Denial-of-Service Attacks Against the 4-way Wi-Fi Handshake

Mathy Vanhoef and Frank Piessens

imec-DistriNet, KU Leuven
firstname.lastname@cs.kuleuven.be

Abstract

Having a secure implementation of the 4-way Wi-Fi handshake is necessary to guarantee that all subsequent Wi-Fi traffic is properly protected. After all, the 4-way handshake negotiates a fresh pairwise key, authenticates both the client and Access Point (AP), and is used by practically all protected Wi-Fi networks. As a result, it has been widely studied, and even formally proven secure. Yet we found that, in practice, many implementations are vulnerable to denial-of-service attacks.

Our first two attacks highlight synchronization issues between when the client and AP install the pairwise key. More concretely, in the first attack the adversary blocks the initial message 4 of the 4-way handshake. This causes the client, but not yet the AP, to install the pairwise key. Therefore the handshake will fail, since the client now expects and sends encrypted handshake messages, while the AP only uses plaintext ones. Our second attack makes the client install the pairwise key before it transmits message 4. Hence, the client will again install the pairwise key, while the AP will not. In the third attack, an adversary waits until the victim completes the 4-way handshake. Then, she initiates a rekey by injecting a malformed message 1 to the client. This causes certain clients to disconnect from the network. Finally, we propose countermeasures against our new attacks.

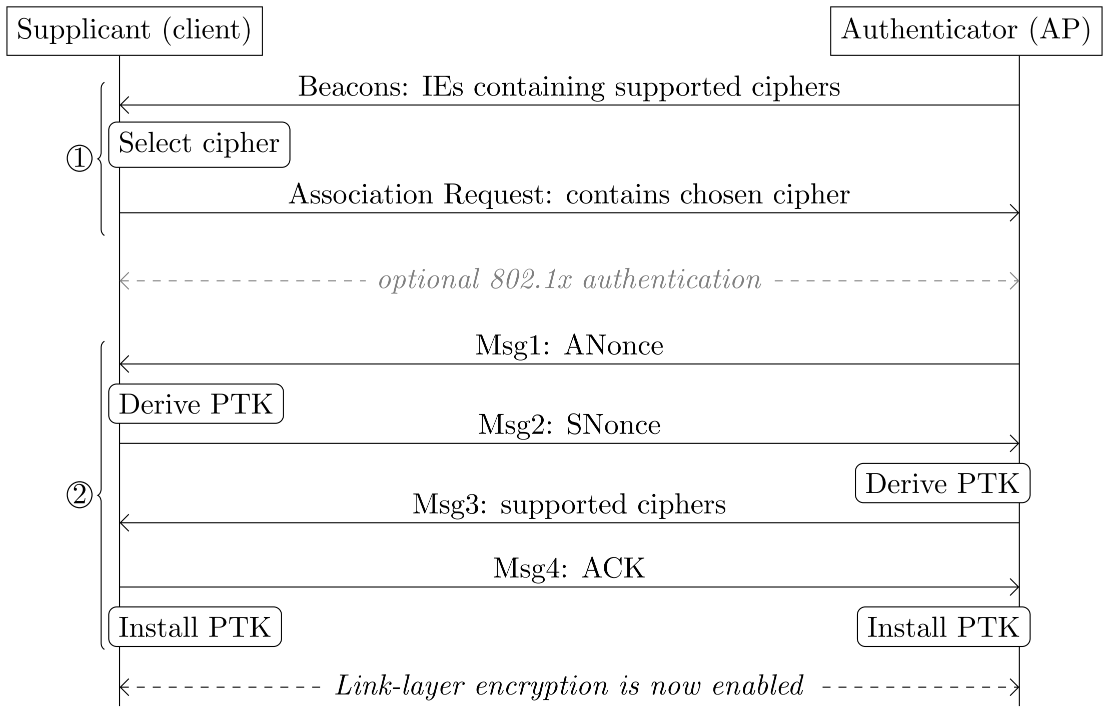
Keywords

Network Protocols, Wi-Fi, 802.11, Denial-of-Service, 4-way handshake

1. Introduction

Nowadays many wireless networks are based on the 802.11 standard, which is more widely known as Wi-Fi. Indeed, this standard is used in both commercial off-the-shelf products, and in reliability-critical industrial systems. However, due to its broadcast nature, adversaries can easily monitor (and interfere with) wireless transmissions. Therefore, it is essential to encrypt and protect transmitted data. Initially, the 802.11 standard offered Wired Equivalent Privacy (WEP) for this purpose. Unfortunately, it has major design flaws, and is considered completely broken [1, 2, 3]. Instead, most modern networks rely on some version of Wi-Fi Protected Access (WPA) to encrypt and protect data [4].

All versions of WPA use a 4-way handshake for mutual authentication, and for the negotiation of a fresh pairwise key. Even protected Wi-Fi networks that use 802.1x authentication, i.e., those that require a username and password, use the 4-way handshake to generate a fresh pairwise key. As a result, it is essential that the 4-way handshake is reliable and does not contain any security flaws. This motivated several researchers to investigate and formally analyze the security of the 4-way handshake [5, 6, 7]. However, even though these works discovered and addressed certain denial-of-service (DoS) attacks, in practice implementations of the 4-way handshake still contain certain deficiencies. In particular, while studying the 4-way handshake, we discovered that several implementations are vulnerable to denial-of-service (DoS) attacks.



Supplicant (client)
Authenticator (AP)
Beacons: IEs containing supported ciphers
①
Select cipher
Association Request: contains chosen cipher
optional 802.1x authentication
Msg1: ANonce
Derive PTK
Msg2: SNonce
②
Derive PTK
Msg3: supported ciphers
Msg4: ACK
Install PTK
Install PTK
Link-layer encryption is now enabled

Figure 1: Simplified overview of the messages exchanged when connecting to a network, including the actions of the client (supplicant) and AP (authenticator). Note that the 4-way handshake corresponds to stage 2 in this figure.

The remainder of this paper is structured as follows. Section 2 introduces the 802.11 standard, the 4-way handshake, and Management Frame Protection (MFP). Our denial-of-service attacks are presented in Section 3, and in Section 4 we propose countermeasures to these attacks. Finally, we present related work in Section 5 and conclude in Section 6.

2. Background

In this section we explain how clients discover nearby networks, introduce the 4-way handshake, and briefly touch upon link-layer confidentiality protocols. Finally, we explain how the Management Frame Protection (MFP) amendment of 802.11 works

2.1. Network Discovery

An Access Point (AP) periodically broadcasts beacons to advertise its presence. This is illustrated in stage 1 of Figure 1. These beacons include the supported link-layer encryption algorithms that are supported by the AP, i.e., the supported cipher suites. This can be either the Temporal Key Integrity Protocol (TKIP), the CTR CBC-MAC Protocol (CCMP), or both. When a client wants to connect to a discovered AP, it first has to select a supported encryption algorithm to use. Once a choice is made, it sends an association request to the AP. This request informs the AP that the client wishes to connect, and contains the selected encryption algorithm.

2.2. The 4-way Handshake

In order to execute the 4-way handshake, the client and Access Point (AP) must possess a shared secret called the Pairwise Master Key (PMK). The PMK is commonly derived from a pre-shared key, or from an 802.1x handshake. The 4-way handshake verifies that both entities possess the same PMK, generates a fresh Pairwise Transient Key (PTK), and confirms the selection of the cipher suite. It also synchronizes the installation of the PTKs. Once a pairwise key (PTK) has been installed, all traffic is encrypted at the link-layer.

The messages exchanged during the 4-way handshake are illustrated in stage 2 of Figure 1. Simplified, the first two messages contain random nonces to generate a fresh pairwise key, and the last two messages protect against downgrade attacks. The handshake messages are defined using EAPOL frames, and we use the notation message n to refer to a specific message of the 4-way handshake. After the client has transmitted message 4, it installs the PTK, while the AP installs the PTK after receiving message 4. Finally, to handle missed frames, the AP will retransmit a handshake message if it did not receive a response. If the client already received the retransmitted message, it will transmit a new response. For example, if the AP does not receive message 4, it will retransmit message 3.

Finally, in an existing connection it is possible to rekey the PTK by executing a new 4-way handshake. During this rekey handshake, the EAPOL frames undergo link-layer encryption using the currently installed PTK.

2.3. Management Frame Protection (MFP)

A client can disconnect from a network by sending a deauthentication or disassociation frame to the AP. By default, these messages are not authenticated. As a result, an adversary can forge them to forcibly disconnect a client from a network. Continuously injecting these forged deauthentication packets results in a denial-of-service attack [8]. Fortunately, this attack can easily be prevented by enabling the Management Frame Protection (MFP) feature defined in the 802.11w amendment. When it is enabled, deauthentication, disassociation, and action frames are authenticated (but not encrypted). Therefore, an adversary can no longer forge these frames in order to disconnect a client.

3. Denial-of-Service (DoS) Attacks

In this section we present three novel denial-of-service (DoS) attacks against implementations of the 4-way handshake. In the first one we make the handshake fail by blocking message 4. The second attack exploits a race condition between installing the pairwise key and sending message 4. Finally, we show how injecting a malformed message 1 causes certain clients to disconnect from the network.

3.1. Blocked Message 4

If message 4 of the handshake does not reach the AP, the handshake will never successfully complete. This is because the client installs the PTK after transmitting message 4, meaning it now only accepts messages that are encrypted at the link-layer. However, the AP will retransmit message 3 without encryption when it did not receive message 4. The client rejects this unencrypted message 3, and hence will not retransmit message 4. Eventually, the AP reaches its retransmission limit, and will abort the handshake. An attacker can abuse this as an efficient denial-of-service attack by selectively jamming message 4. Note that selectively jamming frames is possible using cheap Wi-Fi USB dongles [9].

Figure 2 illustrates this attack. During stage 1 of the attack, the adversary blocks message 4 from arriving at the AP using a selective jammer. Immediately after the victim transmitted message 4, she will install the pairwise key (PTK). From this point onward, the victim only accepts frames that are encrypted at the link-layer. When the AP now

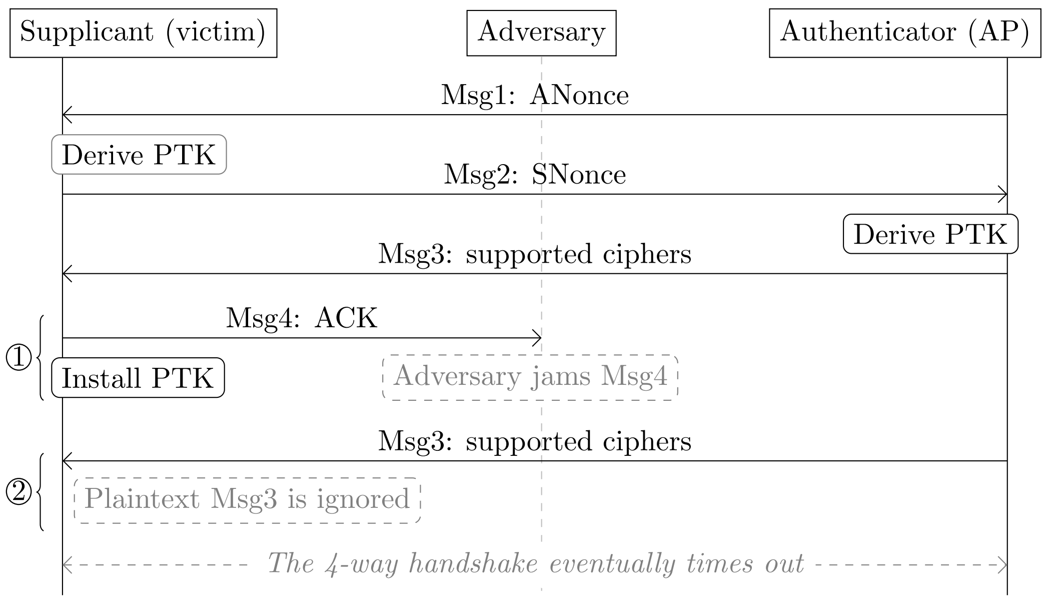


Figure 2: Blocked message 4 attack. The first transmission of message 4 is blocked, after which the victim installs the PTK, meaning retransmissions of message 3 will be ignored.

retransmits message 3 without link-layer encryption, the victim will reject it. This is illustrated in stage 2 of Figure 2. Finally, the AP aborts the handshake after several retransmissions of message 3, since it will never receive message 4 in response.

We do remark that some implementations of the 4-way handshake always accept plaintext handshake messages, even when a PTK has been installed. That is, by testing several implementations, we discovered that some accept plaintext EAPOL frames (i.e., handshake frames) at any moment (see column 2 in Table 1). Allegedly this is done because some implementations always transmit group key EAPOL frames without link-layer encryption, even though they already installed the PTK [10]. Nevertheless, even if plaintext handshake message are always accepted, our attack still results in a denial-of-service. This is because nearly all implementations will transmit message 4 using link-layer encryption once a PTK has been installed, and the AP will reject these frames since it did not yet install the PTK. Interestingly, this behaviour contradicts the 802.11 standard. That is, the standard mandates that message 4 should always be sent without link-layer encryption in the initial 4-way handshake [11, §11.6.6.5]. However, few implementations follow this requirement: we found that only MediaTek retransmits message 4 without link-layer encryption (see column 3 in Table 1). Hence, all other implementations are vulnerable to this attack.

3.2. Encrypted Message 4 Race Condition

In the 802.11 standard, installing the PTK and sending an EAPOL message are both done by calling primitives in the MAC Sublayer Management Entity (MLME). However, in an actual implementation, these MLME primitives do not have to correspond similar interfaces [11, §6.3.1]. In practice, we indeed see that several operating systems use different kernel interfaces for installing the PTK and sending EAPOL messages. For instance, on Linux, the `nl80211` kernel interface can be used to install the PTK, while handshake messages are transmitted by the `sendto` system call. This means there is no guarantee in which order these two actions will be processed by the kernel, and hence the PTK may be installed before message 4 is transmitted [12]. Additionally, message 4 may still be queued for transmission due to a busy medium while the PTK is already being installed. As a

Table 1: Behaviour of several 4-way handshake implementations. The second column shows whether EAPOL frames that did not undergo link-layer encryption are accepted even if a PTK is installed. The third column shows whether message 4 is retransmitted without link-layer encryption during an initial 4-way handshake.

Implementation	Plaintext Reception	Plaintext (Re)transmission
FreeBSD	Yes	No
NetBSD	Yes	No
OpenBSD	No	No
Linux	Yes	No
Android	Yes	No
Windows	No	No
Apple	No	No
MediaTek	Yes	Yes
Broadcom	No	No

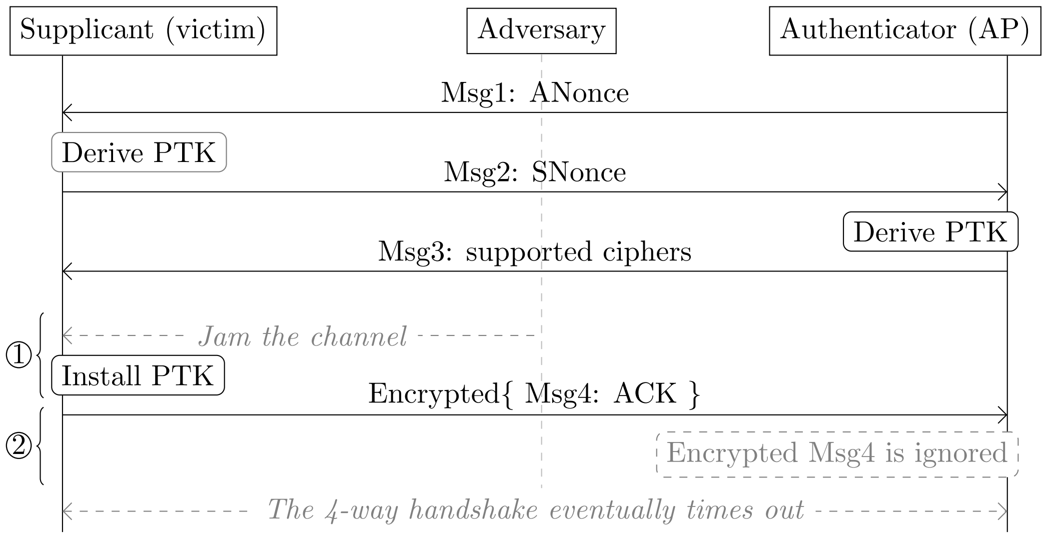


Figure 3: Encrypted message 4 race condition. Here the victim is induced into installing the pairwise key (PTK) before sending message 4.

result, message 4 may undergo link-layer encryption, and will therefore be rejected by the AP. This will eventually cause the handshake to time out.

Figure 3 illustrates this attack more clearly. At stage 1 of the attack, the adversary briefly jams the wireless channel. As a result, the victim queues message 4 for transmission until the wireless channel is no longer busy, i.e., until the adversary stops jamming. While message 4 is queued, the victim already installs the pairwise key. Consequently, when the adversary now stops jamming, the victim will send message 4 using link-layer encryption since the PTK has been installed. However, the AP has not yet installed the pairwise key, and will therefore reject frames that underwent link-layer encryption. This illustrated in stage 2 of Figure 3. Since the AP never receives message 3, the 4-way handshake will eventually time out. We implemented this attack against OpenBSD's `rum` driver, because its transmission path is straightforward to understand and debug. This confirmed that our jamming attack causes the PTK to be installed before message 4 is transmitted. We conjecture that other operating systems and drivers can be attacked in a similar manner.

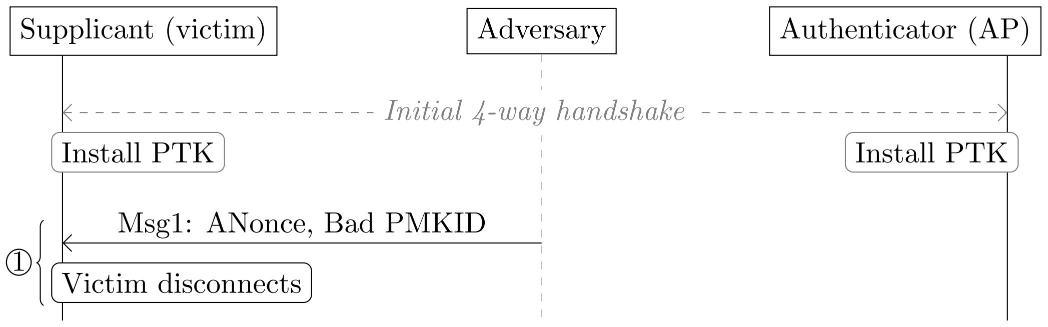


Figure 4: Instigating a failed rekey using a malformed message 1. This messages contains an invalid PMK ID, and in response the victim will disconnect from the AP.

3.3. Failed Rekey Using a Malformed Message 1

In the previous attacks we observed that several implementations accepted plaintext handshake messages, even though they already installed the PTK (recall column 1 in Table 1). We can abuse this by injecting a plaintext message 1 towards to the client, after an initial 4-way handshake was already completed. Note that sending a new message 1 is valid behavior, since the AP can decide at any moment to refresh the pairwise key (PTK) by starting a new 4-way handshake. Additionally, message 1 can indeed be forged, since it is not authenticated. More importantly, if this message contains invalid or malformed data, certain clients will abort the handshake, and subsequently disconnect from the network.

One way to create a malformed message 1, is by including an invalid PMKID alongside the ANonce (see stage 1 of Figure 4). Recall from section 2 that the PMK is the shared secret between the client and AP, and can either be derived from a preshared key, or from an 802.1x authentication. In general though, it is possible for the client and AP to share multiple valid PMKs. Therefore, the first message of the 4-way handshake can optionally include a PMKID that identifies the specific PMK which will be used [11, §11.6.6.2]. Note that the PMKID is essentially just a hash of the secret PMK. Interestingly, we found that many clients will abort the handshake and disconnect from the network when message 1 contains an unknown PMKID. Therefore, an adversary can cause a denial-of-service by injecting a forged message 1 with an unknown PMKID. When done during an initial 4-way handshake, this attack will always succeed. Moreover, if the victim accepts plaintext EAPOL frames even though it already installed a PTK, the attack can also be executed after the initial 4-way handshake has completed. Put differently, in this case our denial-of-service attack still works even though the client has already installed the PTK. This latter variant of the attack is illustrated in stage 1 Figure 4, where the adversary injects a plaintext message 1 after the victim already installed the pairwise key.

In contrast to injecting deauthentication frames to disconnect a client, injecting a malformed message 1 is possible even if Management Frame Protection (MFP) is enabled. Indeed, if MFP is enabled, an adversary cannot forge deauthentication or disassociation frames. However, message 1 of the handshake can still be forged, and hence can still be abused to perform an efficient denial-of-service attack.

Most operating systems that accept plaintext EAPOL frames when a PTK has already been installed, use the open source `wpa_supplicant` utility as their Wi-Fi client. Therefore, we implemented this attack against `wpa_supplicant`. This confirmed that the client will abort the handshake, and subsequently disconnect from the network when it receives a plaintext message 1 containing an invalid PMKID, even when it already installed a PTK.

4. Proposed Countermeasures

In this section we propose countermeasures against our attacks. First we show how to avoid synchronization issues between when the client and AP install the pairwise key. Then we show how to prevent an attacker from forging message 1 to disconnect the client.

4.1. Plaintext EAPOL Frames

The 802.11 standard mandates that, when a PTK is installed, message 4 must still be sent in plaintext during the first 4-way handshake [11, §11.6.6.5]. Unfortunately, this alone does not prevent the first two attacks discussed in Section 3. After all, the client will still drop plaintext retransmissions of message 3 because it already installed the PTK [11, §11.6.6.4]. We also remark that, amendments introduced to avoid synchronization issues during a pairwise rekey [13], do not address our attacks. Instead, we propose to send *all* EAPOL frames in plaintext. Note that this does not negatively impact security, because sensitive information in EAPOL frames is already encrypted, and the full frame is also authenticated. Hence link-layer encryption is not required to protect the messages of the 4-way handshake. Not encrypting EAPOL frames at the link-layer is also consistent with RFC 5247, which states that *"presence or absence of lower layer security is not taken into account in the processing of EAP messages"* [14, §3.4]. Additionally, the formal security proof of the 4-way handshake also does not assume link-layer encryption. In other words, it is safe to send all EAPOL frames without link-layer encryption [5].

Unfortunately, only sending plaintext EAPOL frames introduces some compatibility issues. In particular, some implementations *require* that these frames are encrypted during a rekey. Sending them without encryption would therefore break compatibility. Nevertheless, in a first step, implementations can be modified so they always accept plaintext EAPOL frames. Additionally, they should always send plaintext EAPOL frames during the initial 4-way handshake even if a PTK is already installed. This does not introduce compatibility issues, yet still prevents our first two attacks of Section 3. Once most implementations accept plaintext EAPOL frames at any moment, we can also start sending plaintext EAPOL frames during a rekey.

4.2. Handling a Malformed Message 1

To prevent the malformed message 1 attack of Section 3.3, an implementation should ignore any such malformed messages. Interestingly, OpenBSD already takes this approach. There, if the client receives a malformed handshake message, it is simply dropped. Hence the client remains connected to the network, preventing the attack.

Another modification that can be made is that, during a rekey handshake, the authenticity of message 1 should be validated by the currently installed PTK. Recall that during the initial 4-way handshake, message 1 is sent unauthenticated by the AP, because there is not yet a negotiated pairwise key (recall Figure 1). However, during a rekey we can authenticate message 1 using the PTK that was negotiated in the previous 4-way handshake. This assures an adversary cannot forge any handshake messages once the initial 4-way handshake has completed. To avoid compatibility issues, APs should first be modified such that they transmit authenticated message 1's during a rekey. Once most APs do this, clients can be modified to only accept authenticated messages during a rekey.

5. Related Work

Several researchers have analyzed the security of the 4-way handshake [5, 6, 7, 15, 16, 17]. In particular, He et al. discovered a denial-of-service vulnerability [5, 7], which led to the standardization of a slightly improved design of the 4-way handshake [18]. In their DoS attack, the adversary injects a forged message 1 using a different ANonce than the one of real AP. This causes the client to generate an invalid PTK, making the handshake fail. The solution incorporated into the 802.11 standard is to make the client always use the same SNonce in a specific 4-way handshake, and to verify the ANonce when receiving message 3 of the handshake [5]. Note that message 3 is authenticated, and hence cannot be forged by an attacker. In contrast, our DoS attacks are not caused by making stations generate the wrong PTK, but by synchronization issues between when the client and AP install the PTK.

Recently, Vanhoef and Piessens showed that messages in the 4-way handshake can be manipulated to make the client install the same key multiple times [17]. Each time the key is installed, the incremental transmit packet number (nonce) and receive replay counter are reset. This can be abused to decrypt, replay, and possibly forge arbitrary frames. Similar to our denial-of-service attacks, their attack can be treated as a synchronization issue regarding the installation time of a key.

Researchers also discovered several DoS attack against Wi-Fi networks that abuse different features of the protocol. Arguably the most well-known of these is the deauthentication attack, where an adversary forges deauthentication frames to disconnect the client from the network [19]. This is possible because, by default, these messages are not authenticated. However, nowadays this attack can easily be prevented by enabling protected management frames [11, §4.5.4.9]. In contrast, our attacks remain possible even when protected management frames is enabled.

Several DoS attacks also exploit weaknesses in the TKIP encryption protocol. In particular, Glass and Muthukkumarasamy abuse the TKIP Michael countermeasures to make a Wi-Fi network unusable for one minute [20]. Vanhoef and Piessens improved this attack, by removing the requirement of a man-in-the-middle position [4]. The advantage of our DoS attacks is that they can be executed against any Wi-Fi network, even if the network does not support TKIP.

Finally, Könings et al. found several DoS vulnerabilities in the physical and MAC layer of 802.11 [21]. Some of these make the network unusable for one minute, while others only do so for a brief amount of time. A survey of DoS attacks at the physical and MAC layer is given by Bicakci and Tavli [22]. Generally, these attacks require injecting a large amount of frames, while our attacks only require a low amount of injected packets.

6. Conclusion

We discovered three denial-of-service attacks against implementations of the 4-way handshake. Two attacks can be prevented by always sending and accepting plaintext EAPOL frames. Because this may introduce compatibility issues, we first recommend doing this only during the initial 4-way handshake. This does not introduce compatibility issues, while still preventing our first two attacks. In a second step, implementations can also be modified to send plaintext EAPOL frames during a rekey. Finally, to prevent our third attack, implementations should ignore malformed message 1's during a 4-way handshake.

Acknowledgments

This research is partially funded by the Research Fund KU Leuven and by the imec High Impact Initiative Distributed Trust project.

References

[1] S. R. Fluhrer and D. A. McGrew, "Statistical analysis of the alleged RC4 keystream generator," in *FSE*, 2000.

[2] A. Stubblefield, J. Ioannidis, A. D. Rubin, *et al.*, "Using the fluhrer, mantin, and shamir attack to break wep.," in *NDSS*, 2002.

[3] A. Bittau, M. Handley, and J. Lackey, "The final nail in WEP's coffin," in *IEEE S&P*, 2006.

[4] M. Vanhoef and F. Piessens, "Practical verification of WPA-TKIP vulnerabilities," in *ASIA CCS*, pp. 427–436, ACM, 2013.

[5] C. He and J. C. Mitchell, "Analysis of the 802.1 i 4-Way handshake," in *WiSe*, ACM, 2004.

[6] C. He, M. Sundararajan, A. Datta, A. Derek, and J. C. Mitchell, "A modular correctness proof of IEEE 802.11i and TLS," in *CCS*, 2005.

[7] J. Mitchell and C. He, "Security analysis and improvements for IEEE 802.11i," in *NDSS*, 2005.

[8] S. Park, K. Kim, D. Kim, S. Choi, and S. Hong, "Collaborative QoS architecture between DiffServ and 802.11e wireless LAN," in *Vehicular Technology Conference*, 2003.

[9] M. Vanhoef and F. Piessens, "Advanced Wi-Fi attacks using commodity hardware," in *ACSAC*, 2014.

[10] J. Malinen, "Re: Dealing with retransmitted EAPOL msg 3/4 and 4/4." Retrieved 19 March 2017 from `www.spinics.net/lists/hostap/msg03309.html`, 2017.

[11] IEEE Std 802.11-2012, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Spec*, 2012.

[12] J. Malinen, "wpa keyhandshake question / bug." Retrieved 19 March 2017 from `http://lists.shmoo.com/pipermail/hostap/2005-May/010370.html`, 2005.

[13] R. Stacey, A. Stephens, J. Walker, H. Liondas, and E. Qi, "Rekeying protocol fix text." Retrieved 20 August 2017 from `https://mentor.ieee.org/802.11/dcn/10/11-10-0314-00-000m-rekeying-protocol-fix-text.doc`, 2010.

[14] B. Aboba, D. Simon, and P. Eronen, "Extensible authentication protocol (EAP) key management framework." RFC 5247, 2008.

[15] L. Wang and B. Srinivasan, "Analysis and improvements over DoS attacks against IEEE 802.11i standard," in *NSWCTC*, 2010.

[16] M. Vanhoef and F. Piessens, "Predicting, decrypting, and abusing WPA2/802.11 group keys," in *USENIX Security*, 2016.

[17] M. Vanhoef and F. Piessens, "Key reinstallation attacks: Forcing nonce reuse in WPA2," in *CCS*, 2017.

[18] IEEE Std 802.11-2016, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Spec*, 2016.

[19] J. Bellardo and S. Savage, "802.11 denial-of-service attacks: real vulnerabilities and practical solutions," in *USENIX Security*, 2003.

[20] S. M. Glass and V. Muthukkumarasamy, "A study of the TKIP cryptographic dos attack," in *15th International Conference on Networks*, IEEE, 2007.

[21] B. Könings, F. Schaub, F. Kargl, and S. Dietzel, "Channel switch and quiet attack: New DoS attacks exploiting the 802.11 standard," in *LCN*, 2009.

[22] K. Bicakci and B. Tavli, "Denial-of-service attacks and countermeasures in IEEE 802.11 wireless networks," *Comput. Stand. Interfaces*, vol. 31, no. 5, 2009.