphones and therefore harden the task of identifying them even more.

- Improperly Configured AP

  This type of RAP, as its name suggests is an AP which was improperly configured, there is no adversary involved in the creation process. This can happen when, for instance, an administrator does not use robust authentication and encryption settings. APs can also become vulnerable after some software updates [36] or the lack of upgrades. The issue with this type of RAP is the consequence of possible backdoors in an organization network infrastructure that might enable attackers to bypass authentication, explore and exploit the network.

- Unauthorized AP

  Type of RAP installed by employee or naive user without the network administrator permission. This RAP is connected to the wired side of the network (like a legitimate AP) and thus it is considered part of the WLAN. A RAP of this kind can exist in the network of an organization for the convenience of its employees, i.e., access some external network resources, that otherwise might be blocked by a firewall. Apart from this, it can also be deployed with malicious intentions and thus it will create an entry point in the organization network and compromise its security. Hence, unauthorized users will be able to eavesdrop and launch attacks like in the previous types.

- Compromised AP

  When the shared keys used to secure the network communications gets compromised, the AP becomes a rouge. Thus, the person who obtains the keys (the attacker), will be able to join the network and has previously described, will be able to eavesdrop and launch a series of attacks to the network and its users.

2.4 WiFi Attacks

There are several types of wireless network attacks that target all the pillars of the information security: confidentiality, integrity and availability. For our work we are interested in the RAP type of attacks which target: the access control mechanisms, attempting to penetrate a network, Confidentiality attacks, which attempt to intercept private information sent over the wireless, and Authentication attacks, where legitimate user identities and credentials are stolen to access private networks and services [42].

Thus, we explain the most commonly used techniques to perform WiFi attacks:

- **RAP with stronger signal**: in this specific attack, the RAP behaves as an Evil Twin, as described in 2.3. Here the attacker entices the victim to connect to the Evil Twin AP. This attack does not require any special hardware apart from a wireless adapter with access point capabilities, and in order to accomplish this, it is often used the Aircrack-ng [4] tool.

  The first step in this type of attack is to set the wireless card to monitor or promiscuous mode in order to start capturing traffic and gather information about the near by APs and its clients. With this information an attacker can use the same tool and create a new AP matching the desired victim AP. The common way an attacker performs this attack is by creating the Evil Twin AP with the same SSID, BSSID and channel, and increasing the AP signal power in order to overcome the signal of the legitimate one.

  At this stage, the attacker has a RAP, specifically an Evil Twin, waiting for victims to connect.

[4] Aircrack-ng: https://www.aircrack-ng.org/

An important point for the detection process is that usually in this type of attacks the Evil Twin AP is created with Open authentication which will ease its detection. On the other hand, if the attacker gets the PSK of the network, they will be able to create an exact clone of the legitimate AP which will be more difficult to detect.

These two factors have direct influence in the next topic, deauthentication attacks, because if no authentication (Open) is used, the signal strength will not matter, this is because of the way wireless roaming works, where it will only work when the APs have the same SSID, BSSID, security mode and passphrase. We should also understand that the channel is often different and non-overlapping. On top of that the target station should support roaming. Only with these conditions the target victim would automatically switch to another AP with a better signal strength.

- **Deauthentication attacks**: In order to bypass the stated issue, attackers often use deathentication attacks to force their victims to connect to the RAP.

  This attack is possible because of the special deauthentication frame in the 802.11 standard. The attacker sends this special frame to the desired victim on behalf of the legitimate AP and consequently the target victim will be deauthenticated from that AP. Then, in the process of re-authentication, the victim will automatically reconnect to the strongest known AP.

- **Karma attacks**: In the previous attack, we described that the attacker chooses a specific AP to clone, they monitors the network and selects the desired AP. In karma attacks, things are different, it goes from a manual process of selecting and creating a specific AP to an automatic process where it listens to the networks the wireless devices request and creates them on demand. This is possible because of a vulnerability in the method of discovering available WiFi networks [15].

  When a client wants to connect to a network he must perform a scan for available networks (Probe request) and then select one matching its Preferred Network List (PNL). And as seen in 2.2.1, there are two types of scan: passive and active. Thus, this attack will target the active scans, which are probe requests that are sent with the SSIDs of the client preferred networks. With this information the attacker will be able to respond to the probe request by sending the probe response matching the network requested by the victim.

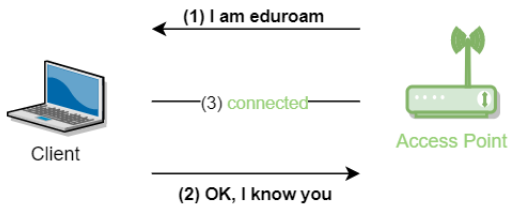


Figure 2.3: Passive Scan

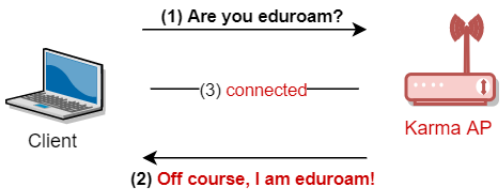


Figure 2.4: Karma attack in Active Scan

Even being an old attack [15], the karma attack can still affect clients that are using active probing authentication. And to perform it, an attacker could use a Pineapple AP [5] or a Pwn Phone [6] that are portable devices that make the attack much simpler to accomplish.

- **Rogue AP with radius server**: The attacks seen so far target private/home and open networks and are the most known and easy to accomplish.

  When facing WPA Enterprise it is also possible to attack this scheme and in particular, impersonate a WPA Enterprise AP. To this purpose there are a couple of tools that can be used by an attacker: `hostapd-wpe` [7] and `freeradius-wpe` [8]. These tools will implement IEEE 802.1X Authenticator and Authentication Server impersonation attacks in order to obtain client credentials and establish connectivity to the client.

  In order to better understand this attack we deployed an RAP using the `hostapd-wpe`. The created RAP had the SSID `eduroam` in order to impersonate the school APs. With this test we could see that in cases where the clients did not had the CA certificate, the tool was capable of retrieving the victims credentials in clear text, this was possible because the inner authentication configured in the `eduroam` APs is Password Authentication Protocol (PAP) which transmits unencrypted passwords over the network.

```
root@k4li-l4b:~# hostapd-wpe /etc/hostapd-wpe/hostapd-wpe.conf
Configuration file: /etc/hostapd-wpe/hostapd-wpe.conf
Using interface wlan0 with hwaddr bc:67:1c:d8:f8:bf and ssid "eduroam"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
wlan0: STA 58:e2:8f:20:7b:6d IEEE 802.11: authenticated
wlan0: STA 58:e2:8f:20:7b:6d IEEE 802.11: associated (aid 1)
wlan0: CTRL-EVENT-EAP-STARTED 58:e2:8f:20:7b:6d
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25
wlan0: STA 58:e2:8f:20:7b:6d IEEE 802.11: disassociated
wlan0: STA 58:e2:8f:20:7b:6d IEEE 802.11: deauthenticated due to inactivity (timer DEAUTH/REMOVE)
wlan0: STA c4:e9:84:1e:3c:4e IEEE 802.11: authenticated
wlan0: STA c4:e9:84:1e:3c:4e IEEE 802.11: associated (aid 1)
wlan0: CTRL-EVENT-EAP-STARTED c4:e9:84:1e:3c:4e
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=21

eap-ttls/pap: Fri Sep 29 01:12:44 2017
        username:       up200802530
        password:       aminhapassword-s3gura
wlan0: CTRL-EVENT-EAP-FAILURE c4:e9:84:1e:3c:4e
wlan0: STA c4:e9:84:1e:3c:4e IEEE 802.1X: authentication failed - EAP type: 0 (unknown)
wlan0: STA c4:e9:84:1e:3c:4e IEEE 802.1X: Supplicant used different EAP type: 21 (TTLS)
wlan0: STA c4:e9:84:1e:3c:4e IEEE 802.11: deauthenticated due to local deauth request
wlan0: STA c4:e9:84:1e:3c:4e IEEE 802.11: authenticated
wlan0: STA c4:e9:84:1e:3c:4e IEEE 802.11: associated (aid 1)
wlan0: CTRL-EVENT-EAP-STARTED c4:e9:84:1e:3c:4e
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=21
```

Figure 2.5: RAP with radius server attack

This is important to notice as some users configure the network connection without adding the CA certificate, and to prevent such cases a strong authentication algorithm should be enforced.

[5] https://hakshop.com/products/wifi-pineapple
[6] https://store.pwnieexpress.com/product/pwn-phone2014b/
[7] http://pkg.kali.org/pkg/hostapd-wpe
[8] http://pkg.kali.org/pkg/freeradius-wpe

This background knowledge was crucial to the development process of the RAP detection tool. The taxonomy guarantees the scope of the project and the type of attacks creates the base cases for the proof of concept.

The Beacon frames information, 2.2.1, was fundamental in understanding what is announced by an AP and consequently detect anomalies in this advertisement, that in some cases represents *rogues*. More details about how these Beacon frames are represented in a Wireshark [9] capture are presented in the Development section, chapter 5.

[9]https://www.wireshark.org/

Chapter 3

Related Work

To proceed with the development of our tool we searched for current work, namely countermeasures and solutions for the Rogue Access Points (RAPs) issue. There are different types of countermeasures, for example it is important to identify passive and active detections, and classify if the solution targets more than one type of RAP. Furthermore, there are techniques that require protocol modifications or some use of specific hardware. In a passive type the detector radio listens on each channel for beacons sent periodically by an Access Point (AP). While in an active type, the detector radio transmits a probe request and listens for a probe response from an AP. The problem with active approaches is that a RAP does not reply to active probing and so passive methods are often preferred.

The solutions that we analyzed and compared, in order to understand the state of the art and to support the development of our tool are divided into two categories: Server-side (or Operator) tools and Client-side ones. It is important to clarify and understand the scope of these two types, for instance, in the Server-side type Intrusion Detection Systems (IDSs) are used, which are implemented on the APs or routers. In this type, the AP, in addition to serving the traffic, also works as an intrusion detector. In the case of Client-side type the main objective is detecting RAPs and for this it is necessary to overcome some challenges, in particular: a client has fewer privileges and has difficulties to gather WLAN traffic from the gateway. Contrasting with this, the Server-side, that is, the operator, has all the privileges to work with the network and overcome these issues. Thus, we can look at the Server-side to identify and if possible, physically locate the RAP, and the Client-type to detect and thereby avoid the connection of a client to a RAP. For these detections, a wireless or wired approach can be used. The wireless approach is used to detect the RAP only through the wireless traffic, and in the wired mode the detection of the RAP works on the wired side, where all the wireless traffic relayed by the switch or router is analyzed.

The majority of techniques to detect RAPs have focussed on Evil Twin RAPs types and consequently will also detect RAP-based deauthentication/disassociation attacks. There are techniques to detect Unauthorized APs, but detection of Compromised APs is scarce. And as revised in [5] there is no single technique to detect all RAP types.

3.1 Available 802.11 security to protect from RAPs

The security mechanisms described in the 802.11 Security subsection, 2.2.2, are not enough to protect against all RAP types. Looking back to Wired Equivalent Privacy (WEP) it has two characteristic weaknesses: the IV is frequently reused, and the secret key is not changed often enough, which makes it difficult to guarantee that key streams will, with great probability, be different. On top of this, an attacker can eavesdrop the IV being transmitted. And in a situation where two messages are encrypted using the same IV along with an original message, the encrypted messages are likely to be decrypted using the XOR operation. Thus, when the attacker gathers the key streams the key can be retrieved [49]. Because of these weaknesses, WEP will fail to protect against RAP.

WiFi Protected Access (WPA) (developed using IEEE 802.1X) was meant to be an access control method for network users. Apart from this, it also provides port security for unauthorized accesses to the network resources. In IEEE 802.1X there are three important components: the supplicant (wireless device that wants to connect to the network), the authenticator (the one responsible for providing access) and the authentication server (responsible for the authentication decisions). The most secure EAP authentication process used in WPA is the EAP Transport Layer Security (EAP-TLS), which uses public key cryptography. In this method, clients and servers use certificates, and it supports mutual authentication and dynamic key derivation [4]. Because of the difficulty in faking an authentication server, this method is capable of protecting against Evil Twin and Compromised APs. Today smartphones, tables and laptops have easy mechanisms to configure WiFi hotspots with these security methods and consequently it should be always used even in a simple task like sharing Internet connection. A drawback that can be stated from this method is that the server certificate validation is optional, and consequently the authentication server may be faked through the capture of the four-way handshake messages [7].

Apart from these security methods, sometimes universities, airports, shopping malls and hotels implement a Web-based Authentication. For example, in our school campus we have a WiFi network of this type, *wifi_eventos*. Also, all of the free WiFi networks provided by the WiFi Internet Service Providers (ISPs) are of this type (*MEO_WIFI* and *FON_ZON* in Portugal). In an authentication of this type, the user is redirected to a web portal where they will need to login with their credentials (previously registered to that service). Some authentication software that comes in this Web-based portal uses open WLAN, and some of the authentication portals are still implemented using HTTP. Furthermore, the authentication process depends on a firewall responsible for redirecting the requests to the login webpage and blocking the other requests. After the user provides the right credentials the access to the network resources is granted. The main issue regarding open WLANs and Web-based authentication resides in the fact that all the communication can be seen by anyone in the network, even without being authenticated or authorized to access the network resources. For this, an attacker can simply use tools such as Wireshark or tcpdump [1] in order to listen to the communications broadcast frames. Hence, this

[1] http://www.tcpdump.org/tcpdump_man.html

method will not protect against RAPs, and furthermore, login web pages are easy to clone which will ease the process of stealing users credentials (Phishing).

Another technology used to protect Internet connections from unsecure environments are Virtual Private Networks (VPNs). In a VPN implementation a tunnel is created over IP and in some types it uses Secure Sockets Layer (SSL) to give an extra layer to the communication process. Even so, there are several unsolved attacks to SSL and consequently the VPN connections may be compromised and aborted [41]. Thus, this method is not able to protect in some RAPs types. Nevertheless, is one of the best countermeasures to protect against all the RAPs types.

We have seen some methods that address the protection of data frames, for instance 802.11i uses 802.1X for authentication/authorization and CCMP for encryption and consequently the frames are encrypted which even if an attacker captures them will not be able to decrypt them and obtain the original data. But 802.11i only protects data frames, thus, management frames are still exposed to attacks, for example if an attacker obtains the MAC address of a client, they can send disassociation requests to the client on behalf of an AP and disconnect the client (a common technique used by attackers in some Evil Twin RAPs approaches and to capture handshakes to brute force PSKs). To address this issue, 802.11w was developed, that is, protect wireless management and control frames. Thanks to this, the deauthentication and disassociation processes are protected and so it is unfeasible to forge these frames. The issue here relates to the deployment of this standard, upgrading the firmware and hardware of millions of WLAN devices in order to make it compatible is extremely complex. Thus, the majority of WLANs still do not implement this standard.

3.2 Available RAP Detection approaches

As we could see, the previous methods are not able to protect against all the RAPs types, in particular they are not even well directed to this purpose, they are mainly security mechanisms of the 802.11. And in order to complement them there are several novel approaches that have been proposed by researchers.

In this section we explain some of the proposed methods as well as some methods available on the market.

- Protection against Evil Twin Coexistence

  As we already saw in section 2.3, the Evil Twin type deploys in the WLAN a RAP which is a clone of the legitimate AP. Table 3.1 presents some of the techniques proposed by researchers regarding the issue of Evil Twin Coexistence, i.e., a RAP that is presented simultaneously in the network with the legitimate AP.

  Taking a look in a chronological order, in [46] we have an approach based on the fingerprint model, the authors used Support Vector Machine (SVM) technique to train and validate

Technique	Year	Mode
Time Interval [29]	2014	Passive
Client-side [40]	2012	Passive
Cipher types [11]	2012	Passive
Duplicate RSSI [30]	2012	Passive
ETsniffer (v2) [54]	2012	Active
WiFihop [38]	2011	Active
DNS server two hops (v2) [25]	2011	Active
RAPiD [43]	2010	Passive
ETsniffer (v1) [47]	2010	Active
DNS server two hops (v1) [24]	2009	Active
Active behavioral [10]	2008	Active
Authentication + SVM [46]	2006	Active

Table 3.1: Evil Twin Coexistence techniques

the precise timing measurement related to the authentication procedure and with this distinguish fingerprints based on this timing.

The fingerprint active model in [10] uses network discovery and takes advantage of security tools like Nmap [2]. The method sends a request frame and waits for the response from where it will be able to determine how the devices react to fragmented or manipulated frames. Once the technique uses an active detection it can be avoided by attackers. Another drawback from this technique is the possible interference with the regular WLAN traffic.

The method described in [25] is a timing-based scheme and explores the expected two hops that happen when the user connects to the DNS server. They used Round Trip Time (RTT) to determine if an AP is legitimate or not. The detection of the RAP happens because in this case the network traffic is relayed through the DNS server of the legitimate AP. This will create a delay resulting from two hops that occur between the user and the RAP, instead of the natural one-hop process.

Taking a closer look to this method we can state that it only focus on a specific type of cause (the delay in a WLAN), where causes like interference, collisions and highly traffic-loaded WLANs may create this kind of delay. Hence, this tool can have a high false positive detection rate.

RAPiD method, described in [43] takes an indirect approach in RAP detection. It uses Local Round Trip Time (LRTT) of TCP packets to measure the delay.

The WiFiHop [38], is another method that uses an active approach, here test packets are sent to see if the RAP relays the packets on a different wireless channel. The proposed method in [30] focus in the case where the attacker device has more than one Received Signal Strength Indication (RSSI). Then the detection happens through the deviation

[2]Nmap: https://nmap.org/

between the two APs Received Signal Strength (RSS). This is one more method focused in the scenario where the RAP relays the traffic through the legitimate AP.

The next approach is [11], where the authors used the authentication and cipher types of the AP to detect RAPs. Here the information present on the authorized APs (SSID, authentication type and cipher type) is stored in a database. With the information stored they are able to sniff the beacon frames and compare its parameters with the ones stored. Thus, if they do not match, an alert will be triggered. The authors designed this method to be implemented on the client side for protection in airports or shopping malls, but as we know the WiFi hotspots in those locations use open authentication and only one cipher type, which makes this approach not practical.

The [40] method addresses the situation where two APs broadcast the same SSID and BSSID, their method checks if the IP addresses are the same and then compares the trace routes to a known machine. A drawback with this method is that it is not able to determine which AP is authorized and which one is not, otherwise would be possible to identify spoofed IPs, as the trace route of same IPs will eventually be different. Once more this is a method where only Evil Twin types are identified, the case where the RAP relays the packets to a legitimate AP.

Last but not least, the approach in [29] is based on the beacon time interval deviation. It takes in consideration the approximated time when beacon frames are sent, (approximately every 100 ms). Then, the time interval between two consecutive beacon frames is used to identify suspicious activities. Unfortunately this approach can not be used in real scenarios, because of the interference within different types of WLAN devices, 802.11b, 802.11g and 802.11n.

- Protection against Coexistence and Replacement Evil Twin

  Here we focus on approaches that solve not only the Coexistence Evil Twin RAP problem but also the Replacement Evil Twin issue.

Technique	**Year**	**Mode**
Prob Request stimuli [32]	2015	Active
Adjacent Channel [32]	2015	Active
CETAD [39]	2014	Active
Clock Skew + Temperature [33]	2014	Passive
Clock Skew [39]	2014	Passive
Radio frequency [51]	2012	Passive
Clock Skew [6] and [27]	2010	Passive
EAP SWAT [20]	2010	Active
EAP SWAT [8]	2008	Active
Radio frequency [50]	2006	Passive

Table 3.2: Coexistence and Replacement Evil Twin techniques

[50] and [51] approaches use a classification scheme that differentiates Ethernet and WLAN

TCP flows based on measurements collected passively. They use an iterative Bayesian inference algorithm to compute the fraction of wireless TCP flows and the degree of certainty that a TCP flow traverses a WLAN inside the network. Even though their algorithm converges to the unique maximum likelihood estimate of the two quantities it is not appropriate to be deployed in real time, because of the time it takes to converge.

The EAP-based authentication methods [8] and [20] are referred by its authors as Simple Wireless Authentication Technique (EAP-SWAT), and it uses the Secure Shell (SSH) trust-on-first-use approach. The trust is certified on the first connection to the AP, then the next connections to the AP are ensured to be authenticated by the coexistence of the certificates. These methods imply a change in the standard or protocol used, and would imply a change to drivers and firmware on the supplicants and APs which is not feasible.

Taking advantage from the hardware fingerprint some authors used it to detect RAPs based on the characteristics that uniquely match a WLAN device. With this in mind, the authors in [6] and [27] propose a clock skewing approach using the timestamp from beacon frames Timing Synchronization Function (TSF). This method compares the TSF generated at the AP with the inter-arrival time of the frame at the user station. And as we already stated on previous methods this will not work properly because of the delays in WLAN medium. Then in [39] and [33] the time skew method uses the TSF in order to differentiate hardware and software based APs. The method in [39] has a combination of ISP-based detection and timing-based detection. In this service the ISP receives a set of IP addresses provided by Internet Assigned Numbers Authority (IANA) (the authority in charge of managing global IP addresses), so they can provide a unique global IP address to clients using this service. The information in each global IP address is publicly available on the web. Hence, in this approach a request is sent to these servers, and with the information response about the requested AP they managed to distinguish Evil Twin APs from legitimate ones. In a ISP-based detection method it is not possible to identify Evil Twin APs sharing the same Internet connection as the legitimate AP, this happens because the Evil Twin AP uses the same Internet service.

- Unauthorized AP Countermeasures

  As previously explained unauthorized APs are connected to the wired side of the network without the permission of the administrator. To prevent such RAPs, in [53] the authors use a verifier placed in the wired part of the network which sends test packets to the wireless side of the network. Consequently, the APs that relay those packets are detected as RAPs and the administrator will be able to locate them. This verifier can monitor the active users on the wired side and with the test packets that it sends it is able to differentiate APs from users, because in the case of an AP it will receive the packet and forward it to the wireless side.

  One of the first approaches is the one proposed in [13] where wireless frames are sniffed and the packets are sent to an analyzing engine which contains a list of legitimate APs (white-list), in fact this approach is the base for further approaches and similar mechanisms

that are present in some known wireless routers [35] and [14]. Once this analyzing engine gets the packet with the information about the new AP it will check it with the authorized list in order to identify suspicious nodes. This approach can be bypassed with a spoofed AP, which will match an authorized AP from the white-list and become a RAP.

The countermeasure presented in [3] has three components: a filtering engine, anomaly detection sensors, and shadow honeypot code. The first component is the first layer of protection, here it uses an authentication list to filter unauthorized APs. Therefore, any traffic sent from a source MAC address that is not on that list will not pass and is considered a RAP. When some traffic passes the first detection mechanism (an authenticated user) it will arrive to the second layer of protection, the anomaly detection sensors. Here the characteristics of the packets will be examined and sent to the inner layer, shadow honeypot. This is the final stage and works in similar way as an anti-virus system, it uses signatures of worms and attacks and compares them with the network trace. As drawbacks, for instance the authors did not take care of the MAC address spoof case, where a spoofed MAC address of a legitimate AP presented on the white-list will pass to the other stages.

- Deauthentication/Disassociation Countermeasures

  As analyzed in 3.1 the security standard used in 802.11 is the IEEE 802.11i [21] which provides data confidentiality, integrity, and mutual authentication in the Media Access Control (MAC) layer. And as seen before, if IEEE 802.11w is not implemented the management frames will be vulnerable to deauthentication and disassociation attacks. Hence, taking advantage of this vulnerability, attackers will use it to their profit, for example they send these packets to disconnect a client from the legitimate AP and then, because of the traditional scheme of increasing the signal strength power of RAPs, the clients will get connected to the RAP.

  Apart from the 802.11w standard which may not be present in a network, there are some approaches which aim to detect deauthentication and disassociation attacks.

  One of the first approaches proposed in the research community was the sequence number, described in [17, 22, 52], where they tried to detect MAC address spoofing coming from deauthentication attacks. The base theory behind this method is the assumption that a wireless user generates a sequence of numbers which will make it difficult for an attacker to manipulate the sequence to get the same match numbers. Eventually this will create a gap between the sequence number from the legitimate AP and the attacker (RAP) and identify a spoofing situation. However, this kind of detection can be traversed by injecting deauthentication frames after the sent frames from a specific user or AP.

  Another kind of approach are the RSSI based ones [12, 45], here the authors try to differentiate WLAN devices based on their location. For this they use the RSSI from the devices, taking in consideration that a wireless device does not increase or decrease its transmission power on a regular basis; it might happen because of multi-path and absorption effects but it is not a general rule or commonly of a great influence. Consequently, obvious changes in RSSI from the same MAC address is considered as an indicator of spoofing and

will identify a RAP. An attacker may overcome this method by deploying the RAP with the same signal of the legitimate AP, i.e., it would increase or decrease the signal strength in order to mimic it. Also, if the attacker is close to the legitimate AP it would be difficult to identify the RAP.

- Protection against multiple RAP types

  Following a chronological order, the authors of [28] present a Distributed Wireless Security Auditor (DWSA) tool which works both on Linux and Windows-based systems providing continuous wireless assessments. This tool also uses the trusted wireless clients as distributed sensors in order to find anomalies. To detect and locate RAPs it uses 3D trilateration [16]. With DWSA it is possible to detect Evil Twin and Unauthorized RAPs.

  In [36] it is possible to detect Evil Twin, Unauthorized and Compromised RAPs types. For such, this hybrid method combines the distributed wireless media surveillance and centralized wired end socket level traffic fingerprinting. A disadvantage of this approach is the fact that it must be packaged with the router or the switch resulting in overloads when dividing the jobs, i.e., serving wireless and running the IDS.

  The method proposed in [48] is another type which strongly depends on APs MAC addresses and therefore is vulnerable for spoofing attacks. The method itself works in a multi-agent form and can detect Evil Twin and Unauthorized RAPs, it is composed by a master agent and a slave. The master regulates the authorization processes in the WLAN, while the slave answers to the master agent identifying active APs in the WLAN. For this, the slave is connected to an AP and gathers its relevant information (SSID, BSSID, vendor name and channel number). When the master agent receives this information it compares it with an authorized list (same as white-lists).

  In some companies wireless sniffing solutions are used where they deploy sensors across the network perimeter in order to gather physical and data link layer information.

To summarize the described techniques, table 3.3 encompasses and organizes some of the previously explained detection techniques for each type of RAPs.

Coexistence	Coexistence & Replacement	Unauthorized	Multiple Attack Types
DNS Server two hops	Radio Frequency	Shadow Honeypot	RAP
WiFiHop	EAP-SWAT	Unauthorized Approach	Elimination
Duplicate RSSI	Clock Skew	RTT Approach	Multi-agent
Cipher types	CETAD	Inter-packet Spacing	DWSA
Client-side	Adjacent Channel		
Time Interval			

Table 3.3: Detection techniques

And in order to compare some of the techniques we also present the summary of the described weaknesses and strengths in table 3.4.

Technique	Strengths	Weaknesses
Non-automated wireless solutions	Passive	Easily evaded
	Minimum infrastructure required	Require considerable effort and time
		Sensors must go through all channels
Wired-active probing	Do not rely on wireless frequency	Active
		RAP may not respond to packets
		Only rely on port switch policies
Hybrid	Passive	Can be evaded from the wired side
	Detect more types of RAPs	
Time approaches	Passive	Need samples (wireless and wired)
	Do not rely on wireless frequency	Assumes wired connection faster than wireless
		Can be evaded by insiders
Identification Approaches	Passive	Can be evaded by insiders
	Do not rely on wireless frequency	
	Do not need samples (wireless and wired)	

Table 3.4: Strengths and weaknesses of existing techniques

Chapter 4

Architecture

In the previous chapters we have seen that there are several proposals of techniques to detect Rogue Access Points (RAPs). We could see that in each countermeasure only one technique is mainly used, which only addresses a specific type of RAP. Moreover, all of them lack a model to estimate the level of certainty of the detected RAP. This is because in some cases the detection that results from the applied method can suffer interference and even be a false positive. Apart from this, we also studied techniques to perform the RAPs deployment and WiFi attacks, as seen in 2.4. With this survey we were able to verify that there are many free available tools to perform the attacks contrasting with the lack of tools to detect them.

The framework that we developed combines the most effective techniques from the state of the art, and complements it with some new approaches in order to accomplish the closest results in the detection process.

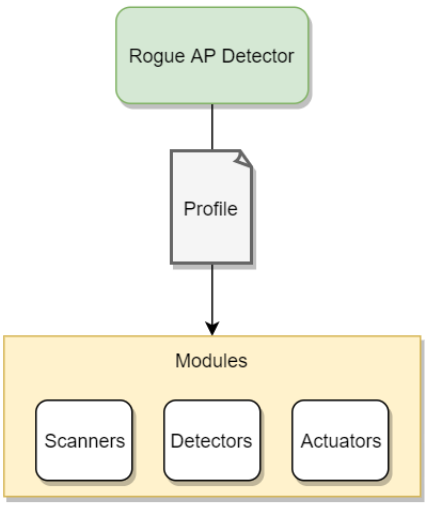


Figure 4.1: Framework architecture

This framework is a RAP Detector that works both in a passive and active mode, and can be used by any type of user, i.e., it can be configured with a profile of a network administrator

with knowledge about the network, or as a simple user with no knowledge of the network.

The main architecture is composed by the main application, which we call the Rogue AP Detector, and it is connected to a set of modules: Scanners, Detectors and Actuators.

In figure 4.1 we present a diagram of the proposed architecture where we follow a single, stationary and centralized approach for the detection.

4.1 Profiles

The application is designed to use configuration files, which we call profiles. Hence, as a starting point before running it, a file should be configured in order to define a matching profile, which can be configured accordingly to the user knowledge about the network.

Lets say that a user wants to create a profile for his/her home network, in this case the user would give a name for that profile, for example: `home-network`, and then add the desired information details: authorized Access Points (APs), expected RSSI or limits, IP ranges and expected traceroute for specific addresses. All of the accepted parameters are explained in the implementation section.

Now lets consider that this user goes to school and wants to run the detector. In this case they are in a different circumstance, now the user has limited knowledge about the network, and so the profile has to be different. Hence, a new profile should be configured, for example: `school-network` with a more limited configuration, where only information like AP name, BSSID and encryption are configured.

Apart from these configuration profiles the application also has a built in profile for open WiFis, this profile contains information about known open WiFis and can easily be updated by the user.

In any case the user configured configuration profiles are optional and the application will run without them, but as expected this will generate much less accurate results in the detection process. Since some of the detection heuristics will not be used.

4.2 Modules

In this section we explain the type of modules present in this framework along side the interoperability between them. We refer to this as type of modules since in the development perspective the actual modules are encapsulated in these type of modules.

The main application is connected with all the modules. The Scanners have methods implemented to monitor the network in order to find nearby APs, during the running time the application will always be scanning, using these modules, which work in a passive mode. The

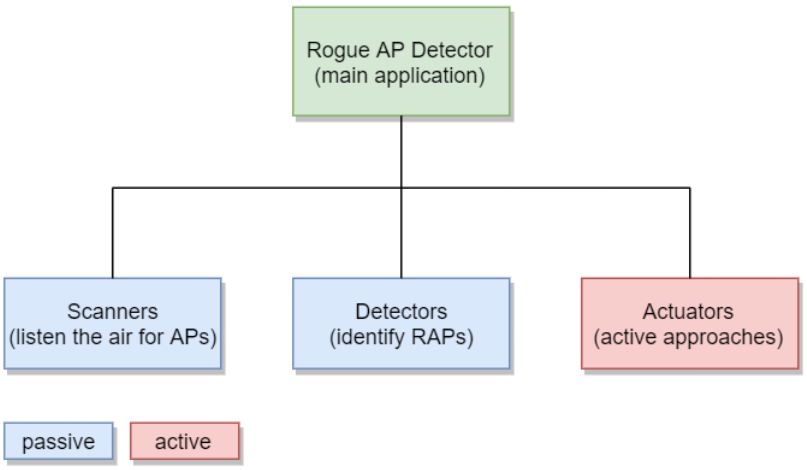


Figure 4.2: Framework Modules

information provided by them will interact with the Detectors, where a set of detection heuristics will sweep the scanned APs and when a specific AP triggers some suspicious level it will interact with Actuators in order to perform some active techniques. Some of the heuristics used by the Detectors make use of information which is stored in configuration files (profiles).

4.2.1 Scanners

The function of this module is to scan the WiFi channels in order to discover the nearby APs. This is a core function in the framework, as to detect something we first need to have something. In order to accomplish this we have configured two different modules: one that uses the `iwlist` [1] Linux command and another that uses the `scapy` [2] packet manipulation framework.

This module gathers all the required information about the nearby APs and sends it to the Detectors module where this information is analyzed.

4.2.2 Detectors

As the name indicates, Detector modules are responsible for identifying RAPs. They sweep the output of the Scanners module and apply to it several heuristics in order to alert the user about near by RAPs and calculate a degree of confidence in the *rogueness* of the AP.

One type of heuristic present in this module is the classic approach of whitelists, where we use the information from the configuration profile regarding SSIDs and BSSIDs.

In this specific case if a scanned AP has an SSID from the authorized list we check if a matching BSSID is on that list too (listing 4.1).

[1] https://linux.die.net/man/8/iwlist
[2] https://github.com/secdev/scapy

```
if ssid in whitelist_ssid and bssid not in whitelist_bssid:
  alert_user()
```

Listing 4.1: Whitelist heuristic

Since this is likely to be bypassed by spoofing the BSSID, we also compare the Encryption type used by the AP, which as seen in the Evil Twin type of RAPs it is often not used in the deployment of this type of RAPs (listing 4.2).

```
if current_ap[encryption] != whitelist_ap[encryption]:
  alert_user()
```

Listing 4.2: Encryption heuristic

If any of these two tests fail it is guaranteed that the analyzed AP is a *rogue* one. Consequently, the alert sent to the user will inform about the SSID and BSSID of the detected RAP and the name of the probable type of attack.

Considering the case where none of the latter tests failed, another detection heuristic is applied to the scanned APs. In this case we test the variation in the signal strength (RSS). As we described, this version of the tool is designed to be stationary, and considering that normal APs are also stationary we try to estimate a baseline for the signal strength and use this information to improve the detection process. Nevertheless, these baselines need to be very well tuned and the generated result will just prompt the user for activating or not a further analysis from the Actuators heuristics. The algorithm for this heuristic is defined as in listing 4.3.

```
if captured_RSSI > auth_rssi+baseline or captured_RSSI < auth_rssi-baseline:
  prompt_user()
```

Listing 4.3: RSSI heuristic

When the *captured_RSSI* activates the RSSI heuristic, the method defined in listing 4.4 will be called.

```
prompt_user():
  answer = raw_input("Strange RSSI! Associate to %s AP?" % suspect_ap)
  activate_associate_module if (answer == "yes") else return

activate_associate_module():
  Process(module.actuators.associate(suspect_ap))
```

Listing 4.4: Call Active Detectors module

The described heuristics presuppose that a configuration profile is loaded at the application run, otherwise they will not be applied. In such case, a *no knowledge* method is always running

by the application, where simple analysis are conducted to the scanned APs. A simple case is looking for APs with same SSID and different encryption.

Another heuristic performed by the application is a free [3] WiFis authenticity validation. This takes advantage from the BSSID pattern of the studied free WiFis. As an example, the free WiFi provided by the ISP NOS (Portuguese media company), is generated from the router of a personal network and consequently the BSSID generated for the free network follows a specific pattern, which in this particular case is the increment of the last byte in one unit. We use this information to analyze the BSSIDs of the scanned free WiFis, and together with the information from free WiFis configuration file we give an authenticity validation for the target AP.

This tool also has a *blacklist* passive heuristic for RAPs generated from known WiFi attacking tools. We configured two methods to detect PineAP [4] RAPs and APs where the BSSID manufacturer is Alfa.

```
def blacklist_detection(scanned_ap):
  alfa_brand = "Alfa"
  pineAP_default_bssid = ":13:37:"
  if (pineAP_default_bssid in scanned_ap['bssid']):
    alert_user()
  elif (alfa_brand in scanned_ap['manufacturer']):
    alert_user()
```

Listing 4.5: Blacklist heuristic

In case of the PineAP the default BSSID always generate values containing the *13:37* pattern, hence this heuristic can identify some RAPs configured by inexperienced attackers. The Alfa condition is supported by the fact that these WiFi cards are mainly used for WiFi attacks and therefore, create RAPs.

4.2.3 Actuators

The Actuators are composed by a set of active detectors, a honeypot and a defensive mechanism. The active detectors will compare the information from the configuration profile and perform tests for each of the entries. For example, the first stage is trying to associate to the AP and further on, apply a series of methods to understand if the configurations are the expected ones.

These active detectors are performed when we have knowledge about the scanned AP, i.e., the scanned AP matches an AP from our list of authorized APs or in case of Open networks that fails the authenticity validation.

This module is separated from the Detectors because we want to segregate passive approaches from active ones. And in this case the detectors under the Actuators type of module perform

[3] WiFi access offered by ISPs
[4] https://www.wifipineapple.com/pages/software

active operations over a target AP, specifically, they associate to the AP and then run the heuristics.

```
def call_active_methods(ssid, bssid, iface):

    internal_IP = active_detectors.get_internal_IP(iface)
    external_IP = active_detectors.get_external_IP()
    print("External IP: %s" % external_IP)
    isp = active_detectors.get_ISP(external_IP)
    print("ISP: %s" % isp)

    if internal_IP not in configuration_profile or external_IP not in
        configuration_profile:
      alert_user("Ip not expected")

    traceroute_internal = active_detectors.traceroute(hostname_int, iface)
    traceroute_external = traceroute(hostname_ext, iface)

    if traceroute_internal not in configuration_profile or traceroute_external
        not in configuration_profile:
      alert_user("Not expected traceroute")

    AP_fingerprint = get_AP_fingerprint()

    if AP_fingerprint not in configuration_profile:
      alert_user("AP wrong fingerprint")
```

Listing 4.6: Active Detectors heuristics

These heuristics will display to the user the obtained results and, if such information is present in the profile configuration files, the correspondent difference will be printed to the user. In detail, if the analyzed AP has Open authentication or is known to that users' device, the *associate* heuristic can be used, and it is defined in pseudo-code 4.7.

```
def associateTo(ssid, pwd, iface):
  try:
    if (pwd):
      nmcli_association(ssid, bssid, pwd, iface)
      call_active_methods(ssid, bssid, iface)
    else:
      nmcli_association(ssid, bssid, iface)
      call_active_methods(ssid, bssid, iface)
  except:
    print("Association Error")
    return
```

Listing 4.7: Association process

The *External IP* and *Internal IP* types will compare the obtained IPs with the configuration profile range of IPs.

The ISP method uses an external API to retrieve the ISP name of the attributed external IP.

The *Traceroute* is the comparison between the original traceroute for a specific address (configured in the profiles) and the actual traceroute for the same address.

Finally, the *Fingerprint* identifies if the OS fingerprint and services of the tested AP corresponds to the ones of an authorized one. As an example, it checks if the gateway of the target AP has the keyword router, which in case of a software based AP this information will not be displayed and corroborate the fact that it is an unauthorized AP. This method can also be used to retrieve the clock-skew of the associated AP and compare it with the one from the configuration file.

The Actuators set of modules is also composed of an hybrid mechanism to detect Karma attacks. In passive mode, we wait for probe responses from the same AP to different probe requests. In active mode, we send probe requests for generated fictitious SSIDs and if an AP sends responses for the different SSIDs a Karma attack is spotted. Again, an alert will be sent to the user, identifying the RAP performing the Karma attack (listing 4.8).

```
phishing_karma = {}
def spot_karma():
   if scanned_ap['mac'] in phishing_karma:
      karmas_len = len(phishing_karma.values())
      karmas=0
      for i in range(karmas_len):
        if SSID in phishing_karma.values()[i]:
          break
        if SSID not in phishing_karma.values()[i]:
          karmas+=1
          break
        if (karmas==karmas_len):
          phishing_karma[scanned_ap['mac']].add(scanned_ap['essid'])
          alert_user("Karma attack detected!")
          break
   else:
      phishing_karma[scanned_ap['mac']] = set([scanned_ap['essid']])
```

Listing 4.8: Detection of Karma attacks

In cases like this, where it is pretty obvious that RAPs are being created with the Karma attack, a defensive mechanism can be used. In this mechanism deauthentication packets are sent to the clients associated with the detected RAP. In any case, this action should be executed with awareness and only when the previous conditions are met, since deauthenticating is not a legal procedure.

Last we have the honeypot module, which is a RAP created and controlled by the application and has similar behavior has a traditional honeypot, once it is started it becomes an authorized AP for the application which in case of an attacker targeting this AP it will immediately trigger an alert to the user. Apart from this, this module was also used to perform tests for some proof of concept scenarios.

Logs

During the development process of this framework it was clear that we needed to store the information about each detection performed: such as starting and ending date and time informations, the results for the detections with the respective profile being used. Apart from this, since some unexpected errors happened during the development, we also store these errors in log messages in order to help in the improvement of the application development and to understand wrong behaviors.

Thus, all the modules: Scanners, Detectors and Actuators, generates log messages in order to monitor and register the described events.

In this chapter we explained the framework architecture, which is composed by a main application that works using a set of modules. These modules have different heuristics that in a first step, scans for near by APs and then apply detection mechanisms on them.

Chapter 5

Development

In this chapter we address the details of the implementation process, specifically we describe topics such as data structures, specific details about the heuristic methods and approaches during the development phase of this project.

For each type of modules we explain the specific methods used, for example we describe what we used to sense the air for APs for the matching scanning module. We also address the implementation details for executing more than one module at the same time. An overview of the Beacon frames using the Wireshark tool is also presented; relevant point to understand the differences between benign APs and *rogue* ones, highlighted in some cases by the difference between the different parameters from a hardware AP and software-based ones.

The code of this project is maintained in a Github [1] repository; it is open-source, so it can be used and contributed by anyone.

5.1 Overview

The main application of this project is developed in Python, specifically version 2.7 [2], this choice is due to the easiness in development provided by the language, which comes from its similarity with pseudo-code that allows a greater dedication in the concept and designing of algorithms for the application logic and less time in programming issues. Apart from this the overall performance of Python is enough for what we want to achieve with this tool.

The growing community and excellent documentation of Python are also relevant factors in its choice, together with a broad amount of modules and frameworks that can simply be incorporated in our program. One example is the Flask Web Framework [3] that can simply transform our command line application in a good looking web application without the need of much work. One of the purposed further work in this project is the implementation of a graphical

[1] https://github.com/anotherik/RogueAP-Detector
[2] Python 2.7 Doc: https://docs.python.org/2/index.html
[3] https://en.wikipedia.org/wiki/Flask_(web_framework)

user interface and this framework is ideal for it.

The developed application is designed to work under Linux systems, most of its core methods operates directly with specific operating system commands, and it should run without any requirements under a Kali Linux distribution. Apart from this, setting up a wireless interface to monitor mode under a Windows system does not come out of the box and so it will be part of a further work.

In order to understand how the application is structured we present a simplified version of the project tree. Figure 5.1 can be mapped with the modules from the architecture chapter. Thus: the main application is the *rogueAP_detector.py*, and the type of modules described in the architecture: Scanners, Detectors and Actuators are in the matching folder, inside the modules folders.

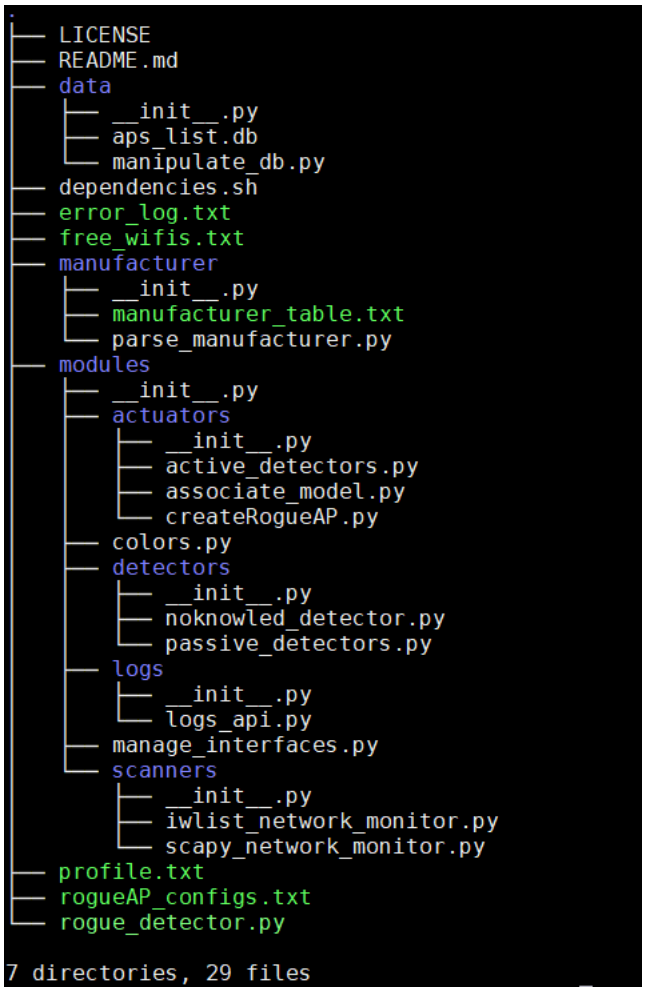


Figure 5.1: Application structure

In this structure we also have the data folder which contains the code related to a future module for database manipulation and storage. The manufacturer folder, is responsible for mapping the MAC address (BSSID) of the scanned AP with the correspondent manufacturer name.

We also have a standalone file, `dependencies.sh`, which is a shell script that installs all the required dependencies of the application based on the specific Linux distribution. It is also though as a further work to create a `requirements.txt` [4] file in order to install all the dependencies with the pip [5] installer.

Main application - *rogueAP_detector.py*:

The main application is configured to set the desired parameters to run the detection tool, in order to do that it has a help menu that explains all the necessary procedures to run and configure it.

The help menu explains how to set the right parameters, namely:

- -i "interface" - set the interface to monitor the network
- -im "interface" - interface to used in monitor mode
- -p "profile" - set the profile to be used
- -s "scanning type" - select the desired scanning type (`iwlist` [6] or `scapy` [7])
- -h "hive mode" - activate honeypot mode (creates a AP)
- -d "deauth mode" - deauthenticates users from target AP
- -a "active mode" - activates random probe requests

The profile parameter will load a set of configurations from a file, that will then be used to help in the detection process.

When the interface, profile and scanning mode are selected the main application will use the *multiprocessing* module to start the respective scanning type. It is inside the scanning modules that the Detectors module heuristics are activated.

The *multiprocessing* module is a package that supports spawning processes using an API similar to the *threading* module. Its advantages over the *threading* module is that it offers both local and remote concurrency, and effectively side-steps the Global Interpreter Lock by using subprocesses instead of threads [8].

[4] https://pip.pypa.io/en/stable/user_guide/#installing-packages
[5] https://en.wikipedia.org/wiki/Pip_(package_manager)
[6] https://linux.die.net/man/8/iwlist
[7] https://github.com/secdev/scapy
[8] https://docs.python.org/2/library/multiprocessing.html

5.2 Scanners

In the Scanners we have implemented two different modules: one that uses the `iwlist` Linux command and another that uses the `scapy` packet manipulation framework.

Either of the selected scanners modules are started has a new process using the *multiprocessing* python module. Their function is *sniffing* the air, i.e., scanning the WiFi channels, in order to discover the nearby APs. When an AP is collected it is stored in a *Array* of data in order to avoid *printing* the same AP more than once. This is because, as already explained, the Beacon frames are transmitted by the APs every 100 *ms* and if we did not perform this filtering, repeated APs would be *printed* to the user's console by the main application.

The generic algorithm that we use for such is presented in listing 5.1.

```
scanned_aps = []
while scanning:
   current_ap = ssid+" "+bssid+" "+channel+" "+encryption+" "+signal
   if current_ap not in scanned_aps:
      scanned_aps.append(ssid+" "+bssid+" "+channel+" "+encryption+" "+signal)
```

Listing 5.1: Scanning APs algorithm

In order to test this algorithm let us consider a scenario where an attacker deploys a *rogue* AP with the same SSID, BSSID, channel and encryption algorithm. This algorithm will most probably print the two APs because of the differences in the signal parameter which hardly would be the same. Apart from this, a drawback needs to be stated, which is the fact that interferences are constantly happening in the air medium, and this will cause to print the same AP several times. To address this we created a baseline for the signal parameter, that reduces the change of printing the same AP because of interferences in the air. Nevertheless, this is susceptible to interferences and needs to be tuned in case the tool is moved to another location (listing 5.2 and 5.3).

```
scanned_aps = []
while scanning:
   current_ap = ssid+" "+bssid+" "+channel+" "+encryption+" "+signal
   if filter_ap(current_ap):
     scanned_aps.append(ssid+" "+bssid+" "+channel+" "+encryption+" "+signal)
```

Listing 5.2: Scanning APs tuned algorithm

```
def filter_ap(current_ap):
  for ap in scanned_aps:
    if (ap["ssid"] == current_ap["ssid"] and ap["bssid"] == current_ap["bssid"]
        and ap["ch"] == current_ap["ch"] and ap["enc"] == current_ap["enc"] and
       (abs(current_ap["signal"]) <= abs(ap["signal"])+20 and
```

```
        abs(current_ap["signal"]) >= abs(ap["signal"])-20 ):
        return False
    return True
```

Listing 5.3: Filter AP algorithm

In listing 5.3 is shown such tunning, specifically it is configured to do not print the AP if the RSSI is between the defined threshold, which where has an upper boundary of more 20 dBm and a lower boundary of less 20 dBm.

Hence, this is the way we track the APs during the scanning process to print them to the user's console.

During the development of these two scanning modules we consider the `iwlist` program preferable over `scapy`. This is due to factors such as difficulties in understanding and parsing the encryption retrieved with `scapy` from the Beacon frames, and its lack of automatic channel hopping during the scan. More details about the Beacon frames parameters and how they can be parsed with `scapy` are explained in the Detectors section, since they are required to detect RAPs.

Next, we explain how the two scanning modules work and how we use them for our needs:

`iwlist`: From the *iwlist* Linux manual entry we can understand that it is responsible for displaying information about a wireless network interface. Its main argument is used to select a category of information and display detailed information about it. In our case the selected parameter is the ***scan****[ing]*.

Thus, using the following command: ***iwlist iface scan***, where the *iface* is substituted by the desired interface, we retrieve a list of APs in range with its respective informations (Beacon frame parameters), that may vary depending on what the WiFi card supports. It is important to notice that the scanning process is a privileged operation, and normal users will only read left-over scans (not permanently searching for Beacon frames). For such, we recommend running our tool with root privileges.

Figure 5.2 shows a typical output from a *iwlist* scan:

```
[anotherik@manjaro ~]$ sudo iwlist wlp0s20u2 scan
wlp0s20u2  Scan completed :
          Cell 01 - Address: 00:05:CA:9E:37:18
                    Channel:11
                    Frequency:2.462 GHz (Channel 11)
                    Quality=49/70  Signal level=-61 dBm
                    Encryption key:on
                    ESSID:"ZON-3710"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 9 Mb/s
                              18 Mb/s; 36 Mb/s; 54 Mb/s
                    Bit Rates:6 Mb/s; 12 Mb/s; 24 Mb/s; 48 Mb/s
                    Mode:Master
                    Extra:tsf=0000000a8a378dbb
                    Extra: Last beacon: 196ms ago
                    IE: Unknown: 00085A4F4E2D33373130
                    IE: Unknown: 010882848B961224486C
                    IE: Unknown: 03010B
                    IE: Unknown: 2A0100
                    IE: Unknown: 32048C98B060
                    IE: Unknown: 2D1A8C0117FFFF000000000000000000000000000000000000000000
                    IE: Unknown: 3D160B0004000000000000000000000000000000000000
                    IE: Unknown: 3E0100
                    IE: WPA Version 1
                        Group Cipher : TKIP
                        Pairwise Ciphers (2) : TKIP CCMP
                        Authentication Suites (1) : PSK
                    IE: IEEE 802.11i/WPA2 Version 1
                        Group Cipher : TKIP
                        Pairwise Ciphers (2) : TKIP CCMP
                        Authentication Suites (1) : PSK
                    IE: Unknown: DD180050F2020101800003A4000027A4000042435E0062322F00
                    IE: Unknown: 0B05040012127A
                    IE: Unknown: 7F0101
                    IE: Unknown: DD07000C4300000000
                    IE: Unknown: DD1E00904C338C0117FFFF000000000000000000000000000000000000000000
                    IE: Unknown: DD1A00904C340B0004000000000000000000000000000000000000
          Cell 02 - Address: 00:05:CA:9E:37:19
                    Channel:11
                    Frequency:2.462 GHz (Channel 11)
                    Quality=50/70  Signal level=-60 dBm
                    Encryption key:off
                    ESSID:"NOS_WIFI_Fon"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 9 Mb/s
                              18 Mb/s; 36 Mb/s; 54 Mb/s
                    Bit Rates:6 Mb/s; 12 Mb/s; 24 Mb/s; 48 Mb/s
```

Figure 5.2: iwlist scan output

From the produced output we can see that each AP corresponds to the "*Cell XY*" section, and then is presented the rest of the Beacon frame information. Over this output we apply a parser to retrieve the informations that we want to use to display to the user and to apply the detection heuristics. The parsing process uses regular expressions in order to retrieve the desired parameters.

Then, the final form presented to the user is the following:

Date	AP Name	BSSID	CH	Brand	Signal	Quality	Encryption	Cipher	Pairwise	Authentication
19:18:16 08/25/17	NOS_WIFI_Fon	00:05:CA:9E:37:19	11	HitronTe	-63	47/70	Open			
19:18:16 08/25/17	ZON-3710	00:05:CA:9E:37:18	11	HitronTe	-63	47/70	WPA2 Version 1	TKIP	TKIP CCMP	PSK

Figure 5.3: Parsed iwlist scan output

To understand some of the *IE: Unknown* parameters from the *iwlist* scan, 5.2, we performed a scan using the Wireshark tool (using the interface in monitor mode) in order to match the unknown types with the right name. It is easily understood that the values after the *IE: Unknown* are the hexadecimal representation of output parameter.

As an example, the first vale of *IE: Unknown:* 00085A4F4E2D33373130 is translated to *ZON-3710* and as we can see from figure 5.4 that is matched with the SSID Tagged parameter. If we follow this process we can retrieve all the unknown values, which basically will match the Tagged parameters of the Wireshark capture. Then, we can store the ones that might be useful for the detection process.

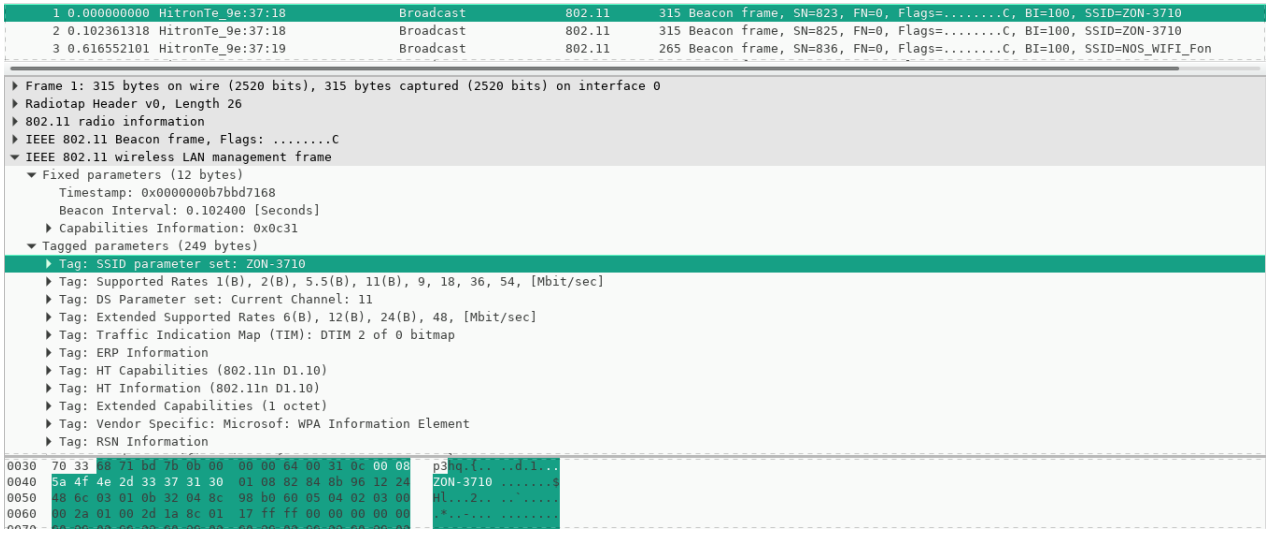


Figure 5.4: Wireshark scan

During the development process and the research to understand how these tools work, we found that the `iwlist` command is considered as deprecated, and it is recommended to use the `iw` command instead. With some tests using the `iw` command we could see that it produces much more detailed information about the AP, and it would be a possible improvement as future work.

`scapy`: The `scapy` is a packet manipulation tool written in Python, and it is able to forge or decode packets of a wide number of protocols. For our tool we are interested in the *Sniffing* process of `scapy`, which is described as: the act of intercepting and logging the packets which flow across the network.

Part of the scanning process using `scapy` is shown in listing 5.4:

```
1  def PacketHandler(pkt):
2      if (pkt.haslayer(Dot11Beacon) or pkt.haslayer(Dot11ProbeResp)):
3        ssid = pkt[Dot11].info
4        bssid = pkt[Dot11].addr3
5        channel = int(ord(pkt[Dot11Elt:3].info))
6        capability = pkt.sprintf("{Dot11Beacon:%Dot11Beacon.cap%}\
7                {Dot11ProbeResp:%Dot11ProbeResp.cap%}")
8        extra = pkt.notdecoded
9        rssi = -(256-ord(extra[-4:-3]))
10     (...)
11 sniff(iface = monitor_interface , prn = PacketHandler)
```

Listing 5.4: Scapy scan snippet

Line 2 contains the most relevant part, where we see if the scanned packet, 802.11 frames, is a Beacon or a Probe Response. The Probe Response packet is used to capture hidden APs.

5.3 Detectors

The Detectors modules are activated inside the Scanners modules, where the heuristics described in 4.2.2 represent the methods of such modules. Hence, this module is initiated in the Scanners and does not require starting a new process, since the scanners only need to use the methods of this module over the current AP being scanned.

As already explained these are passive detectors and the way they are used is as follows:

```
1  scanned_aps = []
2  while scanning:
3     (...)
4      if current_ap not in scanned_aps:
5
6        if (profile):
7          passive_detectors.authorized_aps(current_ap, profile)
8
9        if (active_probing):
10         passive_detectors.send_ProbeRequests(interface_monitor)
11
12       passive_detectors.free_WiFis_detect(current_ap, captured_aps)
13       passive_detectors.spot_karma(current_ap)
14       passive_detectors.spot_PineAP(current_ap)
15       passive_detectors.spot_Alfa(current_ap)
16
17       noknowledge_detector.suspicious_behaviours(current_ap, captured_aps)
```

Listing 5.5: Passive RAP algorithm detection

Figure 5.5 shows the passive detectors used by the application, recapping, the detector from line 7 will call whitelist (listing 4.1), encryption (listing 4.2) and RSSI (listing 4.3) heuristics. In case of the RSSI heuristic, if the captured RSS is above or lower the defined baseline the user will be prompted for further analysis, that uses the heuristics from the active detectors (listing 4.6) from Actuators module.

Then, from line 12 to 17 we have the rest of the passive detectors: validate free WiFis authenticity, identify Karma Attacks, identify RAPs produced by PineAPs and Alfa interfaces.

5.4 Actuators

Some of the Actuators methods require an Association process with the AP. In the way this tool is implemented, we create a new process for the Association and for the other heuristics that are performed. With this approach, we can guarantee that the Scanning process, and consequently the passive detectors are still running in the other process. This happens because the association process uses another interface for the association with the AP, like this, the APs scanning keeps running in the background using the scanning interface.

When the association is completed, the main algorithm of the active detector heuristics ran. The methods implement the heuristics explained in 4.6, namely:

- **get_internal_ip**: parsing the *ifconfig* Linux command to retrieve the internal IP address attributed after the association process and then comparing it with a range of authorized internal ips from the configuration profile
- **get_external_ip**: using the *dig* command to query *opendns* for the machine external ip address after the association process and then comparing it with a range of authorized external ips from the configuration profile
- **find_whois_ip**: using *ip-api.com* API to retrieve Organization and ISP information from the received external ip address (get_external_ip) and comparing it with authorized matching values from the configuration profile
- **traceroute_internal** and **traceroute_external**: using *traceroute* command to count the number of hops for a specific address and compare it with the expected number of hops from the configuration profile
- **get_AP_fingerprint**: using *nmap* security scanner with default script, and OS and version detection. The *nmap* scan first performs a port scan on the target node, probing for known open and closed ports, sending TCP, UDP and ICMP requests. Once the port scan is completed it performs the OS detection. The scan also does a service detection, where for each of the target ports will identify the probable service. Then, with the output we look for configurations and keywords specific from APs and use them to identify possible RAPs.

Apart from this, the Actuators module also has a Karma detector, explained in section 4.2.3. Considering that Karma attacks work by having a *rogue* AP that responds to all probe requests, what we have done in this method is actively probing for random APs and look for the AP that responds to them, if we find the same BSSID responding to these different requests we have found an RAP. We could use this algorithm in a passive format, but in such case it would be more difficult to detect the RAP because it would require wireless clients actively probing instead of our tool.

```
while True:
  ssid = gen_ssid()
  radioTapHeader = RadioTap()
  dot11Header = Dot11(addr1 = broadcast_addr, addr2 = bssid, addr3 = bssid)
  dot11ProbeReq = Dot11ProbeReq()
  dot11Elt = Dot11Elt(ID=0, info=ssid)

  pkt = radioTapHeader / dot11Header / dot11ProbeReq / dot11Elt
  sendp(pkt, iface=interface)
```

Listing 5.6: Random SSID Prob requests

In order to improve the Karma detection it is possible to run our tool with the active mode option on, where it will start sending probe requests for random generated SSIDs. To accomplish this, we use `scapy` to craft a probe request packet, listing 5.6.

Using this active approach we immediately guarantee the detection of a Karma attack, an consequently the source that is generating RAPs.

```
def gen_ssid():
  N = 6
  SSID = ''.join(random.choice(string.ascii_uppercase + string.digits) for _ in
      range(N))
  return SSID
```

Listing 5.7: Generate random SSIDs

The code from listing 5.7 is responsible for generating random SSIDs that will be used in the crafting of the probe request packet. Any time this method is called it will generate a random alphanumeric string with a length of six characters, this value will then be attached to the SSID informational part of the the probe request packet.

Finally, the defense mechanism that we configured also uses the `scapy` module to craft Deauthentication packets and send them to the suspect AP, i.e., the RAP. In this approach we send the Deauthentication packets to all the clients (broadcast_addr) connected to the target AP, but it can simply be modified to target only a specific client.

```
  radioTapHeader = RadioTap()
  dot11Header = Dot11(addr1=broadcast_addr,addr2=target_ap,addr3=target_ap)
  dot11Deauth = Dot11Deauth(reason=7)

  pkt = radioTapHeader / dot11Header / dot11Deauth
  sendp(pkt, iface = interface)
```

Listing 5.8: Deauthentication request

The Deauthentication process could be easily accomplished with `aireplay-ng` [9] tool from the `Aircrack-ng` suite of tools, but in order to not depend on external tools, we configured it with `scapy`, which accomplishes the same goal.

Thus, the developed tool is a modular framework that integrates a set of Scanners, Detectors and Actuators, and enables the creation of other detectors by developing new modules. In the next chapter we present a set of tests to evaluate the success rate of this tool.

[9] http://www.aircrack-ng.org/doku.php?id=deauthentication

Chapter 6

Results

In this chapter we present a set of results that came from a RAP detection performed with the developed tool. These results were obtained from two different scenarios:

1. The RAPs were deployed by an information security specialist impersonating an attacker;
2. The RAPs were configured and deployed by the project researchers in a contained environment.

In the first scenario we did not know which type of RAP would be created, which revealed to be an interesting proof of concept for the accuracy of the tool. In the tests from the second scenario we tried to replicate all the attacks and RAPs covered by the detection tool.

6.1 Proof of Concept

We start by showing the results from the first scenario with the explanation about the performed detection and finally revealing the type of attack that was created.

The Detection dump was stored in a text file and so in the displayed images of the detection for this scenario it is not present the colors associated with a RAP detection.

These tests where conducted in an Enterprise environment, hence the loaded configuration profile used by the detection tool had information about the authorized APs of that network. In such environments the `Corporate` APs are usually configured with WPA2 Enterprise, which usually is associated with directory service credentials to authenticate in the network. This fact is important to understand the possible type of attack in these environments.

Figure 6.1 has the first set of detections produced by the Rogue AP detector. In the screenshot are selected the produced alerts, and in this case we detected two RAPs; the four alerts shown in the figure are because of interferences in the RSSI.

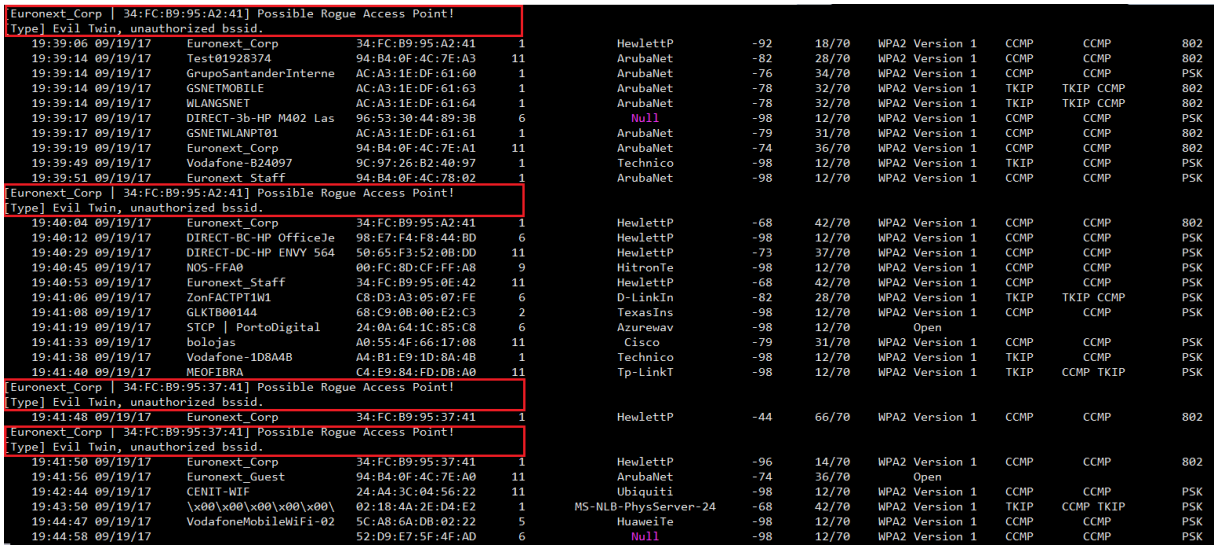


Figure 6.1: Detection 1

In this test case the deployed RAP was detected because it was not an authorized AP, it was almost a perfect clone of one of the authorized APs but it failed the BSSID parameter which caused its spotting.

To bypass this detection the attacker modified the BSSID of the RAP to one of the authorized list, but in order to succeed in the attack, i.e., to have clients trying to associate to the RAP, the RSSI had to be increased. Figure 6.2 shows the new RAP configured with a BSSID from the authorized list and with a much stronger signal than the authorized AP.

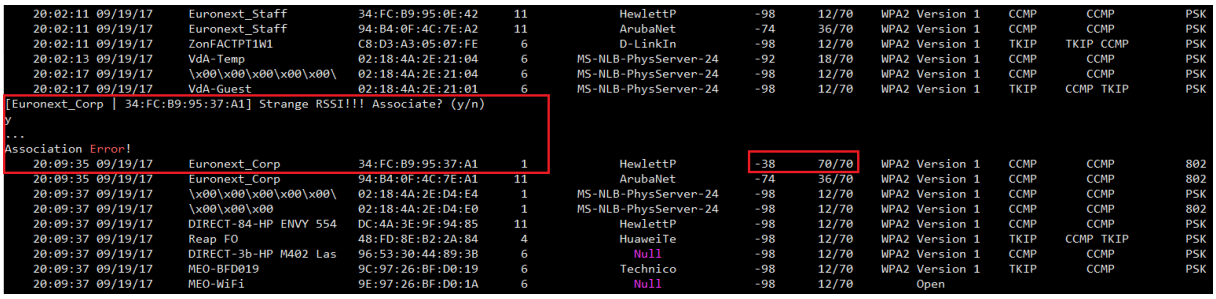


Figure 6.2: Detection 2

When trying to associate to the AP it failed and corroborated the fact that it was an RAP; as if it were the authorized AP it would successfully associate and run the active heuristics to support it, i.e., facing the results with the ones from the configuration profile.

When analyzing the attack type it was understood that in such cases, where a RAP is impersonating a WPA2 Enterprise network the objective is to capture the victim's credentials, and for this it only requires to capture the challenge/response authentication. In order to explore MITM scenarios an attacker would configure an Open RAP, which is usually the case of the Evil Twin attack.

Finally, in Figure 6.3 it was detected a RAP produced by a Pineapple AP and Karma attacks. In case of the Pineapple AP, apart from using the default name of Pineapple which automatically indicates that it is a RAP it also uses the default BSSID which contains the *13:37 - leet speech*, which in our detection tool blacklists the AP.

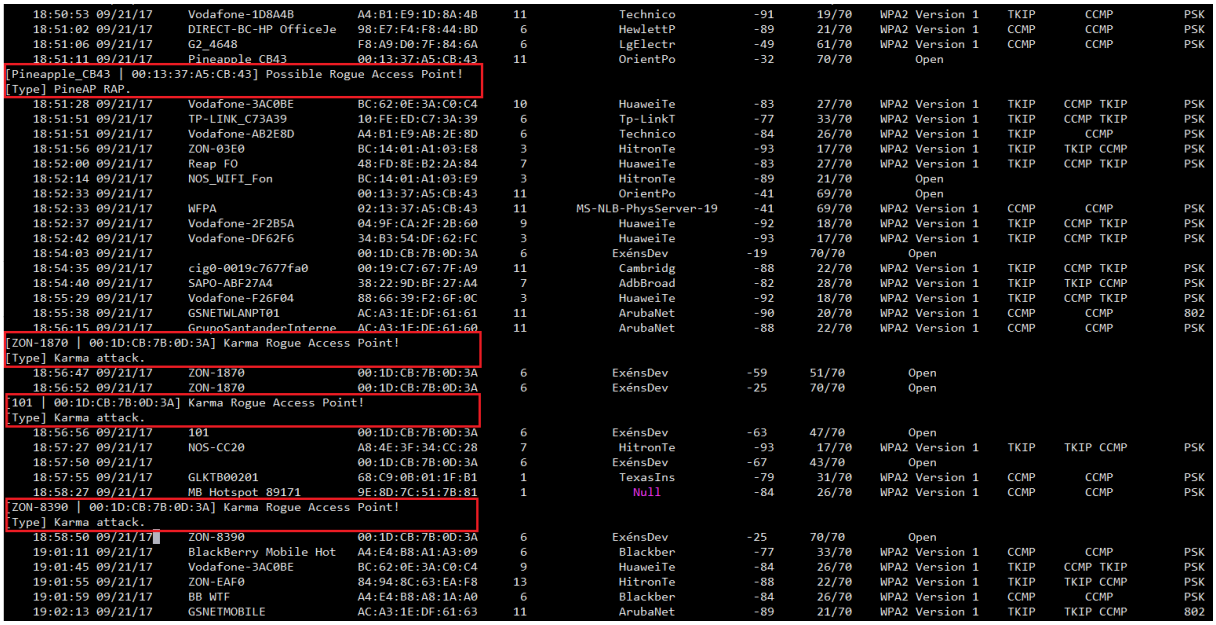


Figure 6.3: Detection 3

It can be seen by the time line that in the Karma attack the BSSID was modified, but since the same BSSID advertise different SSIDs it was easily spotted.

For the second scenario, where the RAPs were configured in a contained environment we show the detections for the not yet used heuristics, specifically: RAP with different encryption and validation of free WiFis authenticity.

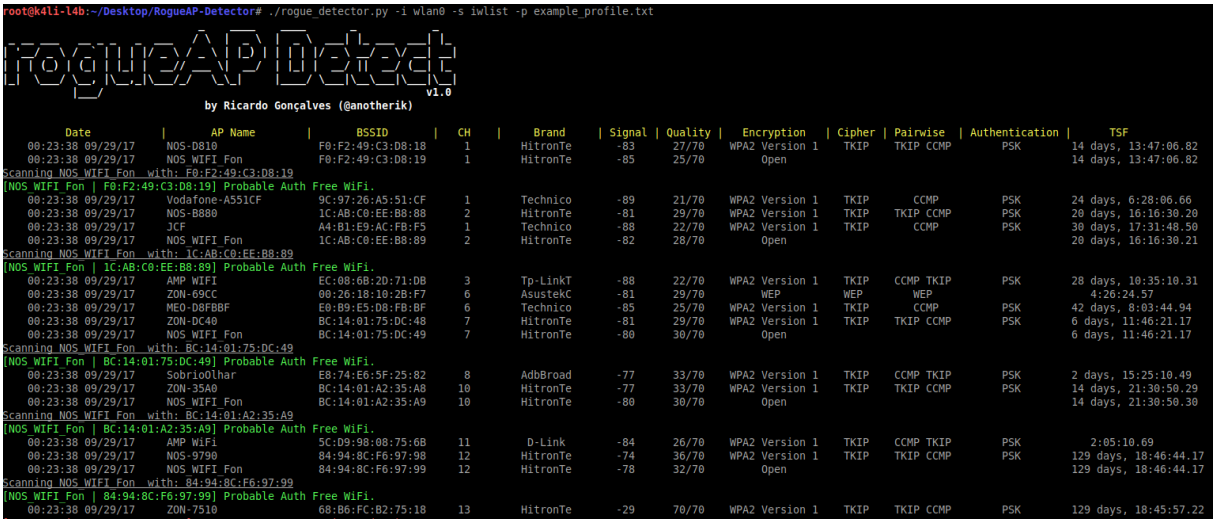


Figure 6.4: Detection 4

Figure 6.4 shows the validated free WiFis following the algorithm described in section 4.2.2 from chapter 4.

The last example, Figure 6.5, shows the case of an Evil Twin attack, here the RAP has the same SSID and BSSID of the authorized AP but it has Open encryption. This type of detection is also possible without using any profile, because of the no knowledge detectors.

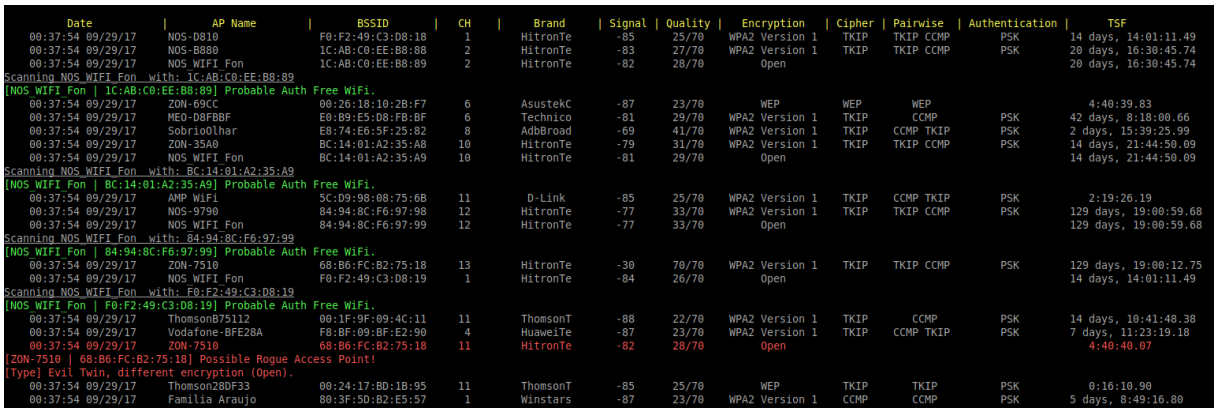


Figure 6.5: Detection 5

An interesting fact that can be enforced here is the TSF produced by RAPs contrasting with the one of the authorized APs. As the TSF keeps the timers for all stations in the same BSS synchronized, each AP will have a different value and, with that we can also relate with the up time of the AP. From Figure 6.5 it can be seen that the newly created RAP have a TSF different from the authorized AP, which even without an authorized list it could be spotted.

Summing up, we performed a set of tests for two different scenarios in order to create a proof of concept for the developed tool. With the presented set of tests we covered the scanning and detection processes (passive and active heuristics).

Name	**Description**
Detection 1	RAP with same SSID and different BSSID
Detection 2	RAP with same SSID, BSSID and Encryption; different RSSI
Detection 3	RAP from PineAP and Karma attack
Detection 4	Validate free WiFis authenticity
Detection 5	RAP with same SSID and BSSID; different Encryption

Table 6.1: Test cases

Chapter 7

Conclusions

In this work we studied the concepts behind the Rogue Access Points (RAPs), we described the types of RAPs and some of the countermeasures studied in the literature. We also studied the way these RAPs are created and explored, e.g., the type of attacks conducted to lure a victim into a RAP.

With this knowledge and during a period of research analyzing the WiFi packets and APs specifications and parameters, we proposed a flexible framework that relies on modules and a set of heuristics that generates a final value to identify a possible RAP.

The modular architecture of this tool is a plus in case of someone wants to incorporate new modules to improve it, from scanning methods to detection heuristics.

The framework addresses Evil Twin types of RAPs, both the case of coexistence and replacement, where the application uses the Actuators association process to identify them; improperly configured APs, unauthorized and compromised APs are also identified with whitelists and blacklist heuristics. It is also possible to detect RAPs generated by attacking tools, like PineAP, Alfa cards and Airbase. On top of this, an heuristic to validate authenticity of free WiFis is also configured in this application.

During the development of the tool and research part of this thesis it became clear the difference between the attacking and defensive worlds. In one side we have an handful of tools and documentation explaining how to perform a WiFi attack and how to deploy a RAP, and in the other side we have much limited information that requires some research. Thus, this point was part of the motivation to start researching and learn more about state of the art proposed detection mechanisms, defensive tools and ultimately create one.

With the set of tests performed to test the application we could retrieve a satisfactory proof of concept that supports the utility and effectiveness of the application.

7.1 Future Work

From the research done during this thesis, it became clear the amount of possibilities to complement and improve this tool. Also, working both as an attacker and defender allowed us to think out of the box and constantly think of new heuristics for detection. Thus, as future work, it would be interesting to expand the framework into a distributed system where the main application would communicate with a set of nodes. The objective of such system would be to cover a larger area, and ultimately, physically locate the RAPs.

Regarding the scanning process we would like to upgrade it to `iw` instead of `iwlist`, this is because the first has more information about the Beacon frames that could be incorporated as new detection heuristics. To the detection part we think of using the TSF parameter to spot newly created AP and face them with possible RAPs from other heuristics in order to give a bigger accuracy in the detection rate.

Also, a weight model for some of the heuristics would give more reliability to the application, considering the fact that some of the detection might be false positives it would help in reduce them.

Finally, the development of a *GUI* for the application would bring some benefits for the most unexperienced user's.

Bibliography

[1] IEEE Computer Society, "IEEE Standard for Local and metropolitan area networks - Port-Based Network Access Control", 2010.

[2] 802.11-2016. IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications, 2016.

[3] Neha Agrawal and Shashikala Tapaswi. Wireless rogue access point detection using shadow honeynet. *Wireless Personal Communications*, 83(1):551–570, 2015.

[4] K. M. Ali and A. Al-Khlifa. A comparative study of authentication methods for wi-fi networks. *In 2011 third international conference on computational intelligence, communication systems and networks (CICSyN)*, pages 190–194, 2011.

[5] Bandar Alotaibi and Khaled Elleithy. Rogue access point detection: Taxonomy, challenges, and future directions. *Wireless Personal Communications*, pages 1–30, 2016.

[6] Chrisil Arackaparambil, Sergey Bratus, Anna Shubina, and David Kotz. On the reliability of wireless fingerprinting using clock skews. In *Proceedings of the third ACM conference on Wireless network security*, pages 169–174. ACM, 2010.

[7] Niemi V. Asokan, N. and K. Nyberg. Man-in-the-middle in tunnelled authentication protocols. *In Security Protocols*, pages 28–41, 2005.

[8] Kevin Bauer, Harold Gonzales, and Damon McCoy. Mitigating evil twin attacks in 802.11. In *2008 IEEE International Performance, Computing and Communications Conference*, pages 513–516. IEEE, 2008.

[9] Kemal Bicakci and Bulent Tavli. Denial-of-service attacks and countermeasures in IEEE 802.11 wireless networks. *Computer Standards & Interfaces*, 31(5):931–941, 2009.

[10] Sergey Bratus, Cory Cornelius, David Kotz, and Daniel Peebles. Active behavioral fingerprinting of wireless devices. In *Proceedings of the first ACM conference on Wireless network security*, pages 56–61. ACM, 2008.

[11] S Chae, H Jung, I Bae, and K Jeong. A scheme of detection and prevention rogue ap using comparison security condition of ap. In *2012 Universal Association of Computer*

and Electronics Engineers international conference on advances in computer science and electronics engineering, pages 302–306, 2012.

[12] Yingying Chen, Jie Yang, Wade Trappe, and Richard P Martin. Detecting and localizing identity-based attacks in wireless and sensor networks. *IEEE Transactions on Vehicular Technology*, 59(5):2418–2434, 2010.

[13] Mohan K Chirumamilla and Byrav Ramamurthy. Agent based intrusion detection and response system for wireless lans. In *Communications, 2003. ICC'03. IEEE International Conference on*, volume 1, pages 492–496. IEEE, 2003.

[14] CISCO. Rogue detection under unified wireless networks. Online, 2007. Accessed 31st December 2016.

[15] Dino A Dai Zovi and Shane A Macaulay. Attacking automatic wireless network selection. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 365–372. IEEE, 2005.

[16] Evgueni Doukhnitch, Muhammed Salamah, and Emre Ozen. An efficient approach for trilateration in 3d positioning. *Computer Communications*, 31(17):4124–4129, 2008.

[17] Daniel B Faria and David R Cheriton. Detecting identity-based attacks in wireless networks using signalprints. In *Proceedings of the 5th ACM workshop on Wireless security*, pages 43–52. ACM, 2006.

[18] Jim Geier. 802.11 beacons revealed. Online, 2002. Accessed 24th December 2016.

[19] Jim Geier. Understanding 802.11 frame types. Online, 2002. Accessed 24th December 2016.

[20] Harold Gonzales, Kevin Bauer, Janne Lindqvist, Damon McCoy, and Douglas Sicker. Practical defenses for evil twin attacks in 802.11. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–6. IEEE, 2010.

[21] IEEE 802.11 Working Group et al. Amendment 6: Medium access control (mac) security enhancements. *IEEE Standard*, 802.

[22] Fanglu Guo and Tzi-cker Chiueh. Sequence number-based mac address spoof detection. In *International Workshop on Recent Advances in Intrusion Detection*, pages 309–329. Springer, 2005.

[23] Piyush Gupta and Panganmala R Kumar. The capacity of wireless networks. *IEEE Transactions on information theory*, 46(2):388–404, 2000.

[24] Hao Han, Bo Sheng, Chiu Chiang Tan, Qun Li, and Sanglu Lu. A measurement based rogue ap detection scheme. In *INFOCOM 2009, IEEE*, pages 1593–1601. IEEE, 2009.

[25] Hao Han, Bo Sheng, Chiu C Tan, Qun Li, and Sanglu Lu. A timing-based scheme for rogue ap detection. *IEEE Transactions on parallel and distributed Systems*, 22(11):1912–1925, 2011.

[26] Changhua He and John C Mitchell. Analysis of the 802.11 i 4-way handshake. In *Proceedings of the 3rd ACM workshop on Wireless security*, pages 43–50. ACM, 2004.

[27] Suman Jana and Sneha K Kasera. On fast and accurate detection of unauthorized wireless access points using clock skews. *IEEE Transactions on Mobile Computing*, 9(3):449–462, 2010.

[28] W JOEL, L NICK, LEENDERT VAN, and David Safford. Autonomic 802.11 wireless LAN security auditing. 2004.

[29] Kuo Fong Kao, Wen Ching Chen, Jui Chi Chang, and Heng Te Chu. An accurate fake access point detection method based on deviation of beacon time interval. In *Software Security and Reliability-Companion (SERE-C), 2014 IEEE Eighth International Conference on*, pages 1–2. IEEE, 2014.

[30] Taebeom Kim, Haemin Park, Hyunchul Jung, and Heejo Lee. Online detection of fake access points using received signal strengths. In *Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th*, pages 1–5. IEEE, 2012.

[31] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Stefanos Gritzalis. Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset. *IEEE Communications Surveys & Tutorials*, 18(1):184–208, 2015.

[32] Fabian Lanze, Andriy Panchenko, Ignacio Ponce-Alcaide, and Thomas Engel. Hacker's toolbox: Detecting software-based 802.11 evil twin access points. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 225–232.

[33] Fabian Lanze, Andriy Panchenko, Benjamin Braatz, and Thomas Engel. Letting the puss in boots sweat: detecting fake access points using dependency of clock skews on temperature. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 3–14. ACM, 2014.

[34] Xinghua Li, Fenye Bao, Shuxin Li, and Jianfeng Ma. Flap: An efficient wlan initial access authentication protocol. *IEEE Transactions on Parallel and Distributed Systems*, 25(2): 488–497, 2014.

[35] Linksys. How to enable rogue ap detection on your linksys wireless-ac access point. Online. Accessed 31st December 2016.

[36] Liran Ma, Amin Y Teymorian, and Xiuzhen Cheng. A hybrid rogue access point protection framework for commodity Wi-Fi networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008.

[37] Guowang Miao, Jens Zander, Ki Won Sung, and Slimane Ben Slimane. *Fundamentals of Mobile Data Networks:*. Cambridge University Press, Cambridge, 002 2016. ISBN: 9781316534298. doi:10.1017/CBO9781316534298.

[38] Diogo Mónica and Carlos Ribeiro. Wifihop-mitigating the evil twin attack through multi-hop detection. In *European Symposium on Research in Computer Security*, pages 21–39. Springer, 2011.

[39] Hossen Mustafa and Wenyuan Xu. Cetad: Detecting evil twin access point attacks in wireless hotspots. In *Communications and Network Security (CNS), 2014 IEEE Conference on*, pages 238–246. IEEE, 2014.

[40] Somayeh Nikbakhsh, Azizah Bt Abdul Manaf, Mazdak Zamani, and Maziar Janbeglou. A novel approach for rogue access point detection on the client-side. In *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*, pages 684–687. IEEE, 2012.

[41] R. M. Pandurang and D. C. Karia. Performance measurement of WEP and WPA2 on WLAN using OpenVPN. *In 2015 international conference on nascent technologies in the engineering field (ICNTE)*, pages 1–4, 2015.

[42] Lisa Phifer. A list of wireless network attacks. Online. Accessed 13th June 2017.

[43] Guangzhi Qu and M Nefcy Michael. Rapid: An indirect rogue access points detection system. In *International Performance Computing and Communications Conference*, pages 9–16. IEEE, 2010.

[44] Timothy R Schmoyer, Yu Xi Lim, and Henry L Owen. Wireless intrusion detection and response: a classic study using main-in-the-middle attack. In *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, volume 2, pages 883–888. IEEE, 2004.

[45] Yong Sheng, Keren Tan, Guanling Chen, David Kotz, and Andrew Campbell. Detecting 802.11 mac layer spoofing using received signal strength. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008.

[46] Bartlomiej Sieka. Active fingerprinting of 802.11 devices by timing analysis. In *CCNC 2006. 2006 3rd IEEE Consumer Communications and Networking Conference, 2006.*, volume 1, pages 15–19. IEEE, 2006.

[47] Yimin Song, Chao Yang, and Guofei Gu. Who is peeping at your passwords at starbucks?-to catch an evil twin access point. In *Dependable Systems and Networks (DSN)*, volume 10, pages 323–332, 2010.

[48] VS Shankar Sriram, G Sahoo, and Krishna Kant Agrawal. Detecting and eliminating rogue access points in ieee-802.11 wlan-a multi-agent sourcing methodology. In *Advance computing conference (IACC), 2010 IEEE 2nd international*, pages 256–260. IEEE, 2010.

[49] Jin Z. Wang, Y. and X. Zhao. Practical defense against WEP and WPA-PSK attack for WLAN. *In 2010 6th international conference on wireless communications networking and mobile computing (WiCOM)*, pages 1–4, 2010.

[50] Wei Wei, Sharad Jaiswal, James F Kurose, and Donald F Towsley. Identifying 802.11 traffic from passive measurements using iterative bayesian inference. In *INFOCOM*, 2006.

[51] Wei Wei, Sharad Jaiswal, Jim Kurose, Don Towsley, Kyoungwon Suh, and Bing Wang. Identifying 802.11 traffic from passive measurements using iterative bayesian inference. *IEEE/ACM Transactions on Networking (TON)*, 20(2):325–338, 2012.

[52] Haidong Xia and José Brustoloni. Detecting and blocking unauthorized access in Wi-Fi networks. In *International Conference on Research in Networking*, pages 795–806. Springer, 2004.

[53] Bo Yan, Guanling Chen, Jie Wang, and Hongda Yin. Robust detection of unauthorized wireless access points. *Mobile Networks and Applications*, 14(4):508–522, 2009.

[54] Chao Yang, Yimin Song, and Guofei Gu. Active user-side evil twin access point detection using statistical techniques. *IEEE Transactions on Information Forensics and Security*, 7 (5):1638–1651, 2012.