

BACHELOR'S THESIS COMPUTING SCIENCE



RADBOUD UNIVERSITY NIJMEGEN

Investigating the behaviour of IoT devices undergoing a deauthentication attack and its countermeasures

Author:
Gabriël Krouwel
s1021541

First supervisor/assessor:
dr. M.G.C. Acar

Second assessor:
dr. ir. E. Poll

May 11, 2023

Abstract

The 802.11 family of standards has been the standard for WiFi security for many decades. However, these standards contain a fundamental flaw that management frames are not authenticated, which we use to execute so-called deauthentication and evil twin attacks. We aim to contribute to the knowledge of attack vectors on some wireless devices, specifically two wireless cameras and two wireless video doorbells. In doing so, we try to find and combine possible countermeasures against these attacks. We investigated the behaviour of these devices during a deauthentication attack combined with an evil twin attack and found that all these devices are susceptible to the attacks. Moreover, we found that we can cause a complete service disruption on these devices, either by sending a continuous stream of deauthentication packets or by setting up an evil twin. We then show that an upgrade in protocol, for example, to the 802.11w standard, is the main solution for these attacks. Nevertheless, for a short-term solution, we can combine countermeasures for detecting these attacks, like detecting an irregular number of packets, to become aware of such an attack. The drawback of this approach is that, although efficient, it does not prevent these attacks and must be dealt with after detection.

Contents

1	Introduction	2
2	Preliminaries	4
2.1	MAC Frames	4
2.2	Device Setup	7
2.2.1	Authentication and Association	7
2.2.2	WPA2 Security	10
3	Deauthentication and Evil Twin Attacks	13
3.1	Deauthentication Attacks	13
3.2	Deauthentication Attack Setup	15
3.3	Network Layout	16
3.3.1	Attack Mitigation	17
3.4	MAC Addresses	18
3.5	Evil Twin Attack	20
4	Results	22
4.1	IP Cameras	22
4.2	Video Doorbells	24
4.3	Discussion	25
4.3.1	Ethics	27
5	Related Work and Countermeasures	28
5.1	Related Attacks	28
5.2	Prevention Countermeasures	29
5.3	Detection Countermeasures	30
5.4	Response Countermeasures	31
6	Conclusions	32
A	Appendix	37
A.1	Deauthentication Packet	37
A.2	Installing Drivers	37
A.3	Connecting Evil Twin to Access Point	38

Chapter 1

Introduction

WiFi allows Internet of Things (IoT) devices to establish a wireless connection between them and access points, which is based on the 802.11 family of standards. There are, however, a few fundamental issues with these standards that can be used to disrupt these devices. For example, some hotels have been fined [1] for using one such flaw to get customers to, instead of connecting to their own network, be forced to connect to the network of the hotel, which is usually a paid service.

In this thesis, we look into these so-called deauthentication attacks, where we disconnect a client from a local network by repeatedly sending specific packets to both the client and the router. In the previous paragraph, we identified a scenario in which these deauthentication attacks can be used. However, deauthentication attacks are the basis of many more such attacks [2] [3].

In fact, in practice, we find that these attacks work on any internet-connected device, as they simply terminate any outgoing internet connection on said device. Deauthentication attacks are thus attacks on availability. This proves to be especially useful on IoT devices that can only serve their purpose with a working connection, for example, wireless cameras and wireless doorbells. In a hypothetical situation, we could, for example, disable all wireless cameras on a single network by executing a deauthentication attack simultaneously on all of these devices. We study the exact consequences of performing such an attack on a few specific devices, namely two wireless cameras and two wireless doorbells.

As we show in this thesis, there are currently no countermeasures implemented against this type of attack if the firmware on the access point has not been upgraded to the 802.11w standard. Moreover, very few tools are required for a deauthentication attack, and these tools are freely available. In order to set up the attack, all we need is the MAC address of the device we want to disconnect and the MAC address of the access point. All of these can be obtained by an adversary through said access point, for example, by a simple MAC-address scan of the network, which we will explain later. We can find these addresses without being connected to the access point, as we can simply scan the frames by being in relative proximity.

When a device wants to disconnect from the network, it sends a deauthentication packet to the router. In this attack, we build these packets to disconnect a client from the network. So, to find proper countermeasures, we cannot simply get rid of these deauthentication packets. We look into a few solutions, most importantly the protected management frames introduced in 802.11w. Nevertheless, this thesis is still very relevant, as 802.11w is rarely used for the security of a network.

Although deauthentication attacks are widely researched, we attack specific IoT devices and look at their behaviour during and after such an attack. Moreover, we combine these attacks with evil twin attacks to block any services the device provides. We do not come up with any new specific countermeasures; however, we do discuss and combine existing countermeasures to prevent these specific attacks. Our two research questions are thus:

- What is the behaviour of IoT cameras and video doorbells when undergoing a deauthentication attack, and does it change when combined with an evil twin attack?
- What countermeasures can be used to prevent or detect a deauthentication attack, and does it work when combined with an evil twin attack on these devices?

We start this thesis by explaining the necessary preliminaries in Chapter 2 in order to understand our attacks. In Chapter 3, we explain how each attack works in technical details and find the required parameters for our attack. Then, in Chapter 4, we show and discuss the results of our attacks. Finally, we discuss some related work and countermeasures for this attack in Chapter 5.

Chapter 2

Preliminaries

2.1 MAC Frames

We start with some preliminaries to be able to understand the setup of our attack, which we discuss in Chapter 3. We are investigating a network within, for example, a household or company, where clients communicate with an access point over the link layer. We are only interested in the frames sent over this connection, so we do not look into, for example, outgoing packets from an access point to another access point outside of the network. These frames are called MAC frames, and they are built up of three parts. A MAC frame consists of a header, a frame body, and a frame check sequence, or FCS for short. The header contains information about the frame itself, which includes the type and sub-type of the frame, whereas the body contains the data that is sent in the frame. Finally, the FCS is calculated over the frame header and body, which can then be used to verify that the frame was received correctly. For this thesis, we are mainly interested in the frame header, but we explain all components of the MAC frame for completeness as we build these MAC frames for our attack. All the notions in this chapter are based on the 802.11 family of standards [4]. The header is structured as follows:

2 bytes	2 bytes	6 bytes	6 bytes
Frame Control	Duration/ID	Address 1	Address 2

6 bytes	2 bytes	6 bytes
Address 3	Sequence Control	Address 4

Figure 2.1: The header of a MAC frame.

It is important to note that Address 4 is not present in every type of MAC frame; this is fully dependent on the 16 bits in the Frame Control. The Frame Control has the following structure:

2 bits	2 bits	4 bits	1 bit	1 bit	1 bit
Protocol Version	Type	Sub-type	To DS	From DS	More Frag

1 bit	1 bit	1 bit	1 bit	1 bit
Retry	Power Mgmt	More Data	WEP	Order

Figure 2.2: Frame Control of a MAC frame.

- **Protocol Version.** This field specifies what the current Protocol Version is. For the network we attack, this is fixed to be 00.
- **Type.** The Type indicates the type of MAC frame, with the possibilities being a management frame (00), a control frame (01), and a data frame (10). For this thesis, we are only interested in the management frames.
- **Sub-type.** Just like the Type field, this field specifies the Sub-type of the MAC frame. Note that each type of MAC frame has specific sub-type meanings, so we only discuss the meaning of sub-types for the management frame.

0000	Association Request
0001	Association Response
0010	Re-association Request
0011	Re-association Response
0100	Probe Request
0110-0111	Reserved
1000	Beacon
1001	ATIM
1010	Disassociation
1011	Authentication
1100	Deauthentication
1101	Action
1110-1111	Reserved

Figure 2.3: Definition of the bits in the sub-type field.

For our attack, only Deauthentication (1100) is needed to perform the attack; however, some of the other frames are required for setting up connections in the local network.

- **To DS and From DS.** These two bits define the meaning of the address fields and whether the fourth address is present. These values are represented in the following table:

x	Address 1	Address 2	Address 3	Address 4
00	DA	SA	BSSID	x
01	DA	BSSID	SA	x
10	BSSID	SA	DA	x
11	RA	TA	DA	SA

Table 2.1: Definition of address field based on DS bits.

In this table, DA stands for Destination Address, SA for Source Address, TA for Transmitting Station Address and finally RA for Receiving Station Address. The Basic Service Set Identifier (BSSID) is the MAC address of the access point in this case. For the deauthentication frames, we will only use the 00 entry in the table.

- **More Fragments.** This bit indicates whether there are other fragments associated with this frame. For our scenario, this is never the case, so we set it to 0.
- **Retry.** If this bit is set, then the current frame is a retransmission. Once again, this is never the case for our attack, hence we set it to 0.
- **Power Management.** When set, this field will change the mode of a station to power-saving mode; otherwise, it will stay active. We will set this to 0, but it is not important for this attack.
- **More Data.** Similar to more fragments, this bit indicates whether the sender sends more data than just the current frame. This bit is also set to 0.
- **WEP.** This field indicates that the standard 802.11 security is used, which is set to 0 for our packets as we are sending them over WPA2 instead of WEP.
- **Order.** Finally, the last field indicates whether the frames must be processed in a strict order. We will only send independent packets multiple times, so this bit is set to 0.

The remaining fields are defined as follows:

- **Duration/ID.** In control frames, this field contains an association identity. Nevertheless, we are only working with management frames, so this field always contains the duration value of the frame.
- **Address fields.** Recall that we will only use the 00 entry of the DS-table. This means that the first address, DA, is the destination address, being either the MAC address of the access point or the device. The second address, SA, is respectively the MAC address of the device or access point. Finally, the third address is simply the MAC address of the access point.
- **Sequence Control.** This field consists of a 4-bit Fragment Number and a 12-bit Sequence Number. The first number simply preserves the order of the fragments within a transmission, whereas the latter tracks the number of transmissions.

Now that we have explained all the fields in the header of the MAC frame, we have two other parts of the frame: the Frame Body and the Frame Check Sequence.

- **Frame Body.** This field contains the data sent with the frame, which is of variable length. We will only be setting the reason code for the deauthentication frame in this field, which is fixed for every fragment and thus a fixed field in our fragments.
- **Frame Check Sequence.** The FCS field contains a 32-bit code to ensure that no errors occur during transmission. We will not explain the formula for calculating this code, as it will be fully automated; however, more information can be found in [4].

2.2 Device Setup

2.2.1 Authentication and Association

Since we are working with wireless IoT devices, we will now explain our network layout for these devices. Consider a scenario where the IoT device, in this case either a camera or a video doorbell, has not been setup to work with an existing network. This network is simply an access point, for example, a router, connected to some devices and serving as a gateway to the internet.

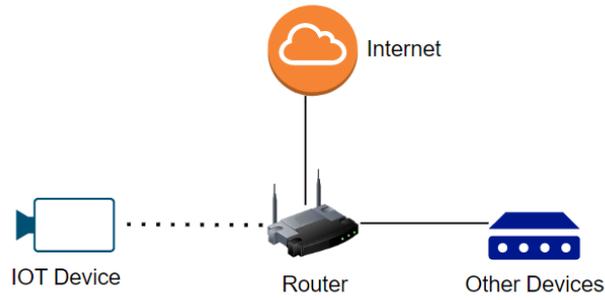


Figure 2.4: The standard setup for a network.

Note that all connected devices (in essence, the "other devices" in the diagram) are in so-called states three or four, whereas the IoT device is in state one, which is defined in Figure 2.5.

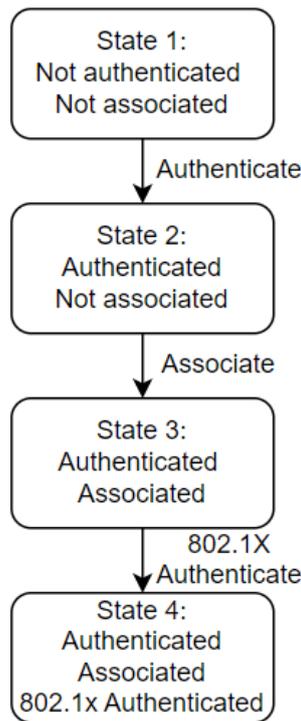


Figure 2.5: The relation between the states.

As soon as a device is in state three, it is able to send data to the access point. However, if the access point has any security measures enabled, like WPA2, the device is required to switch to state four before being able to send data. The protocol used by a device to get from state one to state three is shown in Figure 2.6.

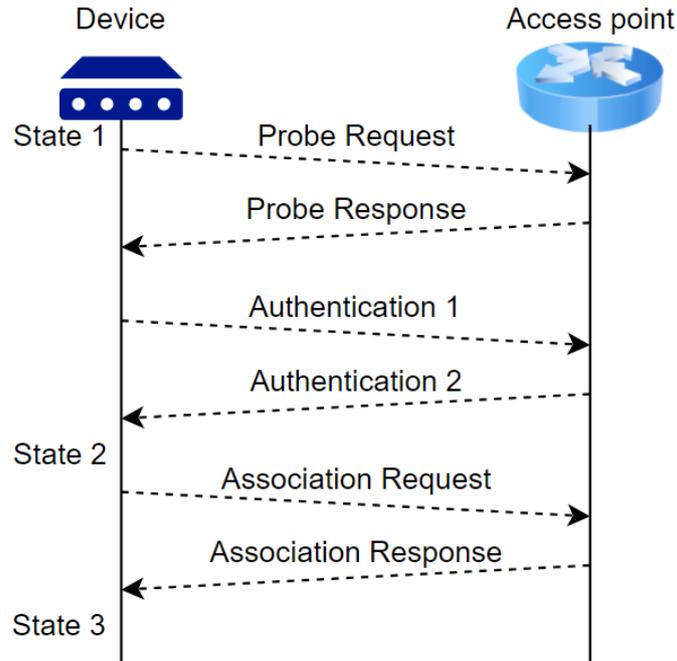


Figure 2.6: The protocol for state 1 to state 3.

- **Probe Request.** The device sends a probe request with information about the supported data rates of the device to FF:FF:FF:FF:FF:FF on the link layer. All access points within a certain distance of the device will respond to this request.
- **Probe Response.** The access point sends back a probe response, including the SSID, compatible data rates, and, if applicable, additional requirements for 802.11 security.
- **Authentication 1.** The device sends an authentication frame to the access point with a specific sequence code (SC).

- **Authentication 2.** The access point opens the authentication and sends back an authentication frame with the sequence code SC+1. If the access point does not receive an authentication frame from the device, it will send a deauthentication frame to the device instead. This does, however, allow any device to be authenticated at multiple access points.
- **Association Request.** The device is now in state two, and it will thus determine which AP it wants to associate with. It then sends an association request, which contains the 802.11 capabilities, to the access point.
- **Association Response.** If the capabilities match, the access point will send an association response, which has a corresponding association ID within the access point for the device. Once again, if the access point does not receive an association frame, it will send a disassociation frame instead. This ensures that any single device cannot be associated with multiple access points at the same time.

After receiving this last frame, the device is in state three, and the protocol is finished. Data can now be exchanged between the access point and the device, unless extra 802.11 security is required.

Note that after sending a deauthentication management frame to the device, it gets kicked back to state one. If we were to send a disassociation management frame to the device, it would go back to state two instead. These disassociation frames can be used in what are known as disassociation attacks [5].

2.2.2 WPA2 Security

We will now discuss the details of the WPA2 protocol, which is part of the 802.11i standard [6], as this is required for a device to switch from state three to state four. A device in state three will send any frames to the access point in plaintext, so this extra layer of protection was introduced to protect the confidentiality and integrity of the data in any frame. However, this protection was not applied to the management frames, thereby meaning these frames are still being sent in plaintext, which allows us to spoof these frames. First, we need some definitions of the keys used in the WPA2 protocol:

- **Pairwise Master Key.** The Pairwise Master Key (PMK) is the pre-shared secret for the device and the access point. This is, for example, the password required when connecting to any private network.
- **Group Master Key.** The Group Master Key (GMK) is a key that only the access point knows and is used to generate the other required keys in the protocol.

- Pairwise Transient Key.** The Pairwise Transient Key (PTK) is the encryption key for all the unicast traffic between a device and the access point, being unique for each device in state three with respect to that access point. The PTK is calculated as follows: $PTK = PRF(PMK + ANonce + SNonce + MAC(AP) + MAC(SA))$, where PRF is simply a pseudo-random function. Note that ANonce and SNonce are just random numbers, generated respectively by the access point and the device.
- Group Temporal Key.** The Group Temporal Key (GTK) is the encryption key for all broadcast and multicast traffic between multiple devices and the access point, which is unique to the access point. The access point generates this key independently of the devices.

Figure 2.7 shows the WPA2 protocol with the discussed definitions.

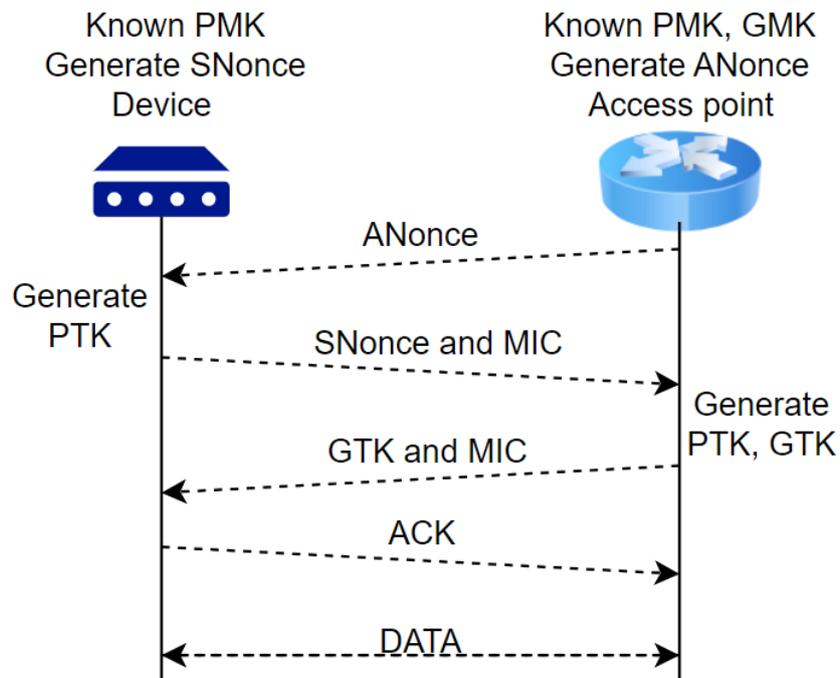


Figure 2.7: The handshake for the WPA2 protocol.

- **Message 1: ANonce.** The access point sends the ANonce to the device, which now has all the information required to generate the PTK. This is because the client knows the MAC address of the access point because the device is associated with it.
- **Message 2: SNonce and MIC.** The device then sends the SNonce to the access point, along with a Message Integrity Check (MIC), which is added to ensure that no message is modified or corrupted during the transmission. With the SNonce, the access point can now calculate the PTK as well, since it knows the MAC address of the device through the association process.
- **Message 3: GTK and MIC.** The access point generates the GTK from the GMK, which it encrypts using the PTK. It then sends the pair of encrypted GTK and MIC back to the device, once again to ensure that the encrypted GTK was not modified or corrupted.
- **Message 4: ACK.** Lastly, the device simply sends an acknowledgment to the access point that all keys have been installed.

After receiving the fourth message, the access point will open the virtual control port for the device, which was previously closed, blocking all traffic coming into the access point. Finally, encrypted data can be sent between the access point and the device. The details of the WPA2 protocol will become especially important in the countermeasures section.

Chapter 3

Deauthentication and Evil Twin Attacks

3.1 Deauthentication Attacks

Now that we have explained the necessary preliminaries, we can focus on the actual deauthentication attacks. A deauthentication attack is a type of denial-of-service attack that disconnects a device from the access point it is connected to. Using our earlier defined definitions, it is the act of putting an authenticated and associated (and potentially 802.1X authenticated) device in state three or four back to unauthenticated and disassociated, which is the first state. In that sense, this attack is quite similar to a disassociation attack, which kicks the device back to the second state instead [5]. Subsequently, a deauthentication attack can be used on this device to kick it back to the first state. Recall that a device in the first state is unable to send data to the access point, meaning that it is unable to communicate with any other device over the LAN using this access point until it is back in states three or four.

Under normal circumstances, when a device wants to disconnect from an access point, it sends a deauthentication packet to the access point it is connected to with a specified reason code in the field body. Several such reason codes exist, but they do not matter for our attack, as we only use reason code 1, which means that the reason is unspecified [4]. Moreover, deauthentication frames are notifications, meaning that the access point will acknowledge the frame and accordingly disconnect the device from the access point, regardless of the reason code. If, instead, the access point wants to disconnect the client from itself, it sends a deauthentication frame with a specific reason code to the device. As this is once again a notification, the device will acknowledge this disconnection. However, the device will try to reconnect to the access point, as we can also see from our packet captures in the next chapter.

As we have mentioned, deauthentication frames are sent over the LAN without using encryption or authentication, meaning they are sent in plaintext. This allows any adversary to read, change, and, most importantly, create these packets and send them over the LAN. An adversary is therefore able to create such a frame with an arbitrary source and destination MAC address, which allows one to spoof any of these MAC addresses. An adversary can now execute a successful deauthentication attack by doing the following:

- (1) The adversary creates deauthentication frames with the source address as the MAC address of the device to be attacked. The destination address is set to the MAC address of the access point, which is also the BSSID. Upon receiving these frames, the access point acknowledges the frames as deauthentication frames sent from the device and henceforth disconnects the device from the access point.
- (2) The adversary also creates deauthentication frames with the source address as the MAC address of the access point, which is also the BSSID. The destination address is then set to the MAC address of the device we want to attack. Once again, the device acknowledges the frames and it thus registers as being disconnected from the access point. Note that the device is not actually disconnected from the access point, as the access point has not sent or received any deauthentication frames.

For a deauthentication attack to be efficient, both types of deauthentication frames must be sent. Suppose that the adversary only sends frames of type (1), the device does not know it is disconnected from the access point and hence keeps sending data to the access point. It thus tries to reconnect instantly, making it less efficient. Suppose instead that the adversary only sends frames of type (2), then the device tries to reconnect to the access point as mentioned earlier. However, as the device is already authenticated and associated with the access point, it is immediately able to send data to the access point.

An adversary is also able to mount a deauthentication attack on the whole network instead of just a single device. The adversary can create packets of type (2), where the destination address is FF:FF:FF:FF:FF:FF instead of the devices' MAC address. This means that the deauthentication packets are broadcast to all devices on the network; hence, it should disconnect all devices from the network. However, as we only want to attack specific devices and do not want to cause any disruption to other devices, we do not broadcast deauthentication frames.

Although a deauthentication attack is an attack on availability, it is also the basis for a few other attacks, of which one of the most prominent is cracking 802.1X authentication. When a device is kicked back to the first state, it will try to re-authenticate and re-associate with the access point it was connected to before the deauthentication attack. It will also perform a re-authentication for 802.1X security, which means that the 4-way handshake, as discussed in Section 2.2.2, will be done again. An adversary is then able to capture this handshake, which can be used for subsequent attacks on the LAN. In fact, if the LAN uses WEP, the captured 4-way handshake is enough to break the encryption on all frames sent over the LAN within a few minutes [3]. For WPA and WPA2, we can use a list of known passwords to break the encryption using the captured handshake [7], if the password is on our list of known passwords.

Another attack that can be used in combination with the deauthentication attack is the so-called Evil Twin attack. When a device gets disconnected from an access point because of a deauthentication packet, it will try to reconnect to the access point. However, an adversary could, in the meantime, setup a rogue access point with the same details as the original access point. The devices could now automatically connect to this rogue access point instead [8], which the adversary can use to, for example, drop or reroute packets being sent through this access point.

3.2 Deauthentication Attack Setup

We shall now delve into the details of how to perform a deauthentication attack. It is very important to note that a deauthentication attack is only effective on a device if it is in state two or three (or state four if required) with respect to the access point. Note that we can execute such attacks from outside the network [9], which allows this attack to work on any network.

An adversary should be able to analyse networks so that it can retrieve MAC addresses from them. We explain some tools that can be used to retrieve them, but a network layout is unique for every situation, thus requiring some understanding of network structure. Moreover, an adversary must be able to install and use these or similar tools to pull off the attacks. Finally, an adversary has to be able to analyse packets over a network, as we can determine the results of our attack and the behaviour of the devices from these packets.

We are attacking IoT cameras and video doorbells. An adversary that wants unauthorised access to a site can use this attack to disable these devices. This could allow such an adversary to perform malicious actions without being recorded, like, for example, stealing items from an otherwise monitored location.

For a deauthentication attack, we need an adapter that is able to both sniff and inject packets on a network. We are using the ALFA AWUS036ACH adapter [10], which supports dual-band operation over both 2.4 GHz and 5 GHz frequencies. None of the currently available adapters can sniff and inject over both frequencies at the same time because an adapter can only be associated with a single access point at the same time, whereas the two frequencies correspond to two different access points. The instructions for installing this adapter can be found in Appendix A.2.

Apart from the adapter, we also need a tool associated with it to sniff and inject packets onto a network. There are several tools available for this, such as ScaPy [11], but we are using aircrack-ng [12]. Aircrack-ng is a suite of tools for network security that, as we have already seen, can be used to crack WEP, WPA1, and WPA2-PSK, but it can also be used to sniff packets and inject deauthentication packets onto a network. Aircrack-ng is already installed on Kali Linux, but it can be installed using the command:

```
sudo apt install aircrack-ng.
```

The command: `sudo airmon-ng start wlan0`, now sets the adapter to monitor mode, allowing it to sniff and inject packets. Using aircrack-ng, we can now launch a deauthentication attack with just a single command:

```
sudo aireplay-ng --deauth 0 -c [MAC of device] -a  
[MAC of access point] [interface]
```

Although the interface names can be found using the command `iwconfig`, the MAC addresses have to be retrieved using other methods, which can be done using other tools from the aircrack-ng suite. As finding these MAC addresses is completely dependent on the layout of the LAN, we first explain the layout we are using.

Summarising all the steps, we must first set our adapter to monitor mode, which allows it to sniff and inject packets. Then, we must retrieve the MAC addresses of the devices and the access point we want to attack. Finally, we can use the `aireplay-ng` command to launch our attack. We will now describe retrieving the MAC addresses, which we do separately as it is specific to the network layout.

3.3 Network Layout

As we have discussed, for our attack to work, we need an adversary that is close enough to the access point to read the frames sent from it. Hence, when combining this with our network, we get the layout in Figure 3.1. We use this specific network as it is the only private network available to us for testing. We avoided testing this attack on other networks for ethical reasons (see Section 4.3.1).

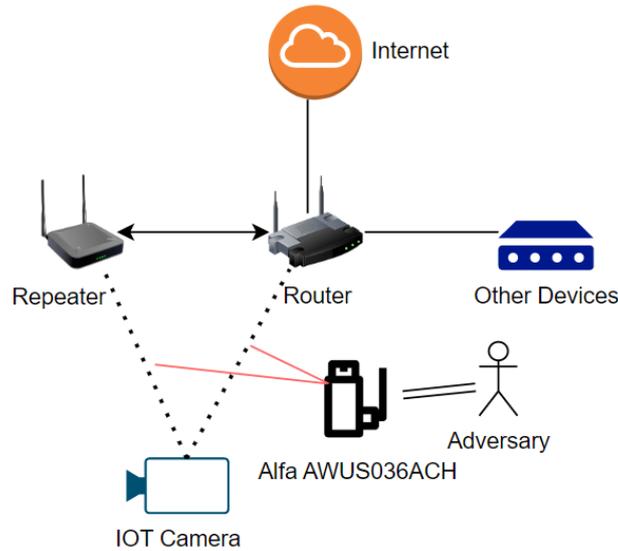


Figure 3.1: Network layout for our attack.

The router is a dual-band FRITZ!Box 7590 router, which supports both 2.4 GHz and 5 GHz frequencies. There is also a repeater, which is basically a WiFi extender, connected to the router. This, however, also means that the IoT device is able to connect to either the router or the repeater if it is in range of both, which complicates the situation, as will be explained in the next subsection. We also have an adversary with an Alfa adapter on the network, able to sniff and inject packets on it.

In this layout, the IoT camera is connected to the router, albeit not directly. The camera uses the access point of the repeater to send data to the router, which is acting as an intermediary. The repeater simply allows an extended connection provided by the router; however, a device is still able to connect directly to the router instead.

3.3.1 Attack Mitigation

Suppose that a device is connected to a repeater. If we tear down this connection using a deauthentication attack, the device can automatically switch to a direct connection with a router instead. If we then execute a deauthentication attack on this connection, it can switch back to the repeater. To solve this issue, we must use a deauthentication attack on both of these connections at the same time. That way, the device cannot connect to the router at all.

The problem from the last paragraph can actually be generalised to broader situations: having more repeaters means that the adversary must deauthenticate more connections for the device to become disconnected. Moreover, if the device supports both 2.4 GHz and 5 GHz frequencies and so does the access point, the device can automatically switch between these two frequencies if either of them is down [13]. Although the devices attacked in this thesis are not capable of supporting both 2.4 GHz and 5 GHz, this problem could be solved using the previously described solution: use a deauthentication attack on both the 2.4 GHz and 5 GHz frequencies at the same time.

In summary, we need to find the MAC addresses of the router, the repeater, and the device. Using these MAC addresses, we can launch two deauthentication attacks on both the router and repeater to disconnect the device. Sometimes, the device's MAC address is on the label on the device; nevertheless, we assume this is not the case and thus find the MAC addresses using the aircrack-ng suite.

3.4 MAC Addresses

Recall that we have already set the adapter to monitor mode in the previous section, so we can use the following command to find all access points in proximity, along with their corresponding MAC address:

```
sudo airodump-ng wlan0.
```

For our setup, this gives the following output:

```
06 -41 2 0 0 1 720 WPA2 CCMP PSK FRITZ!Box 7590 XU 2,4Ghz
33 -19 5 10 0 1 195 WPA2 CCMP PSK FRITZ!Box 7590 XU 2,4Ghz
```

Figure 3.2: Output of the airodump-ng command.

From these results, we can gather that the MAC addresses of the router and the repeater are either XX:33 or XX:06 (we mask the MAC addresses), as our devices are only able to communicate over the 2.4 GHz frequency. Using the signal strength (the second column), we find that the former is the MAC address of the router, whereas the latter is the MAC address of the repeater. Nevertheless, since we have to use a deauthentication attack on both access points, whether either MAC address is the router or repeater does not matter. Finally, we have to retrieve the MAC address of each individual device, which we have to do separately for each device. The devices we attack can be found in Table 3.1.

IoT devices
Eufy Indoor 2K Camera
Monvelli Indoor IP Camera
LG Wireless Video Doorbell
Nikkei BELL4 Smart Video Doorbell

Table 3.1: The IoT devices used in this study.

The first device we attack is the Eufy Indoor 2K Camera. This device only supports 2.4 GHz, so we are able to find the MAC address of this device by sniffing the packets over the access point with MAC address XX:33. To sniff these packets, we can use this command:

```
sudo airodump-ng wlan0 --bssid [BSSID of access point]
```

This gives us the following result:

```
CH 1 ][ Elapsed: 48 s ][ 2023-02-23 07:09
BSSID          PWR RXQ Beacons  #Data, #/s  CH  MB  ENC CIPHER AUTH ESSID
[REDACTED] 33 -19  0    29    39  0  1 195  WPA2 CCMP  PSK FRITZ!Box 7590 XU 2,4Ghz
BSSID          STATION          PWR  Rate  Lost  Frames Notes Probes
[REDACTED] 33 [REDACTED] 5F -21  24e-24e  0    7
[REDACTED] 33 [REDACTED] 08 -20  1e-24e  0    16
```

Figure 3.3: Output of the airodump-ng command on the router.

We can see several MAC addresses as a result of this command. In order to find the right MAC address, we can use a MAC address lookup, like Maclookup [14]. We then find that the MAC address XX:5F belongs to Smart Innovation LLC, which is the vendor of the Eufy camera [15].

The second device we attack is a Monvelli Indoor IP Camera, which once again only supports 2.4 GHz. Using the same command, we get these results:

```
CH 10 ][ Elapsed: 12 s ][ 2023-02-23 07:13
BSSID          PWR Beacons  #Data, #/s  CH  MB  ENC CIPHER AUTH ESSID
[REDACTED] 33 -70    4    51  0  1 195  WPA2 CCMP  PSK FRITZ!Box 7590 XU 2,4Ghz
BSSID          STATION          PWR  Rate  Lost  Frames Notes Probes
[REDACTED] 33 [REDACTED] C6 -31  24e-24e  0    44
[REDACTED] 33 [REDACTED] 08 -18  1e- 1e  0    2
```

Figure 3.4: Output of the airodump-ng command on the router.

Although we could find the MAC address by looking at the unique MAC addresses compared to the results of the previous paragraph, a simple MAC address lookup tells us that XX:C6 is the MAC address of the camera.

The third device is a LG Wireless Video Doorbell, which only supports 2.4 GHz. The command and a similar MAC lookup tells us that the MAC address of this video doorbell is XX:3C.

The final device is a Nikkei BELL4 Smart Video Doorbell, which also only supports 2.4 GHz. The MAC address of this device is XX:26.

3.5 Evil Twin Attack

Before we move on to the results of our deauthentication attack, we study evil twin attacks, which allows us to get more insight into our results. We found some interesting results with an evil twin combined with the deauthentication attack, hence, we are including it here.

The purpose of an evil twin attack is to get any target to connect to a rogue access point, or specifically, an evil twin access point, instead of the original access point it was connected to. An adversary does so by creating an access point with the same details as the original access point and then trying to get a device to automatically connect to it. Suppose a device gets temporarily disconnected from the access point, which can be done via a deauthentication attack; it will reconnect to the access point with the same details. However, as there are two such access points, the device will connect to the access point closest in proximity [8]. The adversary can thus make sure that the evil twin is closest in proximity, such that the device connects to the evil twin.

An evil twin attack can be completely executed using the Airgeddon bash script [16]. We are, however, unable to use this tool in this thesis. This is because Airgeddon launches a deauthentication attack with the destination address set to FF:FF:FF:FF:FF:FF, or broadcasting the message to every device on the network. As we only want to attack a single device, we will manually set up the evil twin access point using airbase-ng. This twin access point simply needs to match the MAC address and the channel of the real access point, but to arouse less suspicion, it should also match the Extended Service Set Identifier (ESSID). This is a unique identifier assigned to a wireless network, which in natural language is just the network name.

For setting up an evil twin access point, we can use the command:
`sudo airbase-ng -a [MAC address of access point] -e [ESSID of access point] -c [Channel of access point] wlan0`

This command will create an access point with the same MAC address, the same ESSID, and the same channel as the access point. If we also want to send these packets to the original access point, in essence, to become a man in the middle, we can set up bridges between our evil twin and the real access point. The commands for this can be found in Appendix A.3. If we set up these bridges, our evil twin serves as a man in the middle for the connection between the device and the real access point. Instead, we are also able to reroute the packets to a completely different access point.

If we are able to get the devices to connect to our fake network, we successfully hijack the connection and can thus read (however, the data packets are still encrypted), change, and drop any packets the device sends. In this scenario, we have our setup shown in Figure 3.5, where AP stands for access point.

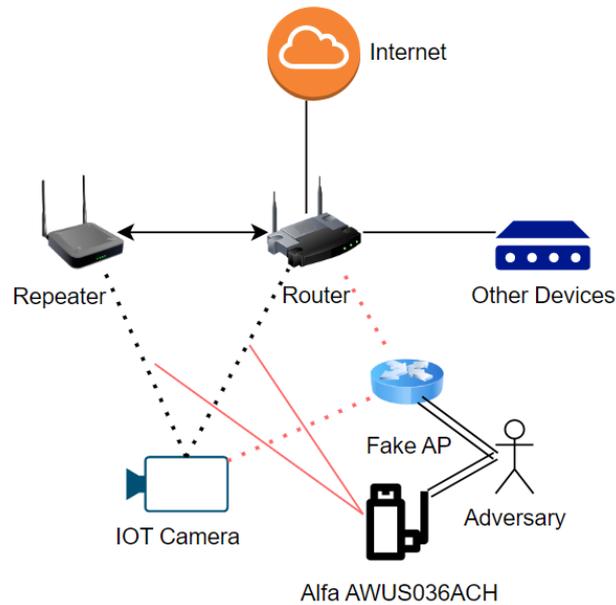


Figure 3.5: The network layout with an evil twin.

In summary, we thus use the Alfa adapter to launch a deauthentication attack on the connections between the camera and the router/repeater (the red lines). Then, we try to set up a connection between the device and our evil twin (the red dotted lines). This way, the adversary is in complete control of all inbound and outbound communication from the device.

Chapter 4

Results

Now that we have acquired the target device's MAC addresses, we use the following command to mount a deauthentication attack:

```
sudo aireplay-ng --deauth 0 -c [MAC of device] -a  
[MAC of access point] [interface]
```

We run this deauthentication attack three times on each individual device to make sure that the behaviour is not random. We ensure that the device is connected to the network and actively streaming footage over the network through the dedicated app before each separate attack.

We analyse our results using Wireshark [17], which can display all frames that our adapter has sniffed from the network. We investigate these frames to determine the behaviour of the devices. We want to find packets being sent between the access point and the device after the final deauthentication packets are sent. Any data packets being sent mean that the device is successfully uploading footage again. Moreover, if we capture certain management frames, like the association and authentication frames, we know that the device tries to get back to state three with respect to that access point, which is the authenticated and associated state.

4.1 IP Cameras

Eufy Indoor 2K Camera. We found that the footage was not transmitted anymore almost instantly after initiating the deauthentication attack on this camera. The device tries to reconnect a few times at the start, as seen in Figure 4.1. However, after several failed attempts at reconnecting, the device will either not try to reconnect and in fact require a complete reset before it is able to reconnect, or it will reconnect after an extended period of time, well over ten minutes. The results after reset are shown in Figure 4.2. We have added in the packet numbers (column 1) and timing (column 2) of the packets to show it does not reconnect automatically, even after several minutes.

1191	12.271467407		5f		06	Deauthentication, SN=21, FN=0, Flags=.....
1194	12.274302605		5f		33	Deauthentication, SN=5, FN=0, Flags=.....
1196	12.275733756		5f			Probe Request, SN=110, FN=0, Flags=.....C,
1197	12.277579376		06		5f	Deauthentication, SN=22, FN=0, Flags=.....
1198	12.279570544		33		5f	Deauthentication, SN=6, FN=0, Flags=.....
1199	12.280106575		5f		06	Deauthentication, SN=23, FN=0, Flags=.....
1201	12.282429056		5f		33	Deauthentication, SN=7, FN=0, Flags=.....
1202	12.283786274		06		5f	Deauthentication, SN=24, FN=0, Flags=.....
1203	12.285178524		33		5f	Probe Response, SN=504, FN=0, Flags=.....C,
1206	12.287625196		5f		06	Deauthentication, SN=25, FN=0, Flags=.....
1209	12.288895750		33		5f	Authentication, SN=256, FN=0, Flags=.....C
1212	12.289841630		5f		33	Authentication, SN=111, FN=0, Flags=...R...C
1214	12.291810812		33		5f	Deauthentication, SN=8, FN=0, Flags=.....
1215	12.293065131		5f		33	Association Request, SN=112, FN=0, Flags=.....
1216	12.294119217		5f		33	Deauthentication, SN=9, FN=0, Flags=.....
1218	12.297633889		06		5f	Deauthentication, SN=26, FN=0, Flags=.....

Figure 4.1: Reconnection attempts from the Eufy Camera.
The marked packets are frames sent by the camera trying to reconnect.

12010	37.552472177		5f		06	Deauthentication, SN=3919, FN=0, Flags=.....
12012	37.556490380		06		5f	Deauthentication, SN=3920, FN=0, Flags=.....
12014	37.560000889		5f		06	Deauthentication, SN=3921, FN=0, Flags=.....
12026	37.662665378		06		5f	Deauthentication, SN=3922, FN=0, Flags=.....
22851	159.7995848...		5f	Broadcast		QoS Data, SN=2, FN=0, Flags=.p.....TC
22856	159.8065872...		5f	Broadcast		Data, SN=3004, FN=0, Flags=.p....F.C
22859	159.8065909...		03		5f	QoS Data, SN=1, FN=0, Flags=.p....F.C
22863	159.8099403...		5f	Broadcast		Data, SN=3589, FN=0, Flags=.p....F.C
23049	161.5179649...		5f	Broadcast		Data, SN=3020, FN=0, Flags=.p....F.C

Figure 4.2: behaviour of the Eufy Camera after a manual reset.
The marked frames are data being sent by the camera.

Monvelli Indoor IP Camera. Even though this camera was also immediately unable to send any more data to the access point, the device did continuously try to reconnect to the access point every time. We can see this from the fact that this camera was also able to reconnect almost instantly without having to reset the camera; the results can be found in Figure 4.3. We can see that the camera only starts sending data a few seconds after the last deauthentication packet.

22974	64.613319140		33		c6	Deauthentication, SN=3938, FN=0, Flags=.....
22975	64.615857093		c6		33	Deauthentication, SN=3939, FN=0, Flags=.....
22976	64.619610881		33		c6	Deauthentication, SN=3940, FN=0, Flags=.....
22981	66.437116494		03		c6	QoS Data, SN=7, FN=0, Flags=.p....F.C
22983	66.437118840		c6		03	QoS Data, SN=1823, FN=0, Flags=.p.....TC
22987	66.462361431		c6		03	QoS Data, SN=1824, FN=0, Flags=.p.....TC
22992	66.473054328		03		c6	QoS Data, SN=8, FN=0, Flags=.p..R.F.C
23030	66.966276269		c6		03	QoS Data, SN=1827, FN=0, Flags=.p.....TC

Figure 4.3: Reconnection of the Monvelli Camera.
The marked frames are the data being sent by the camera.

We also found that both cameras almost immediately connected to the evil twin after a deauthentication attack, even with just a single deauthentication packet sent to the devices. Moreover, we found that as long as this evil twin is online, the camera will not reconnect to a different access point, even though the camera is not able to send data when connected to the evil twin.

4.2 Video Doorbells

Similarly to both cameras, both video doorbells weren't able to send any more data to the access point due to the deauthentication attack.

LG Wireless Video Doorbell. This doorbell only tries to reconnect very sporadically and even requires a reset to reconnect, as can be seen in Figure 4.4.

Nikkei BELL4 Smart Video Doorbell. In contrast to the other doorbell, this one reconnects almost immediately, as can be seen in Figure 4.5.

26642	60.555046524	3c		33	Deauthentication, SN=2125, FN=0, Flags=...
26644	60.561361799	3c		3c	Deauthentication, SN=2126, FN=0, Flags=...
26647	60.725152447	3c		06	Association Request, SN=71, FN=0, Flags=...
26649	60.728457962	06		3c	Deauthentication, SN=3382, FN=0, Flags=...
26651	61.045955089	3c		06	Association Request, SN=72, FN=0, Flags=...
26653	61.045963624	06		3c	Deauthentication, SN=3383, FN=0, Flags=...
26681	61.631207200	3c	Broadcast		Probe Request, SN=74, FN=0, Flags=.....C
26706	61.711577855	3c	Broadcast		Probe Request, SN=76, FN=0, Flags=.....C
26770	62.081544659	3c	Broadcast		Probe Request, SN=85, FN=0, Flags=.....C
26771	62.211400860	3c		33	Probe Request, SN=88, FN=0, Flags=.....C
26775	62.255110356	3c		33	Authentication, SN=89, FN=0, Flags=.....C
26777	62.257718382	3c		3c	Authentication, SN=256, FN=0, Flags=.....C
26779	62.258986967	3c		33	Association Request, SN=90, FN=0, Flags=...
26781	62.262968876	33		3c	Association Response, SN=258, FN=0, Flags=...
26790	62.316883592	33		3c	Action, SN=259, FN=0, Flags=.....C
26792	62.318564944	3c		33	Action, SN=2, FN=0, Flags=.....C
26800	62.330827722	33		3c	Action, SN=260, FN=0, Flags=.....C
26802	62.330832415	33		33	Action, SN=4, FN=0, Flags=.....C
26814	63.016956884	3c		33	Action, SN=261, FN=0, Flags=.....C
26816	63.016958395	3c		3c	Action, SN=7, FN=0, Flags=.....C
27042	64.038442945	33		3c	Action, SN=262, FN=0, Flags=.....C
27044	64.038446175	3c		33	Action, SN=11, FN=0, Flags=.....C
29508	91.410046599	03		3c	QoS Data, SN=410, FN=0, Flags=p...F.C

Figure 4.4: Reconnection of the LG Video Doorbell.
The marked frames include both the authentication and association frames, as well as the 802.1X authentication frames.

22298	62.637183896	06	26	Deauthentication, SN=2784, FN=0, Flags=...
22299	62.639600938	26	06	Deauthentication, SN=2785, FN=0, Flags=...
27493	156.1431006...	26	Broadcast	Probe Request, SN=0, FN=0, Flags=.....C,
27497	156.1638836...	26	Broadcast	Probe Request, SN=1, FN=0, Flags=.....C,
27503	156.1838941...	26	Broadcast	Probe Request, SN=2, FN=0, Flags=.....C,
27510	156.2016564...	26	Broadcast	Probe Request, SN=3, FN=0, Flags=.....C,
27514	156.2265647...	26	Broadcast	Probe Request, SN=4, FN=0, Flags=.....C,
27515	156.2417562...	26	Broadcast	Probe Request, SN=5, FN=0, Flags=.....C,
27552	156.9095279...	26	33	Authentication, SN=0, FN=0, Flags=.....C
27554	156.9115010...	33	26	Authentication, SN=256, FN=0, Flags=.....C
27556	156.9115037...	26	33	Association Request, SN=1, FN=0, Flags=....
27558	156.9155507...	33	26	Association Response, SN=258, FN=0, Flags=.
27574	156.9663046...	33	26	Action, SN=259, FN=0, Flags=.....C
27786	159.1858764...	26	33	Action, SN=3, FN=0, Flags=.....C
27788	159.1872201...	33	26	Action, SN=260, FN=0, Flags=.....C

Figure 4.5: Reconnection of the Nikkei Video Doorbell.

The marked frames include both the authentication and association frames, as well as some of the 802.1X authentication frames.

The evil twin attack on these video doorbells also gives similar results to the cameras; the doorbells connect to the evil twin during the attack, even after a single deauthentication packet. Again, the doorbells stay disrupted and unable to connect to the real access point until the evil twin goes down.

4.3 Discussion

We can summarise our findings, what attacks the devices are susceptible to, and the behaviour of these devices in the following table:

Device brand name	Deauthentication attack	Evil twin attack	Automatic reconnect	Captured reconnect	Single packet for evil twin attack
Eufy	Yes	Yes	No	No	Yes
Monvelli	Yes	Yes	Yes	No	Yes
LifeGoods	Yes	Yes	Yes	Yes	Yes
Nikkei	Yes	Yes	No	Yes	Yes

Table 4.1: Summary of our results.

We note that only two out of the four devices tried automatically reconnecting to the network. We think this is based on designer preference: whether it is preferable to automatically reconnect or to only try to reconnect sporadically (or not at all). The latter could be implemented by limiting the number of reconnection attempts or having a set time between reconnection attempts. Using a set time between reconnection attempts would be preferred, as this causes minimal overhead. However, the device will still try to reconnect, and this method thus minimises the time we can disrupt a device. Using the former method, although it ensures that the device reconnects immediately, does cause unnecessary overhead.

An important result we can gather is that deauthentication attacks send a constant stream of frames over the network, which allows these attacks to be detected. However, combining it with an evil twin attack means that no more packets are being sent over the network, thus showing no sign of an attack happening. We are therefore unable to capture any traffic coming from the device in this scenario. This allows us to completely block the services that the device provides without any activity on the network that could indicate an attack, especially since sending a single deauthentication packet is enough for this to work. Using a micro-SD card with either of these cameras will nonetheless save the footage, and after reconnecting to the real access point, we are able to view this data on the cloud storage medium.

Another interesting result is that with either video doorbell, while reconnecting, we were able to capture both the authentication & association protocols, as well as the 4-way handshake. As we have discussed in Chapter 3, the 4-way handshake could be used to break the encryption on the data packets over the connection between these devices and the access point.

On the other hand, we found that only using a deauthentication attack on the connection between the router and the devices and not between the repeater and the devices (or vice versa) was not successful in disrupting the devices. Wireshark shows no interesting packets are sent over the LAN in this instance, as we had predicted in our setup. Hence, we have not included this in our results.

Although we were able to connect all of the devices to our evil twin, this was also because our evil twin was in closer proximity to the devices than the actual access point. Using signal boosters would allow us to be further away from the devices, as the devices select the access point based on signal strength. This would mitigate the issues with being physically close to the devices

Future work should try to remove all extenders from the network, or if this is not possible, make sure that devices can only connect to a single access point. Multiple access points cause unnecessary overhead, but the attacks still have the exact same setup; we just had to execute them twice each time.

Moreover, finding an adapter that supports monitor mode and works with Kali Linux is not trivial. Most adapters do not work out of the box and thus require a workaround to be able to be set to monitor mode. We first did this attack with a TL-WN722N version 2, but found that even after performing the necessary fixes, it was still quite inconsistent at launching these attacks.

4.3.1 Ethics

We executed all the attacks we discussed on a private network with explicit consent from the owner of the network. Moreover, we own all the devices attacked and made sure to only attack these devices. In doing so, we were able to execute the attacks without causing any damage and with minimal disruptions. We have not tested these attacks on any other networks or devices.

Both the deauthentication and the evil twin attacks are widely studied, as we will also show in the next chapter. We thus do not make use of any new vulnerabilities. In addition to this, because the vulnerability is caused by a flaw in 802.11 and not by the device, and the flaw is well-known, we have not disclosed our findings to the device vendors.

Chapter 5

Related Work and Countermeasures

5.1 Related Attacks

We now discuss some related attacks. Shrivastava et al. [18] describe an access point service blocking attack that uses a similar type of denial-of-service attack and an evil twin to stop the access point from authenticating with devices. They mention that the detection of such an attack is difficult, especially if one combines it with a deauthentication attack. We have contributed to this by actually showing that this combination is indeed able to completely disrupt individual devices.

Tigner et al. [9] investigate deauthentication attacks on online learning environments like Zoom or Google Meet. They find that launching a deauthentication attack against the host's router successfully removes the host from said environment while the adversary is not present on the network. Although this could be fixed by using a wired connection, they suggest that a similar attack could be used on wireless IoT devices. We are expanding on this by performing this attack and describing the exact behaviour of such devices during an attack.

Deauthentication and evil twin attacks are widely researched areas and the combination of them has been used before to attack specific devices [19] [20]. Moreover, a lot of attacks based on deauthentication have been executed on several individual devices, like attacks on rogue UAVs [21], attacks on Bluetooth [22] and attacks on smart farming infrastructure [23]. Although the underlying attacks are similar, we investigate the behaviour on other devices instead.

Dalal et al. [24] investigate possible attack vectors against WPA3, of which one is also a deauthentication attack. The protected management frames are only introduced during the 802.1X authentication, so any device in state three will still process unauthenticated management frames, allowing us to perform our attack. Moreover, they show some attacks for downgrading WPA3 to WPA2, which highlights the relevance of these deauthentication attacks.

Now that we have identified similar attacks, we must explain countermeasures against them. We can split these countermeasures into three categories. First, we have preventive countermeasures, which try to stop the execution of a deauthentication attack on a network. Then we have the detection countermeasures, which focus on detecting a deauthentication attack and solving the problem accordingly. Finally, we have the response countermeasures, which are solutions for devices undergoing a deauthentication attack. Although there are several countermeasures for detecting evil twin attacks [25] [26], we are only interested in the specific case where a device connects to the evil twin as a result of a deauthentication attack.

5.2 Prevention Countermeasures

Prevention-based countermeasures require changes in the underlying protocol, as these measures must change the way deauthentication packets are handled. We discuss a few of these countermeasures below.

The 802.11w standard [27], which, for example, is included in WPA3, adds authentication to certain management frames, the so-called protected management frames. This prevents an adversary from creating and sending, for example, deauthentication frames, completely voiding both of the attacks we discussed in this thesis. The main issue with this solution is that it requires both a protocol change and firmware upgrades. Devices or access points that do not support this standard would thus be unable to support protected management frames.

Bellardo et al. [28] propose delaying the processing of management frame requests. This is based on the idea that if a device wants to disconnect from an access point, it does not send any more data packets. However, with, for example, a deauthentication attack, the device keeps trying to send data packets, which thus indicates a deauthentication attack is happening. By delaying the processing of these management frames, the deauthentication frames can be ignored if other data frames are received. Once again, this solution requires protocol changes and firmware upgrades.

We can see that these preventive countermeasures are a good solution against both of our attacks; however, these measures can only be implemented in the long run. Moreover, specifically for the first countermeasure, the so-called Dragonblood [29] attacks use an exploit on the dragonfly handshake to downgrade WPA3 to WPA2, which does not require 802.11w to function as opposed to WPA3. This would allow us to once again execute our attacks, and hence we require different countermeasures for short-term prevention.

5.3 Detection Countermeasures

Detection-based countermeasures, in contrast to preventive measures, do not require changes in the protocol, nor do they require firmware upgrades. A detection system can be implemented on any network and only requires the hardware it is implemented on to process it. We discuss a few such countermeasures.

Agarwal et al. [30] propose a machine learning-based detection system for detecting deauthentication attacks. This approach opposes the static threshold-based detection system, where after receiving a certain number of deauthentication frames, the system detects the attack. However, with a static threshold, this value could be misjudged and result in a lot of false positives and negatives. Instead, this detection system is trained using model data gathered from a network, which had a detection rate of 96%.

Aslam et al. [31] propose a method based on sequence numbers. Frames sent after each other should have related sequence numbers. So, if deauthentication frames are sent with the wrong sequence number, we can assume that these frames were created by an adversary. This method is based on the fact that timing the correct sequence as an adversary is quite difficult, especially if the number of frames sent is high.

Although these detection-based countermeasures have a high probability of detecting and accordingly dealing with deauthentication attacks, they are not able to counter the evil twin attack used in this thesis. These countermeasures cannot detect deauthentication attacks instantly, so the device is still temporarily disconnected from the access point. Even with the sequence number-based detection, an adversary could correctly predict the next sequence number, especially if these sequence numbers are based on a sequential system, by monitoring the packets sent over the LAN [31]. For pseudo-random numbers, this guess would depend on the bit length of the sequence number. However, a single successful deauthentication frame still results in the device connecting to the evil twin, meaning that the evil twin would have full control over the frames sent by the device.

Shrivastava et al. [18] propose a detection method for evil twins based on client-side detection. When trying to reconnect to an access point after a deauthentication attack with an evil twin on the network, the device receives two messages with an ANonce and a replay counter, as described in the 802.1X security section. With these messages, the device can determine the presence of an evil twin using their EvilScout system. This solution would allow us to detect an evil twin on the network and thus prevent the attack discussed above.

5.4 Response Countermeasures

As mentioned in Chapter 3, deauthentication packets are notifications, and we are thus unable to directly stop an ongoing deauthentication attack. Instead, we want the device to keep working even without a connection. The only solution for the cameras is saving the footage, which can be done using a micro-SD card, as we have discussed in Section 4.3. This is, however, not possible for most video doorbells, as there is no option for local storage on them.

As we have found in the detection-based countermeasures section, it can be quite difficult to differentiate an evil twin from a real access point. Even if we are able to identify that an evil twin is present on the network, we still need to locate and turn off this evil twin. Bhatia et al. [32] propose a method using four antennas to locate both the access point and the evil twin. Using hyperbolic position bounding, they show that the location of an evil twin can be found with a confidence level of 95% in an area of 1/9th of the size of the original grid.

Chapter 6

Conclusions

In this thesis, we found that instead of using just a deauthentication attack, combining it with an evil twin attack is surprisingly effective for completely disrupting the service on a few IoT devices. In Chapter 4, we have shown that all of the devices we used are susceptible to deauthentication attacks, and some devices even require manual interference to work again after such an attack. If we instead allow the devices to connect to our evil twin, the devices will stay connected to the evil twin as long as it is up, meaning that we would have a complete service disruption on these devices.

In chapter 5, we found that it is even more difficult to stop deauthentication in combination with the evil twin attack. The best solution is to upgrade to the 802.11w standard, which includes protected management frames. This completely mitigates both attacks discussed in this thesis. A problem, however, is that this upgrade is unrealistic in the short term, as it requires all devices and access points to be updated. So, if this upgrade is not possible, using a combination of detection countermeasures for both deauthentication attacks and evil twin attacks can make us aware of their presence, but completely stopping these attacks will cause a lot of issues without the preventive measures.

We have also shown that mounting these attacks can be done by anyone with some network attack experience, so future work on preventing these attacks is crucial to stopping them. Although the 802.11w standard introduces protection against these attacks, future work has to be done on the effectiveness of these countermeasures. Moreover, we have discussed an attack that can downgrade WPA3 to WPA2, which allows us to launch a deauthentication attack similar to the attacks we have discussed. More research has to be put in place to completely prevent these attacks.

Bibliography

- [1] Katia Hetter. Marriott fined \$600,000 by FCC for blocking guests' Wi-Fi, 2004. CNN, <https://edition.cnn.com/2014/10/03/travel/marriott-fcc-wi-fi-fine/index.html>.
- [2] Jose Maria Briones, Mario Alejandro Coronel, and Patricia Chavez-Burbano. Case of study: Identity theft in a university wlan evil twin and cloned authentication web interface. In *2013 World Congress on Computer and Information Technology (WCCIT)*, pages 1–4, 2013.
- [3] S Vinjosh Reddy, K Sai Ramani, K Rijutha, Sk Mohammad Ali, and CH. Pradeep Reddy. Wireless hacking - a WiFi hack by cracking WEP. In *2010 2nd International Conference on Education Technology and Computer*, volume 1, pages V1–189–V1–193, 2010.
- [4] 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 - Redline. *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016) - Redline*, pages 1–7524, 2021.
- [5] Baber Aslam, M Hasan Islam, and Shoab A. Khan. 802.11 disassociation dos attack and its solutions: A survey. In *2006 Proceedings of the First Mobile Computing and Wireless Communication International Conference*, pages 221–226, 2006.
- [6] IEEE Standard for information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements. *IEEE Std 802.11i-2004*, pages 1–190, 2004.
- [7] Chia-Mei Chen and Tien-Ho Chang. The Cryptanalysis of WPA & WPA2 in the Rule-Based Brute Force Attack, an Advanced and Efficient Method. In *2015 10th Asia Joint Conference on Information Security*, pages 37–41, 2015.

- [8] Yimin Song, Chao Yang, and Guofei Gu. Who is peeping at your passwords at starbucks? — to catch an evil twin access point. In *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, pages 323–332, 2010.
- [9] Matthew Tigner and Hayden Wimmer. Disruption and Protection of Online Synchronous Learning Environments via 802.11 Manipulation. In *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pages 1–6, 2021.
- [10] Alfa, 2022. Alfa AWUS036ACH, <https://www.alfa.com.tw/products/awus036ach?variant=36473965871176>.
- [11] ScaPy, 2023. ScaPy, version 2.5.0, <https://scapy.net/>.
- [12] Aircrack-ng, 2022. Aircrack-ng, version 1.7, <https://www.aircrack-ng.org/>.
- [13] Bastian Könings, Florian Schaub, Frank Kargl, and Stefan Dietzel. Channel switch and quiet attack: New dos attacks exploiting the 802.11 standard. In *2009 IEEE 34th Conference on Local Computer Networks*, pages 14–21, 2009.
- [14] Maclookup, 2023. Maclookup, version 16.12.0, <https://maclookup.app/>.
- [15] Md Mainuddin, Zhenhai Duan, and Yingfei Dong. Network traffic characteristics of iot devices in smart homes. In *2021 International Conference on Computer Communications and Networks (ICCCN)*, pages 1–11, 2021.
- [16] Airgeddon, 2022. Airgeddon, version 11.10, <https://github.com/visit0r1sh3r3/airgeddon>.
- [17] WireShark, 2023. WireShark, version 4.0.3, <https://www.wireshark.org/>.
- [18] Pragati Shrivastava, Mohd Saalim Jamal, and Kotaro Kataoka. EvilScout: Detection and Mitigation of Evil Twin Attack in SDN Enabled WiFi. *IEEE Transactions on Network and Service Management*, 17(1):89–102, 2020.
- [19] Jose Armando Pratama, Ahmad Almaarif, and Avon Budiono. Vulnerability analysis of wireless lan networks using issaf wlan security assessment methodology: A case study of restaurant in east jakarta. In *2021 4th International Conference of Computer and Informatics Engineering (IC2IE)*, pages 435–440, 2021.

- [20] Sibi Chakkaravarthy Sethuraman, Vaidehi Vijayakumar, and Steven Walczak. Cyber attacks on healthcare devices using unmanned aerial vehicles. *J. Med. Syst.*, 44(1), dec 2019.
- [21] KN Kadripathi, L Yethinder Ragav, KN Shubha, and P Hareena Chowdary. De-Authentication Attacks on Rogue UAVs. In *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pages 1178–1182, 2020.
- [22] Karim Lounis. Cut it: Deauthentication attack on bluetooth. In *2021 14th International Conference on Security of Information and Networks (SIN)*, volume 1, pages 1–8, 2021.
- [23] Sina Sontowski, Maanak Gupta, Sai Sree Laya Chukkapalli, Mahmoud Abdelsalam, Sudip Mittal, Anupam Joshi, and Ravi Sandhu. Cyber attacks on smart farming infrastructure. In *2020 IEEE 6th International Conference on Collaboration and Internet Computing (CIC)*, pages 135–143, 2020.
- [24] Neil Dalal, Nadeem Akhtar, Anubhav Gupta, Nikhil Karamchandani, Gaurav S. Kasbekar, and Jatin Parekh. A Wireless Intrusion Detection System for 802.11 WPA3 Networks. In *2022 14th International Conference on COMMunication Systems & NETWORKS (COMSNETS)*, pages 384–392, 2022.
- [25] Songrit Kitisriworapan, Aphirak Jansang, and Anan Phonphoem. Evil-Twin Detection on Client-side. In *2019 16th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 697–700, 2019.
- [26] Md. Asaduzzaman, Mohammad Shahjahan Majib, and Md. Mahbubur Rahman. Wi-Fi Frame Classification and Feature Selection Analysis in Detecting Evil Twin Attack. In *2020 IEEE Region 10 Symposium (TENSymp)*, pages 1704–1707, 2020.
- [27] 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. *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)*, pages 1–111, 2009.
- [28] John Bellardo and Stefan Savage. 802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions. In *Proceedings of the 12th*

Conference on USENIX Security Symposium - Volume 12, SSYM'03, page 2, 2003.

- [29] Mathy Vanhoef and Eyal Ronen. Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 517–533, 2020.
- [30] Mayank Agarwal, Santosh Biswas, and Sukumar Nandi. Detection of De-Authentication DoS Attacks in Wi-Fi Networks: A Machine Learning Approach. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 246–251, 2015.
- [31] Baber Aslam, M Hasan Islam, and Shoab A. Khan. Pseudo Randomized Sequence Number Based Solution to 802.11 Disassociation Denial of Service Attack. In *2006 Proceedings of the First Mobile Computing and Wireless Communication International Conference*, pages 215–220, 2006.
- [32] Payal Bhatia, Christine Laurendeau, and Michel Barbeau. Solution to the wireless evil-twin transmitter attack. In *2010 Fifth International Conference on Risks and Security of Internet and Systems (CRiSIS)*, pages 1–7, 2010.

Appendix A

Appendix

A.1 Deauthentication Packet

In Figure A.1, we can see the exact structure of the header of a deauthentication frame. As we can see, all flags and values are exactly the same as in Chapter 2.

```
▼ IEEE 802.11 Deauthentication, Flags: .....
  Type/Subtype: Deauthentication (0x000c)
  ▼ Frame Control Field: 0xc000
    .... ..00 = Version: 0
    .... 00.. = Type: Management frame (0)
    1100 .... = Subtype: 12
  ▼ Flags: 0x00
    .... ..00 = DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0 From DS: 0) (0x0)
    .... .0.. = More Fragments: This is the last fragment
    .... 0... = Retry: Frame is not being retransmitted
    ...0 .... = PWR MGT: STA will stay up
    ..0. .... = More Data: No data buffered
    .0.. .... = Protected flag: Data is not protected
    0... .... = Order flag: Not strictly ordered
    .000 0001 0011 1010 = Duration: 314 microseconds
```

Figure A.1: A deauthentication frame sent over the LAN

A.2 Installing Drivers

We will be using Kali Linux 2022.4 to perform the deauthentication attack. The ALFA adapter does not work out of the box with Kali, so we need to use the following commands in order to use our adapter:

- `sudo apt-get update`
`sudo apt-get upgrade`
`sudo reboot`

These commands are used to update and upgrade Kali to the newest version.

- `sudo apt install realtek-rtl88xxau-dkms`
This command installs the driver onto the adapter; however, these drivers do not work with Kali, so we need some extra steps.
- `sudo apt install dkms`
`git clone https://github.com/aircrack-ng/rtl8812au`
These commands install the updated drivers that are compatible with Kali.
- `cd rtl8812au/`
`sudo make`
`sudo make install`
These final commands install the correct drivers for the adapter.

After unplugging and re-plugging, the adapter should be visible using either the command `lsusb` or `iwconfig`, which in our case is under `wlan0`.

A.3 Connecting Evil Twin to Access Point

These commands can be used to set up bridges between the evil twin and an access point.

- `brctl addbr [Name of fake network]`
This command creates an instance of an Ethernet bridge.
- `brctl addif [Name of fake network] [Network interface]`
`brctl addif [Name of fake network] [Evil Twin interface]`
These commands will add the required bridges from our evil twin interface to the actual network through our evil twin.
- `ifconfig [Evil Twin Interface] 0.0.0.0 up`
`ifconfig [Name of fake network] up`
The final two commands will set up the network, allowing all the traffic through the evil twin access point to go to the original access point.