Danmarks Tekniske Universitet

34229 Project Work
Bachelor in Cyber Technology

Midterm Report

Oliver Brisson Bredel (s214684)
Lucas Loua Buhelt (s214691)
Nicklas Thorvald Kiær (s216137)

31. May 2023

Abstract

Written by Oliver, reviewed by Lucas
The main goal of this project is to explore and examine the properties of the 802.11 specification. Wireless communication/security is critical to maintaining the internet infrastructure and therefore a focus of both hackers and network administrators. As data is sent over a shared medium it is possible with passive scanning to obtain traffic and analyze it. By doing so two advanced attacks has been examined. Deauthentication attacks has been exploited in order to deny service for clients(DoS) on the WLAN, whilst WEP secret key cracking has been used to obtain clear text traffic. WEP was one of the first ways to encrypt wireless data, but has glaring flaws in its algorithm. It's been possible to crack the password for a network secured by WEP and shown that by just capturing packets from a single user on an AP, all data that has been sent over the medium has been decrypted. The WLAN has been mapped by a written WiFi-scanner in Scapy (Python library) in connection with our deauthentication attacks that show key information on the WLAN. A deauthentication attack has been successfully completed on a modern device after identifying it with our WiFi-scanner. This shows that even secure networks using technologies in use today, are still at risk of certain attacks.

Contents

1 Introduction

Written by Nicklas & Lucas, reviewed by Oliver
Wireless security has become increasingly critical due to the rapidly expanding use and need for internet access. Many devices use WiFi to connect to the internet, such as smartphones and laptops. As data over WiFi is sent via radio waves, thus using a shared medium there is a coherent risk when communicating. The data is at risk of being captured by bad actors and is prone to manipulation. Thus a need for strong encryption and control of said data is necessary. However, even with security measures in place for wireless networks, there are still vulnerabilities to exploit. One such attack, which is the focus of this report, is a denial-of-service (DoS) attack: the deauthentication attack. It's an attack where a user will be denied access to the network and thereby either disrupt a workplace or pave the way for other more sophisticated attacks such as the Evil Twin Attack[1].

This project will explore the theory and practical applications of deauthentication attacks on networks. Further, password cracking will be explored for secret keys used in the encryption of wireless network traffic using the now outdated WEP protocol. To achieve these goals, information about the wireless local area network (WLAN) has to be known. Thus, another goal for the project is obtain information of available SSIDs, clients connected, signal strength and so on.

When this project is done, the hope is to have explored the the theory and practical applications of deauthentication attacks on networks and the building blocks that make up data-packets in a wireless network. It is important to note that this project is for educational purposes only and should not be used for malicious activities.

2 Functionality

Different parts written by all group members
Our project constists of the following functionalities:

1. Scan the WLAN and identify key information about clients/Access Points
2. Perform deauthentication attack on WiFi devices
3. Capture packets and obtain clear text by Password cracking of WEP

These functionalities aim to be the core of the product that will be created. The first functionality, scanning for local networks, is crucial as it creates the base of the other functionalities.

A typical user roadmap utilizing the scanning of the network in order to deauthenticate the user is shown on figure 1.

A deauthentication attack is a type of Denial-of-Service (DoS) attack that targets communication between a user and a WiFi wireless access point [12]. It exploits a feature of IEEE 802.11 wireless networks that allows devices to disconnect from a network by the deauthentication management frame [19]. A deauthentication frame tells a device to stop using communication with the AP. It can be sent by either the AP or the device itself. It's normally used for legitimate purposes, such as ending a connection or switching to another network (handover); Yet there is no identification check performed in this procedure. Hence an attacker can exploit this feature and send deauthentication frames to the AP with a spoofed client source address. This causes the connection to close making the spoofed device's connection end and probably reconnect. This exploitation has multiple use-cases. If done repeatedly or to multiple devices, this can

[1] An Evil Twin attack is when the adversary creates an AP identical to the victim's, and then either forcing or tricking the victim's devices to connect to the malicious AP, thus obtaining a Man-in-the-Middle state

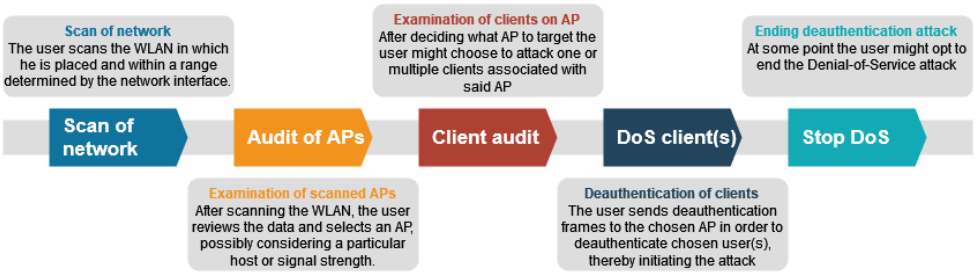

Figure 1: A typical user roadmap

disrupt or disable the network entirely, thus resulting in a DoS Attack. Meanwhile it can also be used to obtain a EAPOL 4-way-handshake for WPA or as a way to misguide a client to a rogue access point created by a malicious user to obtain a Man-In-The-Middle state, known as an Evil Twin Attack.

The last point is password cracking. The product should be able to crack/obtain the WiFi-password in the WEP algorithm. WEP is one of the most basic WiFi algorithms. It is an outdated and non-secure algorithm and the product utilizes Aircrack-ng's PTW method to recreate the passwords on the APs [1]. The method needs data-packets from users on the network containing initialization vectors and those are captured by the WiFi scanner, as that is it's primary functionality. When the scanner has obtained enough packets, around 10.000, the password cracker should then be able to obtain the passwords using Aircrack-ng. This part utilizes a tool that is already known, thus a greater focus here will be largely theoretical.

The implementations are mainly based on the python library Scapy [3] to implement the functionalities. As our network interfaces has to be put in monitor mode a UNIX environment is used. The long-term goal is to use a raspberry pi later, to obtain mobility and a uniform platform. The network interface is a Realtek DWA 131-E1 dongle, as it is cheap and allows for the usage of monitor mode. The UNIX platform has the consequence of limiting the use of the implementation to a specific operating system. Though this should be able to be ported to Windows and MacOs if need be. Here the Linux-sub-system (WSL) on Windows might be exploited [15]. The positive side of using UNIX is that we can implement the program on many types of systems as we use a virtual machine to create the product. Thus making conversion to a raspberry pi easier. The negative side is that we have to use a virtual machine which can create problems with drivers and performance.

2.1 Mapping of networks

This block takes the physical data being sent in the wireless space, and transforms it into information that the other blocks need and use. It contains a table that holds the information as sub-blocks, for instance access points in the WLAN, the signal strength of the AP and which channels are used as well as the clients on said AP. This mapping can be done via beacon management frames used by APs to announce their respective presence. Likewise probe management frames can be used to understand what APs a client has been associated with in the past. The block also contains a way of collecting data-packets containing initialization vectors and sending them to the password cracking block.

2.2 Deauthentication

This block handles deauthentication, wherein we deny clients access of certain APs or all wireless internet usage. Thus making the sub-blocks:

- Client
- AP
- Channel

2.3 Password cracking and WEP/WPA2

This block handles the actual cracking of WEP from data-packets collected from the mapping block. The data should be combined into a single file which can then be analysed and the password/passkey found.

3 Theory

Written by Nicklas & Lucas, reviewed by Oliver
The primary conceptual framework underpinning this project revolves around the 802.11 standard [10]. The 802.11 standard is a set of wireless network protocols developed by the Institute of Electrical and Electronics Engineers (IEEE). The first version of the standard, 802.11, was released in 1997, and since then, several revisions and amendments have been made to improve the technology and increase its capabilities [6].

The 802.11 standard uses radio waves to transmit data between devices within a WLAN without the need for cables or wires. This makes it a popular choice for connecting mobile devices to the internet, especially in homes, offices, and public places such as cafes, airports, and hotels [14].

Throughout the project, programming will be done mainly in python, using the library Scapy for sniffing, analyzing and transmitting packets. Scapy is a program written in Python, that gives the ability to construct, decrypt, send, capture packets and much more [17]. Scapy gives us tools for analyzing 802.11 frames, thus providing the information contained in 802.11 frames. Packets in Scapy are created as objects with layers built on top of each other to define the type of the packet.

There are 3 frame types in 802.11: Management-, control- and data-frames [16, 8]. Figure 2 shows some of the types and subtypes of different frames with a focus on management frames, since that is what our project is focused on. Management frames are used by APs to join and leave the basic service set (BSS). They have a MAC header, a frame body, and a trailer. These frames are needed because it is difficult to connect to a wireless network compared to a wired network. With a wireless connection, all data is sent everywhere and each router and user needs to know which data-packets to ignore and which to receive. This is done via management frames where a user associates with an Access Point (AP) and agree on conditions, such as security and speed. For a user to know which APs are near, it needs to scan the surrounding area. This is done via 2 ways of scanning: Active scanning and passive scanning.

In passive scanning each AP broadcasts beacon frames and a user sends an association request to the AP it wants to connect to. In active scanning the user broadcasts probe requests and each AP responds to this request with a response back to the user. When the user then wants to connect to an AP that has responded it sends an association request to that specific AP.

Types and SubTypes

Various 802.11 Frame Types and Subtypes

Type Value B3..B2	Type Description	Subtype Value B7 .. B4	Subtype Description
00	Management	0000	Association Request
00	Management	0001	Association Response
00	Management	0010	Reassociation Request
00	Management	0011	Reassociation Response
00	Management	0100	Probe Request
00	Management	0101	Probe Response
00	Management	0110	Timing Advertisement
00	Management	0111	Reserved
00	Management	1000	Beacon
00	Management	1001	ATIM
00	Management	1010	Disassociation
00	Management	1011	Authentication
00	Management	1100	Deauthentication
00	Management	1101	Action
00	Management	1110	Action No Ack (NACK)
00	Management	1111	Reserved
01	Control	0000-0001	Reserved
01	Control	0010	Trigger[3]
01	Control	0011	TACK

Figure 2: WiFi typer og subtyper

3.1 WEP & WiFi security standards

Written By Lucas, reviewed by Nicklas
Wireless networks also need security while sending packets, i.e. authentication and confidentiality. To ensure confidentiality, data-frames are encrypted with some sort of security standard. The oldest widespread type being Wired Equivalent Privacy (WEP). It had major security flaws that allowed hackers to bypass the algorithm and read the data being sent [4]. Therefore was WiFi Protected Access (WPA) developed and implemented from 2003 and onwards [13].

WEP uses a secret key to encrypt packets between client and AP [5]. The problem with this is that WEP uses the RC4 encryption algorithm, which is known as a stream cipher. A stream cipher operates by making a short key into an infinite random key-stream. The client uses XOR on the key-stream with the plain data to produce the encrypted data. The AP has a copy of the same used key, and uses it to generate an identical key-stream. The AP then uses XOR on the key-stream with the encrypted data to get the exact same plain data back. This type of encryption is easy to misuse, since by flipping a single bit in the encrypted data, the plain data will also have a corresponding single bit flipped. Also the XOR can be found via statistical analysis of key-streams to thereby recover the plain data.

WEP has incorporated defenses against these 2 types of attacks, but they are implemented poorly. The defense against flipping bits is integrity checks in the packet, implemented as a CRC-32 checksum. Sadly it is linear which means that it is possible to find the difference between 2 CRC's because the bits that are flipped in the encrypted data is deterministic and therefore an attacker can adjust the checksum and then it is believed that the integrity of the packet is kept, when in reality it isn't.

The defense against the statistical attack is an initialization vector to decrease the chance of reuse of key-streams. The vector is only a 24 bit field and that guarantees the reuse of key-streams

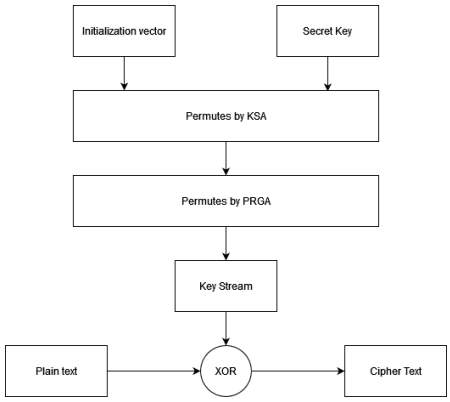

Figure 3: Simplified RC4 algorithm [11]

because the vector is only made up 24 bits and when thousands or millions of data-packets are sent, reuse will happen quickly. It would take around 110.000 data-packets to ensure reuse [7]. Because of this, the attacker can easily pick up enough data to perform statistical analysis of packets using the same key-stream and then recover the plain data.

Fortunately mitigation for the flaws of WEP have become industry standard since the WPA2 protocol, which was a development of the intermediate measure WPA as a security update to fix many of WEP's problems [2][13]. It was released in 2004 in the 802.11i amendment to the original 802.11. WPA2 uses CCMP protocol which is a development of the AES security algorithm. It rids the network encryption of the previously mentioned security vulnerabilities.

3.2 Deauthentication in WLAN

Written by Nicklas
The authentication of clients and APs can be exploited by deauthentication attacks. The theory behind this attack is based on the 802.11 standard, which allows clients to disconnect from an AP using a deauthentication frame.

When a client device is connected to a WiFi network, it sends periodic beacon frames to the access point to maintain the connection. In a deauthentication attack, an attacker sends a series of forged/spoofed deauthentication packets to the target client or AP, pretending to be the legitimate access point or client. This causes the target to disconnect from the network and require reauthentication, disrupting the communication between the client and the access point.

Deauthentication frames are frames of the management type as seen in figure 2, and have the overall structure as such. Therefore the deauthentication frame contains, among other things, 3 addresses. These addresses being the destination address, the source address, and the BSSID of the AP's wirelsess interface, the address of the AP These three addresses are what can be altered in order to perform a deauthentication attack. One of the fields in a deauthentication frame is the reason code, which is 16-bit. This defines the reason for the termination of the connection. There can thus be upwards of 65535 reason codes, yet only about 66 is utilized [18]. Apart from a reason code, the frame body includes vendor specific elements, and the Management MIC IE(MMIE) [9]. Though this is not always present.

802.11w, released in 2009, introduces protected management frames thereby combatting attacks such as deauthentication attacks. This is achieved by ensuring the integrity in the management frame. The integrity check is achieved by the sender calculating a Message Integrity Check (MIC) value, and appending this to the frame. The receiver then calculates the MIC value using the same algorithm, and compares the two. If the two MIC values match, the frame is considered to

not having been tampered with, and thus marking it as authentic. Furthermore a frame sequence number(FSN) is introduced, which is used to prevent replay attacks. Replay attacks are not something we will be covering in this project and therefore it wont be analyzed in this report. The MMIE is thus only in use when protected management frames are enabled [9]. Protected management frames were introduced in order to combat the deauthentication attack. Since the deauthentication frames without protected management frames has no integrity check by nature, this creates a sizeable vulnerability for a network as attacks such as the deauthentication attack can be exploited.

4 Implementation

The implementation is based purely on Python3 and the library Scapy. Figure 4 shows a UML-class-diagram of the two classes utilized in the implementation (without CLI). The Scanner class can hold multiple WiFi objects. The Scanner class can be seen as the main class holding and performing the core functionalities. For now there are some functions that may be bloated in conjunction with a possibility of simplifying the classes. As this is one of the first iterations of the code this is to be expected.

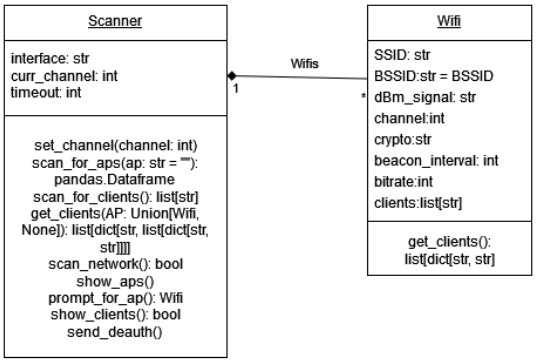

Figure 4: UML diagram of Scanner and WiFi classes

4.1 Mapping of networks

Written by Oliver, reviewed Nicklas
The WiFi-scanner is the backbone of our implementation. It listens to all data that is transferred over WiFi and subsequently creates the lists that the deauthentication uses and captures packets for the password cracker. The point of this functionality is to gather information about APs and the associated clients, the logic for this implementation is showed in figure 9 as a flowchart.

Firstly a system check is performed to ensure that the program is running on a UNIX system, as the program is not compatible with other systems. A possible operating system error is shown on figure 5

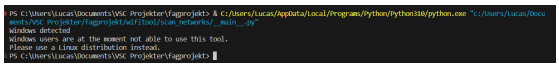

Figure 5: Windows error

Then the program checks if the user is running the program as root, as this is required for the program to function properly. The passive scan is split into two main tasks, the first being the scanning for APs and the second being the scanning for clients. A possible optimization of the program could be combining these two tasks into one, however this would require a

more complex implementation, and would not be as readable. The scanning for APs is done by sniffing for beacon frames, which are sent by APs to announce their presence. The sniffing is done by using the Scapy function "sniff", which takes a filter as an argument, this function is used when scanning for APs. The filter "type mgt, subtybe beacon" is applied in order to only sniff for beacon frames when looking for APs, and the function is set to run for a specified timeout duration. This is done to ensure that the program does not run indefinitely, and thereby potentially causing issues with the system. The code for part of this is shown in figure 6.

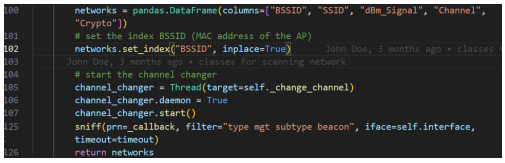

Figure 6: Code for the sniffing with Scapy

The scanning is done on each network channel in turn, where it scans for beacon frames from the APs. It returns the BSSID (MAC-address), SSID, signal strength, channel and the cryptography of each AP.

To obtain information about clients we use the APs that we found and then look at packets being sent and check if the packet is sent from a client. This is then returned with the code shown in figure 7. We can see that `WiFi.get_clients()` is the code that gets the clients for each AP.

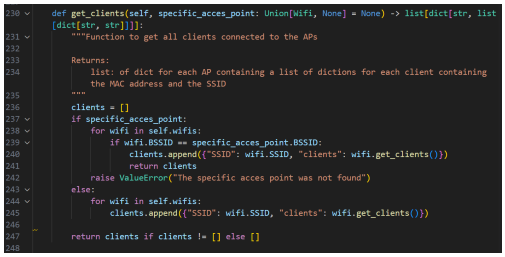

Figure 7: Code for getting the clients for each AP

When scanning for clients all 802.11 packets are examined as it is rare that there is communication between APs. However the DS fields in the 802.11 header is exploited, as they indicate the direction of the frame. The code for this is seen on figure 8.

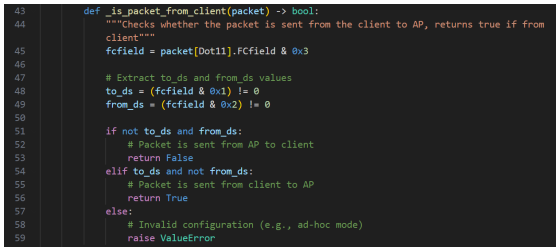

Figure 8: Code for examining packets DS field

Further optimization can be done to create better filters and most importantly use all data sent over the medium to create a more complete map of the WLAN. This however has not been done in the current level of implementation and might be deemed unnecessary as the wanted data is obtained.

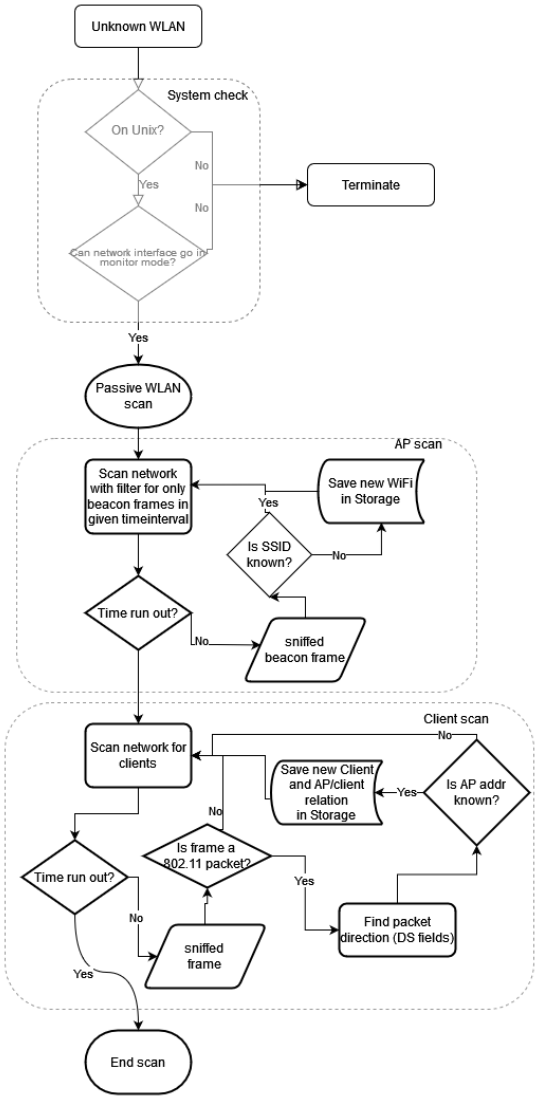

Figure 9: Flowchart for mapping of WLAN

4.1.1 Limitations

The implementation is limited to only being able to scan for APs and clients in the 2.4 GHz band. This is a physical limitation due to the fact that the dongle used for the implementation only supports the 2.4 GHz band. The mapping of the network is done by passively scanning the medium, thus resulting to the implementation being limited to only having the ability to scan for APs and clients that are actively sending data. This is a limitation that is inherent to passive scanning. This can be overcome by using active scanning e.g. via a Replay Attack (which can force a device to send packets). When doing the deauthentication attack there is actually a form of active scanning as the system is sending packets to the AP. This is however not a reliable way of scanning for APs and clients, as the current implementation of the attack is not built for this purpose and rather builds upon the passive scanning. But the limitation of using passive scanning is also one of its positives: Since there is no active participation in the communication, the implementation is close to impossible to be detected by a WIDS (WiFi Intrusion Detection System). This is a positive as it allows for the implementation to be used for reconnaissance without being detected, whereas active scanning is noisy. NMAP a known tool for mapping networks (on ethernet) is very noisy and easy to detect.

4.2 Deauthentication

Written by Nicklas, reviewed by Lucas
Our implementation of the deauthentication functionality relies on the ability to create and send packets with the use of Scapy. Thereby forging our own deauthentication packet, thus gaining the ability to set our own parameters for the packet. As shown in figure 10a, our implementation of the deauthentication is as follows. The user is first met with a menu of the known APs, gained from the mapping of the network, that are eligible for attempting an attack. We have then decided to show another prompt menu which gives the user the option of choosing the target from a list of the already known clients, if so desired. If not, the user can either input a MAC-address, or choose to use the broadcast address, resulting in all clients receiving the deauthentication packets.

```python
def send_deauth():
    target_ap = prompt_for_ap()
    scanner.set_channel(target_ap.channel)

    # Check if we have clients on the AP. If so, prompt user for client to deauth
    if target_ap.clients:
        print("Choose client to deauth from list:")
        for i, client in enumerate(target_ap.clients):
            print(f"{i}. {client}")
        print(f"{len(target_ap.clients)+1}. User defined client")
        print(f"{len(target_ap.clients)+2}. Deauth all clients")
        client_to_deauth = int(input("Input choice: "))

        if client_to_deauth == len(target_ap.clients) + 1:
            target_client = input("Input client MAC for deauth: ")
        elif client_to_deauth == len(target_ap.clients) + 2:
            target_client = "ff:ff:ff:ff:ff:ff"
        else:
            target_client = target_ap.clients[client_to_deauth]

    # If we don't have clients on the AP, prompt user for client to deauth
    else:
        target_client = input("Input client MAC for deauth: ")

    deauth(INTERFACE, target_ap.BSSID, target_client)
```

(a) Code for prompt and call of attack

```python
from scapy.all import sendp, Dot11, RadioTap, Dot11Deauth


def deauth(iface: str, BSSID: str, client: str, reason: int = 7):
    packet = RadioTap() / \
        Dot11(type=0, subtype=12, addr1=client, addr2=BSSID, addr3=BSSID) / \
        Dot11Deauth(reason=reason)
    print(f'SENDING DEAUTH to {BSSID}')

    sendp(packet, iface=iface, loop=1, inter=0.01)
```

(b) Code for deauthentication attack function

Figure 10

After a target AP, and a target client has been defined, the "deauth" function, seen in figure 10b, is then called. This function is what creates the appropriate deauthentication packets. The instantiation of the packet works by stacking layers on top of each other. In this case the packet is created by first making a RadioTap layer, which is responsible for sending the packet we create. Then a Scapy "Dot11" layer is made, which represents a basic 802.11 frame. Here the type i set

to 0, indicating a management frame, and the subtype to 12 indicating a deauthentication frame. From here the addresses are defined for the target of the frame. Finally the last layer is added, this being the "Dot11Deauth" layer which here defines a reasoncode for the deauthentication. This reasoncode is set to be 7 for default, which corresponds to the reason "Class 3 frame received from non-associated STA" [18]. In a communications relation, the reasoncode provides important information about the cause of the deauthentication. However it is not an integral part of the deauthentication action itself, therefore the reasoncode is just given as a random number we have chosen. Although it does not have a functional use, the reasoncode could in practice improve a deauthentication attack, by using appropriate reasoncodes, thus further concealing the fact that it is of malicious nature. Subsequently a line is printed to the console notifying the user that a deauthentication attack will commence, combined with the target for the attack. Ultimatly the "sendp" function, sends the beforehand created deauthentication frame.

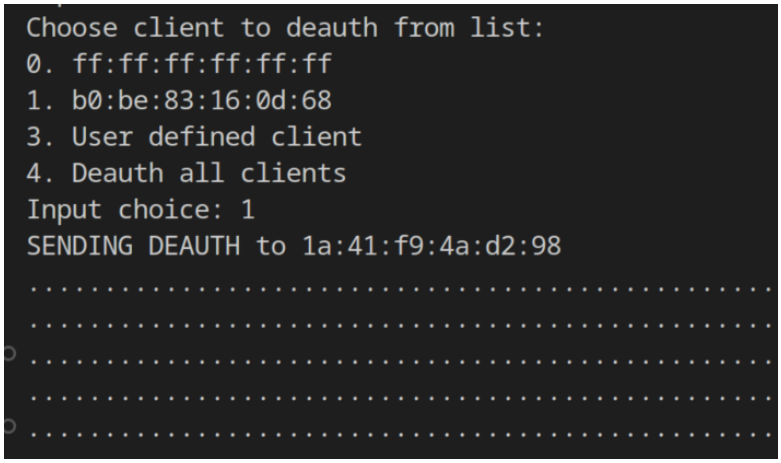

Figure 11: Output when running deauthentication attack

Figure 11 shows the output presented to the user, when performing the deauthentication attack. The user can here choose to selectively deauthenticate a single client. Alternately the user possesses the option to deauthenticate all clients from the targeted AP. Upon the user inputting a choice, the program begins transmitting the packet constructed. The deauthentication attack is continuous and will only stop when interrupted by the user. Our deauthentication implementation has undergone limited testing, highlighting the need for further investigation. Our objective is to assess its functionality in various scenarios, including both successful and unsuccessful cases. Furthermore, we aim to evaluate the performance of the code by conducting tests on edge cases. Additionally, we plan to examine the behavior of the implementation when multiple clients are connected, observing whether all of them are successfully deauthenticated.

4.3 Password cracking

Written by Lucas, reviewed by Oliver

Aircrack-ng is made up of commands that is used to manipulate data, it it not code written directly. That has the consequence that most of the implementation will be in the mapping of networks part. Thus no actual implementation has been done yet in the cracking of passwords, although all commands that are needed have been found. A succesful test has also been completed with the use of Airodump-ng which is a part of Aircrack-ng.

5 Tests

Written by Nicklas, reviewed by Oliver & Lucas
We have tested our implementation.

5.1 Mapping of networks

We have done limited testing of our program. First we tested the mapping of networks as that is a requirement for the deauthentication. We tested on DTU campus which should give us APs that are connected to DTUsecure, eduroam, DTUdevice and DTUguest. It should also give us all clients nearby as their MAC address. The result is shown on figure 12.

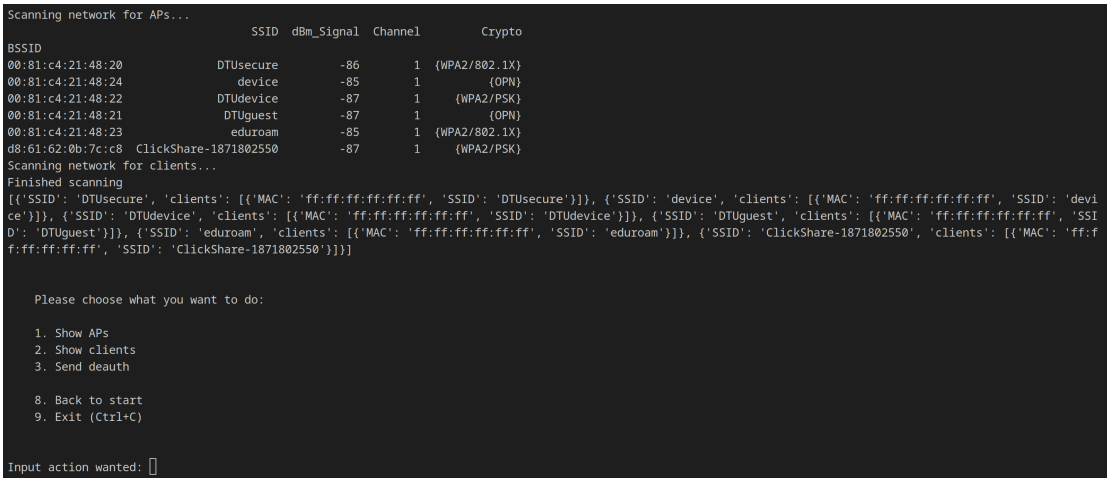

Figure 12: Testing the network mapper

As we can see the test was successfully completed and DTUsecure, eduroam, DTUdevice and DTUguest showed up, together with some other APs unconnected to DTU. We have also found clients nearby. This shows that the mapping of networks has been successfully implemented in its simplest form.

5.2 Deauthentication

Then we tested the deauthentication of a Macbook. It was connected to a smartphone set up as an AP and both the Macbook and the AP showed up. We started a constant ping on the Macbook and afterwards started the deauth attack. The pinging i shown on figure 13

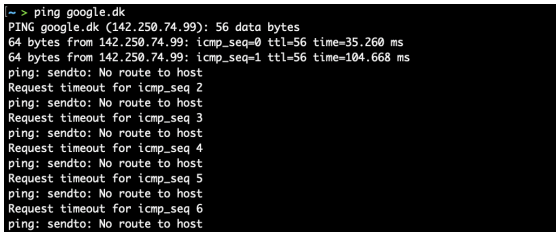

Figure 13: Result from being attacked (Mac)

Figure 13 shows the result of the attack. This attack was run by deauthenticating a single client. it is evident that the pinging to google works in absence of complications, however when the deauthentication attack is deployed, the pinging encounters issues and outputs ”No route

to host". Thereby denying the client service, and disconnecting the client from the internet. . We then tested on a Windows PC as seen on figure 14.

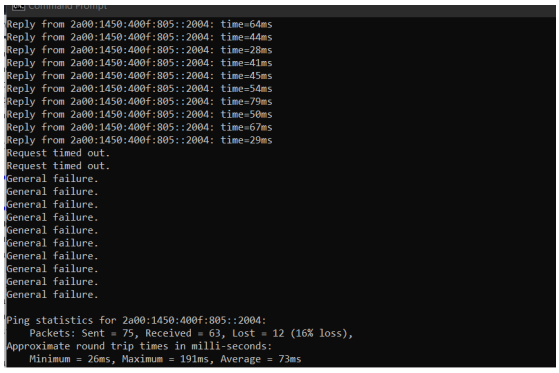

Figure 14: Result from being attacked (Windows)

We can conclude that the deauthentication works on both Windows and MacOS.

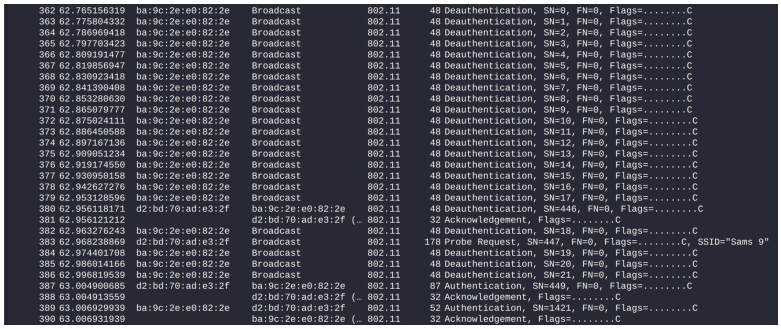

Figure 15: pcap of deauthentication

As well as testing the deauthentication by pinging, we have examined the packets themselves using wireshark. In figure 15 the deauthentication packets sent out can be observed. It is evident that as expected, our program transmits deauthentication packets for the designated target. The deauthentication packets were, in this test, broadcasted in order to deauthenticate all clients connected to the AP. Notice packet number 381, which is an acknowledgement, thus confirming the deauthentication of the client.

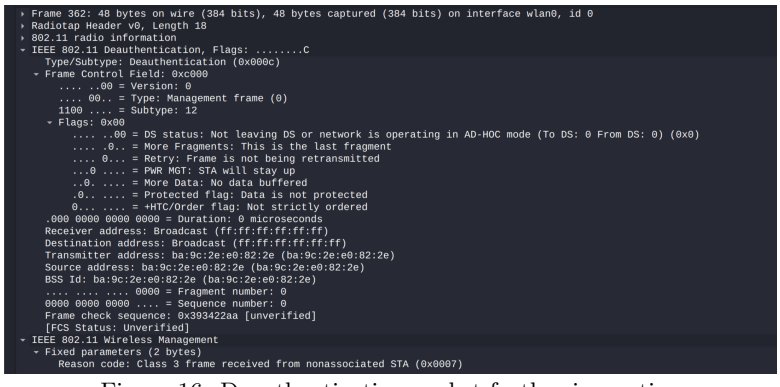

Figure 16: Deauthentication packet further inspection

Delving deeper into a particular deauthentication packet transmitted, as shown in figure 16,

we can observe the specific parameters set during the deauthentication. The reciever, and destination addresses are both set to the broadcast address, and the source address is set to the AP the clients will be disconnected from. Additionally the aforementioned reasoncode for the deauthentication.

5.3 Password Cracking

Written by Lucas

Written by Lucas, reviewed by Oliver
A test has been successfully done where, using Airodump-ng for capturing packets, a network has been cracked and the password found.

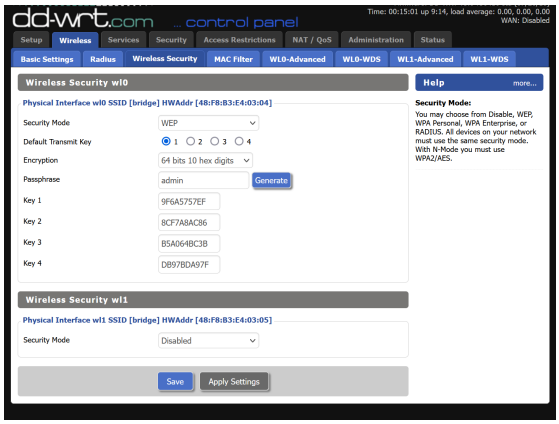

Figure 17: The passwords on the AP

The test was that an AP, with the capability to use WEP, was set up and connected to a wider network on the internet and then made to use WEP with passkeys, as seen on figure 17. Then a computer using Linux was connected to that AP and a constant stream of data was created by watching YouTube. Then another computer was setup to monitor these data-packets and capture those that contain an initialization vector (IV).

There was a small problem with the connection between the AP and the wider internet, so we had to constantly pull the ethernet plug and replug. This was probably because the wider network didn't like using WEP, and that was also why YouTube did not actually work. So the computer sent and received data-packets from the AP only. Enough data-packets where still created and so a large .pcap file was made with about 10.000 IV packets.

Then Aircrack-ng was used on this .pcap file to find the passkey, which it did after 6 secounds. The passkey corresponds with one of the chosen passkeys on the AP. We can see the result on figure 18.

Hopefully we can capture packets containing IV's using the WiFi-scanner and Scapy and place them in a combined file. That would make it simple to crack the WEP algorithm using Aircrack-ng and thereby recreating the passkeys and allowing access to the data-packets.

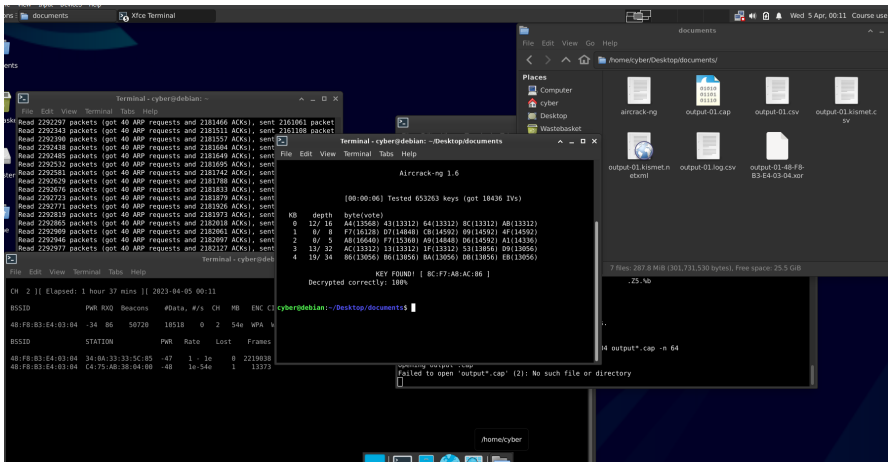

Figure 18: Aircrack-ng finding the password

6 Main Challenges

Written by the group as a whole
Our main challenge in this project is the fact that internet traffic is complicated and made up of many interlocking parts. There is a lot of data being sent constantly and most of it is not relevant to what we are doing. As we are working with a shared medium it can be hard to differentiate between what we are creating and what is part of the standard traffic. That makes it difficult to figure out what is going on 'under the hood' when testing our implementations. Thus it has so far taken much longer to perform some of the basic steps of mapping the network and simple deauthentication. Still this is very much a learn by doing project in the way that we obtain solutions to either tackle or overcome the obstacles and after implementing our WiFi-scanner we have gained valuable knowledge about how data-packets are built.

Likewise the theory behind most of the 802.11 management frames and such used in this project is very new to us, since only a short overview of 802.11 has been provided in our courses where the focus mainly has been on the EAPOL 4-way-handshake within WPA. Thus there is a great gap in knowledge we need to fill as some of the implementation parts require a deep understanding of how e.g. the procedure for deauthentication works.

When cracking WEP, the main challenge is figuring out how to create enough packets consistently that contain IV's and are use the same passkey. Also if this is successfully implemented we also have to recreate Aircrack-ng's PTW method to crack the WEP algorithm. Then this needs to be integrated with the rest of the code.

The code that we have written is also hard to understand for outsiders, so we have to focus on rewriting the code to make it more understandable for everyone. It would also make expansion of features easier in the future.

7 Timeplan

Written by Lucas, reviewed by Oliver & Nicklas
Figure 19 shows the current timeplan for the 3-week period. We are planning on writing on the report concurrently with working on the implementation and tests. We are probably not working on the report every day, but at least multiple times a week. We have set time off for the last week to work on the presentation and on the report to finish it and that we should be done with coding at this time.

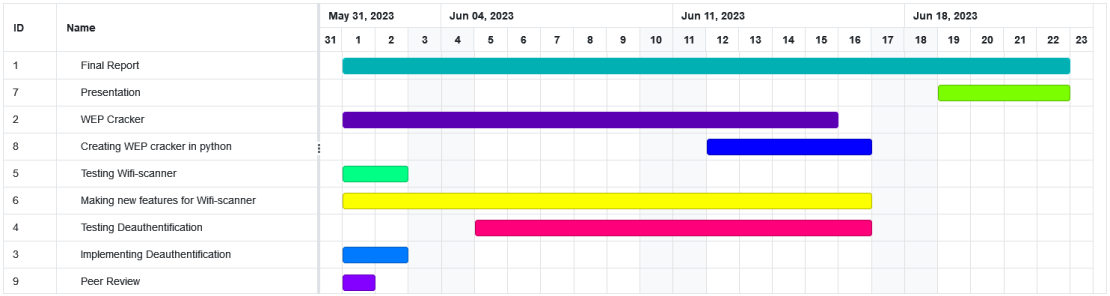

Figure 19: Timeplan

8 Conclusion

Written by whole group
In this project we have investigated network traffic by mapping local WiFi networks using python and the tool Scapy to show all APs connected and the clients connected to these networks. By obtaining knowledge of the WLAN it was managed to send deauthentication frames from a spoofed source address, hence doing a deauthentication attacks to disconnect one or more clients from using wireless networking. By doing this we gained advanced knowledge about different types of frames and subframes in the 802.11 standard as well as a thourough understanding of the attacks. The security algorithm in the 802.11, namely WEP has been examined and exploited to understand fundamental methods as statistical analyzis to exploit flaws in the algorithm. Yet there is still work left in order to explore mitigations for the different attacks. Likewise there is work left in order to ensure stability and integrity of the attacks, and anonymity when performing the attacks. During this project we also increased our knowledge of python programming and networking toolboxes that use python, which gives a broad understanding of how packets are structured.

References

[1] aircrack-ng.org. *Aircrack-ng*. Mar. 2023. URL: https://www.aircrack-ng.org/doku.php?id=aircrack-ng.

[2] Paul Arana. "Benefits and Vulnerabilities of Wi-Fi Protected Access 2 (WPA2)". In: 2006.

[3] Philippe Biondi. *Scapy*. Mar. 2023. URL: https://scapy.net/.

[4] Nikita Borisov, Ian Goldberg, and David Wagner. "Intercepting Mobile Communications: The Insecurity of 802.11". In: *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM* (Sept. 2001). DOI: 10.1145/381677.381695.

[5] Nikita Borisov, Ian Goldberg, and David Wagner. *Security of the WEP algorithm*. Sept. 2001. DOI: 2023-03. URL: https://web.archive.org/web/20160219010026/https://www.isaac.cs.berkeley.edu/isaac/wep-faq.html.

[6] Engineering and Technology History Wiki. *Wireless LAN 802.11 Wi-Fi*. Mar. 2023. URL: https://ethw.org/Wireless_LAN_802.11_Wi-Fi.

[7] Philippe Flajolet and Andrew M. Odlyzko. "Random Mapping Statistics". In: *Advances in Cryptology — EUROCRYPT '89*. Ed. by Jean-Jacques Quisquater and Joos Vandewalle. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 329–354. ISBN: 978-3-540-46885-1.

[8] M. Gast. *802.11 Wireless Networks: The Definitive Guide*. A Nutshell handbook. O'Reilly Media, 2005. ISBN: 9780596100520. URL: https://books.google.dk/books?id=9rHnRzzMHLIC.

[9] "IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 4: Protected Management Frames," in IEEE Std 802.11w-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, and IEEE Std 802.11y-2008) , vol., no., pp.1-111, 30 Sept. 2009, doi: 10.1109/IEEESTD.2009.5278657." In: (2009), p. 12.

[10] "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". In: *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)* (2021), pp. 1–4379. DOI: 10.1109/IEEESTD.2021.9363693.

[11] Sally Jarkas. *RC4 Encryption Algorithm*. URL: https://www.geeksforgeeks.org/rc4-encryption-algorithm/. (accessed: 30.05.2023).

[12] Yogi Kristiyanto and Ernastuti Ernastuti. "Analysis of Deauthentication Attack on IEEE 802.11 Connectivity Based on IoT Technology Using External Penetration Test". In: *CommIT (Communication and Information Technology) Journal* 14 (May 2020), p. 45.

[13] Arash Habibi Lashkari, Mir Mohammad Seyed Danesh, and Behrang Samadi. "A survey on wireless security protocols (WEP, WPA and WPA2/802.11i)". In: *2009 2nd IEEE International Conference on Computer Science and Information Technology*. 2009, pp. 48–52. DOI: 10.1109/ICCSIT.2009.5234856.

[14] Laura Leininger. *2022 Public Wi-Fi Statistics*. Mar. 2023. URL: https://www.highspeedinternet.com/resources/public-wi-fi-statistics.

[15] Microsoft. *Windows Subsystem for Linux Elevation of Privilege Vulnerability*. Jan. 2018. URL: https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2018-0743.

[16] Amit Pindoria. *802.11 Frame Format and Types*. Mar. 2023. URL: https://dot11ap.wordpress.com/802-11-frame-format-and-types/.

[17] Rohith Raj S et al. "SCAPY- A powerful interactive packet manipulation program". In: *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*. 2018, pp. 1–5. DOI: 10.1109/ICNEWS.2018.8903954.

[18] Cisco Systems. *Deauthentication Reason Code Table*. Mar. 2023. URL: https://www.cisco.com/assets/sol/sb/WAP371_Emulators/WAP371_Emulator_v1-0-1-5/help/Apx_ReasonCodes2.html.

[19] Wikipedia. *Wi-Fi deauthentication attack*. Mar. 2023. URL: https://en.wikipedia.org/wiki/Wi-Fi_deauthentication_attack.