

February 2024

Bypassing Wi-Fi Authentication in Modern WPA2/3 Networks

How vulnerabilities in Wi-Fi software put users at risk, despite the recent release of new security standards like WPA3

Authors:

Mathy Vanhoef
Héloïse Gollier

This report is supported by
a grant from Top10VPN

EXECUTIVE SUMMARY

We performed a security evaluation of two widely-used open-source Wi-Fi implementations, namely `wpa_supplicant` and Intel's iNet Wireless Daemon (IWD). Our focus was on identifying logical vulnerabilities that can be abused to bypass authentication. This revealed two new vulnerabilities in `wpa_supplicant` and IWD. In both cases, the root cause of the vulnerability is that certain messages in the authentication protocol can be skipped, leading to a bypass of the authentication procedure:

CVE-2023-52160 (“Phase-2 bypass”): A vulnerability in `wpa_supplicant` v2.10 and lower, which is used in practically all Android and Linux devices, allows an attacker to trick a victim into connecting to an adversary's malicious clone of a Wi-Fi network. The adversary can subsequently intercept the victim's traffic. The vulnerability can be exploited against Wi-Fi clients that are not properly configured to verify the certificate of the authentication server, which unfortunately still often occurs in practice, in particular with ChromeOS, Linux, and Android devices.

The technical summary is that the vulnerability allows an adversary to skip Phase-2 authentication in Enterprise WPA2/3 networks that use PEAP for authentication. The attack works against any Phase-2 inner authentication method, including the widely-used MS-CHAPv2 method.

CVE-2023-52161 (“4-way bypass”): A vulnerability in IWD v2.12 and lower allows an adversary to gain unauthorized access to a protected Wi-Fi network. The adversary can subsequently use the Wi-Fi network as a normal user. For instance, the adversary can then use the network to access the Internet, connect to internal devices, attack other clients on the network, and so on. The only requirement to perform the attack is that the target Wi-Fi network must be using IWD.

The technical summary is that the vulnerability allows an adversary to skip message 2 and 3 of the 4-way handshake, enabling an adversary to complete the authentication process without knowing the network's password.

Both vulnerabilities were reported to the vendors and have meanwhile been patched. For `wpa_supplicant`, the patch will be part of the next v2.11 release, but at the time of writing it is unclear when this version will be released.

The vulnerability in `wpa_supplicant` can be mitigated by configuring clients to always verify the server's certificate. The vulnerability in IWD only is present when it is operating in AP mode, and to the best of our knowledge, the only defense is to run a patched version.

CONTENTS

Executive Summary	1
1 Introduction	4
2 Technical Background	4
2.1 Home WPA2 Networks	5
2.2 Enterprise WPA2 and WPA3 Networks	6
2.3 Home WPA3 Networks	8
3 Analysis of wpa_supplicant	8
3.1 Usage in Practice	8
3.2 Vulnerability Details	9
3.3 Exploiting the Vulnerability	10
3.3.1 Typical Attack Prerequisites	10
3.3.2 Main Attack Prerequisite: Establishing TLS Tunnel	11
3.3.3 Attack Details	12
3.4 Defenses	13
4 Analysis of IWD	13
4.1 Introduction to IWD	13
4.2 Vulnerability Details	14
4.3 Exploiting the Vulnerability	15
4.4 Attack Limitations	16
4.5 Defense	17
5 Related Work	17
6 Conclusion	18

1 INTRODUCTION

This report covers two new vulnerabilities that were discovered in `wpa_supplicant` and `IWD`. These were assigned the identifiers CVE-2023-52160 and CVE-2023-52161, respectively.

Both vulnerabilities were found while looking for logical implementation flaws. In particular, we looked for flaws where an implementation wrongly accepts an out-of-order packet. We specifically focused on authentication protocols that provide mutual authentication, that is, protocols that both: (1) verify the authenticity of the network; and (2) verify the identity of the client and whether this client is authorized to access the network.

The vulnerability in `wpa_supplicant` is present in its PEAP implementation, which is an authentication protocol that is widely-used in Enterprise WPA2/3 networks. An adversary can exploit this vulnerability to bypass Phase-2 authentication, independent of which Phase-2 authentication method is being used. One requirement to perform the attack is that the client must be (mis)configured to not check the authenticity of the authentication server. Unfortunately, even recent work from 2022 has shown that this is still commonly the case in practice [18].

The vulnerability in `IWD` is present in its implementation of the 4-way handshake, which is used when connecting to any protected Wi-Fi network for the first time. It is exploitable when `IWD` is operating in Access Point (AP) mode. For context, `IWD` is an open-source Wi-Fi daemon created by Intel that was first released in 2018.

In the next section, we will provide a more detailed introduction to these authentication protocols. For both vulnerabilities, we will also describe in detail when they can be exploited, what the practical impact is, and how users can defend themselves against attacks.

2 TECHNICAL BACKGROUND

In this section, we introduce the security protocols that are used in modern Wi-Fi networks. We specifically focus on how clients verify the authenticity of a protected Wi-Fi network and how in turn the Wi-Fi network verifies the identity of the user.

2.1 Home WPA2 Networks

In home WPA2 networks, authentication is done using a shared password. Here the 4-way handshake provides mutual authentication between the client and AP. In other words, in the 4-way handshake the client proves to the AP that it possesses the password, and at the same time the AP also proves that it knows the password. This assures that unauthorized users cannot access the network and that the client is connecting to the legitimate network.

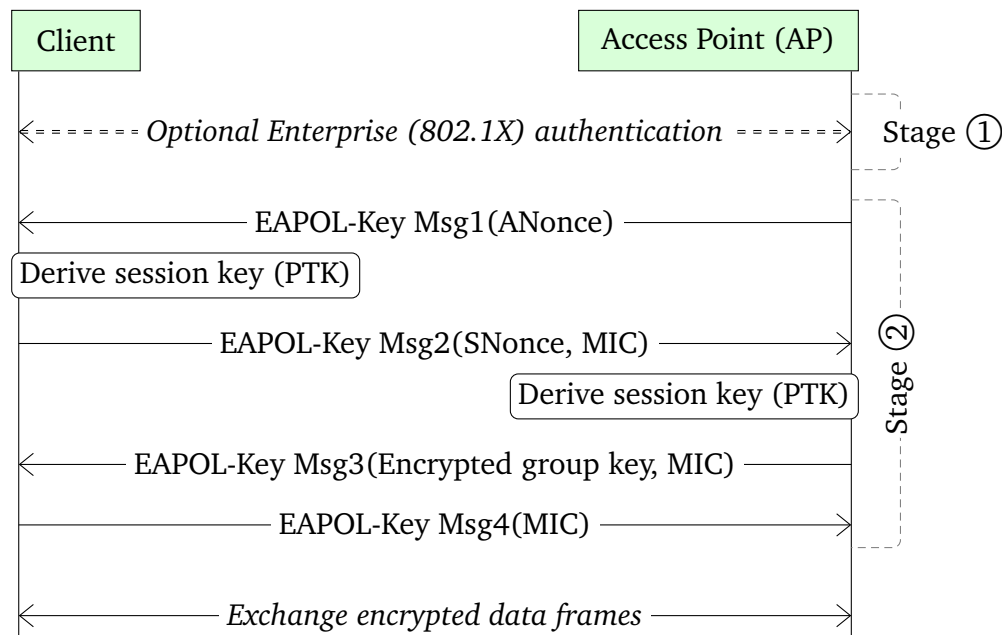


Figure 1: Illustration of the different handshake stages when connected to a (home or Enterprise) WPA2 network or an Enterprise WPA3 network.

The messages exchanged in the 4-way handshake are shown in stage ② of Figure 1. Apart from providing mutual authentication, the 4-way handshake also generates a fresh session key between the client and AP. This session key is called the PTK (Pairwise Transient Key) and is used to encrypt data frames once the 4-way handshake has been completed. The PTK is calculated by combining a random number generated by the client, called the SNonce, a random number generated by the AP, called the ANonce, and the shared password. These two random numbers, the SNonce and ANonce, are transported in message 1 and 2 of the 4-way handshake, respectively. What is notable to remark is that the client can already derive the PTK after reception of message 1 of the 4-way handshake. In contrast, the AP will derive the PTK after reception of message 2 (see Figure 1).

Once the client or AP has derived the PTK, they will authenticate 4-way handshake messages by adding a Message Integrity Code (MIC). The secret session key, also called PTK, is used to calculate the MIC. As shown in Figure 1, in practice this means that all messages except message 1 are protected by a MIC. A receiver, i.e., a client or AP, will only process all other handshake messages (apart from message 1) after first verifying the MIC. This prevents an adversary from modifying messages 2 through 4: the adversary does not know the PTK and therefore cannot compute a valid MIC value, meaning they cannot create or modify handshake messages that require a MIC.

Source	Destination	Protocol	Info
02:00:00:00:00:00	02:00:00:00:01:00	EAPOL	Key (Message 1 of 4)
02:00:00:00:01:00	02:00:00:00:00:00	EAPOL	Key (Message 2 of 4)
02:00:00:00:00:00	02:00:00:00:01:00	EAPOL	Key (Message 3 of 4)
02:00:00:00:01:00	02:00:00:00:00:00	EAPOL	Key (Message 4 of 4)

Figure 2: Illustration of a normal EAPOL 4-way handshake in Wireshark.

Figure 2 contains a screenshot of the 4-way handshake, also sometimes called the EAPOL 4-way handshake, as shown by Wireshark. Notice that the AP, which here has MAC address 02:00:00:00:01:00, initiated the handshake by sending message 1.

The 4-way handshake also transports the group key to the client in message 3 (see Figure 1). This group key is used by the AP to encrypt broadcast and multicast packets.

2.2 Enterprise WPA2 and WPA3 Networks

In an Enterprise WPA2 or WPA3 network, an 802.1X EAP-based handshake is performed before executing the 4-way handshake. The advantage of using EAP is that it supports a wide range of authentication methods. For example, users can be authenticated based on their username and password, based on client certificates, based on their mobile SIM card, and so on.

As an example, Figure 3 shows an EAP handshake where PEAP with MS-CHAPv2 is used to authenticate the user based on their username and password. The full details of this handshake are not important to understand our attacks. What is notable is that first the anonymous identity of the user is required. This anonymous identity informs the AP which authentication server to use to authenticate the user.

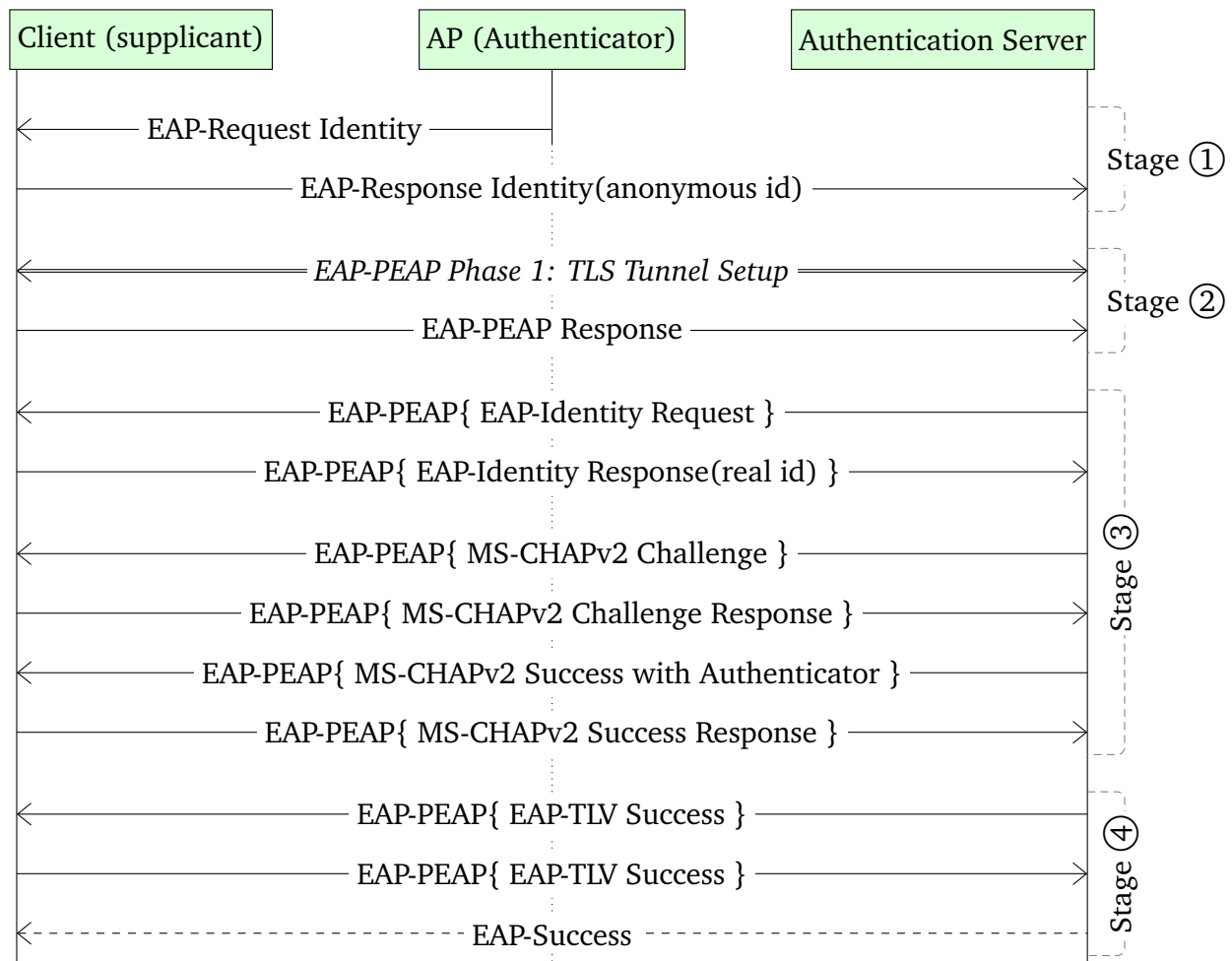


Figure 3: Illustration of a PEAPv0 handshake with as Phase 2 authentication MS-CHAPv2. Note that EAPOL is used to transfer EAP packets between the client and AP, and that RADIUS is used to transfer EAP packets between the AP and authentication server.

When using PEAP, stage ② of the handshake will establish a TLS connection between the client and the authentication server. During this phase, the client can optionally also verify the authenticity of the authentication server, and thereby also the authenticity of the Wi-Fi network, by verifying the TLS certificate of the server.

Next, in stage ③ of Figure 3, the Phase-2 authentication method is executed between the client and authentication server. These Phase-2 methods are executed within the TLS tunnel. This is done because some Phase-2 methods, such as MS-CHAPv2, have known vulnerabilities, and executing them within a TLS tunnel mitigates these known vulnerabilities. In practice, the most common Phase-2 authentication method is MS-CHAPv2, but others like EAP-SIM or EAP-AKA can also be used. These three Phase-2

methods provide mutual authentication: the client and authentication server prove to each other that they both possess the user's secret credentials, e.g., the user's password. In case either the client or authentication server does not know the user's credentials, then all these three Phase-2 methods will fail.

Finally, in stage ④ of the EAP handshake, the authentication server will send a Success message to the client to indicate that authentication has been completed successfully. The client will in turn also reply with a Success message. After this stage, the 4-way handshake is executed to negotiate a session key (PTK), so that encrypted data frames can then be exchanged.

2.3 Home WPA3 Networks

In home WPA3 networks, the Simultaneous Authentication of Equals (SAE) handshake, also called the Dragonfly handshake, is performed before executing the 4-way handshake. In other words, in WPA3 networks the 4-way handshake is *still* being used. However, first executing the SAE handshake mitigates several known weaknesses in the 4-way handshake. Namely, WPA3 provides forward secrecy and is resistant to dictionary attacks, unlike the older WPA2 protocol.

Additionally, when using WPA3, both the client and AP are required to use Management Frame Protection (MFP). This, among other things, prevents the infamous Deauthentication attack. Without using MFP, an adversary can spoof Deauthentication packets to forcibly disconnect a client from the network, hence the name Deauthentication attack.

3 ANALYSIS OF WPA_SUPPLICANT

In this section, we introduce the `wpa_supplicant` Wi-Fi client, and discuss the authentication bypass vulnerability that was discovered in it. This vulnerability was assigned the identifier CVE-2023-52160.

3.1 Usage in Practice

The `wpa_supplicant` client is the default Wi-Fi client in all Android devices and Linux distributions. It is an open-source client and was initially released in 2003, meaning it has been in use for more than two decades. This client is also often used in routers to implement repeater or client functionality and is used in ChromeOS devices.

3.2 Vulnerability Details

We discovered that against `wpa_supplicant`, the Phase-2 stage in the Enterprise PEAP authentication can be skipped. This implies that the vulnerability is present when using Enterprise WPA2/3 networks that use PEAP for authentication, which is the most common authentication method for Enterprise networks. In other words, stage ③ in Figure 3 can be completely skipped, and a malicious AP can instead immediately send the TLV-Success message to complete the PEAP handshake with `wpa_supplicant`.

To better understand the root cause of the vulnerability, we inspected the implementation of PEAP in `wpa_supplicant`. The vulnerable code is shown in Listing 1. What is notable is that the Phase-2 authentication can only be skipped if it has not been started yet. In case part of the Phase-2 authentication was already started, variable `phase2_eap_started` will have been set to true, and then the TLV-Success message is only accepted if the Phase-2 authentication has been completed successfully.

Listing 1: Vulnerable PEAPv1 code in `wpa_supplicant`. A success indication is accepted if Phase-2 authentication was never started or was fully completed. The connection is only aborted if the Phase 2 authentication was started but not yet completed. This can be abused to skip Phase 2 authentication and (more easily) impersonate a network.

```
1 static int eap_peap_decrypt(/*[...]*/) {
2     // [...]
3     case EAP_CODE_SUCCESS:
4         wpa_printf(MSG_DEBUG, "EAP-PEAP: Phase 2 Success");
5         if (data->peap_version == 1) {
6             /* EAP-Success within TLS tunnel is used to indicate
7              * shutdown of the TLS channel. The authentication has
8              * been completed. */
9             if (data->phase2_eap_started &&
10                !data->phase2_eap_success) {
11                 wpa_printf(MSG_DEBUG, "EAP-PEAP: Phase 2 "
12                    "Success used to indicate success, "
13                    "but Phase 2 EAP was not yet "
14                    "completed successfully");
15                 ret->methodState = METHOD_DONE;
16                 ret->decision = DECISION_FAIL;
17                 wpabuf_clear_free(in_decrypted);
18                 return 0;
19             }
```

In practice, an adversary can abuse this vulnerability to (more easily) create a rogue clone of a trusted Enterprise network. In particular, by skipping the Phase-2 authentication, the adversary can now trick the victim into connecting to the rogue network, even if the adversary does not know the password of the victim. We elaborate on this below.

3.3 Exploiting the Vulnerability

When a `wpa_supplicant` client is tricked into (trying to) connect to a malicious clone of an Enterprise Wi-Fi network, the vulnerability can be abused to make `wpa_supplicant` skip Phase-2 authentication. Under the conditions described below, this can be abused to make the client connect successfully, such that the client will subsequently send data to the adversary's malicious Enterprise network.

3.3.1 Typical Attack Prerequisites

The first requirement that must be fulfilled is unsurprising: the adversary needs to know which network they want to impersonate. More technically, the adversary needs to know the name (SSID) of at least one Enterprise Wi-Fi network that the victim will (automatically) connect to. The adversary also needs to be within range of the victim. However, the attacker does *not* have to be near the legitimate Enterprise Wi-Fi network that is being impersonated. The victim can also be located anywhere, e.g., at home, while traveling, at work, and so on, the victim does not have to be within range of the legitimate network that is being impersonated by the adversary.

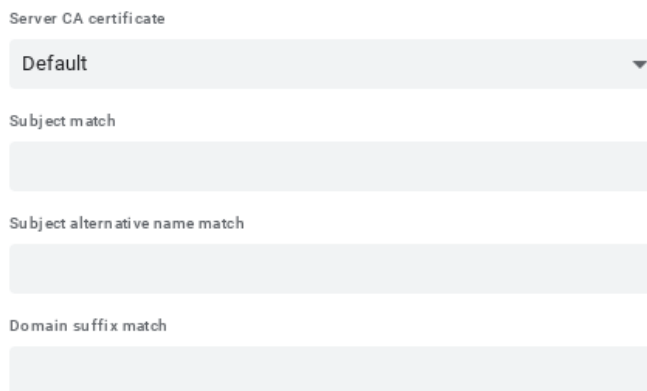
Also important to note is that no user interaction of the victim is required to exploit the vulnerability. The attacker only needs to create their own malicious Wi-Fi network with the same name (SSID) as an Enterprise Wi-Fi network that the victim previously used. When targeting specific individuals, e.g., employees of a company, an adversary can easily learn the Wi-Fi network name by walking around the company's building. When targeting random users, the adversary can advertise popular Enterprise network names such as `eduroam`, `Vodafone Homespot`, `TelenetWiFree`, `Unitymedia WifiSpot`, and so on, and wait until unsuspecting users connect to it. More generally, common network names can be found on websites such as `WiGLE`, which have a large database of Wi-Fi networks world-wide [20]. Since most devices will automatically connect to a previously-used Wi-Fi network, the user will not be aware of the attack.

If the legitimate Wi-Fi network is also within range of the victim, then techniques like spoofing Channel Switch Announcements can be abused to make the victim switch to the adversary’s malicious clone of the network [17, 9]. Additionally, if the victim is already connected to the Wi-Fi network, then the adversary can forcibly disconnect the victim by spoofing Deauthentication frames. If the legitimate network is using Management Frame Protection, which prevents spoofing deauthentication frames, then the adversary can still abuse implementation flaws to disconnect the victim from the legitimate network [14].

3.3.2 Main Attack Prerequisite: Establishing TLS Tunnel

The main condition to exploit the vulnerability in `wpa_supplicant` is that it must be configured to *not* verify the authentication server’s TLS certificate. Unfortunately, on many systems that use `wpa_supplicant`, this type of misconfiguration is a known issue:

ChromeOS. On ChromeOS, it is also difficult to properly configure the verification of the TLS certificate. Either the user selects “Do not check” so that the TLS certificate is not verified, or the user selects “Default”, in which case the user also needs to enter a so-called “Subject alternative name match” or “Domain suffix match” (see Figure 4 on page 11). We expect that many users will not know what these latter two fields mean, and as a result they will simply select to not check, i.e., to not verify, the certificate. In fact, many (university) guides instruct users to select “Do not check” [2, 11, 10, 15, 16, 8]. This implies that many ChromeOS users are likely vulnerable.



The image shows a configuration dialog for TLS certificate verification. It contains four fields:

- Server CA certificate:** A dropdown menu with "Default" selected.
- Subject match:** An empty text input field.
- Subject alternative name match:** An empty text input field.
- Domain suffix match:** An empty text input field.

Figure 4: Dialog when configuring ChromeOS to securely verify the TLS certificate of an Enterprise Wi-Fi network. The user needs to properly fill in these fields, otherwise the TLS certificate will not be properly verified, making it vulnerable to our attack.

Android. Previous research [3, 1, 7], including a recent study from 2022 [18], has shown that many Android users (unknowingly) configure their device not to verify the TLS certificate of the authentication server. This is because users have to manually configure the expected TLS certificate which is a tedious task for most users. As a result, we expect that many Android users are vulnerable to our attack.

Linux Distributions. On many Linux distributions, such as Ubuntu, Fedora, Debian, and so on, the user also has to manually configure the TLS certificate of the authentication server. As a result, similar to Android and ChromeOS, we expect that many users skip this configuration step, and will therefore be vulnerable to our attack.

3.3.3 Attack Details

The resulting attack is illustrated in Figure 5. In the attack, the adversary acts as a malicious Enterprise Wi-Fi network, and induces the victim into connecting to this network. While the victim is connecting, the Phase-2 authentication method can be trivially skipped because of the vulnerability by sending the EAP-TLV Success packet instead of starting Phase-2. Once connected, the adversary can intercept and manipulate all Internet traffic of the victim.

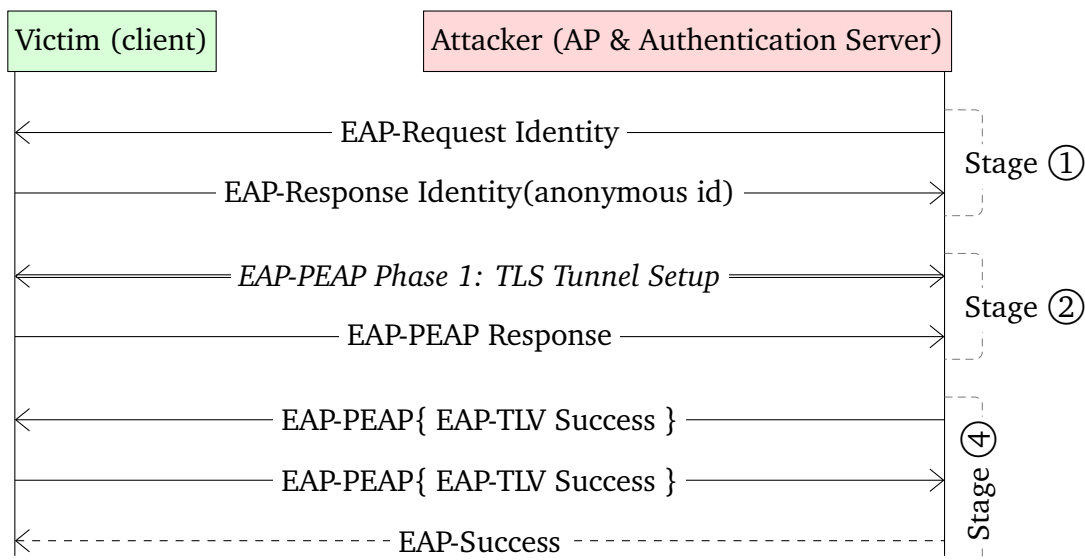


Figure 5: Attack against `wpa_supplicant` where Phase-2 authentication is skipped. Notice how there is no stage ③ in contrast with Figure 3. This allows the adversary to impersonate a network without needing to know the victim’s password.

3.4 Defenses

To prevent the vulnerability, `wpa_supplicant` was patched to more carefully handle packets that indicate PEAP authentication was successful [12]. When such packets are now received, the patched `wpa_supplicant` will first verify that the Phase 2 handshake was successfully completed, and if that is not the case, the connection attempt is aborted. More precisely, when receiving the EAP-TLV Success packet in the PEAPv0 TLS tunnel, or when receiving the EAP-Success packet within the PEAPv1 TLS tunnel, `wpa_supplicant` will reject this message if Phase 2 authentication was not yet successfully completed.

For backward compatibility, the patch provides a new option “`phase2_auth`” to still revert to the old behavior where Phase 2 authentication can be skipped. Additionally, the patched `wpa_supplicant` will, by default, not require Phase 2 authentication when client certificates are used to establish the PEAP TLS connection.

The patch will be part of the next release of `wpa_supplicant`, which will be version 2.11. At the time of writing, it is unknown when this version will be released. In the meantime, we strongly encourage vendors to backport the patch to older (and existing) versions of `wpa_supplicant` [12], so that users can already be protected instead of having to wait for the next version of `wpa_supplicant`.

4 ANALYSIS OF IWD

In this section, we introduce Intel’s open-source IWD daemon and subsequently analyze its security. This led to the discovery of a logical implementation vulnerability that allows an adversary to gain unauthorized access to a protected Wi-Fi network. This vulnerability was assigned the Common Vulnerabilities and Exposures identifier CVE-2023-52161.

4.1 Introduction to IWD

Intel’s iNet Wireless Daemon (IWD) is an open-source Wi-Fi client developed by Intel for Linux-based systems. It was designed with the goal of simplifying and improving the Wi-Fi experience on Linux, so that configuring Wi-Fi on Linux would become more user-friendly. The development of IWD started in 2014 [5] and the first public release of version 0.1 was on February 2018 [4, 6].

A few months after releasing IWD version 0.1, support for access point mode was added in version 0.4 which was released on July 25, 2018 [6]. Nowadays, IWD is available in the package managers of numerous Linux distributions such as Arch Linux, Debian, Ubuntu, and so on. An advantage of IWD is that it is designed to be lightweight so it can run on a diverse set of systems.

More recent releases of IWD even have a built-in DHCP server, making the creation of a usable Wi-Fi network very straightforward. For example, a system administrator can create a simple configuration file, as shown in Listing 2, and this is sufficient to start a protected Wi-Fi network using IWD.

Listing 2: Example config file of an IWD network. The filename contains the network name which here is “testnetwork”. Since the AP mode of IWD contains a built-in DHCP server, we can immediately also configure the IP addresses that are handed out to clients.

```
1 $ cat /var/lib/iwd/ap/testnetwork.ap
2 [Security]
3 Passphrase=password123
4
5 [IPv4]
6 Address=192.168.250.1
7 Gateway=192.168.250.1
8 Netmask=255.255.255.0
9 DNSList=8.8.8.8
```

4.2 Vulnerability Details

When inspecting the source code of IWD’s 4-way handshake implementation, we noticed that it did not verify in which order message 2 or 4 of the handshake are received. Put differently, IWD is not storing or checking what the expected next message in the handshake is. It will simply accept any message.

The code in Listing 3 contains the vulnerable code. The vulnerability is in the function `eapol_auth_key_handle`, which is called whenever a 4-way handshake message is received by the AP. In Line 9 it is checked whether the AP has already sent message 1 of the 4-way handshake, i.e., whether a handshake is in progress. However, in lines 12

to 16 it is not checked whether the AP now expects message 2 or 4. Instead, whichever message arrives is processed. This means that when an adversary is connecting to an IWD network, they can skip message 2 and immediately send message 4.

Listing 3: Vulnerable code in Intel’s iNet Wireless Daemon (IWD). The AP does not verify whether message 4 of the 4-way handshake is expected or not. An adversary can send an unexpected message 4 to complete the handshake without needing to know the password of the network, thereby gaining unauthorized access to the network.

```
1 static void eapol_auth_key_handle(struct eapol_sm *sm,
2                                 const struct eapol_frame *frame) {
3     size_t frame_len = 4 + L_BE16_TO_CPU(frame->header.packet_len);
4     const struct eapol_key *ek = eapol_key_validate((const void *) frame,
5                                                     frame_len, sm->mic_len);
6
7     // [...]
8
9     if (!sm->handshake->have_anonce)
10        return; /* Not expecting an EAPoL-Key yet */
11
12    key_data_len = EAPOL_KEY_DATA_LEN(ek, sm->mic_len);
13    if (key_data_len != 0)
14        eapol_handle_ptk_2_of_4(sm, ek);
15    else
16        eapol_handle_ptk_4_of_4(sm, ek);
17 }
```

4.3 Exploiting the Vulnerability

We now know that an adversary, after receiving message 1 from IWD, can immediately reply with message 4 of the 4-way handshake. However, IWD will still try to verify the MIC of the received message 4. Interestingly, IWD will now use an all-zero PTK key to verify the MIC value. This is because IWD never received message 2 and therefore never derived a valid PTK key (recall Figure 1) and instead uses a “default” all-zero key.

To exploit the vulnerability, an adversary has to send a message 4 where the MIC is calculated using an all-zero PTK. This attack is illustrated in Figure 6. When IWD receives this message, it will process message 4 due to the vulnerability we discovered, and it will then verify the MIC using an all-zero key. This verification will succeed, after which

the handshake is complete, and IWD will then accept encrypted data frames from the adversary. These encrypted data frames are also encrypted using the all-zero PTK key, meaning an adversary now has full access to the Wi-Fi network.

Source	Destination	Protocol	Info
02:00:00:00:00:00	02:00:00:00:01:00	EAPOL	Key (Message 1 of 4)
02:00:00:00:01:00	02:00:00:00:00:00	EAPOL	Key (Message 4 of 4)
02:00:00:00:01:00	Broadcast	802.11	QoS Data, SN=15, FN=0, Flags=.p.....T
02:00:00:00:01:00	Broadcast	802.11	Data, SN=79, FN=0, Flags=.p....F.
02:00:00:00:00:00	02:00:00:00:01:00	802.11	Data, SN=80, FN=0, Flags=.p....F.

Figure 6: Successful authentication bypass attack against an IWD AP, where message 2 and 3 of the 4-way handshake are skipped. This is followed by the exchange of encrypted data frames between the attacker and vulnerable AP

We implemented the attack by modifying `wpa_supplicant` to send message 4 after receiving message 1, and to use an all-zero key to calculate the MIC value of message 4. Listing 4 shows selected debug output of our modified `wpa_supplicant`, where the all-zero PTK is installed after sending message 4. All combined, this allows the adversary to have full access to the network, i.e., the adversary can now send and receive encrypted packets to and from the network.

Listing 4: The attacker's `wpa_supplicant` is modified to use an all-zero Pairwise Transition Key (PTK) after completion of the 4-way handshake with a vulnerable IWD network.

```

1 wlan1: WPA: Installing PTK to the driver
2 wpa_driver_nl80211_set_key: ifindex=15 (wlan1) alg=3 addr=0x2d90afe0 key_idx=0
3 set_tx=1 seq_len=6 key_len=16 key_flag=0x2c link_id=-1
4 nl80211: NEW_KEY
5 nl80211: KEY_DATA - hexdump(len=16): 00 00 00 00 00 00 00 00
6                                     00 00 00 00 00 00 00 00
7 nl80211: KEY_SEQ - hexdump(len=6): 00 00 00 00 00 00
8 addr=02:00:00:00:00:00
9 pairwise key
10 EAPOL: External notification - portValid=1
11 wlan1: WPA: Key negotiation completed with 02:00:00:00:00:00 [PTK=CCMP GTK=CCMP]

```

4.4 Attack Limitations

One minor limitation of the attack is that the adversary will not receive the group key that is normally sent in message 3 of the handshake. This key is used to decrypt broadcast

packet *sent by the AP*. When the client broadcasts a packet, it sends it as a unicast Wi-Fi frame to the AP, and the AP will then broadcast it to all devices. This means that a client can still use the all-zero PTK to encrypt and send broadcast or multicast packets. All combined, only the decryption of broadcast or multicast packets is not possible, the adversary can still transmit such packets to the network.

4.5 Defense

To prevent the vulnerability, a patch for IWD was written that tracks whether the next expected 4-way handshake message is either message 2 or 4. As a result, when IWD now receives message 4 after sending message 1, this unexpected message 4 will be rejected. The patch will be part of the next release of IWD, which will be **version 2.12**.

5 RELATED WORK

Notable and highly recommended related work is that of Dominic White as part of Orange Cyberdefense's SensePost team [19]. He discovered that the MS-CHAPv2 authentication method, which is often used as a Phase-2 authentication method in PEAP, was improperly implemented in Apple products. Namely, it was possible to skip certain messages of the MS-CHAPv2 protocol, and similar to our attack, this could be abused to make a victim connect to a malicious copy of an Enterprise Wi-Fi network.

In his work, White also tested `wpa_supplicant` for this vulnerability, but found that it was not vulnerable. We confirmed this observation: `wpa_supplicant` indeed requires Phase-2 authentication to be completed *once it is started*. More precisely, in the tests that White performed, the Phase-2 MS-CHAPv2 handshake was started, after which certain messages were skipped.¹ This caused the attack to fail, whereas in our attack we do not start the Phase-2 handshake at all, which is the key to discovering and exploiting the vulnerability. Also note that our vulnerability in `wpa_supplicant` can be abused to bypass not just MS-CHAPv2, but can be abused to bypass *any* Phase-2 authentication method.

Researchers have also discovered various weaknesses in MS-CHAPv2 [13]. They showed that by capturing a MS-CHAPv2 handshake, and using specialized FPGA equipment, the

¹See the parameter `wpe_enable_return_success` in the `hostapd-wpe` patch.

MD4 hash of the password can be recovered. Even with this specialized equipment it took on average half a day, and at most one day, to recover the MD4 hash [13]. If the user has a short or weak password, this MD4 hash can then also be broken to reveal the user's original password. With this password it is then also possible to trick the user into connecting to a malicious clone of an Enterprise network. In comparison, our attack can be executed without needing specialized hardware, and without the computationally expensive steps of recovering the MD4 hash and subsequently cracking it, making our attack significantly easier to execute in practice.

6 CONCLUSION

Discovering and avoiding logical implementation bugs in cryptographic protocol remains a difficult challenge in practice. Even when using modern protocols, such implementation vulnerabilities can severely compromise security, as demonstrated by this report. In practice, it is therefore recommended to have multiple layers of defense. In other words, apart from using Wi-Fi encryption, users should ensure that websites are using HTTPS, or use a VPN to ensure that nearby adversaries cannot intercept or manipulate your traffic.

ACKNOWLEDGEMENTS

This report was supported by a research grant of Top10VPN. We would also like to thank the Research Fund KU Leuven and the Flemish Research Programme Cybersecurity for funding this line of research.

REFERENCES

- [1] Alberto Bartoli, Eric Medvet, Andrea De Lorenzo, and Fabiano Tarlao. (in) secure configuration practices of wpa2 enterprise supplicants. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–6, 2018.
- [2] Cal Poly Pomona University. Wireless access: Chromeos set-up guide using SecureW2. Retrieved 20 December 2023 from https://cpp.service-now.com/ehelp/ess?id=kb_article&sys_id=2aeba6fe877039104f5098e73cbb35f7&spa=1, December 2023.

- [3] Aldo Cassola, William K Robertson, Engin Kirda, and Guevara Noubir. A practical, targeted, and stealthy attack against WPA enterprise authentication. In *NDSS*, 2013.
- [4] Jonathan Corbet. iwd: simplifying WiFi management. Retrieved 28 December 2023 from <https://lwn.net/Articles/770991/>, November 2018.
- [5] Marcel Holtmann. Initial revision. IWD commit 344a33268ce1, April 2014.
- [6] Marcel Holtmann. The new Wi-Fi experience for linux. Retrieved 28 December 2023 from <https://www.youtube.com/watch?v=QIQT2obSPDk>, October 2018.
- [7] Man Hong Hue, Joyanta Debnath, Kin Man Leung, Li Li, Mohsen Minaei, M Ham-mad 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.
- [8] Shivan Kaul. Authentication certificate rejected locally error while trying to connect to wifi on chromebook. Retrieved 20 December 2023 from <https://shivankaul.com/blog/authentication-certificate-rejected-locally-error-on-chromebook>, December 2023.
- [9] 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.
- [10] Leeds Beckett University. How do i connect my chromebook to eduroam? Retrieved 20 December 2023 from <https://libanswers.leedsbeckett.ac.uk/faq/180415>, December 2023.
- [11] Louisiana State University. Chromebook: Connecting to eduroam (wireless). Retrieved 20 December 2023 from <https://moodle3.grok.lsu.edu/article.aspx?articleid=17398>, December 2023.
- [12] Jouni Malinen. PEAP client: Update Phase 2 authentication requirements. Hostap commit 8e6485a1bcb0. Retrieved 28 December 2023 from <https://w1.fi/cgit/hostap/commit/?id=8e6485a1bcb0baff>, July 2023.

- [13] Moxie Marlinspike. Divide and conquer: Cracking MS-CHAPv2 with a 100 Retrieved 10 December 2023 from <https://web.archive.org/web/20160316174007/https://www.cloudcracker.com/blog/2012/07/29/cracking-ms-chap-v2/>, 2021.
- [14] 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*, pages 245–256, 2022.
- [15] Texas A&M University. Q. how do i log in to chromebooks borrowed from the library? Retrieved 20 December 2023 from <https://askus.library.tamu.edu/faq/383010>, December 2023.
- [16] University of York. Wifi: Connecting chrome OS - manual setup. Retrieved 20 December 2023 from <https://support.york.ac.uk/s/article/Wifi-Connecting-Chrome-OS-Manual-Set-up>, December 2023.
- [17] Mathy Vanhoef and Frank Piessens. Release the kraken: new cracks in the 802.11 standard. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 299–314, 2018.
- [18] Kailong Wang, Yuwei Zheng, Qing Zhang, Guangdong Bai, Mingchuang Qin, Donghui Zhang, and Jin Song Dong. Assessing certificate validation user interfaces of WPA supplicants. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 501–513, 2022.
- [19] Dominic White. Understanding PEAP in-depth. Retrieved 21 December 2023 from <https://sensepost.com/blog/2019/understanding-peap-in-depth/>, 2019.
- [20] WiGLE. WiGLE Stats: SSID. Retrieved 28 December 2023 from <https://wigo.net/stats#ssidstats>, 2023.

APPENDIX

Listing 5: Commands used in our experiments to configure IWD in AP mode.

```
1 wget https://mirrors.edge.kernel.org/pub/linux/network/wireless/iwd-2.9.tar.xz
2 tar -xvf iwd-2.9.tar.xz && cd iwd-2.9
3 ./configure && make
4 sudo mkdir -p /etc/iwd/
5
6 # Without the following command, starting IWD will result in the D-Bus error
7 # "Name request failed". This copies iwd-dbus.conf to $DATADIR/dbus-1/system.d
8 sudo /usr/bin/install -c -m 644 src/iwd-dbus.conf \
9     `pkg-config --variable=datadir dbus-1`/dbus-1/system.d
10 sudo reboot # Reboot after executing the above command.
11
12 # Optional system-wide IWD settings
13 sudo vim /etc/iwd/main.conf
14
15 # See Listing 2 for example content of testnetwork.ap
16 sudo mkdir -p /var/lib/iwd/ap/
17 sudo vim /var/lib/iwd/ap/testnetwork.ap
18
19 sudo ./src/iwd -i wlan0
20 sudo ./client/iwctl
21 # Now execute the following two commands in iwctl to enable AP mode:
22 #     device wlan0 set-property Mode ap
23 #     ap wlan0 start-profile testnetwork
```

Listing 6: Configuration of the Kankun smart power socket so that it can properly connect to modern WPA2 networks that only support the CCMP cipher.

```
1 # Modify /etc/config/wireless as follows:
2 config wifi-iface
3     option device 'radio0'
4     option mode 'sta'
5     option ssid 'your ssid'
6     option encryption 'psk2+ccmp'
7     option network 'wan'
8     option key '***'
9
10 # Append to /etc/config/network the following:
11 config interface 'wan'
12     option proto 'dhcp'
```
