Testing and Improving the Correctness of Wi-Fi Frame Injection

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

Xianjun Jiao
IDLab, imec - Ghent Univeristy
Gent, Belgium
xianjun.jiao@imec.be

Wei Liu
IDLab, imec - Ghent Univeristy
Gent, Belgium
wei.liu@imec.be

Ingrid Moerman
IDLab, imec - Ghent Univeristy
Gent, Belgium
ingrid.moerman@imec.be

ABSTRACT

Investigating the security of Wi-Fi devices often requires writing scripts that send unexpected or malformed frames, to subsequently monitor how the devices respond. Such tests generally use Linux and off-the-self Wi-Fi dongles. Typically, the dongle is put into monitor mode to get access to the raw content of received Wi-Fi frames and to inject, i.e., transmit, customized frames.

In this paper, we demonstrate that monitor mode on Linux may, unbeknownst to the user, mistakenly inject Wi-Fi frames or even drop selected frames instead of sending them. We discuss cases where this causes security testing tools to misbehave, making users believe that a device under test is secure while in reality it is vulnerable to an attack. To remedy this problem, we create a script to test raw frame injection, and we extend the Radiotap standard to gain more control over frame injection. Our extension is now part of the Radiotap standard and has been implemented in Linux. We tested it using commercial Wi-Fi dongles and using openwifi, which is an open implementation of Wi-Fi on top of software-defined radios. With our improved setup, we reproduced tests for the KRACK and FragAttack vulnerabilities, and discovered previously unknown vulnerabilities in three smartphones.

CCS CONCEPTS

• **Security and privacy** → **Mobile and wireless security**; • **Networks** → *Protocol testing and verification.*

KEYWORDS

802.11, monitor mode, packet injection, radiotap

ACM Reference Format:
Mathy Vanhoef, Xianjun Jiao, Wei Liu, and Ingrid Moerman. 2023. Testing and Improving the Correctness of Wi-Fi Frame Injection. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '23), May 29-June 1, 2023, Guildford, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3558482.3581779

WiSec '23, May 29-June 1, 2023, Guildford, United Kingdom

ACM ISBN 978-1-4503-9859-6/23/05…$15.00
https://doi.org/10.1145/3558482.3581779

1 INTRODUCTION

A large number of Wi-Fi experiments are performed using Linux scripts with commercial off-the-shelf network cards. This is especially the case in security research, where often unexpected or malformed frames are sent towards a device under test. For instance, the aircrack-ng toolsuite, which is arguably one of the most well-known tools for offensive Wi-Fi research, is designed to be used on Linux with commodity Wi-Fi dongles. The same is true for other Wi-Fi tools such as mdk4. However, these commercial network cards are often black-boxes over which researchers have no control, meaning there is no guarantee that injected frames are transmitted as intended by the (user space) Linux scripts.

Overall, Wi-Fi scripts on Linux generally make use of monitor mode to have control over the full (raw) frames that are transmitted, but the reliability of raw Wi-Fi frame injection on Linux has never been studied. This means it is unclear how reliable such scripts are in practice. We provide a sobering answer to this question: depending on the experimental setup, various problems may arise when injecting Wi-Fi frames, and these problems can cause scripts to misbehave. To understand the extent of this problem, we study Wi-Fi frame injection in detail, and propose, implement, and standardize improvements to make frame injection more reliable.

To evaluate the correctness of raw Wi-Fi frame injection, we create a tool to perform various injection tests under different configurations of the network card, and this tool checks whether frames are transmitted as intended [19]. These tests revealed that header fields may be overwritten before transmission, that injected frames may be reordered, that some types of injected frames may get unexpectedly dropped by the driver or network card, and that frames may be unintentionally sent when the receiver is in sleep mode. In this paper, we discuss how these injection problems may impact the correctness of Wi-Fi experiments and tests.

To improve the correctness of frame injection, we extend Radiotap, which is a cross-platform Application Programming Interface (API) that is used when injecting and capturing raw Wi-Fi frames. We implement these extensions on Linux and test them with off-the-shelf Wi-Fi dongles and with openwifi, which is an open Wi-Fi implementation that uses software defined radios [12]. Our extensions have been included in an update to the Radiotap standard, and our new Radiotap code and injection fixes are meanwhile part of the official Linux kernel since kernel version 5.11.

To demonstrate the impact of the injection fixes, we replicate recent Wi-Fi attacks to test (new) devices in a more reliable manner.

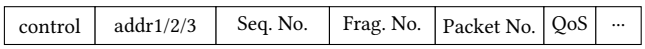

Figure 1: Simplified header of an encrypted Wi-Fi frame.

In particular, we reproduce tests for the KRACK and FragAttacks vulnerabilities [18, 21]. This led to the discovery of previously unknown vulnerabilities in three smartphones, which we disclosed to the affected vendors. To summarize, our contributions are:

- We test the injection of various Wi-Fi frames under different conditions and give an overview of issues that can occur when injecting raw Wi-Fi frames (Section 3).
- We extend the Radiotap standard to better control frame injection, implement our extensions on Linux, and make the extensions compatible with openwifi [12] (Section 4).
- We evaluate our injection improvements by replicating test for the KRACK and FragAttacks vulnerabilities. This allowed us to (more reliably) discover three new vulnerable devices.

Finally, we give an overview of related work regarding frame injection in Section 6, and conclude in Section 7.

2 BACKGROUND

This section introduces the IEEE 802.11 standard that underpins Wi-Fi and focuses on monitor mode and injection of raw frames.

2.1 Header layout of encrypted 802.11 frames

Figure 1 shows the layout of an encrypted 802.11 frame. The frame starts with a control field that contains flags indicating whether the client is entering sleep (power-save) mode and whether there are more fragments of the frame to follow. Next are the transmitter, sender, and final destination MAC address. The 12-bit sequence number (Seq. No.) is used to identify and ignore retransmissions, while the 4-bit fragment number (Frag. No.) is used to reassemble a fragmented frame. The packet number contains the nonce used to encrypt and authenticate the transported data. Finally, the 802.11e 0oS (Quality-of-Service) field contains a flag to indicate whether this is an aggregated (A-MSDU) frame and contains the Traffic Identifier (TID) which represents the frame's priority.

2.2 The Radiotap standard

Normally, the operating system fills in all fields of a Wi-Fi frame. However, some operating systems also support injection of raw 802.11 frames, where the user has full control over the frame's content. When injecting such raw 802.11 frames, various parameters must also be provided: the transmission bitrate, the channel bandwidth to use, whether to retransmit the frame if it was not acknowledged, whether to use collision-avoidance methods such as CTS-to-self, etc. User-space programs specify these parameters by prepending the Wi-Fi frame with a Radiotap header. This header contains control info to specify exactly how the injected frame is transmitted [2]. Note that the kernel also prepends received frames with a Radiotap header containing metadata related to the received frame, such as the bitrate, signal strength, and so on. As a result, Radiotap is typically used by specialized tools to inject and monitor traffic, but it is also used by standard programs such as hostap.

Radiotap is a de facto standard. New fields are added by proposing a change, implementing it in wireshark or tcpdump, adding support for the new field to at least one driver, and posting the proposal to the Radiotap mailing list [2]. If the proposal withstands discussion, the proposed change is re-posted in its final form, after which it is adopted one week later if there are no further objections.

2.3 Frame injection on Linux

The Linux kernel has built-in support for raw frame injection [10]. To use it, the wireless interface needs to be set in monitor mode, after which user space processes use the `nl80211` interface to inject, i.e., transmit, raw Wi-Fi frames over this interface. Commands sent over `nl80211` are handled by the `cfg80211` kernel module, which will in turn call a transmit callback function that was registered by the driver during the initialization of the network interface.

There are two types of Linux Wi-Fi drivers: Full-MAC and Soft-MAC ones. A Full-MAC driver implements the MAC sublayer Management Entity (MLME) in hardware or firmware. The MLME layer is responsible for handling operations such as scanning, (open) authentication, association, and so on. A Soft-MAC driver relies on the `mac80211` kernel module to handle (parts of) the MLME in software. A Full-MAC driver registers callbacks in `cfg80211` using the `cfg80211_ops` structure, while a Soft-MAC driver registers callbacks in `mac80211` using the `ieee80211_ops` structure.

All combined, with Full-MAC drivers, injected frames are directly passed by `cfg80211` to the driver, meaning the driver or hardware will parse the Radiotap header. In contrast, with Soft-MAC drivers, injected frames are passed by `cfg80211` to the `mac80211` kernel module, meaning `mac80211` will parse the Radiotap header.

When injecting a frame, it will appear twice when capturing packets on the interface used to inject the frame. The first copy appears before and is independent of whether the frame is actually transmitted, and it represents the frame as injected by the user space process. The second copy is the frame that is actually transmitted, containing extra details in the Radiotap header such as the bitrate that was used and whether an acknowledgement was received.

Various flags exists to control how an interface in monitor mode behaves [6]. An important flag is the `activate` flag. When set, the network card actively acknowledges incoming unicast frames if they match the configured MAC address. However, at the time of writing, few network cards on Linux support this flag, and recent works instead used virtual interface to assure that incoming frames are acknowledged [15, 18] (see also Section 3.2).

2.4 Virtual interfaces on Linux

Linux can use a network card in several modes, such as client or Access Point (AP) mode, while simultaneously operating the card in monitor mode. This is done using virtual interfaces: one virtual interface implements the client or AP behavior, while a second virtual interface can be used to monitor and inject frames. Recent works used this ability to quickly prototype proof-of-concepts [15, 18].

When the network card is only used in monitor mode, then we say it is operating in *pure monitor mode*. When the network card is used by one or more (virtual) interfaces in client or AP mode, and also by one or more (virtual) interfaces in monitor mode, then we say that the network card is operating in *mixed monitor mode*. In case of mixed monitor mode, the term *non-monitor interface(s)* refers to the interfaces that are operating in client or AP mode.

3 TESTING FOR FRAME INJECTION ISSUES

In this section, we test the correction of raw frame injection, and discuss cases where injected frames are incorrectly transmitted.

3.1 Experimental setup

To test the correctness of frame injection for various network cards, we wrote a Python script using the Scapy library to inject several kinds of frames [19]. A second independent network card is used in pure monitor mode to capture injected frames and verify whether the frame has been sent without any unexpected modifications. To reliably detect the injected frame, a unique random identifier is appended to every injected frame. If the frame was not detected, it is retransmitted once, meaning the frame is injected at most twice.

In our experiments we consider various configurations. First, we test the network card in pure monitor mode. Second, we test the card in mixed monitor mode. Here we perform the injection tests before the client is connecting, while the client is connecting, and once the client has successfully authenticated to the AP. In these scenarios, the injection tests were done both when the non-monitor interface is in client mode and when it is in AP mode.

We tested the network cards recommended by the KRACK and FragAttack tools [18, 21]. In particular, we tested the TP-Link TL-WN722N, Intel Wireless-AC 3160, and Alfa AWUS036ACM. These internally use an Atheros, Intel, and MediaTek chip, respectively. We also tested a network card with a RT5572 chipset. We used a (second) TL-WN722N and AWUS036ACM for the network card that monitors whether the network card under test is correctly injecting frames. All experiments are performed using an ZBook Power G8 running Arch Linux with kernel version 5.4.223-1-lts.

3.2 Acknowledgements and retransmissions

Our first observation is that the retransmission behavior depends on the type of network card being used. When no retransmissions are executed, this means that injected frames may not have arrived at their destination, while an excessive number of retransmissions may impact the reliability of time-sensitive experiments because unnecessary retransmissions delay the reception and transmission of other frames. Our experiments indicate that the retransmission behavior depends on: (1) the type of network card; (2) whether pure or mixed monitor mode is used (recall Section 2.4); and (3) whether the transmitter MAC address of the injected frame equals the MAC address of a non-monitor interface that is associated to the network card. We suspect that these behavioral differences could be abused to fingerprint and detect the network card used by an attacker.

Whether an interface in monitor mode acknowledges received frames also depends on various factors. In theory, when in pure monitor mode, acknowledgements should only be sent if the `active` flag was set (recall Section 2.3). However, none of our tested network cards support this flag. An inspection of the Linux kernel source code indicates that only two out of more than 60 Wi-Fi drivers support this flag: the MediaTek mt76 and mt7601u driver.[1] Interestingly, we found that the WN-722N acknowledges received frames in pure monitor mode without setting the `active` flag.

Not acknowledging frames can impact time-sensitive experiments and can cause experiments to fail. For instance, since 2002, the `hostapd` AP requires that authentication and association responses are acknowledged [13]. Otherwise, the client will be disconnected, at least if this info is accurately provided. Since acknowledgement frames cannot be injected in a timely manner from user space, they must be generated in hardware. Hence, when the monitor interface does not acknowledge frames, one cannot simulate a client that can successfully connect to APs with such behavior.

[1] This was determined by searching for the word `NL80211_FEATURE_ACTIVE_MONITOR`.

3.3 Handling sleeping clients

Another issue we observed is that scripts may inject frames when the client is in sleep mode. This can be problematic because this may lead some testing scripts, such as the FragAttacks tool, to wrongly believe that the client being tested is not vulnerable [18]. Previous works mitigated this issue by using the network card in mixed monitor mode [15, 18]. That is, one interface is operating as an AP while a second interface is in monitor mode. When a Wi-Fi frame is then injected on the monitor interface, the kernel will effectively add the frame to the AP's transmission queue, meaning the injected frame is only sent once the client wakes up. Unfortunately, even in mixed mode we observed that some injected frames did not arrive at the client, or arrived significantly later than expected. It is unclear what caused this.

We are not aware of a method to force clients to wake up. Nevertheless, one can use a second network card in monitor mode to detect when the client under test wakes up, and more importantly when the injected frame is sent. If the frame is sent much later than expected, the user can be warned. Similarly, if the injected frame was not acknowledged, a warning can also be shown. Though this does not fundamentally solve the problem of handling sleeping clients, it does make the user aware that a test needs to be re-run.

3.4 Order of injected frames

When using a network card in mixed monitor mode, we found that frames may be transmitted in a different order compared to their injection from user space programs. We observed this behavior on all tested network cards. A code inspection of the Linux kernel revealed that frames may get reordered based on their QoS TID. This may cause Wi-Fi tools to misbehave and give incorrect results. For instance, in some of the implementation-specific FragAttacks vulnerabilities, a device is only vulnerable *before* the 4-way handshake has completed. To test whether a device is vulnerable, a (plaintext) frame is sent during the handshake by the user space script. Unfortunately, because injected frames may get reordered, in reality the (plaintext) frame may get sent by the network card *after* the handshake completes. This would cause the script to think that a device may not be affected by a vulnerability, while in reality it is.

3.5 Sequence number and fragment number

One of the most blatant issues is that the sequence and fragment number of injected frames was modified by *all* tested network cards when operating in mixed monitor mode. This was caused by kernel the function `ieee80211_tx_h_sequence` in the `mac80211` module, meaning only Soft-MAC drivers are affected. The problem is that this function overwrites the sequence and fragment number if there is a non-monitor interface associated to the network card, regardless of whether the frame being processed is an injected

frame or not. Note that this kernel function does not overwrite the sequence number when the network card is in pure monitor mode.

After fixing the above issue, the WN-722N card, which is managed by the `ath9k_htc` driver, still overwrites the sequence number (but not the fragment number) when in mixed monitor mode. Moreover, this network card even overwrites the sequence number when operating in pure monitor mode. By inspecting the open source firmware code of this card, we identified that the function `ath_tgt_tx_seqno_normal` was overwriting the sequence number (but not the fragment number) of all transmitted frames.

The FragAttack tool bypassed the above issues by patching the TL-WN722N firmware and the Linux kernel to not overwrite non-zero sequence numbers [18]. In Section 4 we generalize these fixes by extending Radiotap, so user space can inform the kernel whether a new sequence number should be assigned or not.

3.6 Hardware decryption

When operating in mixed monitor mode, the non-monitor interface may enable hardware crypto, meaning frames get decrypted in hardware before they arrive at the driver. Unfortunately, with the network cards we tested, the virtual monitor frame then no longer has access to the original encrypted frame when using hardware decryption. Moreover, hardware encryption may also remove the packet number from the frame header (see Figure 1). This is problematic in case access to the packet number (also called nonce or initialization vector) of received encrypted frames is essential. For example, the KRACK scripts require access to the packer number to detect nonce reuse and key reinstallations.

One work-around is to disable hardware decryption. Unfortunately, the driver for the AWUS036ACM has no parameters to disable hardware crypto, even the latest Linux kernel, which at the time of writing was version 6.0. In fact, we found that since kernel 5.8, the ability to disable hardware crypto was also removed for the Intel AC-3160. Because the AWUS036ACM removes the packet number from the frame header during hardware decryption, and this cannot be disabled by its current driver, this network card cannot be used out-of-the-box with the KRACK test scripts.

In Section 4.2 we discuss how we added an option the Linux kernel to disable hardware cryptography in all Soft-MAC drivers.

3.7 Dropped injected frames

When using mixed monitor mode, the Linux kernel drops normal data frames that are injected before a client has finished authenticating. This is due to a bug in the function `ieee80211_tx_dequeue` in the `mac80211` kernel module, meaning this issue only occurs when using Soft-MAC drivers. This behavior is problematic because a user space script expects that injected frames will be transmitted no matter when they are sent.

We also observed that the Intel AC-3160 did not transmit injected frames in mixed monitor mode when the sender address of the injected frame was different from the MAC address of the non-monitor interface. Instead, these injected frames got silently discarded. We also found that the Intel AC-3160 did not correctly inject A-MSDU frames: sometimes they were transmitted in a malformed manner (with two extra random bytes in the middle) and sometimes injected A-MSDU frames were not transmitted at all.

The Intel AC-3160 and RT5572 chipset did not transmit injected frames with the More Fragments (MF) flag set. This could be solved by, after injecting the frame with the MF flag set, immediately injecting a dummy frame *without* the MF flag. With the RT5572, this dummy frame must also have the same QoS TID as the injected frame, but all other fields of the dummy frame do not matter.

3.8 Unexpected Block Ack procedure

When injecting fragmented frames in mixed mode after the client is connected with the AP, the AWUS036ACM starts a Block Ack procedure after injecting the first fragment. This is problematic because any frame transmitted between two fragmented frames may interfere with their reassembly [18]. Initiating the Add Block Ack procedure also caused the receiver to buffer the second fragment, making it appear as if the second fragment never arrived at the receiver. The WN-722N card started a Block Ack before the first fragment and did not send the second fragment, which FragAttacks avoided by making the driver not advertise the `AMPDU_AGGREGATION` feature. Finally, the RT5572 also started a Block Ack procedure, which again interfered with the correct transmission of fragmented frames.

To avoid Block Acks we can disable 802.11n in the non-monitor interface, since Block Acks are part of 802.11n. Unfortunately, this work-around cannot always be used, as some experiments require that the non-monitor interface advertises support for 802.11n. For instance, this injection issue was not noticed during the FragAttack research because 802.11n was disabled by default in all tests. However, disabling 802.11n means that the aggregation vulnerability was initially not detected in OpenBSD [18], because OpenBSD only processes aggregated (A-MSDU) frames when the client negotiated to support 802.11n. Interesting future work would be to patch the Linux kernel to avoid this issue without having to disable 802.11n.

4 IMPROVED WI-FI INJECTION

In this section we discuss how we extended the Radiotap standard to gain more control over how injected Wi-Fi frames are transmitted. We implement our extensions on Linux and discuss how software-defined radio platforms can be used as a more flexible, albeit more costly, alternative to commercial network cards.

4.1 Extending the Radiotap standard

We cannot just update the Linux kernel to implement the desired frame injection behavior because some programs rely on the current injection behavior. For instance, hostap assumes that the Linux kernel overwrites the sequence number of injected frames with a fresh value. To assure backwards-compatibility, we extend Radiotap with two flags that inform the kernel how to transmit injected frames. The first flag, `no-seqnum`, instructs the kernel *not* to overwrite the sequence number. The second flag, `no-reorder`, instructs the kernel not to reorder the injected frame relative to other injected frames that have this flag set. Both new transmission flags have meanwhile been accepted as part of the Radiotap standard.

While extending Radiotap, we noticed that Linux also supports a `noack` flag to indicate that injected frames do not have to be acknowledged. Surprisingly, this flag was not yet part of the Radiotap standard, and during our work we also standardized this flag so other operating systems can implement support for it as well.

4.2 Linux fixes for Wi-Fi injection & reception

We implemented the Radiotap extension in the Linux kernel. This involved parsing the new Radiotap flags and assuring they are respected when processing an injected frame. Among other things, we modified the `ieee80211_tx_h_sequence` function to prevent overwriting the sequence number if the `no-seqnum` flag was set. Additionally, we modified `ieee80211_select_queue_80211` to always use the same transmission queue for injected frames that have the `no-reorder` flag set. This assures that injected frame with this flag set are not reordered relative to each other.

We contributed our Radiotap code to the Linux kernel and it has been included in the official kernel. In particular, the code to prevent sequence number overwrites is part of the Linux kernel since version 5.9, and the code to prevent frame reordering is part of the Linux kernel since version 5.11.

To overcome the problems related to hardware decryption mentioned in Section 3.6, we added an option to the `mac80211` kernel module to disable hardware cryptography for *all* Soft-MAC drivers. When enabled, this forces the driver to disable hardware decryption and encryption, and instead perform these crypto operations in software using the built-in crypto support in `mac80211`. As an example, this change allowed us to use the `mt76` network cards, such as the AWUS036ACM, with the KRACK attack scripts.

Our global parameter to disable hardware crypto did not work with the `iwlwifi` driver of the Intel AC-3160. The problem is that, when disbaling hardware crypto is forced, the `iwlwifi` driver will mistakenly drop non-decrypted frames. We solved this issue by porting older code for `iwlwifi` to disable hardware decryption, which was removed in kernel 5.8 onwards, to the latest long-term-support kernel (version 5.15). With these changes, network cards using the `iwlwifi` driver can again disable hardware crypto.

4.3 Adding support for openwifi

4.3.1 Background. Another disadvantage of commercial off-the-shelf Wi-Fi dongles, besides being closed-source, is that after some years specific models may no longer be available on the market. Additionally, new hardware versions of a Wi-Fi dongle may contain a radio chip of a different vendor, meaning its frame injection and transmission behavior may suddenly change. Overall, relying on off-the-shelf dongles can be troublesome in practice.

A better solution is using open source Wi-Fi implementations that run on Software Defined Radios (SDRs), such as openwifi [12]. Openwifi offers a physical and low MAC layer implementation of Wi-Fi on FPGA and provides an open-source Soft-MAC driver for Linux. From the perspective of a user, it provides an ordinary Wi-Fi network interface, with the advantage that its full implementation is open-source. This means openwifi gives complete control over the transmission behavior of injected frames, while at the same time being compatible with user space Wi-Fi scripts and tools.

Openwifi runs on the ARM processor of the Xilinx Zynq SoC, and the low-layer Wi-Fi functionality, such as sending acknowledgements and (re)transmitting frames, is implemented in Verilog and runs on the FPGA that is part of the same SoC. The latest version of openwifi is based on the Analog Devices Linux distribution called ADI-Linux [4], since it provides the most reliable support for the AD9361 RF fort-end. Openwifi uses version 2019_R1 of ADI-Linux, and this distribution is in turn based on Linaro 14.04, which is a variant of Ubuntu that uses Linux kernel 4.14.0.

4.3.2 Incorporating our Radiotap extensions. To reuse (most of) our existing Linux kernel patches that implement the previously-discussed Radiotap extensions, we first upgraded openwifi to run on a newer ADI-LINUX release, namely version 2021_R1. Doing so also makes it easier to run and test the latest Wi-Fi security tools such as KRACK and FragAttacks. This new release of ADI-Linux is based on Raspberry Pi OS 11.2, which in turn is based on Debian 11.2, and uses Linux kernel 5.10.0. As a result, the new `no-seqnum` Radiotap flag (recall Section 4.2) is supported by default. The patches to support the `no-reorder` flag had to be included manually since they are not yet part of Linux 5.10. During the upgrade to 2021_R1, the openwifi driver had to be adapted for some kernel API changes. The openwifi device tree was also updated according to the new Analog Devices ARM peripheral address space allocation. Finally, the Verilog FPGA code also had to be adapted to be compatible with the new Xilinx Vivado 2021.1 toolsuite. All these changes have been contributed to the public openwifi project.

Thanks to this upgrade of openwifi to a new Linux distribution, we can now directly run our injection tests and Wi-Fi tools on it. For instance, the KRACK and FragAttacks scripts run out-of-the-box on the upgraded openwifi release, and we could reliably test commodity Wi-Fi devices such as the iPhone 13, Dell XPS15 2016, and AR93XX PCIe card. We validated that these scripts correctly injected frames towards these devices in line with our expectations.

5 EVALUATION

In this section, we evaluate the impact of our injection fixes. We do this by testing for vulnerabilities, where our injection fixes enable us to more reliably discover vulnerable devices.

5.1 Tested vulnerabilities

We evaluate our injection improvements by testing whether devices are (still) affected by (variants of) the FragAttacks vulnerabilities [18]. In our tests we focus on attacks where our injection fixes make detection of a vulnerable device more reliable. In particular, we focus on two vulnerabilities:

- Transmission of broadcast frames: we perform tests where broadcast frames are sent to the device under test. Since broadcast frames are not automatically retransmitted, it is essential that the frame is injected when the client is awake. This can be done by using mixed monitor mode such that the Linux kernel will only transmit injected frames once the client is awake (recall Section 3.3).
- Transmission of frames during the handshake: we perform tests where plaintext frames are sent during the handshake. Since handshake messages are typically sent using a different QoS TID priority, it is essential that the injected frame is not reordered relative to the handshake messages, which our injection patches will guarantee.

Since our first test case is aimed towards clients that may enter sleep mode, we focus our tests on smartphones. Concretely, we tested the Huawei Y6 prime, Nexus 5X, Samsung i9305, iPhone XR, iPad Pro 2, OnePlus 6 A6000, and Pixel 4 XL.

We also tested our patches to disable hardware encryption for all Soft-MAC drivers by running the KRACK script. We used the AWUS036ACM to run the KRACK script against our phones.

5.2 Vulnerability test results

We discovered three new vulnerable devices that accept (fragmented) plaintext data frame before the client has authenticated:

OnePlus 6 This device accepts and processes plaintext broadcast data frames while it is still authenticating, which is a new variant of the implementation-specific FragAttack vulnerabilities. Note that to correctly inject broadcast Wi-Fi frames we set the `no-ack` flag in the Radiotap, instructing the sender not to wait for an acknowledgment (and thereby not retransmitting the frame). We reported this vulnerability to the vendor and it was assigned CVE-2021-36196.

Pixel 4 XL and OnePlus 6 Both devices accept plaintext unicast Wi-Fi data frames while it is still authenticating to a network, i.e., during the 4-way handshake. We reported both vulnerabilities to the vendors, and the vulnerability in the OnePlus 6 was assigned the identifier CVE-2021-36197.

Huawei Y6 Prime This device accepts plaintext unicast and broadcast frames both during and after this handshake. As a result, this vulnerability is trivial to exploit.

Finally, only the Samsung i9305 was vulnerable to KRACK. Since this is an old phone that has not received updates after the disclosure of this attack, this is an expected result, and it confirms that our patches to disable hardware decryption are working properly.

6 RELATED WORK

Bellardo et al. modified an iPAQ H3600 containing a Dlink DWL-650 card to be able to inject control frames [1]. Others created an open firmware version for Broadcom network cards, giving researchers a high level of control over transmitted frames [11]. To enable monitor mode and frame injection on Android, researchers modified the closed-source firmware of Broadcom chips [16].

In another work, the `ath9k_htc` driver and firmware was modified to prevent modification to forwarded frames [20]. The same driver has also been modified to acknowledge a range of MAC addresses in order to launch denial-of-service attacks [22]. Users also modified drivers to more easily inject frames in the 5 GHz band [17].

There have been various open implementations of the 802.11 standard on top of software-defined radio platforms. One of the first is an implementation of 802.11b by BBN technologies using GNU Radio [5, 9]. This was extended by the Signal Processing Across Networks (SPAN) lab to improve reliability [5, 7]. In 2010, researchers from Forschungszentrum Telekommunikation Wien (FTW) created an implementation of 802.11p using GNU Radio and a USRP2 [8]. Bloessl et al. implemented 802.11p in GNU Radio in2013 [3]. Finally, Jiao et al. created an open implementation of 802.11 on an System-on-Chip [12], and Nuand created an open implementation of 802.11 that can run on the FPGA of a software defined radio [14]. These projects are mainly for study and experimentation of the 802.11 communication protocols and are rarely used in the context of testing Wi-Fi implementations by injecting unexpected or malformed frames.

7 CONCLUSION

When performing Wi-Fi experiments, it is *essential* to use an additional, independent, Wi-Fi dongle to monitor whether frames are transmitted correctly and as expected. We also encourage vendors to open-source the firmware of network cards, as this would give researchers more control over the network card in experiments.

We created a script to test the correctness of Wi-Fi frame injection which is available online [19]. Our frame injection improvements are now part the Radiotap standard, we implemented them on Linux, and our patches are now part of the Linux kernel since version 5.11. Our changes provide support for a selection of commercial Wi-Fi dongles as well as software defined radio platforms, such as openwifi. By having greater control over how and when frames are injected, we were able to more reliably test wireless cards for vulnerabilities, leading to the discovery of three vulnerable devices.

ACKNOWLEDGMENTS

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

REFERENCES

[1] John Bellardo and Stefan Savage. 2003. 802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions. In *USENIX Security*.
[2] Johannes Berg. 2022. Radiotap: De facto standard for 802.11 frame injection and reception. Retrieved 14 November 2022 from https://www.radiotap.org/.
[3] Bastian Bloessl, Michele Segata, Christoph Sommer, and Falko Dressler. 2013. Towards an Open Source IEEE 802.11 p stack: A full SDR-based transceiver in GNU Radio. In *2013 IEEE Vehicular Networking Conference*. IEEE, 143–149.
[4] Analog Devices. 2022. ADI-linux. Retrieved 17 November 2022 from https://github.com/analogdevicesinc/linux.
[5] Ryan Dobbins, Saul Garcia, and Brian Shaw. 2011. *Software defined radio localization using 802.11-style communications.* Bachelor's Thesis.
[6] Thomas d'Otreppe de Bouvette. 2017. aircrack-ng: iw monitor mode flags. Retrieved 15 November 2022 from https://aircrack-ng.blogspot.com/2017/02/monitor-mode-flags.html.
[7] Hamed Firooz. 2013. SPAN Lab. https://web.archive.org/web/20130412163400/http://span.ece.utah.edu/pmwiki/pmwiki.php?n=Main.80211bReceiver.
[8] Paul Fuxjäger, A Costantini, D Valerio, P Castiglione, G Zacheo, T Zemen, and F Ricciato. 2010. IEEE 802.11p transmission using GNURadio. In *WSR*. 1–4.
[9] GNURadio. 2013. Projects – The Comprehensive GNU Radio Archive Network (CGRAN). Retrieved 14 November 2022 from https://web.archive.org/web/20131108081814/https://www.cgran.org/wiki/Projects.
[10] Andy Green. 2022. How to use packet injection with mac80211. Retrieved 15 Nov. 2022 from kernel.org/doc/html/v6.0/networking/mac80211-injection.html.
[11] Francesco Gringoli and Lorenzo Nava. 2022. OpenFWWF: Open FirmWare for WiFi networks. Retrieved 17 November 2022 from https://web.archive.org/web/20220121031132/http://netweb.ing.unibs.it/openfwwf/.
[12] Xianjun Jiao, Wei Liu, Michael Mehari, Muhammad Aslam, and Ingrid Moerman. 2020. openwifi: a free and open-source IEEE802.11 SDR implementation on SoC. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 1–2.
[13] Jouni Malinen. 2002. Fixed auth/assoc TX callback handling to really use information about whether the transmit was successful or not. Retrieved 15 November 2022 form https://w1.fi/cgit/hostap-history/commit/?id=095f76f19170.
[14] Nuand. 2022. bladeRF-wiphy. Retrieved 17 November 2022 from https://www.nuand.com/bladerf-wiphy/.
[15] Domien Schepers, Mathy Vanhoef, and Aanjhan Ranganathan. 2021. A framework to test and fuzz Wi-Fi devices. In *ACM WiSec*. 368–370.
[16] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. 2017. Nexmon: The C-based Firmware Patching Framework. https://nexmon.org
[17] twisteroidambassador. 2022. linux-ath-user-regd. Retrieved 17 November 2022 from https://github.com/twisteroidambassador/arch-linux-ath-user-regd.
[18] Mathy Vanhoef. 2021. Fragment and Forge: Breaking Wi-Fi Through Frame Aggregation and Fragmentation. In *USENIX Security*. USENIX Association.
[19] Mathy Vanhoef. 2023. Injection test script. github.com/vanhoefm/wifi-injection.
[20] Mathy Vanhoef and Frank Piessens. 2014. Advanced Wi-Fi attacks using commodity hardware. In *ACSAC*. 256–265.
[21] Mathy Vanhoef and Frank Piessens. 2017. Key reinstallation attacks: Forcing nonce reuse in WPA2. In *ACM CCS*.
[22] Mathy Vanhoef and Eyal Ronen. 2020. Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. In *IEEE S&P*. IEEE.