Cut It: Deauthentication Attacks on Protected Management Frames in WPA2 and WPA3

Karim Lounis[1(✉)], Steven H. H. Ding[2], and Mohammad Zulkernine[1]

[1] Queen's Reliable Software Technology Lab, Queen's University, Kingston, ON, Canada
{karim.lounis,mz}@queensu.ca

[2] L1NNA Research Laboratory, School of Computing, Queen's University, Kingston, ON, Canada
steven.ding@queensu.ca

Abstract. Deauthentication attacks on Wi-Fi protocol (IEEE 802.11) were pointed out in early 2003. In these attacks, an attacker usually impersonates a Wi-Fi access point (a.k.a., authenticator) and sends spoofed deauthentication frames to the connected Wi-Fi supplicants. The connected supplicants receive the frames and process them as if they were sent by the legitimate access point. These frames instruct - connected Wi-Fi supplicants to invalidate their current association and authentication to the access point and get disconnected from the Wi-Fi network. This is possible due to the absence of authentication in management frames (which includes deauthentication frames) in the currently used Wi-Fi security mechanisms (i.e., WPA and WPA2). To thwart these attacks, as well as, many other Denial-of-Service attacks, in 2009, an amendment, standardized IEEE 802.11w, was published as a set of new security mechanisms and procedures to enforce authentication, data freshness, and confidentiality on certain management frames. This amendment uses PMF (Protected Management Frames) to provide authentication of management frames and prevent the occurrence of many management frame spoofing-related attacks, including deauthentication attacks. Although only a few Wi-Fi-certified devices have incorporated IEEE 802.11w as an optional mechanism, the new Wi-Fi security mechanism, WPA3, has made IEEE 802.11w mandatory to provide a better security against those Denial-of-Service attacks. In this paper, we demonstrate through various attack scenarios the feasibility of deauthentication attacks on PMF-enabled WPA2-PSK and WPA3-PSK networks. We provide interpretations to explain the reason behind the feasibility of the attacks and describe possible countermeasures to prevent the attacks.

Keywords: Wi-Fi security · WPA3 security · Wi-Fi attacks · PMF · Wi-Fi Denial-of-Service attacks · Deauthentication attack

1 Introduction

Wi-Fi networks have been susceptible to Denial-of-Service (DoS) attacks at both the physical layer (e.g., jamming attacks) and the MAC-layer (e.g., deauthentication and disassociation attacks [12]). Also, tools to launch these attacks are freely available on the Internet. Technically, there are two main reasons why Wi-Fi networks have been vulnerable to DoS attacks: (1) The wireless medium is not confined by physical boundaries like it is in wired Ethernet networks. Therefore, attacks can be generated from an outside range of an access point (e.g., from a nearby building or from inside a parked vehicle). (2) Wi-Fi management and control frames are neither encrypted nor authenticated as per the IEEE 802.11 specifications. In fact, the original reasoning was that there are management frames that are expected before a Wi-Fi supplicant is associated with an access point, and hence protecting frames with encryption and authentication and sending them to Wi-Fi supplicants that knew nothing about the access point credentials did not sound logical for IEEE 802.11 designers. As a consequence, the lack of authentication in these frames allowed attackers to spoof the frames and generate various types of Wi-Fi Denial-of-Service (DoS) attacks [12].

To mitigate these Denial-of-Service attacks, in particular, deauthentication attacks, in 2009, an amendment, standardized IEEE 802.11w [1], was published to provide a set of new security mechanisms to augment certain management frames with authentication, data freshness, and confidentiality. This amendment uses PMF (Protected Management Frames) to provide authentication of certain management frames, called RMF (Robust Management Frames), and prevent the Denial-of-Service attacks that rely on spoofing management frames. Even though the standard has been around since 2009, it is really unfortunate to find that many Wi-Fi devices in 2021 still not have incorporated the IEEE 802.11w amendment. Only a few number of Wi-Fi certified devices have implemented IEEE 802.11w as an optional mechanism to be used in WPA2 networks. Fortunately, the IEEE 802.11w standard has been made mandatory in WPA3 certified devices to provide a higher security against many Denial-of-Service attacks.

In this paper, we demonstrate different attack scenarios to cause deauthentication of PMF-enforced WPA2 and WPA3 supplicants. We analyze the causes of the attacks and provide possible countermeasures to prevent the attacks.

The remainder of this paper is organized as follows. In Sect. 2, we discuss the related work. In Sect. 3, the IEEE 802.11w amendment is briefly presented. We demonstrate various deauthentication attack scenarios in Sect. 4. We conclude the paper in Sect. 5.

2 Related Work

There has been some research work that demonstrated that IEEE 802.11w was not completely effective. Ahmad et al. [2] demonstrated three Denial-of-Service attacks on IEEE 802.11w, namely, BIP (Broadcast Integrity Protocol) vulnerability, Security Association (SA)-query manipulation, and association starvation. The first one is an insider attack where a malicious supplicant uses the shared broadcast key (which is supposed to be used only by the access point) to generate protected broadcast deauthentication and disassociation frames. The second attack consists of maliciously initiating an SA-query procedure and jamming the legitimate supplicant to prevent it from responding to the SA-query requests causing a deadlock. The third attack consists of preventing a supplicant from associating to an access point by sending a fake association frame with Reason Code 30 and a large association come-back time, e.g., 300 s. Eian et al. [3] outlined the feasibility of an authentication attack, where a spoofed open system authentication request would cause the access point to disassociate the supplicant and drop its received data. This would force the supplicant to re-associate and restart the 4-way-handshake. Nevertheless, most Wi-Fi device manufacturers have fixed this issue. A new authentication request would not change the status of an associated supplicant. Wang et al. [4] briefly discussed some known Denial-of-Service attacks that are still possible on IEEE 802.11w during the 4-way-handshake. For example, by injecting a fake EAPoL message (a.k.a., EAPoL M_1[1]) during the 4-way-handshake and before the supplicant replies to the first legitimate EAPoL message that it receives from the access point, an attacker could force the supplicant to derive the pairwise transient key (PTK) each time it receives a newly forged EAPoL message. Additionally, injecting spoofed deauthentication frames during the 4-way-handshake, e.g., after exchanging the EAPoL M_1, would abort the authentication process. Valli et al. [5] performed a formal security analysis of the IEEE 802.11w during the 4-way-handshake phase using CasperFDR. They pointed out the feasibility of man-in-the-middle attacks to compromise certain security properties of the 4-way-handshake and disclose keys used for group communication and protected broadcast messages. Schepers et al. [6] developed a framework to test and fuzz Wi-Fi devices for vulnerabilities. They used the developed tool to demonstrate that certain 802.11w capable access points are vulnerable to deauthentication by exploiting the vulnerability CVE-2019-16275. This vulnerability makes certain access points reply with a protected broadcast deauthentication frame when a spoofed association request frame is injected with a destination address set to broadcast. They used the tool to detect whether certain devices were vulnerable to KRACKs [14]. Ram et al. [7] discussed through a patent how an attacker can disconnect a PMF-enforced supplicant from an access point by forcing the supplicant to switch the radio channel through a spoofed probe response (with a channel switching element) during the execution of an SA-query procedure. This is to prevent the supplicant from responding to the SA-query requests that are being sent on the original channel. The SA-query procedure would time out, causing the disconnection of the supplicant. Lounis et al. [8–11] demonstrated various Denial-of-Service attacks on WPA2-PSK and WPA3-PSK when PMF is enforced. These attacks target the authentication phase by injecting spoofed authentication messages in a race condition to prevent and deprive the supplicant of successfully getting authenticated and associated with the access point.

[1] There are 4 EAPoL messages that are exchanged between the supplicant and the authenticator during the 4-way-handshake. Based on their order, these messages are often referred to as EAPoL M_1, M_2, M_3, and M_4.

As IEEE 802.11w is a set of MAC-layer procedures, physical-layer threats, e.g., jamming, are not concerned and hence are still feasible. Last but not least, it is important to note that most of the attacks presented in [2–7,9–11] are attacks that need to be launched before or during the execution of the 4-way-handshake, where the session keys are derived at the end. This means that most of them would not work if the supplicant is already associated with an access point and is exchanging encrypted data. In this paper, the attacks that we present target PMF-protected supplicants that are associated and are exchanging encrypted data to cause their disconnection.

3 IEEE 802.11w Amendment

Before IEEE 802.11w[2] (a.k.a., Protected Management Frames, or PMF[3]), only data frames could be protected in Wi-Fi. Management and control frames were used without any protection. The IEEE 802.11w amendment came to provide certain protection to some specific management frames, known as Robust Management Frames (RMF). These frames include, deauthenticaiton frames, disassociation frames, and certain action frames, e.g., QoS action frames and Block ACK frames. Also, the mechanism provides protection, through Security Association teardown protection (a.k.a., Security Association Query Procedure, cf., next subsection), to association and authentication frames exchanged during an existing connection to prevent disconnection of connected Wi-Fi supplicants.

The IEEE 802.11w provides data integrity and freshness for broadcast and multicast robust management frames through the use of the Broadcast Integrity Protocol (BIP). This protocol uses the Message Integrity Code (MIC) to protect the integrity of the frames and provide freshness to prevent the replay of old frames. Tampered or replayed frames are passively discarded when they are detected. This for example mitigates broadcast deauthentication attack, where all connected supplicants get instantly disconnected after processing (without any verification) a spoofed deauthentication frame. On the other hand, unicast robust management frames benefit from data confidentiality in addition to data integrity and data freshness protection.

Because IEEE 802.11w provides protection to only some management frames, DoS attacks based on other management frames (i.e., Class 1 frames) are unfortunately still possible (e.g., race condition-based attacks [8–11]). Additionally, attacks based on control frames (e.g., RTS/CTS[4]-based attacks [12]) are still

[2] IEEE 802.11w only applies to Wi-Fi networks running Robust Security Networks (RSN), i.e., using WPA-TKIP or WPA-CCMP (WPA2 and WPA3).

[3] Note that PMF should not be confused with Cisco MFP (Management Frame Protection), which was developed in 2005. In MFP, there are two modes: (1) Infrastructure mode, where the access point sings beacon frames and other broadcast management frames (to detect Rogues). (2) Client mode, where the AP signs management frames that are sent to the client in addition to beacon and broadcast management frames.

[4] The request to send (RTS) and clear to send (CTS) is a mechanism used to reserve the radio channel to send time-sensitive packets and prevent collisions.

possible since IEEE 802.11w deals only with management frames. Furthermore, if an attacker manages to crack the network password (and hence the keys), it will be able to forge authenticated management frames and may succeed in generating DoS attacks. This also means that an insider malicious supplicant may abuse the mechanism and run successful DoS attacks since it knows the network password (although may need to capture some 4-way-handshakes).

3.1 Security Association Query Procedure

IEEE 802.11w amendment introduced an association spoofing protection mechanism to prevent replay attacks from tearing down an existing Wi-Fi supplicant's association to an access point. It consists of two mechanisms: (1) Association come-back time, and (2) SA-query procedure.

When an authenticator (i.e., access point) receives an association request from a supplicant that is already associated with the authenticator (i.e., in IEEE 802.11 State 3[5]), the latter responds with a rejective association response stating the reason "Association rejected temporarily; try again later (Code 30)". This association response incorporates an association come-back time, a.k.a., timeout interval value (TIV), that informs the supplicant to comeback and re-associate after the expiration of that association come-back time and in the case where the SA-query procedure is unsuccessful. In fact, just after the rejection, the authenticator initiates the SA-query procedure by sending SA-query requests (which are 12-byte protected action frames) until it receives a valid SA-query response (also a 12-byte protected action frame) from the supplicant or the association come-back time expires. If no valid SA-query response is received and the association come-back time expires, the access points consider that the supplicant is no longer associated and requires a re-association. Otherwise, if a valid SA-query response was received, the authenticator drops the received association request and considers it as a spoofed request that was generated by an attacker. This maintains the association of the supplicant. This SA-query procedure against a spoofed association request is illustrated by the MSC[6] of Fig. 1.

The SA-query procedure is basically used for the following: (i) Prevent an attacker from tearing down an existing supplicant's association using spoofed association frames. (ii) Allow a previously associated supplicant to securely re-associate to an authenticator after loosing the keys or encountering a local failure. (iii) Prevent an attacker from disassociating/deauthenticating associated supplicants from an access point using disassociation/deauthentication frames.

[5] There are three IEEE 802.11 states in which a supplicant can be: (1) State 1, where the supplicant is not authenticated and not associated with any access point. (2) State 2, where the supplicant is authenticated but not associated. (3) State 3, where the supplicant is both authenticated and associated.

[6] MSC (Message Sequence Chart) is a graphical language for the description of the interaction between different components of a system. This language is standardized by the ITU (International Telecommunication Union).

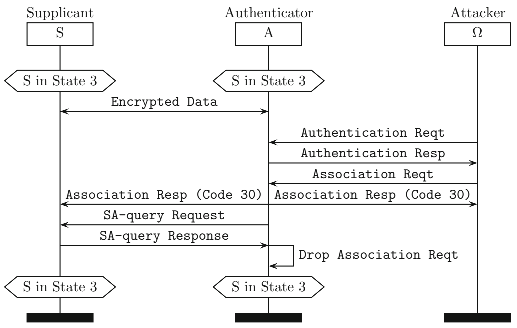

Fig. 1. Security association-query procedure initiated against a spoofed association request sent by an attacker Ω on an associated supplicant S, where State 3 indicates the IEEE 802.11 state of "Authenticated & Associated". State 1 and State 2 indicate the state of "Unauthenticated & Unassociated" and "Authenticated & Unassociated".

Overall, if an unprotected frame is received, the SA-query procedure is used to authenticate the communicating parties and take the correct decision. Receiving unprotected frames could happen due to the presence of an attacker spoofing management frames, or legitimate supplicants having lost their keys for some reasons.

4 Deauthentication Attacks on IEEE 802.11w

In this section, we present different deauthentication attacks on IEEE 802.11w, in general, and Protected Management Frames (PMF), in particular. These attacks are of type Denial-of-Service (DoS) as they all aim to disconnect a Wi-Fi supplicant from an access point when PMF is used in WPA2-PSK and WPA3-PSK. We first present the experimental environment that we have used to generate the attacks, and then individually present each attack, how it was generated (for reproducibility), and provide our interpretations w.r.t. the feasibility of the attack. We also discuss how each attack can be mitigated. In Subsect. 4.1, we present deauthentication attack scenarios that are based on the use of unicast deauthentication frames. In Subsect. 4.2, we present deauthentication attack scenarios that are based on fake authentication sessions and association frames.

Table 1 illustrates the Wi-Fi devices (with their characteristics) that we have used during the experiments. Additionally, Table 2 shows the estimated time in seconds to succeed in different deauthentication attack scenarios. The average time (avg) is computed over 20 consecutive and independent attack attempts for each attack scenario. In the next paragraph, we present the experimental environment that we have used to generate the attacks and analyze the causes of their feasibility.

Experimental Environment. To realize our attacks, we have used two types of Wi-Fi networks, one operating WPA2-PSK with PMF enabled on a Cisco

Table 1. Wi-Fi devices (with their characteristics) used during the experiments.

Wi-Fi device	Operating system or Firmware version	Device type	Wi-Fi security	PFM capable
Apple MacBook Pro M1	Apple macOS Big Sur (versions 11.4, 11.5.1, & 11.5.2)	Laptop (Supplicant)	WPA2-PSK & WPA3-PSK	Yes
Apple MacBook Pro i5	Apple macOS Big Sur (version 11.5.2)	Laptop (Supplicant)	WPA2-PSK & WPA3-PSK	Yes
Apple iPhone 11 Pro Max	Apple iOS (version 14.7.1-18G82)	Smartphone (Supplicant)	WPA2-PSK & WPA3-PSK	Yes
Huawei Nova 5T	Google Android (version 10.0)	Smartphone (Supplicant)	WPA2-PSK	Yes
Cisco WAP150	WAP150-A-K9-NA V02 (version 1.1.3.2)	Access point (Authenticator)	WPA2-PSK	Yes
TP-Link AX6000	TP-Link 1.2.3 Build 20210511 rel.76452(5553)	Access point (Authenticator)	WPA2-PSK & WPA3-PSK	Yes
HP ProBook 6560b	Linux Ubuntu (version 20.04 LTS)	Laptop (Attacker)	WPA2-PSK	No

WAP150 access point[7], and the second network running WPA3-PSK (PMF enforced by default) on a TP-Link AX6000 wireless router. Moreover, we have used different types of supplicants as illustrated in the first group of rows of Table 1. These supplicants are PMF-capable. Further, for the attacker, we have used an HP ProBook 6560b laptop that runs Linux Ubuntu 20.04LTS. We have used *airodump-ng*, *aireplay-ng*, *macchanger*, and some *Scapy-based python scripts* to launch the attacks and capture the wireless traffic. We have analyzed the traffic using the *Wireshark* packet analyzer.

4.1 Deauthentication Using Unicast Deauthentication Frames

Observation. In IEEE 802.11w, when an associated supplicant/access point receives an unprotected deauthentication frame, it starts the SA-query procedure to check whether the access point/supplicant has truly sent that deauthentication frame (e.g., in the case where the access point/supplicant has lost the session keys) or the frame was sent by an unauthorized party that is impersonating the access point or supplicant. If the access point or supplicant responds correctly to the SA-query request, the supplicant/access point concludes that the received frame was a spoofed one and discards it. Otherwise, if no response was received within an SA-timeout, the supplicant/access point resend the SA-query request again. If no response is received for a second time, the supplicant/access point assumes that the access point/supplicant has lost the session keys (for some reason) and considers the unprotected deauthentication frame as a legitimate frame. The supplicant/access point usually sends a protected disassociation frame to conclude the session. The number of SA-query requests that are sent during the

[7] The Cisco WAP150 is a Wi-Fi access point that uses MFP (Management Frame Protection), which is the Cisco implementation of PMF.

Table 2. Estimated time in seconds to succeed in different deauthentication attack scenarios. The average time (avg) is computed over 20 consecutive and independent attack attempts for each attack scenario.

Deauthentication attack scenario on IEEE 802.11w Supplicants		WPA2-PSK (Cisco WAP150)	WPA3-PSK (TP Link AX6000)
Send bidirectional spoofed unicast and unprotected deauthentication frames (Command 1 in Table 3)	Apple MacBook Pro M1	[03–32] (avg: 15.70)	[05–28] (avg: 14.45)
	Huawei Nova 5T	[03–27] (avg: 11.70)	Unsupported
	Apple iPhone 11 Pro Max	[03–56] (avg: 14.40)	[03–44] (avg: 24.15)
	Apple MacBook Pro i5	[05–41] (avg: 17.80)	[06–50] (avg: 23.10)
Send bidirectional spoofed unicast and unprotected disassociation frames (Code 1 in Table 3)	Apple MacBook Pro M1	[03–29] (avg: 08.30)	[03–60] (avg: 26.10)
	Huawei Nova 5T	[03–18] (avg: 08.85)	Unsupported
	Apple iPhone 11 Pro Max	[04–33] (avg: 14.50)	[04–60] (avg: 25.80)
	Apple MacBook Pro i5	[04–27] (avg: 10.80)	[06–46] (avg: 25.50)
Send spoofed unicast deauthentication/disassociation frames to the access point (Code 2 & 3 in Table 3)	Apple MacBook Pro M1	[10– 30] (avg: 16.05)	[05–9] (avg: 11.25)
	Huawei Nova 5T	[03–56] (avg: 14.70)	Unsupported
	Apple iPhone 11 Pro Max	[08–53] (avg: 22.70)	[03–58] (avg: 21.20)
	Apple MacBook Pro i5	[05–25] (avg: 14.80)	[08–43] (avg: 20.50)
Send spoofed unicast deauthentication/disassociation frames to the supplicant (Code 2 & 3 in Table 3)	Apple MacBook Pro M1	No disconnection	No disconnection
	Huawei Nova 5T	No disconnection	Unsupported
	Apple iPhone 11 Pro Max	No disconnection	No disconnection
	Apple MacBook Pro i5	No disconnection	No disconnection
Use complete fake open system authentication and association (Command 3 in Table 3)	Apple MacBook Pro M1	[04–10] (avg: 06.50)	[03–20] (avg: 10.70)
	Huawei Nova 5T	[07–123] (avg: 47.35)	Unsupported
	Apple iPhone 11 Pro Max	[04–10] (avg: 07.35)	[05–60] (avg: 19.10)
	Apple MacBook Pro i5	[04–18] (avg: 06.40)	[07–32] (avg: 16.70)
Use injected association request frames with capability 0×0431 (Code 2 in Table 3)	Apple MacBook Pro M1	[02–09] (avg: 03.90)	[03–04] (avg: 03.40)
	Huawei Nova 5T	[03–24] (avg: 14.50)	Unsupported
	Apple iPhone 11 Pro Max	[03–10] (avg: 05.20)	[03–06] (avg: 04.40)
	Apple MacBook Pro i5	[02–08] (avg: 03.60)	[03–05] (avg: 03.95)
Use injected association response frames with reason code $0 \times 001e$ (Code 3 in Table 3)	Apple MacBook Pro M1	[02–06] (avg: 03.60)	[02–07] (avg: 04.05)
	Huawei Nova 5T	No disconnection	Unsupported
	Apple iPhone 11 Pro Max	[03–13] (avg: 05.02)	[03–06] (avg: 04.60)
	Apple MacBook Pro i5	[02–08] (avg: 04.15)	[04–10] (avg: 05.75)
Use injected association response frames with reason code $0 \times 001f$ (Code 6 in Table 3)	Apple MacBook Pro M1	[02–09] (avg: 03.50)	[02–05] (avg: 04.05)
	Huawei Nova 5T	No disconnection	Unsupported
	Apple iPhone 11 Pro Max	[03–08] (avg: 04.95)	[04–08] (avg: 04.75)
	Apple MacBook Pro i5	[02–08] (avg: 03.75)	[04–08] (avg: 04.65)

SA-query procedure may depend on the implementation of IEEE 802.11w on Wi-Fi certified devices by different manufacturers.

As part of our experiments, we have discovered that it was possible to cause a deauthentication and force the PMF-enforced supplicants to get disconnected using spoofed deauthentication and disassociation frames. We have observed that by generating a large number of spoofed unprotected unicast deauthentication frames or disassociation frames, sent to both the access point and the supplicant (i.e., bidirectional injection), the access point usually ends up sending a protected disassociation frame to the supplicants. It then ignores the supplicant's protected action frames (which are encrypted SA-query requests/responses). The supplicants continue sending their frames (encrypted SA-query requests/responses) to the access point, and since the latter is not responsive, the SA-procedure times out and the supplicants disconnect from the access point by sending a protected disassociation frame. It is important to note that sending spoofed frames to both the access point and the supplicants would initiate the SA-query procedure on both sides. Interestingly, we have also discovered that using spoofed unicast and unprotected deauthentication/disassociation frames, sent only to the access

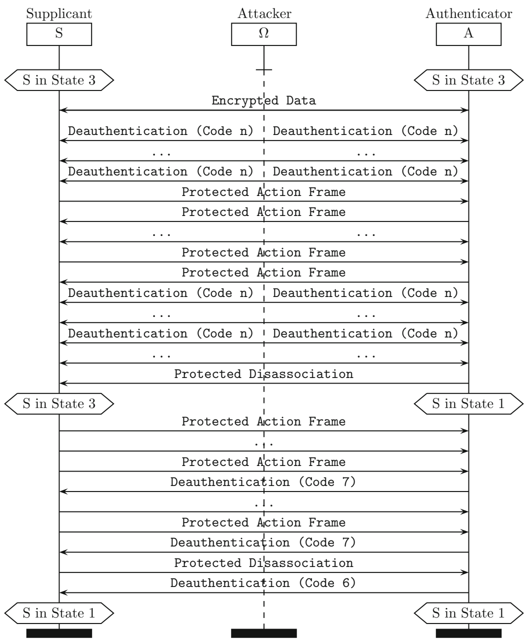

Fig. 2. Deauthentication attack using unprotected unicast deauthentication frames on PMF (WPA2 and WPA3), where State 1 and State 3 indicate the IEEE 802.11 state of "Unauthenticated & Unassociated" and "Authenticated & Associated", respectively. State 2 is "Authenticated & Unassociated".

point on behalf of the supplicants, would cause a disconnection of the supplicants. However, when we have sent these frames to the supplicants on behalf of the access point, the attack did not succeed and no disconnection was observed.

Attack Generation. To generate the attack, we have used the attacker laptop (HP ProBook 6560b) and configured it to impersonate both the access point and the supplicants by setting its MAC address to the ones of the spoofed parties. Then, by connecting the supplicants to the access points, we have generated a large number of spoofed unprotected unicast deauthentication frames[8] (using Command 1 in Table 3) and captured the subsequent wireless traffic (using Command 2 in Table 3). After few seconds, we have managed to disconnect the supplicants from the access points. Next, with the help of *Wireshark*, we have analyzed the exchanged wireless packets and tried to understand the reason that caused the disconnection. Additionally, we have used different *scapy scripts* (viz., Code 1, 2, and 3 in Table 3) to achieve the same goal of disconnecting the supplicants. For example, we have used Code 1 to send a flood of unicast and bidirectional disassociation frames. We have found that after

[8] We have used different Reason Codes [0–254] and the impact was the same. For the experiments of Table 2, we have used Reason Code 10.

Table 3. Commands and codes used for generating deauthentication attacks. We have made the complete codes (Code 1–6) publicly available over a GitHub repository [13]. Commands 1, 2, and 3, are part of the *aircrack-ng* Linux toolset.

Command & Code	**Command/Code Syntax**	**Command/Code Semantics**
Command 1	*aireplay-ng −0 5 −a* mac_{ap} *−c* mac_{sp} *−−deauth-rc n wlan0*	In this command, the option *−0 5* indicates deauthentication to be run 5 times, −a the access point MAC address, −c the supplicant's MAC address, *−−deauth-re n* the reason code (e.g., *n*=7 is "Class 3 frame received from non-associated STA"), and *wlan0* the Wi-Fi interface
Command 2	*airodump-ng −c 6 −−bssid* mac_{ap} *−w ./file.pcap wlan0*	The option *−c 6* indicates the radio channel 6 to listen on (i.e., the one used by the access point), *−−bssid* the access point MAC address, −w the pcap file location where to store the captured wireless traffic, and *wlan0* the Wi-Fi interface
Command 3	*aireplay-ng −1 5 −a* mac_{ap} *wlan0*	In this command, the option *−1 5* indicates the generation of fake authentications using the IEEE open system (i.e., no security) each 5 s (reassociation), *−a* the access point MAC address, and *wlan0* the Wi-Fi interface
Code 1	*dot11=Dot11x(type=0, subtype=10, addr1=bssid, addr2=supp, addr3=bssid)* *dot11=Dot11y(type=0, subtype=10, addr1=supp, addr2=bssid, addr3=bssid)* *framex=RadioTap()/dot11x/Dot11Disas()* *framey=RadioTap()/dot11y/Dot11Disas()* *sendp(framex, iface=wlan0, count=500, inter=0.1)* *sendp(framey, iface=wlan0, count=500, inter=0.1)*	This scapy-based python code snippet creates and sends 500 disassociation frames on both directions, i.e., to the access point (on behalf of the supplicant) and to the supplicant (on behalf of the access point). This code has the same impact and consequences as those of Command 1
Code 2	*dot11=Dot11(type=0, subtype=12, addr1=bssid, addr2=supp, addr3=bssid)* *frame=RadioTap()/dot11/Dot11Deauth(reason=254)* *sendp(frame, iface=wlan0, count=500, inter=0.1)*	This scapy-based python code snippet creates and sends 500 deauthentication frames with reason code 254 (unknown) to the access point on behalf of the supplicant. To send the frame to the supplicant on behalf of the access point, we switch the values of *addr1* and *addr2*
Code 3	*dot11=Dot11(type=0, subtype=10, addr1=bssid, addr2=supp, addr3=bssid)* *framex=RadioTap()/dot11/Dot11Disas()* *sendp(frame, iface=wlan0, count=500, inter=0.1)*	This scapy-based python code snippet creates and sends 500 spoofed unicast disassociation frames to the supplicant on behalf of the access point. To send the frame to the supplicant on behalf of the access point, we switch the values of *addr1* and *addr2*
Code 4	*dot11=Dot11(type=0, subtype=0, addr1=bssid, addr2=supp, addr3=bssid)* *frame=RadioTap()/dot11/Dot11AssoReq(cap=0x0431, listen_interval=0x000a)/Dot11Elt(ID=0, info="SSID")* *sendp(frame, iface=wlan0, count=500, inter=0.1)*	This scapy-based python code snippet creates and sends 500 association request frames to the access point on behalf of a PMF-capable supplicant. This code proved to be more efficient than Command 3 in succeeding the attack within a shorter time. For example, when launched against the Apple MacBook Pro M1, it took between 3 and 4 s to disconnect it (3.40 s on average)
Code 5	*dot11=Dot11(type=0, subtype=1, addr1=supp, addr2=bssid, addr3=bssid)* *frame=RadioTap()/dot11/Dot11AssoResp(cap=0x0431, status=0x001e)/Dot11Elt(ID=0, info="SSID")* *sendp(frame, iface=wlan0, count=500, inter=0.1)*	This scapy-based python code snippet creates and sends 500 association response frames with Reason Code 30 to the supplicant on behalf of the access point. This code proved to affect more the re-association of the supplicant once disconnected. This is probably due to the association come-back time
Code 6	*dot11=Dot11(type=0, subtype=3, addr1=supp, addr2=bssid, addr3=bssid)* *frame=RadioTap()/dot11/Dot11AssoResp(cap=0x0431, status=0x001f)/Dot11Elt(ID=0, info="SSID")* *sendp(frame, iface=wlan0, count=500, inter=0.1)*	This scapy-based python code snippet creates and sends 500 association response frames with Reason Code 31 to the supplicant on behalf of the access point. This code had the same impact as Code 5

sending a certain amount of spoofed frames (around 130 frames), the supplicants got disconnected. Furthermore, using Code 2 and 3, we were able to cause the disconnection by sending unidirectional spoofed deauthentication/disassociation frames. The attack flow using bidirectional deauthentication frames is illustrated in the MSC of Fig. 2 (where $\mathtt{n} \in \{0, \ldots, 254\}$ is arbitrary chosen reason code).

Attack Interpretation. When analyzing the wireless traffic that we have captured using *Wireshark*, we have observed that there was a large number of protected action frames (SA-query requests and responses) exchanged between the access points and the supplicants during the injection of the spoofed deauthentication/disassociation frames. Most of the time, the supplicants were not responding to any of the requests. After few seconds, the access points sent a protected disassociation frame to the supplicants, which set the supplicants' status at the access points' association table to "non-associated and unauthenticated" (i.e., IEEE 802.11 State 1). Right after that, the supplicants started sending protected action frames (SA-query requests/responses) which got rejected by the access points using deauthentication frames with a reason "Received Class 3 frame from non-associated STA (Code 7)" (since action frames are Class 3 frames). After multiple rejections, the supplicants concluded with a protected disassociation frame (to disassociate themselves from the access point) since no response was received and the SA-query procedure timed out. The access points replied with a deauthentication frame with a reason "Received Class 2 frame from unauthenticated STA (Code 6)".

As per the IEEE 802.11w, a party that is involved in an SA-query procedure would send a protected disassociation frame if the latter does not receive any response to its SA-query requests and the SA-query procedure timeout elapses. In our experiments, we believe that the access points (and sometimes the supplicants) concluded the session due to the fact of not receiving SA-query responses to their SA-query requests. There are many hypotheses as to why this has occurred:

1. One party could not respond to new SA-query requests as long as their locally generated SA-query requests have not yet been sent or responded to. In fact, the specification that not explicitly dictate what a party that has initiated the SA-query procedure does if it receives an SA-query request from the other party. Nevertheless, the success of the attack using unidirectional deauthentication/disassociation frames (i.e., using Code 2 and 3), makes this hypotheses weaker since the SA-query procedure is only initiated on one side.
2. One party is not able to access the channel on time and to send their SA-query responses due to the flood of spoofed frames generated by the attacker. This would make the SA-query timeout expire and cause disassociation.
3. Some of the SA-query responses or requests got into a collision with the attacker's frames making the party that is expecting SA-query responses believe that the requested party cannot respond to their SA-query requests, which would timeout the SA-query procedure and cause the disassociation.
4. It is possible that the implementation of IEEE 802.11w on certain devices (including access points) is not robust enough to perfectly handle a large number of SA-query requests and responses mixed along with the spoofed management frames, which would cause the disassociation.
5. If the quality of the radio signal is weak (e.g., due to long distance or noise), it is possible that many of the SA-query requests and responses get lost and do not reach their destination. This would lead to the expiry of the SA-query procedure timeout and cause a disassociation.

Table 2 (Row 1, 2, 3, and 4), shows the time it took to successfully accomplish deauthentication attack using different attack patterns. For example, in Row 1 (bidirectional unicast spoofed and unprotected deauthentication frames) and in the case of the Apple MacBook Pro M1, it took between 3 and 32 s for the attack to succeed on the Cisco WAP150 running WPA2-PSK with PMF enabled. In 20 successful attempts, the average time was 15.70 s. It took between 5 and 28 s for the same attack to succeed on the TP-Link AX6000 running WPA3-PSK. Where in the case of the Huawei Nova 5T, it took between 3 and 26 s for the attack to succeed (12.35 s on average) on the Cisco WAP150. Since the Huawei Nova 5T does not support WPA3-PSK, the attack on this particular supplicant could not be evaluated on the TP-Link AX6000 access point.

Furthermore, as per the impact of the attacks, we note that after a successful deauthentication, we have observed that the supplicants had serious difficulties to rejoin the network again when the attacks continue. In fact, each time the supplicants try to re-authenticate and re-associate to the access points, the supplicants as well as the access points, get distracted by the flood of deauthentication frames and fail to accomplish the authentication and remain disconnected.

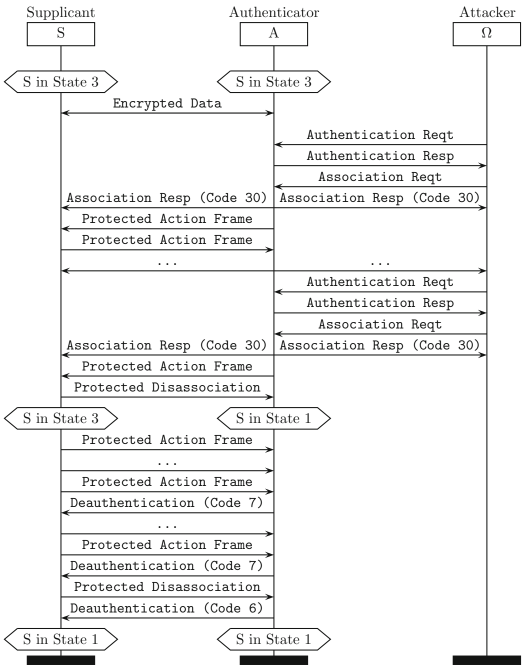

Fig. 3. Deauthentication attack using fake open system authentication on PMF (WPA2 and WPA3), where State 1 and State 3 indicate the 802.11 state of "Unauthenticated & Unassociated" and "Authenticated & Associated", respectively. State 2 (not used here) is "Authenticated & Unassociated".

4.2 Deauthentication Using Fake Authentication Session

Observation. In IEEE 802.11w, when an access point receives an unprotected associated frame from a supplicant that is already associated with it, it starts the SA-query procedure with an association come-back time to check whether the supplicant has truly sent that association frame (in the case where the supplicant has lost the session keys) or the frame was sent by an unauthorized party that is impersonating the supplicant. If the supplicant responds correctly to access point's SA-query requests before the association come-back time is up, the access point concludes that the received frame was a spoofed one and discards it. Otherwise, if no response was received for any of the generated SA-query requests and before the association come-back time runs out, the access point assumes that the supplicant has lost the session keys and allows a re-association from the supplicant after the association come-back time.

During our experiments, in particular, when we have used the Apple devices as supplicants (viz., Row 1, 2, and 3 in Table 1), we have discovered that it was possible to quickly disconnect the supplicants from the access points due to what it seems to be an implementation flaw. By initiating a fake authentication session using the IEEE 802.11 open system mode, the access points rejected the association with a reason "Association Request Rejected Temporarily; Try Again Later (Code 30)". This has made the access points and the supplicant exchange protected action frames (encrypted SA-query requests/response) to check the legitimacy of the new association, which totally conforms to the standard (i.e., SA-query procedure of 802.11w). However, we have noticed that the supplicants (specifically, Apple devices) did not always react to the rejected association frame as per the standard (i.e., wait for an SA-query request and respond to it), but rather sent a protected disassociated frame to the access points and then started sending their protected action frames (possibly, SA-query responses) as if they remained associated. The access points processed the disassociation frame and changed the status of the supplicants in the access points' association table to Unauthenticated (State 1). For each action frame sent by the supplicants after their disassociation, the access point replied by sending unprotected deauthentication frames with a reason "Received Class 3 frame from non-associated STA (Code 7)". Subsequently, the supplicants continued sending protected action frames (which indicates that they are still considering themselves associated with the access point, i.e., in State 3) and then concluded with another protected disassociation frame (to disassociate themselves from the access points). The access points replied with a deauthentication frame with a reason "Received Class 2 frame from unauthenticated STA (Code 6)".

With respect to the Huawei Nova 5T, the attack was successful although it took longer for the disconnection to take place compared to the case of Apple devices. Nevertheless, the disconnection in this case occurred in the same way as it had occurred in the deauthentication attacks presented in the previous section. Due to the absence of SA-query responses, a timeout occured, causing the disconnection of the supplicant.

Attack Generation. By connecting the supplicants to the access points, we have started generating fake open system authentication sessions with the access point (using Command 3 in Table 3) and captured the subsequent wireless traffic (using Command 2 in Table 3). After few seconds, we have managed to disconnect the supplicants from the access points. Furthermore, to optimize the attack, we have used some *scapy-based scripts* (viz., Code 4, 5, and 6 in Table 3). Code 4 performs what Command 3 does but without going through the authentication phase. It only sends a spoofed association request to the access points on behalf of the supplicants to receive a legitimate association response from the access point. Code 5 and 6 save the transmission of 3 frames by only sending the association response frame with Reason Code 30 (i.e., "Association request rejected temporarily; try again later") or 31 (i.e., "Robust management frame policy violation"), respectively. Nevertheless, these two last attack patterns (i.e., Code 5 and 6) succeeded only on Apple devices and did not cause a disconnection on the Huawei Nova 5T. The attack flow using complete fake authentication and association on Apple devices is illustrated in the MSC of Fig. 3. We provide our interpretations in the next paragraph.

Attack Interpretation. After analyzing the wireless traffic that we have captured, we have observed that all Apple devices supplicant do not always react, as per the standard, to a an association response with Reason Code 30 sent by the access point. Indeed, we have discovered that after a couple of fake authentications and associations (sometimes at the first attempt), these devices sent a protected disassociation frame to the access point after receiving a protected action frame (an encrypted SA-query request), changing their status in the access point's association table. This has made all future supplicants' frames (mostly Class 3 frames, e.g., action frames) being ignored by the access point, which made the supplicants disconnect after several attempts. This seems to be an implementation flaw as it is completely incorrect to send a protected disassociation frame (declaring disassociation) and then start replying to SA-query requests. This does not conform to the standard and it is making the attack accomplishment quicker compared to other devices from a different vendor. Furthermore, we believe that this incorrect behavior of sending a protected disassociation frame is related to the SA-query procedure implementation. In fact, we managed to reproduce the same behavior by just injecting spoofed association responses with status code 30 and 31. These two reason codes are only used in IEEE 802.11w (viz., Code 5 and 6 in Table 3). We have reached out to Apple Product Security and they asked us to run the attacks on their latest macOS version (macOS Monterey v12.0 Beta) [15]. We have tried the attacks on this latest version after we installed it on MacBook Pro i5. The incorrect behavior of sending a protected disassociation frame and remaining associated seemed to be fixed in this new version of macOS. These attacks did not succeed. Notwithstanding, deauthentication attacks that we have discussed in the previous section were still successful as they engender a different behavior.

Table 2 (Row 5, 6, 7, and 8), shows the time it took to successfully accomplish deauthentication attacks using fake open system authentication and some of its variant and optimized attack patterns. For example, in Row 5 (use complete fake open system authentication and association) and in the case of the Apple MacBook Pro M1, it took between 4 and 10 s for the attack to succeed on the Cisco WAP150 running WPA2-PSK with PMF enabled. In 20 successful attempts, the average time was 6.50 s. Additionally, it took between 3 and 20 s for the same attack to succeed on the TP-Link AX6000 running WPA3-PSK. The average time was even lower, between 3 to 4 s, when we have used Code 4, 5, and 6. This codes are optimized versions of the attack pattern of Command 3. The execution time of Command 3 on Huawei Nova 5T was longer. It took between 7 and 123 s (47.35 s on average) to disconnect the supplicant from the Cisco WAP150. A much better average execution time of 14.50 s was obtained using Code 4 for this supplicant. Code 5 and 6 did not cause any disconnection.

4.3 Further Result Analysis

Based on the obtained experimental results, we do not deny that IEEE 802.11w is indeed a security amendment for the IEEE 802.11i standard to prevent many Denial-of-Service attacks, including, deauthentication attacks, to be successfully executed within one second. However, we do claim that the current implementations of IEEE 802.11w do not stand against certain attack patterns. We have demonstrated how it was possible to disconnect associated supplicants within one minute using a flood of spoofed unprotected management frames.

With respect to the vulnerability that we have discovered on certain Apple devices, the vulnerability seemed to be fixed in the upcoming version of Apple operating systems, such as, macOS Monterey v12.0 Beta and iOS 15 Beta. This has been confirmed by Apple Product Security department [15]. Thus, until these Beta versions become available to the public as an update, it is still possible to cause deauthentication of certain PMF-enforced Apple devices within 3 to 4 s. As a countermeasure to this vulnerability, we strongly urge Apple device users to update their systems as soon as the update becomes available to be immune from these attacks.

As per the attacks that rely on creating a flood of deauthentication or disassociation frames to cause the disconnection, we have placed five hypotheses in Sect. 4.1 (although we have weakened Hypothesis 1) as for why the disconnection had occurred. Since in most cases, if not all, the SA-query procedure is aborted by the access point by sending a protected disassociation frame (possibly after the SA-query procedure timeout expires), we thought that the issue may reside on the access point. Thus, it may be a good idea to use the Apple MacBook Pro i5 that runs the Beta version of macOS (which is claimed to be secure), as an access point (i.e., Wi-Fi hotspot) and try the attacks. To that end, we have used the Apple MacBook Pro M1 as a supplicant and run the first three attacks of Table 2. The results were as follows:

- When bidirectional spoofed deauthentication frames were used (Command 1), there were 20 disconnections out of 21. It took between 5 and 141 s to cause the disconnection. The average time was around 46.90 s, which is considerably

longer than the case of the TP-Link AX6000 access point for the same supplicant.
- When bidirectional spoofed disassociation frames were used (Code 1), all attack attempts caused a disconnection. It took between 5 and 129 s to cause the disconnection. The average time was around 51.60 s, which is also longer than the case where the TP-Link AX6000 access point was used.
- When unidirectional spoofed deauthentication and disassociation frames were used (Code 2 & 3), we have found that 40% of the attack attempts did not cause a disconnection. Based on the cases where a disconnection occurred, the average time was around 103.80 s, which is much longer than 11.25 s that we have obtained on the TP-Link AX6000 access point.

Although all 5 hypotheses presented in Sect. 4.1 are logical, these latter results propel us to claim that Hypothesis 4 is more likely to be true. As Apple's latest version of macOS (Monterey v12 Beta) proved to have a more robust resilience against these attacks when used as a Wi-Fi hotspot, it is clear that the implementation of the SA-query procedure by different vendors has indeed an impact on hardening or easing the feasibility of those attacks. Furthermore, the fact that the access point generally disassociates the supplicants after the expiry of the SA-query timeout, implies that it is not receiving the expected SA-query responses on time. This could indicate a lack of robustness by the supplicants' current implementation of IEEE 802.11w in handling a large number of SA-query requests that are interfered with other frames.

Therefore, we recommend to device manufacturers to consider evaluating the robustness of their implementation of the IEEE 802.11w amendment and perform intensive testings as of whether their implementations could handle non-standardized behaviors, such as floods of SA-query requests/responses.

5 Conclusion

Deauthentication attacks on Wi-Fi networks constituted a tiresome security threat for many years. Attackers were able to remotely disconnect legitimate devices from a secured Wi-Fi network by merely sending spoofed management frames of type deauthentication and disassociation. In 2009, the IEEE 802.11w amendment came to put an end to many Wi-Fi Denial-of-Service attacks, including deauthentication attacks, through the use of PMF (Protected Management Frames). Later on, some researchers demonstrated the feasibility of certain Denial-of-Service (DoS) attacks on IEEE 802.11w Wi-Fi network. Most of these attacks target the authentication and association phase to deprive devices from getting successfully connected to the network. Only a few of these attacks aimed to cause the disconnection of already connected PMF-enforced devices.

In this paper, we have demonstrated, through various attack patterns, the feasibility of deauthentication attacks on IEEE 802.11w Wi-Fi networks that adopt either WPA2 or WPA3. We have started by briefly presenting the most important concepts of IEEE 802.11w amendment. Then, through numerous experiments, we have demonstrated different deauthentication attack scenarios on PMF-enforced Wi-Fi networks. As part of our experiments, we have identified a vulnerability on certain APPLE devices that could make deauthentication happen within 4 s. After coordinating with Apple products security department, the vulnerability has been fixed in the upcoming version of their systems. Furthermore, we have discussed some hypotheses to why some of the presented attacks were successful. We have recommended to device manufacturers to carefully evaluate the robustness of their implementation of IEEE 802.11w on their devices w.r.t. handling a large number of SA-query requests and responses. In fact, on certain IEEE 802.11w implementations, we have observed a better resilience against the attacks compared to other implementations.

References

1. IEEE. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY). Amendment 4: Protected Management Frames. IEEE Std. 802.11w-2009 (2009)
2. Ahmad, M.S., Tadakamadla, S.: Short paper: security evaluation of IEEE 802.11w specification. In: Proceedings of the 4th ACM Conference on Wireless Network Security, pp. 53–58 (2011)
3. Eian, M.: Fragility of the robust security network: 802.11 denial of service. In: Proceedings of the 7th International Conference on Applied Cryptography and Network Security, pp. 400–416, Springer, Berlin (2009). https://doi.org/10.1007/978-3-642-01957-9
4. Wang, W., Wang, H. Weakness in 802.11w and an improved mechanism on protection of management frame. In: Proceedings of the 2011 International Conference on Wireless Communications and Signal Processing, pp. 1–4 (2011)
5. Valli, K.V., Krishnam, R.K.V.: Formal verification of IEEE 802.11w authentication protocol. In: The 2nd International Conference on Communication, Computing & Security (ICCCS-2012), vol. 6, pp. 716–722, Elsevier (2012)
6. Schepers, D., Vanhoef, M., Ranganathan, A.: DEMO: a framework to test and fuzz Wi-Fi devices. In: Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2021, pp. 368–370, ACM (2021)
7. Ram, M., Kaushik, A.: Deauthenticating and disassociating unauthorized access points with spoofed management frames. United States Patent: US9681299B2, pp. 1–17 (2017)
8. Lounis,K.: Security of short-range wireless technologies and an authentication protocol for IoT. Ph.D. thesis, Queen's University (2021)
9. Lounis, K., Zulkernine, M.: Exploiting race-condition for Wi-Fi denial of service attacks. In: 13th International Conference on Security of Information and Networks, SIN 2020, Istanbul, Turkey, 4–7 November 2020, pp. 1–8 (2020)
10. Lounis, K., Zulkernine, M.: Bad-Token: denial of service attacks on WPA3. In: Proceedings of the 12th International Conference on Security of Information and Networks, Article no. 15, pp. 1–8, ACM (2019)
11. Lounis, Karim, Zulkernine, Mohammad: WPA3 connection deprivation attacks. In: Kallel, Slim, Cuppens, Frédéric., Cuppens-Boulahia, Nora, Hadj Kacem, Ahmed (eds.) CRiSIS 2019. LNCS, vol. 12026, pp. 164–176. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-41568-6_11
12. Lounis, K., Zulkernine, M.: Attacks and defenses in short-range wireless technologies for IoT. IEEE Access J. **8**, 88892–88932 (2020)

13. Lounis, K.: Python-based Scapy scripts for deauthentication attacks on PMF (2021). https://github.com/KarimLounis/Scapy-Scripts
14. Vanhoef, M., Piessens, F.: Key reinstallation attacks: forcing nonce reuse in WPA2. In: Proceedings of the ACM Conference on Computer and Communications Security, pp. 1313–1328, ACM (2017)
15. Lounis, K., Nick: A possible security vulnerability in Wi-Fi PMF on MacOS. Private Email Communications, July 24th to September 1th (2021)