Framing Frames: Bypassing Wi-Fi Encryption by Manipulating Transmit Queues

Domien Schepers and Aanjhan Ranganathan, *Northeastern University;*
Mathy Vanhoef, *imec-DistriNet, KU Leuven*

https://www.usenix.org/conference/usenixsecurity23/presentation/schepers

This paper is included in the Proceedings of the
32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Proceedings of the
32nd USENIX Security Symposium
is sponsored by USENIX.

Framing Frames: Bypassing Wi-Fi Encryption by Manipulating Transmit Queues

Domien Schepers
Northeastern University
schepers.d@northeastern.edu

Aanjhan Ranganathan
Northeastern University
aanjhan@northeastern.edu

Mathy Vanhoef
imec-DistriNet, KU Leuven
mathy.vanhoef@kuleuven.be

Abstract

Wi-Fi devices routinely queue frames at various layers of the network stack before transmitting, for instance, when the receiver is in sleep mode. In this work, we investigate how Wi-Fi access points manage the security context of queued frames. By exploiting power-save features, we show how to trick access points into leaking frames in plaintext, or encrypted using the group or an all-zero key. We demonstrate resulting attacks against several open-source network stacks. We attribute our findings to the lack of explicit guidance in managing security contexts of buffered frames in the 802.11 standards. The unprotected nature of the power-save bit in a frame's header, which our work reveals to be a fundamental design flaw, also allows an adversary to force queue frames intended for a specific client resulting in its disconnection and trivially executing a denial-of-service attack.

Furthermore, we demonstrate how an attacker can override and control the security context of frames that are yet to be queued. This exploits a design flaw in hotspot-like networks and allows the attacker to force an access points to encrypt yet to be queued frames using an adversary-chosen key, thereby bypassing Wi-Fi encryption entirely.

Our attacks have a widespread impact as they affect various devices and operating systems (Linux, FreeBSD, iOS, and Android) and because they can be used to hijack TCP connections or intercept client and web traffic. Overall, we highlight the need for transparency in handling security context across the network stack layers and the challenges in doing so.

1 Introduction

In many scenarios, Wi-Fi devices will opt to buffer or queue packets that arrive from the upper layers before they are transmitted. One of the most common use cases is to conserve power on devices such as mobile phones and laptops. The first release of the 802.11 standard already contained power-save mechanisms that allow clients to enter a *doze or sleep* power state to consume low power. When a client enters a sleep state, the Access Point (AP) buffers eligible frames destined for the client. The buffered frames are later transmitted to the client following a specific protocol. Frames might also be queued by the hardware when the sender is waiting for available medium access or while waiting for an acknowledgment of a transmitted frame that may require retransmission. Frames are also buffered at the receiver. For example, frame fragmentation allows large frames to be broken up into smaller fragments and transmitted. At the receiver, frames are buffered until all fragments are received for reassembly.

Although queues by themselves are standard data structures, managing the security context of the 802.11 frames queued in Wi-Fi devices is non-trivial. We use the term security context as a synonym of what is called a *security association* in the 802.11 standard [23, §12.6.1.1]. A security context contains all information needed to securely communicate with a peer, such as the negotiated session keys, the encryption protocol used, the current packet counters, etc. The security context is used both for sending and receiving frames. Most recently, researchers pointed out design flaws in how receivers processed fragmented frames without any regard for the relevant security context [44]. Specifically, receivers reassembled fragments that were decrypted using different keys. Since a decryption key is part of the security context when processing a fragment, the underlying problem was that receivers were mixing fragments of different security contexts.

So far, no work has rigorously analyzed how a station manages the security context of queued frames at the transmitter. The Kr00k attacks showed some devices can be tricked into sending frames queued in the hardware chip using an all-zero encryption key [34]. However, they did not systematically study the security impact related to uninitialized or mishandled security contexts, e.g., in different queues in the network stack. Our study reveals that the problem is more fundamental. An adversary can actually force the queueing of frames, which we show gives an adversary better control over the attack timing, meaning more frames will be buffered and leaked. Overall, in this work we investigate the (mis)management of the security context when transmitting queued or buffered

Wi-Fi packets over all queues in the network stack, and argue that the root cause of the discovered flaws is a vague standard.

Specifically, we make the following contributions. First, we observe the standard lacks explicit guidelines on how security contexts of buffered frames should be managed. There is no guidance on handling changing security contexts, e.g., when a device disconnects or reconnects to the network. Motivated by this, we investigate open-source network stack implementations in FreeBSD, Linux, and hardware dongles with open-source firmware and find most stacks do not securely manage their transmit queues. We discover vulnerabilities that allow an adversary to exploit the power-save functionality and place frames into the transmit queue of the AP. Once in the transmit queue, we manipulate their security context, which we refer to as *framing the frame*, and force the AP to transmit all the frames in the queue thereby revealing its contents. Our attacks show that several APs can be tricked to transmit frames in plaintext or encrypted using the group key or an all-zero key.

Second, we show how an adversary can abuse the power-save mechanism to cause clients to disconnect and more broadly execute denial-of-service attacks. Particularly, we exploit the unprotected power-save bit in the frame header to falsely inform a client's power state and trigger the AP's power-saving mechanisms for that client, i.e., queue frames intended for the client resulting in disconnection of the client. Our attack highlights the need to authenticate the power-save bit in the header of frames and the challenges in doing so, e.g., affecting backward compatibility. We demonstrate our attack against Linux and popular iOS and Android smartphones. Additionally, we demonstrate an attacker can force queueing of data frames leading to a trivial denial-of-service attack that is effective against any network configuration including WPA3.

Third, we investigate whether an adversary can influence or control the security context of frames that the AP has not yet received. Particularly, we explore whether an attacker can abuse the fact that the security context of a frame is based on the sender/destination address. We answer this question positively by demonstrating how an adversary can change the security context of the whole transmit queue and transmit the frames under an attacker-controlled security context. Our evaluations reveal that all tested APs are vulnerable to our security-context override attacks, i.e., an adversary can force frames not yet in the queue to be encrypted using an adversary-controlled security context (e.g., keys) thereby *bypassing encryption at the Wi-Fi layer.*

Finally, we elaborate on the practical impact of intercepting frames through several use cases such as hijacking TCP connections, and intercepting client and web traffic. Through this work, we highlight the complexity of the network stack and the need for a defense-in-depth approach where the user-space, kernel, firmware, and hardware all have to be transparent in terms of security context management. Since frames can be queued at various layers in the network stack, each layer must properly handle changes in the security context.

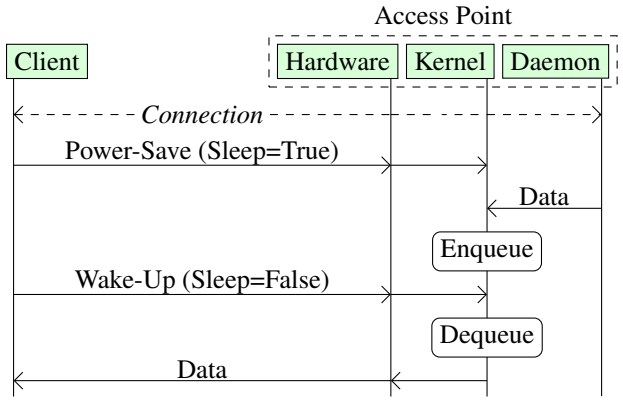

Figure 1: Power-saving mechanisms enable a client station to enter power-save mode. Until the client wakes up, the access point will queue all bufferable frames destined for the client.

This complicates the analysis of systems since every layer must be carefully inspected, which is non-trivial because the behavior of one layer may influence the security of another.

Coordinated Disclosure We disclosed all identified vulnerabilities to the affected vendors. An overview and timeline of the disclosure process is available on our public repository.[1]

2 Background

In this section, we provide an overview of the power-saving mechanism and Wi-Fi Management Frame Protection (MFP).

2.1 Wi-Fi Power-Saving Mechanisms

Power-save mechanisms have been part of the IEEE 802.11 standard since its first release in 1997. With the introduction of the IEEE 802.11e amendment, defining Quality-of-Service (QoS) enhancements for wireless LAN networks, the standard adopted more advanced power-save mechanisms [23, §11.2].

2.1.1 Frame Queuing in the Wi-Fi Stack

For this paper, it is important to understand when and where frames are queued (e.g., where in the network stack). In Figure 1, we illustrate a typical exchange of power-save messages between a client station and an access point. At any moment during the connection, the client station can indicate it will enter power-save mode. In order to do so, the client can set the power-bit in the IEEE 802.11 header frame format. When the access point receives frames to transmit to the client, they will be buffered in the kernel's transmit queue until the client wakes up (e.g., polls for data) or the frame's lifetime gets exceeded (i.e., a frame is queued only for a maximum duration).

[1] https://github.com/domienschepers/wifi-framing

How long frames can stay queued is considered beyond the scope of the standard [23, §11.2.3.10]. At this point in time, all the queued frames are stored in plaintext [23, Fig. 5-1]. Once the client indicates it is waking up, the access point's kernel can dequeue all the buffered frames, apply any encryption operations, and transmit them to the client station.

In addition to the transmit queue within the kernel, frames can be queued by the hardware. This can be due to the need to wait for access to the wireless channel or to allow for retransmissions when the receiver did not acknowledge the transmitted frame. Depending on the hardware capabilities (e.g., support for hardware encryption) the queued frames may be stored in plaintext or encrypted in hardware.

2.1.2 Bufferable Frames

While a client station is in power-save mode, the access point will buffer its frames, however, not all frames are bufferable. In order to standardize which frames are bufferable, the IEEE 802.11 standard defined a Bufferable Unit (BU). Bufferable units are eligible to be queued using a power-saving mechanism, while all others are to be delivered immediately.

Bufferable units include the MAC Service Data Unit (MSDU), Aggregate MSDU (A-MSDU), and MAC Management Protocol Data Unit (MMPDU). Since not all MMPDU frames are bufferable, the standard defines a number of classifications [23, §11.2.2]. In one of its classifications, it is defined that MMPDU Action frames (except for Fine Timing Measurement frames), disassociation, and deauthentication are bufferable:

> *An MMPDU that is carried in one or more Action (except for Fine Timing Measurement frame and Fine Timing Measurement Request frame), Disassociation, or Deauthentication frame.*

This clearly indicates not all frame types are bufferable, with notable examples of non-bufferable frame types being the authentication and association response messages. An important example of frame types that are eligible to be buffered according to the standard are Security Association (SA) Query messages. These messages are used to protect the SA under Wi-Fi Management Frame Protection (MFP).

2.2 Wi-Fi Management Frame Protection

Wi-Fi Management Frame Protection (MFP) standardized protection mechanisms for management frames including data confidentiality, integrity, origin authenticity, and replay protection. Wi-Fi MFP is defined in the 802.11w amendment and incorporated in the 2012 version of the 802.11 base standard. The new protection mechanisms increase the security of robust management frames, which represent a subset of all management frames. Notable examples are robust action frames (e.g., used for spectrum management) and (re)association and deauthentication frames. With the new defenses, networks are protected against a variety of known attacks that abuse management frames, for example, deauthentication attacks which disconnect clients from the network [40]. The Wi-Fi Alliance made Wi-Fi MFP mandatory in all its certification programs since 2020 [16] and Wi-Fi Protected Access 3 (WPA3) [3], though a backwards-compatible transition mode exist where stations are capable of Wi-Fi MFP but do not yet require its usage. As a result of these standardization efforts, researchers observed a growing adoption of both WPA3 and MFP [39].

When Wi-Fi MFP is used between two capable stations, selected managements frames, such as association and disassociation frames, are protected under the current Security Association (SA). The security association is a set of policies and cryptographic keys used to protect information within a connection and is stored by each party in the association. When a station has an existing security association, then any association frame will be temporarily rejected by the access point to prevent denial-of-service attacks. Subsequently, the access point will initiate the SA Query procedure to determine whether the client still has an active security association with the AP [23, §11.13]. In an SA Query procedure, the stations exchange a protected query request and response. If the procedure succeeds, the security association is determined to still be valid and the stations can safely continue using the association. Consequently, unprotected association frames can be safely discarded. If the procedure fails or times out, it indicates there might be a mismatch in the association (e.g., due to an unplanned reboot of the client station) and therefore the security association can be torn down (i.e., the client station can be disconnected from the network).

3 Leaking Frames from the Wi-Fi Queue

In this section, we study how access points behave when the security context of queued frames changes. We first analyze the standard and find that it does not explicitly define how to handle changing security contexts. Based on this, we describe a general attack strategy that forces an AP to queue frames in order to subsequently change the security context and leak the queued frames. We then analyze open-source network stacks and provide instantiations of our strategy which causes implementations to leak frames in plaintext, encrypt them using the group key, or encrypt frames using an all-zero key.

3.1 Motivation: Under-Specified Standard

A Wi-Fi device cannot instantly transmit packets that arrive from the upper layer, and in practice, frames will be queued before transmission. For instance, they may be queued by the hardware while the receiver is in power-save mode, while the sender is waiting for available air time, or while frames are not yet acknowledged and therefore may need to be retransmitted. This raises an important question: how should a sender behave

if the security context changes between the time when a frame was queued and the time where it is actually (re)transmitted?

To answer this question, we inspected the 802.11 standards and found only indirect guidance on how a transmitter should manage buffered frames. The standard mentions an aging function to delete frames which have been buffered for an excessive amount of time but does not define its precise behavior: "The exact specification of the aging function is beyond the scope of this standard" [23, §11.2.3.10]. Though there are service specifications that assure a station should not send frames in plaintext once encryption is enabled, we did not find explicit instructions on how to handle changing security contexts, nor that buffered frames should be removed (or temporarily stored) when a device disconnects or reconnects to the network. To analyze how devices implement edge cases in practice, we investigate open-source operating systems and hardware dongles with open-source drivers and firmware.

3.2 Threat Model

We consider an attacker with the goal of leaking frames from the access point destined to a victim client station. The attacker can inject and intercept frames and manipulate the victim client's security context, e.g., injecting unprotected management frames such as authentication and association frames. Furthermore, the attacker can tamper with the power-save status of the victim client by spoofing frames that use the power-save bit (e.g., an unprotected null-data frame). Typically, the attacker does not require knowledge of network credentials and therefore is an outside threat. However, under certain conditions, leaked frames are protected with the network group key, in which case network credentials are required to obtain the respective key.

3.3 General Attack Strategy and Methodology

The high-level strategy behind our transmit queue leak attack relies on the observation that most Wi-Fi stacks do not adequately dequeue or purge their transmit queues when the security context changes. As an attacker, we can abuse this behavior by spoofing certain management frames under the MAC address of a victim client, thereby cleverly interfering in the security context and power-save mechanism.

3.3.1 Attack Strategy

Consider a client who is connected to an access point with power-management support. In Figure 2, we illustrate how an attacker can trick a vulnerable access point into leaking frames, destined for the victim client, from its transmit queues.

1. The attacker spoofs an unprotected power-save frame to trick the access point into believing the victim client is entering power-save mode. As a result, the access point queues (i.e., buffers) all frames for the client until the client indicates it has left power-save mode. Importantly, at this stage, all frames are queued in plaintext.

2. The attacker now aims to trick the access point into removing the encryption keys of the victim client (i.e., the pairwise encryption key). While it may take certain creativity to achieve this goal, generally management frames such as authentication and (re)association requests will cause an access point to update its security context. When successful, the access point no longer has access to a pairwise encryption key for the victim client.

3. The attacker spoofs an unprotected wake-up frame to trigger the access point into transmitting all the frames queued for the client. Since the vulnerable access point no longer has a pairwise encryption key for the client it may resort to using no encryption (i.e., plaintext) or fall back to the group-addressed encryption key.

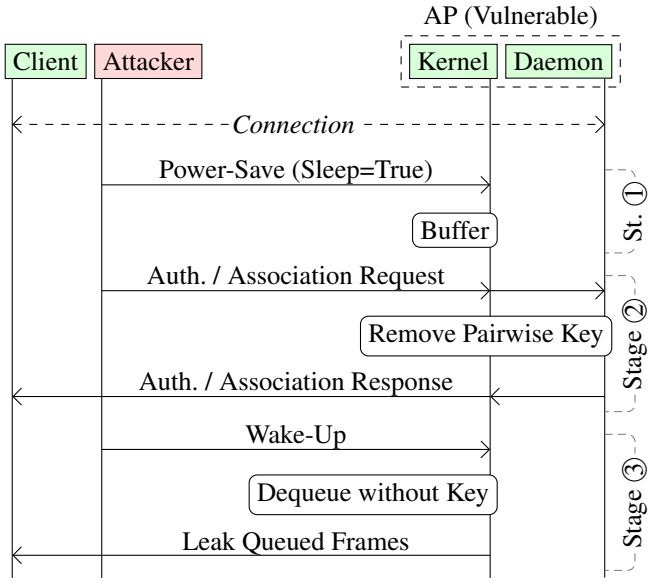

Figure 2: An attacker can leak the queue of a vulnerable access point by putting the client in power-save mode (Stage 1). The attacker then tricks the access point into removing the pairwise key (Stage 2) using authentication or association messages. Finally, the attacker wakes up the client causing the access point to send frames without encryption (Stage 3).

As a result of the attack, anyone within the communication range of the vulnerable access point can intercept the leaked frames in plaintext or encrypted using the group-addressed encryption key, depending on the respective implementation of the stack (i.e., user-space daemon, kernel, driver, firmware).

3.3.2 Instantiating the Attack

In order to investigate whether such an attack is feasible in practice, we audited open-source network stacks. This re-

vealed various implementation-specific behaviors that must be taken into account when trying to successfully execute the attack. For instance, the type of frame used in stage 2 to remove the pairwise key depends on the implementation being targeted. Additionally, the frame can be leaked in plaintext, may be encrypted using an all-zero key, can have a seemingly valid encryption header while having plaintext content, may be erroneously encrypted using WEP, and so on. This results in various different instantiations of our attack strategy.

In the remainder of this section, we evaluate and discuss open-source operating systems which support an operating mode for access points with power-management support and we evaluate hardware dongles that have open-source firmware. Specifically, we evaluate FreeBSD 13.0 and 13.1, Linux 5.5.0 to 5.17.6, and hardware dongles with the Atheros AR9271 hardware chipset (*ath9k_htc* driver). Since operating systems such as NetBSD do not offer power-management support (i.e., when operating as an access point) we did not analyze them.

3.4 Frame Leaks in FreeBSD

To begin our investigation we inspect if our attack is feasible against access points operating on FreeBSD 13.0 and 13.1. To fully understand the network stack and its behavior, we inspect the kernel code as well as vendor-specific drivers. Our investigation reveals two major issues in the kernel and drivers that must be taken into account when instantiating our attack:

1. Drivers are often programmed to call upon default functions in the FreeBSD kernel. For example, the `ieee80211_crypto_get_txkey()` function is used to select the encryption key. Interestingly, this function will fall back on the group key if no pairwise key is available. An attacker can exploit this behavior to force the vulnerable access point into encrypting queued frames using the group encryption key.

2. Modern devices are often capable of offloading encryption operations to hardware, including its key selection. While specific implementations are therefore unavailable for inspection, we find devices with hardware encryption may fall back on plaintext or an all-zero encryption key.

We find that if the vendor-specific driver implementation falls into one of these two categories, an attacker is able to leak the queue in either plaintext or the group encryption key.

3.4.1 Instantiating the Attack

The resulting attack against FreeBSD access points closely follows the attack strategy outlined in Section 3.3 and illustrated in Figure 2. Specifically, the attacker has to use a re-association request to trick the vulnerable access point into removing the pairwise key of the victim client (Stage 2), but the adversary needs to be aware that frames can also be leaked using the group key. Note that if the queue is leaked using the group key, the attacker must have valid network credentials in order to obtain the respective encryption key.

Table 1: Queue leak vulnerabilities within FreeBSD where Hardware Encryption (HWE) is enforced (●) or optional (◑).

Ver.	Vendor (*Driver*)	Leakage	HWE
13.0	Ralink (*run*)	Plaintext	●
13.1	Ralink (*run*)	WEP with All-Zero Key	●
13.1	Ralink (*rum*)	CCMP with Group Key	●
13.1	Realtek (*rtwn*)	CCMP with Group Key	◑

3.4.2 Evaluation

Whether an access point is vulnerable depends on its driver implementation and therefore we evaluate a variety of dongles using hardware by Ralink (*run* and *rum* drivers) and Realtek (*rtwn* driver). In Table 1, we present an overview of our evaluation results. Note that certain drivers do not allow the configuration of hardware encryption, that is, hardware encryption is enforced (represented with ●) or optional (◑). For Ralink (*run*) we evaluated the ALFA Network AWUS036NH, Belkin F5D8053 v3, and Linksys WUSB600N v1. For Ralink (*rum*) we evaluated the Sitecom WL-172 v1. For Realtek (*rtwn*) we evaluated the Edimax EW-7811Un v2, TP-Link TL-WN722N v3, and TP-Link TL-WN725N v3.8.

We find that Ralink (*rum*) and Realtek (*rtwn*) call the vulnerable key selection function, and as a result, their queue is leaked using the network's group encryption key. However, Ralink (*run*) does not call the kernel function. Instead, the driver offloads both encryption and decryption of data frames to its hardware and as a result, we are unable to inspect its source code. However, from observing network traffic we find the queue transmits frames in plaintext in FreeBSD 13.0, thereby trivially leaking the frames. Specifically, we find the encryption flag is set in the frame header, however, the encryption operation is not performed before transmission. Furthermore, against FreeBSD 13.1 we find the hardware falls back to Wired Equivalent Privacy (WEP) encryption using an all-zero encryption key, thereby also leaking queued frames.

While we were unable to evaluate additional hardware, we find drivers such as Atheros (*ath* driver) indirectly call the vulnerable key selection function. Therefore, we conjecture such devices also leak frames using the group encryption key.

3.5 Frame Leaks in Linux and Hardware

Our investigation reveals a number of attack variations against access points on Linux. For each attack variation, the attacker successfully leaks frames from the transmit queue in plaintext.

3.5.1 Leaks on Encrypted Links without Encryption Key

Earlier Linux kernel implementations did not drop data frames when the encryption key was no longer available on encrypted links. Instead, its data frames were transmitted in plaintext. In a legitimate scenario, this may occur when the station disconnects while there are still data frames in the transmit queue of the access point. In order for this attack variant to be successful, the access point must not be capable of maintaining full client state (i.e., the hardware driver does not support this feature). This applies to older Linux kernels (i.e., the `mac80211` kernel module enables full access point client state by default since Linux 4.5.0) and may apply to FullMAC driver implementations (i.e., FullMAC drivers make use of their own `mac80211` module implementation). As a result of not maintaining full client state, the user-space daemon (e.g., hostapd) will not delete the entire state of a client station (including its transmit queue) upon receiving, for example, an authentication request. An attacker can leverage this behavior to delete the pairwise key and leak all frames from the transmit queue in plaintext.

Attack Instantiation Consider a client connected to an access point that does not maintain full client state. Similar to the attack outline in Figure 2, an attacker can leverage this behavior to leak frames from the transmit queue in plaintext. After forcing the access point to buffer frames for the victim client (Stage 1), the attacker transmits an authentication request. Since the access point does not maintain full client state, the pairwise encryption key is now deleted without purging the transmit queue (Stage 2). Finally, the attacker can send a wake-up frame to initiate the dequeuing process (Stage 3) and a vulnerable access point will leak all its frames in plaintext.

Evaluation We evaluated Linux 5.5.0 with the hostapd 2.9, hostapd 2.10, and IWD 1.28 user space daemons and confirm the attack is successful. In order to be successful against the IWD implementation, a re-association request must be sent (instead of the authentication request) to remove the pairwise encryption key. From Linux 5.6.0 onward, data frames are dropped by the kernel if no encryption key is available on encrypted links, effectively preventing this attack method.

3.5.2 Leaks due to Race Conditions in Hardware

Finally, we identified two attack variations against Linux when the encryption operations are offloaded to the hardware. Specifically, we find race conditions in the (re)transmission mechanisms that may cause frames to be leaked in plaintext. We evaluated both attacks on Linux 5.8.0 using a TP-Link TL-WN722N v1.10 dongle with an Atheros AR9271 hardware chipset. Since the hardware has an open-source driver and firmware code (*ath9k_htc*), we can inspect its implementation.

Retransmissions In this attack variation, we find frames leak in plaintext after a number of retransmission attempts. Consider a legitimate client which fails to acknowledge the frame (e.g., due to frame collisions or poor radio channel conditions), then the hardware will retransmit the frame. If these retransmissions are unsuccessful the hardware will transmit request-to-send frames, and if a clear-to-send frame is received, queued frames are retransmitted once more. Note an attacker can use commodity hardware to selectively jam [45] the initial transmission to trigger retransmissions. Now consider an attacker who transmits an authentication request without sleep-bit to remove the pairwise key and wake up the client. The initial transmission will use the appropriate encryption as expected, however, after the request-to-send procedure, the encryption key has been deleted causing the final retransmission to be in plaintext.

Race Condition In this attack, the adversary transmits an authentication request without sleep-bit. We observe a race condition causing the encryption key to be deleted from hardware, which is removed by writing to memory-mapped registers, while other state does not appear to get updated. As a result, all the queued frames are transmitted in plaintext.

3.6 Defenses

The standard defines explicit procedures to follow up on the receipt of management frames, for example, in [23, §11.3.4.3] the procedure is given for processing an authentication frame. However, the standard does not define any behavior for how an access point should manage its transmit queues when the security context changes. From our results, it becomes clear such behavior must be defined. We identify two approaches that can be used to secure the behavior of transmit queues:

1. Prior to the deletion of the pairwise encryption key, the transmit queue must be dequeued regardless of the power-save status of the receiving client station. That is, the access point makes a final best-effort attempt to deliver all frames stored in the transmit queue.

2. Prior to the deletion of the pairwise encryption key, the transmit queue should be purged. That is, the access point drops all frames stored in the transmit queue.

Then and only then can the pairwise encryption key be deleted. Note these approaches cover the rekeying procedures as well, i.e., delete an expired and install a fresh pairwise key.

4 Abusing the Queue for Network Disruptions

In addition to manipulating the security context to leak frames from the queue, an access point can be tricked into enabling its power-saving mechanisms for a victim client. That is,

since the power-bit in the header of a frame is unprotected, an attacker can trivially enable power-saving mechanisms and thereby selectively queue frames at the access point. Consequently, client stations are vulnerable to disconnection and denial-of-service attacks. While frames can be blocked using other techniques, for example, jamming or a multi-channel MitM, abusing the queue requires fewer resources. Performing a multi-channel MitM requires two wireless network cards, while our attacks only require one. Similarly, jamming requires more specialized hardware, while our attack can be performed with any network card that supports frame injection. An adversary can also combine these jamming techniques to improve the overall success rate of blocking frames.

4.1 Queueing of SA Query Requests

In our first attack abusing the transmit queue, we show how the SA Query procedure (Section 2.2) can be exploited to disconnect a client from the network. Recall the standard defines which management frames are bufferable and thus allowed to be queued using a power-saving mechanism [23, §11.2.2] (Section 2.1.2). SA Query request and response frames are transmitted as robust (i.e., protected) Action frames [23, §9.4.1.11]. However, the standard does not explicitly exclude them from being bufferable (i.e., Action frames are bufferable with certain exceptions). When a station queues either the request or response message, it can cause a timeout on the receiving end since it is a time-sensitive security procedure. As a consequence the security association must be torn down, effectively disconnecting the victim client from the network. Given this behavior, access points are exposed to a novel queue-based disconnection attack against its clients, even if the network configuration enforces Wi-Fi MFP (e.g., WPA3).

4.1.1 Attack Outline

To demonstrate how the standard definition can be exploited, we identified the following attack. Consider a victim client which is connected to a vulnerable access point, where the stations have agreed to enforce Wi-Fi MFP protection. In Figure 3, we demonstrate how an attacker can target a vulnerable access point to disconnect the victim client in three stages.

1. The attacker spoofs an association request from the victim client, and sets the sleep-bit in the frame header. This will mark the victim client as asleep, causing buffereable frames to be queued at the access point. Since Wi-Fi MFP is required, the association request is rejected.

2. As part of Wi-Fi MFP protection mechanisms, the access point initiates the SA Query procedure. It transmits a number of SA Query requests, which are now queued by the kernel. As a result, the access point will never receive a response and therefore the procedure will time out, and now the daemon will allow new association requests.

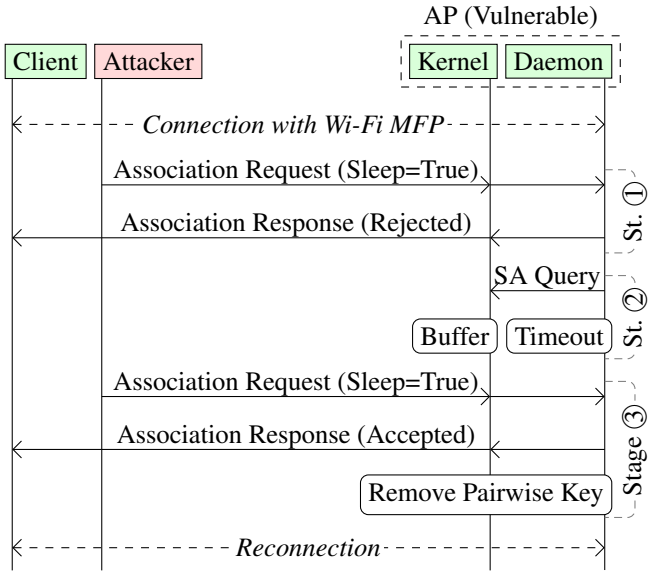

Figure 3: An attacker triggers the Security Association (SA) Query procedure with an association request and marks the victim client as asleep (Stage 1) causing SA Query requests to be queued (Stage 2). After timeout, a new association request will clear the pairwise key, disconnecting the client (Stage 3).

3. The attacker spoofs a second association request, which is now allowed by the access point. As defined by the association procedure, the access point now removes the victim client's pairwise encryption key.

Note the client silently discards all unsolicited association responses. Furthermore, following the three stages, the access point may start a new 4-way handshake procedure. However, since the client still has its pairwise keys installed, it will respond with encrypted handshake messages. The access point can not read these messages since it previously removed the pairwise encryption key. As a result the handshake will fail, effectively disconnecting the victim client and access point.

4.1.2 Evaluation

We evaluated Linux 5.17.6 with the hostapd 2.10 user space daemon and confirm the attack is successful, independently of hardware encryption and thus the hardware used by the access point. Furthermore, the attack is successful against Apple's iOS 15.5 and macOS 12.4 when operating as an access point (e.g., "Personal Hotspot" on iOS). Note the attack could not be tested against Android since its hotspot does not yet support clients which require Wi-Fi MFP protection.

4.1.3 Discussion

The standard specifies SA Query procedures for Sub 1 GHz (S1G) stations which enter power save mode [23, §11.13].

The operation of S1G stations, defined in the IEEE 802.11ah amendment, enable communication on a longer range with lower energy consumption [42]. As such, it serves a great purpose for large sensor networks, internet-of-things applications, and smart cities [24]. According to the SA Query procedures, the S1G station must wake up within a duration that is not larger than some predefined maximum timeout interval, otherwise the security association can be torn down. Interestingly, the standard does not define behavior for stations operating on the more commonly-used 2.4 and 5 GHz band. Regardless, with the defined procedures, stations remain vulnerable to our attack. Consider a legitimate client which is asleep, an attacker can simply spoof a wake-up frame to dequeue the requests. No valid responses will be sent since these will not be received by the client, resulting in the teardown of the SA.

As such, our attack exposes shortcomings in the standard. We hypothesise that, in order to prevent such an attack, it is necessary to (1) authenticate the sleep-bit and polling frames, preventing that an attacker is capable of manipulating the state of the queue, and (2) a sleeping client must always poll the access point within the predefined maximum timeout interval, ensuring buffered SA Query requests are received in time. Until then stations remain vulnerable to our SA Query attack.

4.2 Queueing of Wi-Fi FTM Requests

Wi-Fi Fine Timing Measurement (FTM), defined in IEEE 802.11mc and incorporated in IEEE 802.11-2016, is used to measure the physical distance between two stations. The protocol makes use of round-trip time-of-flight measurements, and allows for meter-level ranging accuracy in low-multipath environments [10, 22, 40]. The IEEE 802.11 standard defines that Wi-Fi FTM frames are not bufferable, and are to be delivered without reference to a power-saving mechanism [23, §11.2.2] (Section 2.1.2). That is, even though Wi-Fi FTM frames are constructed as an action frame, they are explicitly listed as an exception of bufferable frames. In practice, the queuing mechanism affects only the Wi-Fi FTM Request frame, which is transmitted by a station to initiate the distance measurement session with another capable device. The Wi-Fi FTM Response frames, which are responsible for measuring the round-trip time-of-flight, are processed by the firmware. That is, any critical time-sensitive operation is performed on the hardware device and therefore not affected by power-saving mechanisms. Nonetheless, when a request is queued, the station will not initiate a new distance measurement session. This behavior may expose clients to potentially (security-sensitive) distance measurement disruptions.

Attack Outline Consider the scenario where a client station is connected to a network and needs to regularly report its physical distance to an access point in a geo-fencing context (an example application provided by the Wi-Fi Alliance [4]). At any point, the attacker can spoof a frame towards the access point indicating the client is entering power-save mode. Consequently, the measurement session will not be initiated because its requests are queued by the kernel. This may result in a timeout on the higher-layer application (e.g., wireless daemon) which in turn could trigger the access point to remove the victim from the geo-fenced area (i.e., the access point concludes the client is irresponsive and has left the geo-fenced area), or disconnect the client from the network altogether.

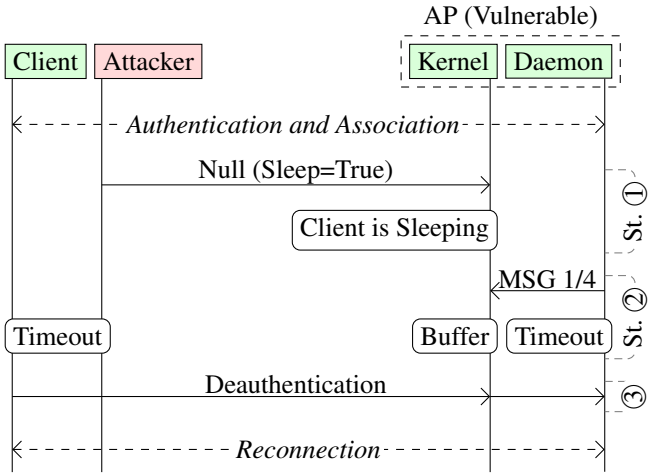

Figure 4: An attacker can spoof messages which set a victim client to be asleep (Stage 1) causing the first 4-Way Handshake message to be queued (Stage 2). After not receiving the message the victim client timeouts and disconnects (Stage 3).

Evaluation We evaluated Linux 5.17.6 with a custom-built application to initiate distance measurement sessions, and find the kernel buffers Wi-Fi FTM frames in practice. Since Wi-Fi FTM applications are most often deployed on Linux-based systems [41], we conjecture they may all be vulnerable.

4.3 Queueing of 4-Way Handshake Messages

In addition to action frames targeted before, an attacker can force data frames to be queued. When targeting messages in the 4-way handshake, that is, during a connection procedure, it can trivially lead to a queuing-based denial-of-service attack. Since this attack targets the 4-way handshake, it is effective against any type of network configuration (e.g., WPA3).

4.3.1 Attack Outline

Consider a victim client which is connecting to a vulnerable access point. In Figure 4, we demonstrate how an attacker can target a vulnerable access point to prevent the victim client from connecting in three stages.

1. After the association procedure, the attacker spoofs a null-frame, and sets the sleep-bit in the frame header.

This causes the kernel to mark the victim client as asleep.

2. The vulnerable access point will now queue all bufferable frames, including the first 4-way handshake message (which is sent as a data frame). This behavior leads to a handshake timeout at both the access point and victim client, since the handshake messages are never sent.

3. The victim client expects the first 4-way handshake message, however never receives any. Consequently, a timeout will abort the handshake procedure causing the victim client to send a deauthentication message.

Since the access point is still buffering frames, its deauthentication message is not transmitted to the victim client either. Consequently, the client will time out and disconnect from the network. Following the three stages, the stations may restart the connection procedures if they are configured as such.

4.3.2 Evaluation

We evaluated Linux 5.17.6 with the hostapd 2.10 and IWD 1.28 user space daemons and confirm the attack is successful. Furthermore, the attack is successful against Apple's iOS 15.5 and macOS 12.4 when operating as an access point (e.g., "Personal Hotspot" on iOS), as well as hotspots on Android 12 and Android 13 beta (i.e., "Wi-Fi hotspot").

5 Overriding the Victim's Security Context

In this section, we show how to fully control the security context that an AP uses to protect frames sent toward a victim. First, the adversary connects to the AP and negotiates a new security context. Then, the AP is tricked into associating the now attacker-controlled security context with frames belonging to a victim. As a result, the adversary can decrypt frames meant for the victim. This attack abuses design flaws in how a security context is associated with outgoing frames and is exploitable in hotspot-like networks.

5.1 Threat Model

In this section, the attacker's goal is to make the AP transmit frames under a security context controlled by the adversary. In order for the adversary to create a security context on the AP, they must be able to successfully connect to the AP. Therefore, we will target hotspot-like networks where users distrust each other and for which the adversary possesses valid credentials. An example of such networks are enterprise Wi-Fi networks such as eduroam, or Wi-Fi Passpoint networks where users can, for instance, authenticate using their mobile SIM card [2]. Another example is WPA3's new SAE-PK networks, which are hotspots where users can connect using a pre-shared password, but an adversary cannot use this password to create rogue clones of the network [50]. We assume that these networks are configured to disallow client-to-client communication, which prevents one client from intercepting the traffic of another client by establishing Machine-in-the-Middle positions using techniques such as ARP poisoning [31].

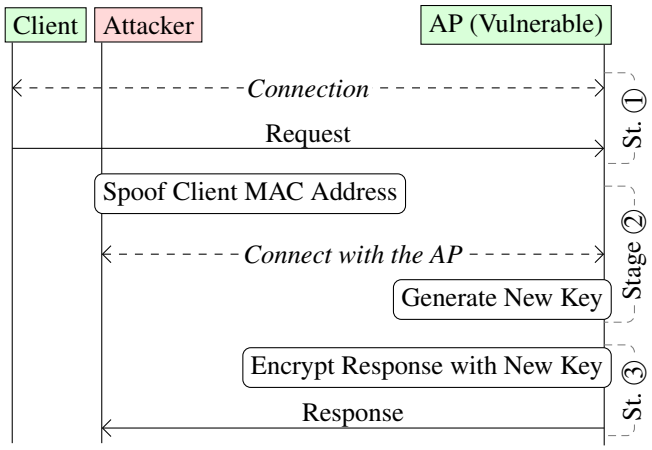

Figure 5: Illustration of the security context overriding attack.

5.2 Overriding a Client's Security Context

When an AP transmits a frame, the corresponding security context is looked up based on the sender's MAC address and the frame's destination. This assures that the correct session keys are used to send frames to each client and that the group key is used to protect broadcast and multicast frames. However, if an attacker can override the security context that the AP associates to a particular destination MAC address, frames (not yet) in the transmit queue will be protected using this attacker-controlled security context.

Attack Description. The adversary can override a security context that the AP associates with a client by spoofing the MAC address of the client and then connecting to the AP. This causes the AP to encrypt traffic towards this client using session keys that the adversary possesses. In Figure 5, we illustrate the individual stages behind this attack:

1. We assume the victim is currently connected to the AP and is about to receive a frame that the adversary wants to intercept. For instance, the victim may send an HTTP request without using TLS, and the adversary wants to intercept the HTTP response. Note that traffic analysis techniques [29, 35] can be used to detect requests for which the adversary wants to intercept the response.

2. Before the client receives the response, the adversary spoofs the victim's MAC address and connects to the AP using its own credentials (e.g., its personal credentials in WPA2 or WPA3 enterprise networks). As a result, the

adversary overwrites the security context that the AP associates with the MAC address of the victim with the security context controlled by the adversary.

3. The AP now sends frames to the victim's MAC address under the security context of the attacker. In other words, packets intended for the victim, such as HTTP responses, are now encrypted by keys that the adversary possesses.

Since packets intended for the victim are now encrypted using keys that the adversary possesses, the adversary can trivially decrypt them. That is, *we can bypass encryption at the Wi-Fi layer*. Note that even if the response is encrypted at a higher layer using a protocol such as TLS, our attack will still leak the IP addresses that the victim is connecting to. In Section 6, we further discuss the impact of intercepting responses in this manner and we will also demonstrate how this enables an adversary to subsequently inject data towards a victim.

A precondition of our attack is that the adversary must be able to connect under the MAC address of the victim, which is why we target hotspot-like networks. This also implies that the legitimate client must first be disconnected from the network. When MFP is not used, this can be accomplished by spoofing deauthentication frames. In case the victim is using MFP, prior work has shown that implementation vulnerabilities can usually still be abused to disconnect clients [40]. Moreover, in Section 5.3, we show how to perform our attack without having to first disconnect the victim, meaning the attack can be performed whether or not MFP is being used.

One practical limitation of the attack is that it might take a substantial amount of time for the adversary to connect with the AP in stage 2 of the attack, causing the response to be missed (Figure 5). For instance, when authenticating using EAP-based protocols, a remote RADIUS server is typically used to authenticate the client, and communicating with this server may incur high delays. If the response that we want to intercept is sent over TCP this is no issue: the sender will automatically retransmit it when not acknowledged. However, for protocols such as UDP, this may be problematic, as the response may be missed. To overcome this limitation, we will discuss fast (re)connection techniques in Section 5.3.

Our attack is independent of the authentication and encryption scheme used, with the main precondition being that the adversary must possess credentials to connect to the hotspot-like network. In particular, the attack works against both WPA2 and WPA3 *independent* of their precise settings.

Experiments We evaluated our security context overriding attacks against professional and home APs that support enterprise WPA2 or WPA3, i.e., authentication using 802.1X. We force the victim to send a UDP packet to our server and our server echoes this packet back. The attack was considered successful if the adversary was able to intercept the UDP response. Table 2 lists the devices that we tested, where our attack was successful against all devices. Especially relevant are professional APs (the first four APs in Table 2) as these are more commonly used in hotspot-like networks such as eduroam or Passpoint. We also tested the open-source hostapd daemon, which is internally used in some professional APs, when running on top Linux kernel 5.18.1 on Arch Linux.

Table 2: Overview of APs tested against Security Context Overriding (SCO) and Fast Reconnect (FR) attacks. Devices behaved the same when configured using WPA2 or WPA3.

Hardware	Software	SCO	FR
LANCOM LN-1700	10.42.0255	●	●
Aruba AP-305/7008	ArubaOS 8.4.0.0	●	●
Cisco Catalyst 9130	IOS XE 17.2.1.11	●	○
Hostapd on Linux	Version 2.10	●	●
Asus RT-AC51U	3.0.0.4.380_8591	●	○
D-Link DIR-853	ET853pnp-1.05-b55 [1]	●	—
D-Link DIR-853	OpenWRT 22.03	●	●
Cisco WAG320N	V1.00.08	●	—
Asus RT-N10	Tomato 1.28	●	—

[1] This software did not support 802.1X or SAE-PK meaning fast reconnect attacks could not be tested. Overriding attacks were simulated using a pre-shared password.

5.3 Fast (Re)Connection Attack

Our default attack is limited by how fast the adversary can connect with the AP to override the victim's security context. However, the 802.11 standard provides several methods to cache a security association and resume it later without having to perform a full handshake. The oldest such method is Pairwise Master Key (PMK) caching. When using PMK caching, the AP will store the negotiated PMK that resulted from the 802.1X or SAE-PK authentication. A PMK Identifier (PMKID) is used to refer to this security association.

Attack Description We assume the network consists of at least two different Basic Service Set Identifiers (BSSIDs). In other words, we assume that the network is broadcast by at least two APs, or by a single AP on two or more channels. Figure 6 illustrates the attack:

1. The adversary first spoofs the MAC address of the victim and uses their own credentials to connect with AP1 of the network. AP1 will cache the PMK that has been negotiated during the (often slow) 802.1X authentication.

2. The victim connects with AP2, which caches the PMK that the client negotiates during its 802.1X handshake. Most APs only store one cached PMK for every client MAC address. It is therefore important that the victim does not connect to the same AP as the adversary, as

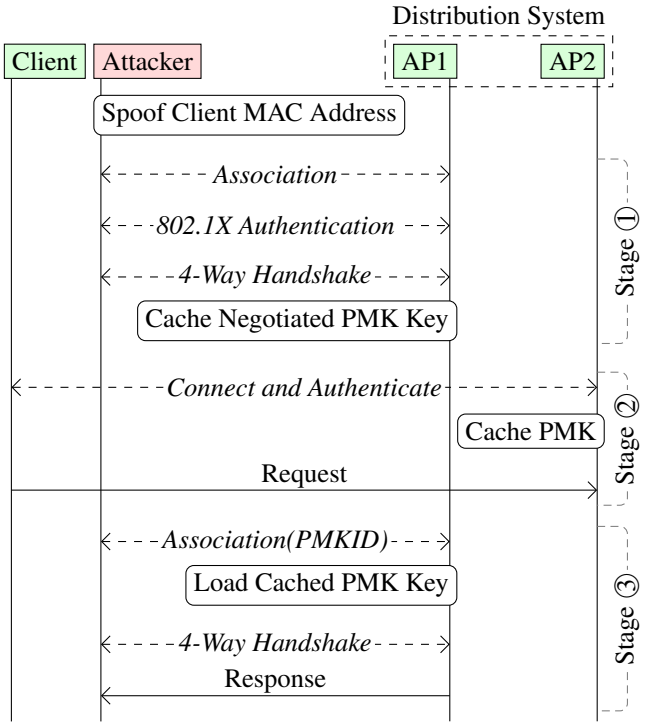

Figure 6: Fast security context overriding attack where the attacker first creates a cached PMK with AP1. After the victim connects to AP2 and sends a request, the attacker reconnects with AP1 using the cached PMK. The two APs can also be different BSSIDs, on different channels, hosted by a single AP.

otherwise the cached PMK of the adversary will be overwritten. After connecting, the victim will send a request to a server of which we want to intercept the response.

3. To intercept the response of the server, the adversary uses the cached PMK to reconnect to AP1. This is accomplished by including the PMKID in the association request. Because the cached PMK is (still) available on AP1, 802.1X authentication can be skipped, and the 4-way handshake can immediately be performed to negotiate session keys. Once connected, the response will be sent by AP1 to the adversary.

Notice that the victim transmitted the request through AP2, but the response was sent by AP1 to the adversary. This is possible because both APs are connected to the same distribution system, i.e., to the same back-end network. In general, the AP with which the client last authenticated will be the AP that transmits the response to the client.

In this variant of the attack, it is not required to disconnect the victim from the network. The distribution system will forward all traffic to the latest AP with which the client's MAC address was last authenticated. As a result, the attack works even when the victim remains connected to another AP, meaning the usage of MFP has no impact on the attack.

Experiments We tested the attack against three professional APs and hostapd (see Table 2). The open-source hostapd AP was configured to broadcast two BSSIDs using two wireless network cards. The LANCOM and Cisco Catalyst consist of one physical device that can broadcast multiple BSSIDs on different channels. The Aruba setup consists of an Aruba 7008 Mobility Controller that was managing two physically separate Aruba AP-305 access points. The attack failed against the Cisco device because the cached PMK was shared between both BSSIDs, meaning the victim overwrites the cached PMK of the adversary. The attack was successful against the LANCOM, Aruba, and Hostapd setup.

We also tested home routers against our attack. The older WAG320N and RT-N10 are not affected because they can broadcast only one BSSID on a single channel. The default firmware of the DIR-853 did not support 802.1X authentication and did not support SAE-PK. As a result, we could only test the RT-AC5U, and the DIR-853 when flashed with OpenWRT. On OpenWRT, we enabled RSN preauthentication to simulate a hotspot that was optimized to efficiently handle roaming clients. We observed that the RT-AC5U did not support PMK caching, causing the fast reconnect attack to fail. However, the attack was successful against OpenWRT.

We evaluated the connection time by performing 100 fast reconnections in the 5 GHz band with our LANCOM and Aruba APs. With LANCOM the average reconnect time was 19.65 ms with a standard deviation of 3.72 and a minimum of 12.36 ms. This can be optimized by speeding up the creation of message 2 of the 4-way handshake, as this took on average 5.73 ms. With Aruba, the average connect time was 24.51 ms with a standard deviation of 5.27 and a minimum of 12.76 ms. As reference, the average latency for transatlantic connections is 70.78 ms, and for connections within Europe this is 13.30 ms [49]. Since an adversary can usually repeat the attack until it is fast enough, this indicates most responses sent over the Internet can be intercepted on time.

5.4 Defenses

To mitigate attacks against hotspot-like networks, an AP can temporarily prevent clients from connecting if they are using a MAC address that was recently connected to the AP. This prevents an adversary from spoofing a MAC address and intercepting pending or queued frames towards a victim. When it can be guaranteed that the user behind a MAC address has not changed, the client can be allowed to immediately reconnect. Note that this check must be done over all APs that are part of the same distribution system.

To securely recognize recently-connected users, an AP can store a mapping between a client's MAC address and their cached security associations (e.g., their cached PMK). A client is allowed to immediately (re)connect under a recently-used MAC address by proving that they posses their cached security association, e.g., by connecting use the correct cached PMK.

Another method to securely recognize recently-connected users is based on the EAP identity they used during 802.1X authentication. An AP can securely learn the EAP identity from the RADIUS server that authenticated the client [37], and can keep a mapping of recently connected MAC addresses and their corresponding EAP identity. When a client connects, the AP checks whether its MAC address was recently used. If it is not, or if it is and the client is using the same EAP identity as before, the client can connect as normal. However, if the same MAC address is used under a different EAP identity, the client is forced to wait a predefined amount of time before being able to successfully connect.

The above defenses assume that, after a certain delay, no more pending packets will arrive for the victim. To prevent leaks beyond this delay, clients can use end-to-end encryption, such as TLS, with the services they communicate with.

6 Discussion on Impact and Lessons Learned

In this section, we discuss the practical impact of our attacks, lessons learned and summarize the fundamental changes needed in the standard to improve security.

6.1 Practical Impact

To demonstrate the practical impact of our attacks, we discuss possible consequences of intercepting frames towards a client.

6.1.1 Hijacking TCP Connections

In the security context overriding attack (Section 5), the adversary will receive frames that are part of TCP connections that the victim previously initiated. The adversary can acknowledge these incoming TCP packets in order to receive all pending data that the server is trying to send to the client. Moreover, these leaked TCP packets reveal the sequence numbers being used, which can be exploited to hijack the TCP connection and inject data to either the client or server. An adversary can use their own Internet-connected server to inject data into this TCP connection by injecting off-path TCP packets with a spoofed sender IP address [13, 27]. This can, for instance, be abused to send malicious JavaScript code to the victim in plaintext HTTP connections with as goal to exploit vulnerabilities in the client's browser.

To spoof TCP responses toward the client after learning the TCP sequence numbers, the client must reconnect to the AP and must do so before it closes the TCP connection. To assure this, the adversary can send a deauthentication frame to the client with a disconnection reason "class 2 frame from non-authenticated station" after performing the attack. Experiments indicated that this causes Linux, Windows, and iOS to reconnect to the AP in less than half a second. Android reconnected in less than 3.5 seconds, which is fast enough to subsequently inject data into active TCP connections.

6.1.2 Intercepting Client Traffic

An adversary can use the identified vulnerabilities to learn the port and transaction identifier that a client is using in DNS requests, which in turn can be abused to spoof DNS responses and intercept most traffic sent by the victim. To perform this attack, the adversary uses our discovered vulnerabilities to intercept a real DNS response towards the victim. If the victim is using encrypted DNS, the adversary can try to block these connections to induce the victim into falling back to plaintext DNS over UDP. Packets can be blocked using methods from Section 4 or by using a multi-channel MitM position [43, 45].

Once the adversary intercepted the DNS response, the random client port and transaction identifier can be extracted. With this information, the adversary can send spoofed DNS responses from their own server on the Internet toward the victim. Note that packet-in-packet routing protocols can be exploited to easily spoof the source IP address of this DNS response [27]. Against Linux, Windows, and iOS, the random port and transaction identifier remain valid for at least 6 seconds, which is long enough for the victim to reconnect to the AP and for the attacker to spoof the DNS response. When testing Android using a Pixel 4 XL the random port and transaction identifier remain valid for 5 seconds. All combined, this provides sufficient time for the adversary to induce the victim to reconnect and inject the spoofed DNS response.

6.1.3 Intercepting Web Traffic

Using our attacks we can intercept any packet that is sent towards a client. This enables various attacks, with a notable use-case of intercepting web traffic. For instance, if the victim visits plaintext HTTP websites, the attacker can steal the victim's cookies. Although the adoption of HTTPS keeps growing [18], roughly 10% of website visits still occur over plaintext connections.[2] An attacker can also intercept traffic to local intranet websites, which may more often use plaintext HTTP. When the victim uses higher-layer encryption protocols such as TLS, the adversary still learns the IP addresses that a victim connects to. We conjecture this can be combined with new traffic analysis techniques, e.g., variations of [29], to learn the website and webpage that a victim is visiting.

6.2 Lessons Learned

We believe the root cause of the identified issues is that the standard is not sufficiently explicit in handling security context changes. In Table 3, we provide an overview of our attacks and their properties: whether we leak or block frames, whether it requires abusing the sleep-bit, and whether the attacker is an insider or outsider. Most notably, an adversary can leak a victim's frames by manipulating the security context of

[2] See https://transparencyreport.google.com/https/overview for weekly statistics on HTTPS adaption as measured by Google.

Table 3: Properties of our attacks. The columns represent whether the targeted frame is already enqueued or still incoming, the goal of the attack, whether the attack relies on spoofing the sleep-bit, the type of attacker, and the last column contains the sections describing the corresponding attack(s).

Frame Location	Goal	Sleep	Attacker	Section(s)
Enqueued	Leak	●	Outsider	3.3, 3.4
Enqueued	Leak	●	Mixed	3.5.1
Enqueued	Leak	○	Outsider	3.5.2
Enqueued	Block	●	Outsider	4.1, 4.2, 4.3
Incoming	Leak	○	Insider	5.2, 5.3

(en)queued frames. While upper-layer security mechanisms such as TLS and HTTPS can limit the risks caused by leaking Wi-Fi frames, our attacks would, for example, leak the IP addresses a client is communicating with, which can reveal sensitive or personal information. For instance, an IP address can often reveal the website that a victim is visiting [32]. Furthermore, we consider it important to include queues and security context updates in formal models of WPA2. For instance, a recent security proof of WPA2 models transmission queues where the transmission key may get updated before a frame is dequeued and transmitted [14]. However, they did not model the *removal* of the transmission key during a disconnection, meaning their model would not discover our attacks.

7 Related Work

A variety of implementation and design flaws were identified in the 802.11 standard. The works that come closest to ours are the FragAttacks [44] and the Kr00k vulnerabilities [34]. FragAttacks demonstrated three fundamental design flaws involving frame fragmentation and aggregation features, as well as various implementation flaws. Specifically, they showed how receivers i) can be tricked into processing the encrypted transported data (lack of authentication of the *is aggregated* flag), ii) reassemble fragments that were decrypted using different keys, and iii) are not required to delete fragments from the receive queue once a client disconnects leading to the possibility to inject forged packets. All these vulnerabilities are caused by flaws in how received frames are processed. In contrast, this paper focuses on vulnerabilities related to frame transmission. The Kr00k vulnerabilities allowed an attacker to decrypt data frames by forcing a devices hardware chip to encrypt frames with an all-zero encryption key. The vulnerabilities were based on the key reinstallation attacks [46, 47] that demonstrated the ability of an attacker to trick a victim into reinstalling an already-in-use key by exploiting the 4-way handshake messages. In this paper, we leverage the entire network stack and use power-save mechanisms to force multiple data frames into both the software and hardware transmit queue. These frames are then leaked in plaintext or under the group or an all-zero encryption key. Researchers have also demonstrated downgrade and denial-of-service attacks in the Dragonfly handshake of WPA3 [15, 48]. These flaws allowed an attacker to recover the password of the wireless network even if configured with WPA3. Furthermore, researchers analyzed the security of enterprise networks and identified several weaknesses [9, 36], for example, widespread security issues due to evil twin attacks [6] and poor configurations [5, 21].

Denial-of-service attacks against Wi-Fi networks are ubiquitous [8, 19, 26], have existed since the first version of the 4-way handshake [20, 30], and even affect WPA3 [12, 28]. Furthermore, denial-of-service attacks may target channel switching mechanisms [25] or jam the physical-layer [7, 11, 33]. Even though Wi-Fi Management Frame Protection (MFP) protects against forged deauthentications, numerous attacks remained present in practice [40]. In this paper, we presented novel queue-based disconnection attacks targeting Security Association (SA) Query requests and 4-way handshake messages. Unlike [1, 17], our attack does not require an attacker to jam the wireless channel to cause a SA Query procedure timeout. In addition to denial-of-service attacks disrupting the connection, our attacks can cause disruptions in time-sensitive operations such as Wi-Fi Fine Timing Measurement (FTM). Our queue-based disruption attacks, targeting measurement requests, extend the FTM vulnerabilities identified in [38, 41].

8 Conclusion

In this work we provided a rigorous analysis of how the security context is managed when Wi-Fi devices buffer frames across various layers of their network stack before transmission. Primarily, we discovered that modern operating systems and devices fail to manage the security context of their transmit queues in a secure manner thereby allowing an adversary to intercept frames. Furthermore, we showed how an adversary can abuse transmit queues to execute denial-of-service attacks such as forcing a client to disconnect even when they use management frame protection. We then explored the feasibility for an attacker to manipulate and control the security context of yet to be queued frames resulting in an adversary forcing vulnerable access points to encrypt frames using adversary-controlled secrets leading to a complete by-pass of higher-layer encryption. Altogether our work highlights the need for the standard to consider queuing mechanisms under a changing security context. Finally, we published our proof-of-concept code to the broader research community after completion of the responsible disclosure process.

Acknowledgments

This research is partially funded by the Research Fund KU Leuven and the Flemish Research Programme Cybersecurity.

References

[1] Md Sohail Ahmad and Shashank Tadakamadla. Short paper: security evaluation of IEEE 802.11 w specification. In *Proceedings of the fourth ACM conference on Wireless network security*, pages 53–58, 2011.

[2] Wi-Fi Alliance. *Passpoint Specification Ver. 3.2*, 2020.

[3] Wi-Fi Alliance. Security | wi-fi alliance. https://www.wi-fi.org/discover-wi-fi/security, 2022 (Accessed 07 June 2022).

[4] Wi-Fi Alliance. Wi-Fi aware | Wi-Fi alliance. https://www.wi-fi.org/discover-wi-fi/wi-fi-aware, 2022 (Accessed 07 June 2022).

[5] Alberto Bartoli, Eric Medvet, Andrea De Lorenzo, and Fabiano Tarlao. (in)secure configuration practices of WPA2 enterprise supplicants. In *WiSec*, 2018.

[6] Alberto Bartoli, Eric Medvet, and Filippo Onesti. Evil twins and WPA2 enterprise: A coming security disaster? *Computers & Security*, 74:1–11, 2018.

[7] John Bellardo and Stefan Savage. 802.11 Denial-of-Service attacks: Real vulnerabilities and practical solutions. In *12th USENIX Security Symposium (USENIX Security 03)*, 2003.

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

[9] Sebastian Brenza, Andre Pawlowski, and Christina Pöpper. A practical investigation of identity theft vulnerabilities in eduroam. In *WiSec*, 2015.

[10] Markus Bullmann, Toni Fetzer, Frank Ebner, Markus Ebner, Frank Deinzer, and Marcin Grzegorzek. Comparison of 2.4 GHz WiFi FTM-and RSSI-based indoor positioning methods in realistic scenarios. *Sensors*, 20(16):4515, 2020.

[11] Aldo Cassola, William K Robertson, Engin Kirda, and Guevara Noubir. A practical, targeted, and stealthy attack against WPA enterprise authentication. In *NDSS*, 2013.

[12] Efstratios Chatzoglou, Georgios Kambourakis, and Constantinos Kolias. How is your wi-fi connection today? dos attacks on wpa3-sae. *Journal of Information Security and Applications*, 64:103058, 2022.

[13] Weiteng Chen and Zhiyun Qian. Off-Path TCP exploit: How wireless routers can jeopardize your secrets. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1581–1598, 2018.

[14] Cas Cremers, Benjamin Kiesl, and Niklas Medinger. A formal analysis of IEEE 802.11's WPA2: Countering the kracks caused by cracking the counters. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1–17. USENIX Association, August 2020.

[15] Daniel de Almeida Braga, Pierre-Alain Fouque, and Mohamed Sabt. Dragonblood is still leaking: Practical cache-based side-channel in the wild. In *Annual Computer Security Applications Conference*, pages 291–303, 2020.

[16] Philipp Ebbecke. Protected management frames enhance Wi-Fi network security. https://www.wi-fi.org/beacon/philipp-ebbecke/protected-management-frames-enhance-wi-fi-network-security, 2020 (Accessed 07 June 2022).

[17] Martin Eian. Fragility of the robust security network: 802.11 denial of service. In *International Conference on Applied Cryptography and Network Security*, pages 400–416. Springer, 2009.

[18] Adrienne Porter Felt, Richard Barnes, April King, Chris Palmer, Chris Bentzel, and Parisa Tabriz. Measuring HTTPS adoption on the web. In *26th USENIX security symposium (USENIX security 17)*, pages 1323–1338, 2017.

[19] Stephen Glass and Vallipuram Muthukkumarasamy. A study of the TKIP cryptographic DoS attack. In *2007 15th IEEE International Conference on Networks*, pages 59–65. IEEE, 2007.

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

[21] Man Hong Hue, Joyanta Debnath, Kin Man Leung, Li Li, Mohsen Minaei, M Hammad Mazhar, Kailiang Xian, Endadul Hoque, Omar Chowdhury, and Sze Yiu Chau. All your credentials are belong to us: On insecure wpa2-enterprise configurations. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1100–1117, 2021.

[22] Mohamed Ibrahim, Hansi Liu, Minitha Jawahar, Viet Nguyen, Marco Gruteser, Richard Howard, Bo Yu, and Fan Bai. Verification: Accuracy evaluation of wifi fine time measurements on an open platform. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 417–427, 2018.

[23] IEEE Std 802.11. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, 2020.

[24] Evgeny Khorov, Andrey Lyakhov, Alexander Krotov, and Andrey Guschin. A survey on IEEE 802.11 ah: An enabling networking technology for smart cities. *Computer communications*, 58:53–69, 2015.

[25] 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. IEEE, 2009.

[26] Sokjoon Lee and Byung Ho Chung. Denial of service attack against IEEE 802.11 WLAN fast initial link setup technology. *International Journal of Computer Science and Electronics Engineering (IJCSEE)*, 3(4):335–337, 2015.

[27] Yannay Livneh. Spoofing IP with IPIP. In *PoC || GTFO*, volume 0x02, 2022.

[28] Karim Lounis and Mohammad Zulkernine. Bad-token: denial of service attacks on WPA3. In *Proceedings of the 12th International Conference on Security of Information and Networks*, pages 1–8, 2019.

[29] Brad Miller, Ling Huang, Anthony D Joseph, and J Doug Tygar. I know why you went to the clinic: Risks and realization of HTTPS traffic analysis. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 143–163. Springer, 2014.

[30] CHJC Mitchell and Changhua He. Security analysis and improvements for ieee 802.11 i. In *The 12th Annual Network and Distributed System Security Symposium (NDSS'05) Stanford University, Stanford*, pages 90–110. Citeseer, 2005.

[31] Alberto Ornaghi and Marco Valleri. Man in the middle attacks. In *Blackhat Conference Europe*, volume 1045, 2003.

[32] Simran Patil and Nikita Borisov. What can you learn from an ip? In *Proceedings of the Applied Networking Research Workshop*, pages 45–51, 2019.

[33] Konstantinos Pelechrinis, Marios Iliofotou, and Srikanth V Krishnamurthy. Denial of service attacks in wireless networks: The case of jammers. *IEEE Communications surveys & tutorials*, 13(2):245–257, 2010.

[34] ESET Experimental Research and Detection Team. Kr00k: A serious vulnerability deep inside wi-fi encryption. https://www.eset.com/int/kr00k/, 2020 (Accessed 07 June 2022).

[35] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. Automated website fingerprinting through deep learning. *arXiv preprint arXiv:1708.06376*, 2017.

[36] Pieter Robyns, Bram Bonné, Peter Quax, and Wim Lamotte. Short paper: exploiting WPA2-enterprise vendor implementation weaknesses through challenge response oracles. In *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*, pages 189–194, 2014.

[37] Allan Rubens, Carl Rigney, Steve Willens, and William A. Simpson. Remote Authentication Dial In User Service (RADIUS). RFC 2865, June 2000.

[38] Domien Schepers and Aanjhan Ranganathan. Privacy-preserving positioning in Wi-Fi fine timing measurement. *Proceedings on Privacy Enhancing Technologies*, 2022(2):325–343, 2022.

[39] Domien Schepers, Aanjhan Ranganathan, and Mathy Vanhoef. Let numbers tell the tale: measuring security trends in Wi-Fi networks and best practices. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 100–105, 2021.

[40] Domien Schepers, Aanjhan Ranganathan, and Mathy Vanhoef. On the robustness of Wi-Fi deauthentication countermeasures. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '22, page 245–256, New York, NY, USA, 2022. Association for Computing Machinery.

[41] Domien Schepers, Mridula Singh, and Aanjhan Ranganathan. Here, there, and everywhere: Security analysis of Wi-Fi fine timing measurement. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '21, page 78–89, New York, NY, USA, 2021. Association for Computing Machinery.

[42] Weiping Sun, Munhwan Choi, and Sunghyun Choi. IEEE 802.11 ah: A long range 802.11 WLAN at sub 1 GHz. *Journal of ICT standardization*, 1(1):83–108, 2013.

[43] Manesh Thankappan, Helena Rifà-Pous, and Carles Garrigues. Multi-channel man-in-the-middle attacks against protected Wi-Fi networks: A state of the art review. *arXiv preprint arXiv:2203.00579*, 2022.

[44] Mathy Vanhoef. Fragment and forge: Breaking Wi-Fi through frame aggregation and fragmentation. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 161–178, 2021.

[45] Mathy Vanhoef and Frank Piessens. Advanced Wi-Fi attacks using commodity hardware. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 256–265, 2014.

[46] Mathy Vanhoef and Frank Piessens. Key reinstallation attacks: Forcing nonce reuse in WPA2. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1313–1328, 2017.

[47] Mathy Vanhoef and Frank Piessens. Release the kraken: new KRACKs in the 802.11 standard. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 299–314, 2018.

[48] 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. IEEE, 2020.

[49] Verizon. IP latency statistics. https://web.archive.org/web/20220514091144/https://www.verizon.com/business/terms/latency/, 2022 (Accessed 07 June 2022).

[50] Wi-Fi Alliance. WPA3 specification version 3.0. https://www.wi-fi.org/file/wpa3-specification, 2020 (Accessed 07 June 2022).