Injection attacks on 802.11n MAC frame aggregation

DOI: 10.1145/2766498.2766513
Handle: http://hdl.handle.net/1942/19047

Injection Attacks on 802.11n MAC Frame Aggregation

Pieter Robyns, Peter Quax, Wim Lamotte
iMinds/tUL/UHasselt
Expertise Centre for Digital Media
Wetenschapspark 2
3590 Diepenbeek, Belgium
{pieter.robyns, peter.quax, wim.lamotte}@uhasselt.be

ABSTRACT

The ability to inject packets into a network is known to be an important tool for attackers: it allows them to exploit or probe for potential vulnerabilities residing on the connected hosts. In this paper, we present a novel practical methodology for injecting arbitrary frames into wireless networks, by using the Packet-In-Packet (PIP) technique to exploit the frame aggregation mechanism introduced in the 802.11n standard. We show how an attacker can apply this methodology over a WAN – without physical proximity to the wireless network and without requiring a wireless interface card. The practical feasibility of our injection method is then demonstrated through a number of proof-of-concept attacks. More specifically, in these proof-of-concepts we illustrate how a host scan can be performed on the network, and how beacon frames can be injected from a remote location. We then both analytically and experimentally estimate the success rate of these attacks in a realistic test setup. Finally, we present several defensive measures that network administrators can put in place in order to prevent exploitation of our frame injection methodology.

Keywords

Wireless security, injection attack, frame aggregation

Categories and Subject Descriptors

C.2.0 [**Computer-communication networks**]: Security and protection.

General Terms

Experimentation, Security

1. INTRODUCTION

Throughout the years, several amendments have been made to the original Institute of Electrical and Electronics Engineers (IEEE) 802.11 standard in order to meet the increasing throughput demands of wireless networks. The first amendment to increase the data rate as the result of improvements on both the Physical (PHY) layer and the Medium Access Control (MAC) layer is 802.11n [13, 20]. Here, a key improvement on the PHY layer is the introduction of Multiple Input, Multiple Output (MIMO) technology, where multiple spatially separated antennas are used on both the receiver and transmitter to create multiple spatial streams [14] and reduce multipath interference. Another addition is the optional usage of an increased channel bandwidth of 40 MHz instead of 20 Mhz [13, 20].

Besides these improvements on the PHY layer, the efficiency of the MAC layer needed to be improved as well in order to go beyond speeds of 100 Mbps. Here, the most notable efficiency improvement is frame aggregation [20], which allows a station to transmit or receive multiple MAC frames in a single PHY frame, reducing overhead due to headers and interframe spacing. All of the improvements on the PHY and MAC layer combined increase the theoretical maximum raw data rate of 802.11n to 600 Mbps [26, 13], which ultimately resulted in 802.11n devices dominating the Wi-Fi market in 2013 [1].

Despite the beneficial effect of PHY and MAC layer improvements on the data rate, these additions introduce new opportunities for security issues to arise. In the case of 802.11n, frame aggregation is especially interesting to look at. Although the standard specifies aggregation principles, frame structures and the general mechanisms, it provides mere guidelines for the actual implementation [14]. It is up to developers to implement their own aggregation schemes that determine when, why and even how to aggregate individual frames [27, 8]. Such schemes are typically implemented in the device driver or on the device firmware.

In this paper, we will explore the frame aggregation mechanism introduced in the 802.11n standard. We will show how the frame aggregation algorithm provided by the 802.11n standard introduces a remote arbitrary frame injection vulnerability on MAC hardware that implements this algorithm. In the remainder of this section, we will detail all of our contributions, and describe several related works. Section 2 describes the properties and implementation of the 802.11n frame aggregation mechanism, which is required knowledge for understanding our attack. Next, in Section 3, we describe the attack itself, along with a feasibility study and a discussion of the results and potential impact. In Section 4 we propose several measures that can be put in place to defend against our attack. Finally, Section 5 describes our conclusions and potential future work.

DOI: http://dx.doi.org/10.1145/2766498.2766513.

1.1 Related work and contributions

Goodspeed et al. introduced the Packet-In-Packet (PIP) technique [12] in 2011, and applied it to IEEE 802.15.4 [10, 12]. Additionally, they used this technique to inject raw PHY layer frames into 802.11b networks for data rates of up to 2 Mbps [11]. Barisani et al. demonstrated in 2013 how, in rare cases, the PIP technique can be applied to 802.3 wired links [2]. In 2014, Jenkins et al. used the PIP technique for fingerprinting 802.14.5 and ZigBee receivers[15]. Similar PIP principles have been used by Dabrowski et al. in order to embed barcodes inside other barcodes [6]. Ossmann et al. propose a defense against PIP in the field of radio communications by using error correcting codes [19]. Finally, Sassaman et al. describe how formal language theoretic methods can be used as a defense against injection attacks in general [25].

Our main contribution is a methodology for remote arbitrary frame injection into wireless networks that implement the 802.11n standard or newer standards such as 802.11ac. This is accomplished by applying PIP principles to the MAC frame aggregation mechanism. To the best of our knowledge, this approach has not been demonstrated before. Previously, the PIP technique could only be used to perform frame injection attacks on 802.11b networks [11] by tricking the radio into interpreting a raw PHY layer packet which is embedded in the payload of another packet. As Goodspeed et al. mentioned in their work, several complications and challenges exist that limit the practical feasibility of this approach:

- The symbol set used in the Physical Layer Convergence Protocol (PLCP) Protocol Data Unit (PPDU) must be included in the symbol set used for the injected payload. Otherwise it will not be possible to inject a valid PHY layer header. For example: if the data portion of the frame is modulated using Complementary Code Keying (CCK), it will be difficult for an attacker to inject a valid PLCP preamble, which is modulated using Differential Binary Phase Shift Keying (DBPSK).
- Even if a favorable symbol set is used, the data rates must be compensated for. The 802.11 standard specifies different data rates for different sections of the frame. For example, the PLCP preamble must always be transmitted using the 1 Mbps DBPSK modulation. The `SIGNAL` field of a PPDU however, can then indicate to use a higher data rate for the remainder of the frame [14].
- 802.11 PLCP Service Data Units (PSDUs) are scrambled using a 127-bit sequence, which must be accounted for by the attacker.
- If the modulation technique uses differential signaling, the attacker must account for this as well.

The methodology presented in our work removes all of these complications by operating on the MAC layer. Consequently, our injection method can be applied regardless of the chosen data rate, symbol set, signal whitening, or modulation technique. Hence, the practical exploitability and attack surface of frame injection attacks is increased. To justify our claim of practical feasibility, we present two realistic proof-of-concept attacks and estimate their success rate. Since this success rate depends on the aggregation behavior of the wireless Network Interface Controller (NIC), this aspect is briefly discussed as well. Finally, we propose several defensive measures which can be applied to prevent exploitation of our attack.

2. BACKGROUND

Before we discuss our injection attack, we will give a brief overview of some of the relevant PHY and MAC layer features introduced in 802.11n. These features have notable implications for Goodspeed's PIP attack, as well as the injection attack that we will discuss in Section 3.

2.1 PHY features

On the PHY layer, two features added in 802.11n have some interesting properties for this research. First, there are the added PLCP frame formats, which have additional fields and hence additional capabilities in comparison to previous versions of the standard. Secondly, there is a set of new Modulation and Coding Schemes (MCSs) for higher data rates which we used throughout our experiments.

2.1.1 PLCP frame format

The PLCP acts as an interface between the PHY and MAC layers. It defines a frame format named the PLCP Protocol Data Unit (PPDU), which consists of a PLCP Preamble, PLCP Header, and a PSDU. In 802.11n, the PPDU can have several formats depending on the capabilities of the transmitting device:

- Non-High Throughput (HT) format: legacy frame format as specified by previous versions of the standard.
- Mixed format: format that is backwards compatible with the 802.11 a/g format.
- Greenfield format: HT frame format that can only be used by devices that are 802.11n compatible.

The fields specified in the PPDU contain the parameters that determine how the device hardware should transmit a packet. Examples of such parameters are the modulation scheme, transmission length, and aggregation flag. Figure 1 shows the generic PLCP frame format. Naturally, a frame transmitted using a certain set of parameters can only be received by a device that supports said parameters. This set of supported parameters is also referred to as the capability set of the device.

2.1.2 MCS

The MCS is determined by the index specified in the `MCS` field of the PPDU. The mapping between indices and the corresponding mandatory rates is given in Table 1. The last column of the table indicates the data rate when a short Guard Interval (GI) is used.

2.2 MAC features

802.11n related extensions of the MAC layer include changes in the frame format and the addition of two types of frame aggregation that were introduced in order to reduce MAC layer overhead: MAC Service Data Unit (MSDU) aggregation or A-MSDU, and MAC Protocol Data Unit (MPDU) aggregation or A-MPDU.

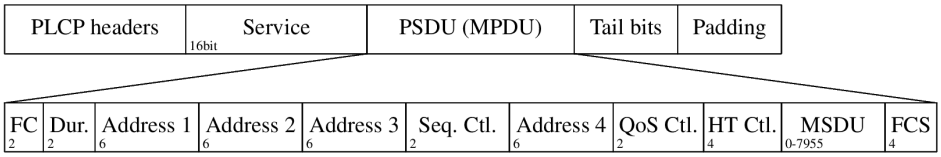


Figure 1: PLCP and MAC frame structures

MCS	Modulation	Coding	Data rate (Mb/s) GI	SGI
0	BPSK	1/2	6.5	7.2
1	QPSK	1/2	13.0	14.4
2	QPSK	3/4	19.5	21.7
3	16-QAM	1/2	26.0	28.9
4	16-QAM	3/4	39.0	43.3
5	64-QAM	2/3	52.0	57.8
6	64-QAM	3/4	58.5	65.0
7	64-QAM	5/6	65.0	72.2

Table 1: MCS parameters for mandatory 20 MHz [14].

2.2.1 MAC frame format

The MAC frame format is defined in the MPDU. Its original frame structure is extended with an optional `HT Control` header field as shown in Figure 1. The presence of this field is indicated by the `Order` subfield of the `Frame Control` field [14]. Other new additions are the HT Capabilities and HT Operation Information Elements (IEs), which are included in the Beacon frames of the Access Point (AP), and in the Probe Requests and (Re)Association Requests of stations [8].

The MPDU of the MAC layer is equivalent to the PSDU of the PHY layer. The higher layer payload of the frame is included in the `MSDU` field of the MPDU.

2.2.2 Aggregate MSDU

With A-MSDU aggregation, the transmitter collects multiple MSDU subframes from the LLC sublayer and prepends them with an A-MSDU subframe header, which is structurally equivalent to an 802.3 header: it contains the Destination Address (DA), Source Address (SA), and Length of the MSDU subframe as shown in Figure 2a.

The MSDU subframes are aggregated and transmitted in a single MPDU when either the maximum A-MSDU size (7935 bytes) is reached, or when the transmission delay reaches a predefined threshold. Each A-MSDU subframe must be followed by a number of zero padding bytes, so that the length is a multiple of 4 bytes. The final A-MSDU subframe has the `A-MSDU Present` flag in the QoS header set to true. Furthermore, the `DA` and `SA` fields of the MSDUs must be identical to respectively the `Receiver Address (RA)` and `Transmitter Address (TA)` fields of the MPDU [14].

A disadvantage of this aggregation method is that it performs poorly in situations where the packet error rate is high [9]. This is due to the fact that there is only a single Cyclic Redundancy Check (CRC) for the entire aggregate frame.

2.2.3 Aggregate MPDU

A-MPDU aggregation collects multiple frames from the MAC sublayer and aggregates them in a single PHY frame of maximum 65,535 bytes. Here, each subframe is prepended with an A-MPDU delimiter as shown in Figure 2b. As with A-MSDU aggregation, each subframe is padded with zero bytes so that its length is a multiple of 4.

For the subframes to be valid, they must have the same `RA` and `Duration` fields, i.e. the receiving host and frame lengths must be the same, and the same `KeyID` field must be provided in case encryption is enabled. The maximum subframe size is 4095 bytes [14].

The A-MPDU delimiter of a subframe is shown in Figure 2b, and contains the following fields:

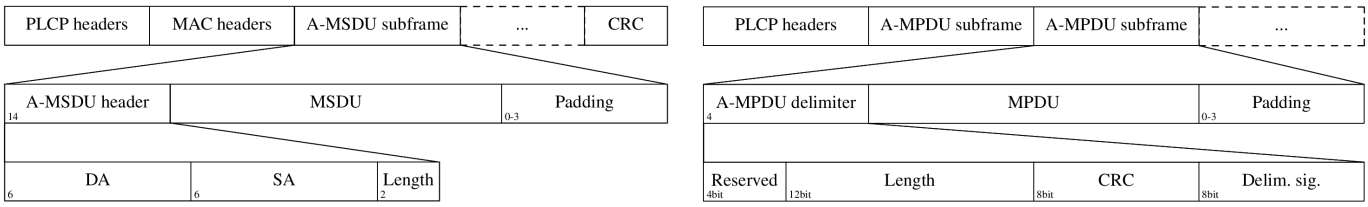
- Reserved: Unused bits with a possible future application.
- Length: Length of the A-MPDU subframe in bytes. This length can be 0 bytes, in which case the A-MPDU delimiter is used for padding purposes.
- CRC: 8-bit CRC of the `Reserved` and `Length` fields.
- Delimiter signature: Pattern that indicates the start of an A-MPDU subframe. This pattern is defined as the ASCII value for the character “N”.

Note that contrary to A-MSDU aggregation, a packet error in one subframe does not result in the entire aggregate frame being dropped. Instead, only the erroneous subframe is dropped. If the subframe error occurred in the A-MPDU delimiter, the 802.11n specification suggests that the next `Delimiter signature` should be searched for according to the algorithm in Listing 1.

```
void parse_mpdu (int length)
{
  int offset = 0;
  while (offset+4 < length)
  {
    if (valid_mpdu_delimiter(offset) &&
      mpdu_len(offset) <= (length - (offset+4)))
    {
      recv_mpdu(offset+4, mpdu_len(offset));
      offset += 4 + 4 * ((mpdu_len(offset)+3)/4);
    }
    else
    {
      offset += 4;
    }
  }
}
```

Listing 1: A-MPDU scanning and parsing algorithm [14, p. 2661]

In essence, this algorithm searches for any A-MPDU delimiter signature on a 4-byte boundary. If a valid delimiter signature has been found, the `recv_mpdu` function will be called. This function will check whether the value specified



(a) A-MSDU frame structure

(b) A-MPDU frame structure

Figure 2: Frame structure of A-MSDUs (a) and A-MPDUs (b).

in the `CRC` field of the A-MPDU delimiter is correct, and if so, strips the delimiter and sends the number of bytes specified in the `Length` field of the A-MPDU delimiter as an individual MPDU up to the MAC protocol driver for further processing. Note that because this algorithm is performed by the hardware MAC component of the chip, it is not possible to observe the A-MPDU in its entierity on the host. Only the individual MPDUs will be visible to the firmware and driver of the device.

Our hypothesis is that the above algorithm can be exploited by an attacker. Observe that since the symbol set and data rate used by the payload data is the same as the symbol set and data rate used by the A-MPDU delimiter, a vulnerability is introduced: if one subframe has an incorrect delimiter, the scanning algorithm will overflow into the payload and parse this payload as if it were header data. An attacker can therefore define their own subframe boundaries by using a specially crafted payload.

3. FRAME INJECTION ATTACK

We have investigated whether our hypothesis is correct and whether the A-MPDU frame aggregation mechanism can be exploited in practice. We found that this is indeed the case and demonstrate a proof-of-concept remote frame injection attack. This attack allows an attacker to inject arbitrary frames into 802.11n networks from any location, by leveraging the PIP technique [12] at the data link layer. For this attack to succeed in practice, a number of conditions must be true:

- The last hop between the attacker and the victim transmits packets wirelessly.
- Encryption is disabled. In other words, the AP is an open network (e.g. hotspot, internet cafe, airport, or other open AP).
- The AP and associated victim are configured for 802.11n operation.

In order to determine whether the network is vulnerable, the first two conditions are trivial to determine. The last condition can be evaluated based on the presence of the HT Capabilities IE (see Section 2.2.1). In all of the following experiments we assume that these conditions are true.

3.1 Experimental Setup

While testing the applications and success rate of our frame injection attack, we used the setup depicted in Figure 3. Here, the attacker's Linux machine is connected to the internet via an Ethernet port. An AP is used to provide internet access for the wireless network to which the

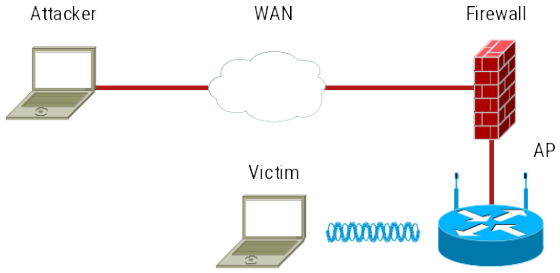


Figure 3: Experimental setup

victim is connected. For our experiments we used four different models: a MikroTik CRS109 (AR9344 chip), a Linksys E1200 (BCM5357C0 chip), a Sitecom WLR-3100 (MediaTek MT7620N chip), and a Linux machine running `hostapd` (AR9271 chip). Each AP uses Network Address Translation (NAT) to translate between internal addresses and its external address, and is protected by a firewall.

Our victim's Linux machine is associated to the AP with a TP-Link TL-WN722N USB dongle, which uses the Atheros AR9271 wireless chip with the default open source `ath9k_htc` firmware[1]. The setup was placed in an office environment without any actively interfering stations.

3.2 Injection Method

We now present our method for performing the frame injection attack. As a first step, the attacker is required to craft a valid A-MPDU delimiter, subframe, and padding as shown in the bottom part of Figure 4. The `Padding` fields are necessary to align the current and any subsequent A-MPDU delimiters to a 4-byte boundary as specified by the standard. Note that the length of the first `Padding` field is dependent on the length of all previous bytes of the frame, and hence on the link layer protocol used from the AP to the victim's machine as well. Based on the knowledge that the used link layer protocol will be 802.11 on the target network, the attacker can calculate the correct padding size. The second `Padding` field can simply be calculated as $4 - (\texttt{mpdu\_len} \bmod 4)$.

Our crafted A-MPDU subframe can now be transmitted towards the victim. The attacker can embed this subframe in any higher layer protocol payload, such as an HTTP request. Ideally, we would like to trigger frame aggregation from the AP to our victim, and inject as many frames as possible. Therefore, a possible approach is to rapidly transmit repeated sequences of the crafted subframe to the victim.

[1]This firmware is available for download at `https://github.com/qca/open-ath9k-htc-firmware`.

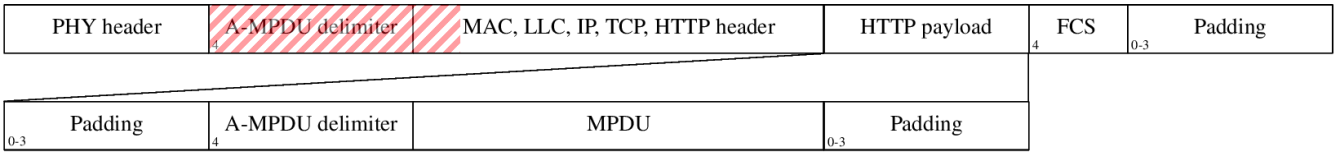


Figure 4: A-MPDU subframe injection

When the payload is transmitted over the wireless link at the last hop, there is a non-negligible probability that part of the aggregated frame becomes corrupted, for example due to frame collisions, transmission errors, or interference from other radio protocols. If the corruption of any subframe is limited to only the A-MPDU delimiter, its `CRC` field will be invalid, and the algorithm from Listing 1 will be performed by the hardware of the wireless NIC in order to recover any following uncorrupted subframes of the aggregated frame. Hence, if a valid A-MPDU subframe delimiter crafted by the attacker is present in the payload at a 4-byte boundary, this data will be interpreted as an A-MPDU delimiter instead of payload data. Figure 4 (top frame) demonstrates an aggregated frame where the A-MPDU delimiter of a subframe is corrupted.

After parsing the attacker's A-MPDU delimiter, the hardware will pass the malicious subframe to the host or device firmware. It should be mentioned that the Frame Check Sequence (FCS) of the injected frame must be correctly calculated by the attacker, or the frame will be dropped due to an invalid FCS at the MAC layer. However, this is trivial since the attacker has complete control over the injected MPDU. A library can be used to calculate the correct CRC over the injected MPDU.

In the work presented by Goodspeed et al.[11], frame injection in 802.11b networks is achieved by embedding an entire PHY frame in a higher layer protocol. Using our methodology however, it is not required to include PHY layer headers in the payload, because the injection happens at the MAC layer. The data rate and scrambler state do not have to be guessed, and the same symbol set is used for the distinction between frame header and frame data in our case. This increases the attack surface and chance of success from a practical point of view. The attacker only needs to make sure that A-MPDU aggregation is performed when the AP transmits the frames to the victim. Exactly when or why frames should be aggregated is not defined in the standard, and therefore depends on the device driver or firmware implementation [8, 27]. In our experience, an efficient method to trigger A-MPDU aggregation is to rapidly transmit small packets of the same size to the AP. This was determined by means of a number of experiments that we will discuss in Section 3.4.

3.3 Applicability

In Section 3 we mentioned that only open wireless networks, such as hotspots and internet cafes, are vulnerable to our injection attack. At the time of writing, this is about 10.61% of the total number of known wireless networks [30]. An attacker would therefore have a reasonable chance of being able to inject packets into a random network if they would probe a random IP address range. Note that this may happen from any location or via a Wide Area Network (WAN) such as the internet.

Device name	**Chipset**
Intel Dual Band Wireless-AC 7260	7260HMW
TP-Link TL-WN722N	AR9271
Netgear WNA1100	AR9271
CastleNet RTL8188CTV	RTL8188CTV
K11 Mini	RT5370
TL-WDN3200	RT5572
Nexus 5	BCM4339
MikroTik CRS109	AR9344
Linksys E1200	BCM5357C0
Sitecom WLR-3100	MT7620N

Table 2: Tested devices vulnerable to A-MPDU subframe injection.

Besides targeting an open network, a second requirement is that the victim implements A-MPDU frame aggregation in a 802.11n standard compliant manner. Since according to a research article by ABI Research 802.11n devices hold the largest market share of the consumer Wi-Fi device shipments in 2013 [1], and since A-MPDU reception support is a mandatory requirement in 802.11n [8], we believe a significant number of devices will be vulnerable to our demonstrated attacks. It should be noted that newer 802.11 standards, such as 802.11ac, implement A-MPDU frame aggregation as well. Therefore, devices that implement these standards will also be vulnerable. Table 2 shows the devices we tested. All of them are indeed vulnerable to our attack.

3.4 Optimal aggregation triggering

The 802.11n standard specifies that the frame size or a timer could be used for determining how many frames should be aggregated [14]. However, in practice it is very difficult to determine exactly which packets will be aggregated and which packets will be sent in the regular fashion, as this depends on the vendor's implementation. Moreover, the attacker has no means of measuring the properties of the wireless link at the remote location, such as the Packet Delivery Ratio (PDR), which can affect aggregation behavior as well [17].

From the attacker's point of view it would be interesting to see whether a generic, optimal transmission speed and frame size can be selected in order to maximize the probability of aggregation at the remote AP, as this will increase the probability of successful injection. We experimentally compared the aggregation behavior of the four APs from our setup with respect to these parameters. In each experiment, we transmitted 250,000 frames from the attacker to

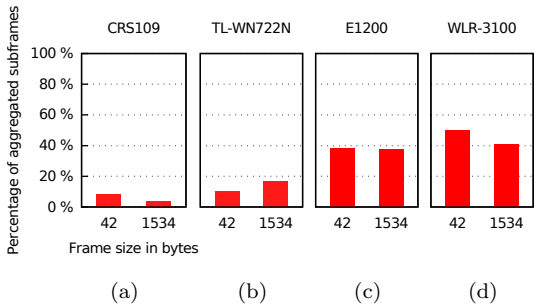


Figure 5: The percentage of aggregated subframes received per 250,000 transmitted frames from attacker to victim in our test setup shows that except for the TL-WN722N (`hostapd`) device, using small frames results in a slightly higher aggregation rate.

the victim, and checked how many of them were forwarded by the AP as A-MPDU subframes.

The MikroTik AP and `hostapd` machine respectively use the AR9344 and AR9271 Atheros Wi-Fi chipsets, which have a number of hardware queues that receive frames from the firmware (AR9271) or driver (AR9344). Both chips aggregate frames depending on the total number of frames currently in these queues [4, 24]. Note that neither a timer is used nor are frames aggregated based on their size. Figure 5a shows the precentage of A-MPDU subframes transmitted by the MikroTik AP, which appears to favor small frames for aggregation.

Figure 5b shows the percentage of A-MPDU subframes received by the `hostapd` AP using the same setup. Despite the fact that this device implements the same aggregation strategy, it is much slower at emptying the transmission queues, which leads to increased aggregation – especially for larger frames. Therefore, the performance of the device itself impacts aggregation as well, and is an additional unknown to the attacker.

Finally, the Linksys AP (Figure 5c) and Sitecom AP (Figure 5d) respectively use the Broadcom and MediaTek chipsets, which appear to use frame aggregation more frequently. Since we observed that in most cases, using small frames results in a higher aggregation rate and since the aggregation rate decreases if the delay between successive transmissions increases, we can conclude that crafting small payloads and transmitting them rapidly is a good strategy.

3.5 A-MSDU injection

In Section 2.2.2, A-MSDU aggregation was briefly discussed. A similar vulnerability in these kind of aggregated frames would allow an attacker to craft their own MSDUs. However, we determined that this aggregation method is not vulnerable to our injection attack.

To see why this is the case, let us consider the A-MSDU header from Figure 2a, which is different from the A-MPDU delimiter. For the attack to succeed, a random bit error must cause the `Length` field to become smaller, so that the attacker's payload is interpreted as the next A-MSDU header. This random `Length` value cannot be predicted by the attacker, which is a first issue that complicates exploitation.

Another consequence of not knowing the corrupted values from the previous fields is that the FCS of the MPDU cannot be calculated beforehand. An incorrect FCS would then automatically lead to the entire frame being dropped, resulting in a failed injection attack.

It should be noted that although the injection of A-MSDUs is unlikely to work in practice because of the above reasons, there is a possible use case for them in context of A-MPDU injection: the 802.11n standard specifies that an A-MSDU can be embedded inside an A-MPDU subframe [14] in order to increase the maximum size of the subframe. An attacker can therefore use this in order to increase the injection payload of a single subframe.

3.6 Attack scenarios

We have proposed a methodology for frame injection that can be used to inject arbitrary frames into a remote network. In this section, we will discuss two practical attacks that can be performed using this methodology. Our implementation is written in Python, uses the Scapy library for crafting the payload, and is open source [21].

3.6.1 Host scan

In our first attack, we apply our frame injection methodology to perform a remote scan for all active hosts on an internal network. We assume that the target network to scan is behind a remote AP, and that at least one service is accessible through this AP. This ensures that our payload will be sent wirelessly over the air. In our case, we configured a web server on the victim of our experimental setup. In practice this can be any service or open port, as long as the last hop between the attacker and the victim is a wireless link. For this experiment, the Sitecom AP was used, having its firewall configured to explicitly only allow HTTP packets and drop ingress ICMP requests.

In the above scenario, an attacker can craft HTTP POST requests containing A-MPDU subframes with ICMP echo requests as the payload, and transmit them to our victim machine. Note that since we are injecting at the MAC layer, the MAC address of the AP must be known[2] and used as the TA, or else the packet will be dropped by the victim. For the RA, the broadcast MAC address can be used.

After several repeated transmissions, we observed that frame corruption indeed caused the inner ICMP packet to be successfully received and replied to by the victim instead of the original payload, successfully bypassing the configured firewall rule. By checking which clients reply, we can then iterate over each destination IP address to perform a full host scan on the target network. A packet trace of this experiment is available at [23]. Here, we were able to scan 52% of a /24 network in 122 seconds using 665 MB of data.

When testing this attack on our Linksys and Mikrotik APs, we noticed that the ICMP replies by the victim were not forwarded to the attacker. The reason is that these APs use stateful firewalls, and hence block outgoing ICMP replies if there is no associated request. Nevertheless, even if stateful firewalls are used, the ICMP requests were successfully injected and interpreted by the victim for all tested APs and receivers.

Naturally, the injection of data frames is not limited to ICMP requests. An attacker can choose any frame in order to perform other attacks such as a Denial of Service (DoS) attack, ARP poisoning, or the injection of 802.11 manage-

[2]This could be done by looking up the Service Set Identifier (SSID) or location of the AP in a wardriving database such as WiGLE.net [30]

ment frames such as Beacons and Deauthentication frames. For each of the *data frame* related attacks to succeed, the attacker would need to know the Basic Service Set Identification (BSSID) to which the targeted host is associated. This is essentially the MAC address of the AP's wireless interface. If the BSSID is incorrect, the targeted host will drop the frame, which makes random attacks against an IP range infeasible. However, a determined attacker could look up the BSSID in a wardriving database such as WiGLE.net[30], and perform a targeted attack against a specific BSSID.

3.6.2 Beacon injection

Section 3.6.1 gave an example of how our frame injection attack can be applied if the attacker has access to a service on the internal wireless network. However, in practice this scenario is not very prevalent. It would therefore be more interesting for an attacker to set up their own service, and lure the victim into accessing it. As an example, the attacker can set up a web server and serve a binary file containing malicious frames, which would be automatically downloaded by anyone who visits the web page. As with our previous attack, these frames will occasionally be interpreted by the victim due to interference on the wireless link.

The above scenario was tested using the same experimental setup from Figure 3. We set up a web server on the attacker's machine and created a `jpg` image containing Beacon frames with a specific SSID as the A-MPDU subframe payload, though any other type of payload could have been used. Despite the fact that sending management frames inside an A-MPDU is not standard compliant [14], the frames were accepted by the victim machine upon injection. A packet trace of this experiment is available at [22]. The trace shows a first injected beacon frame after 47 seconds and 16 MB of transmitted data.

Contrary to our previous attack from Section 3.6.1, the attacker would need no prior knowledge of the AP's MAC address, since for Beacon frames the DA is always `ff:ff:ff:ff:ff:ff` and the TA can be spoofed. The impact of such attack can range from rather harmless, such as displaying a message in a remote victim's SSID list, to injecting a beacon frame with a malformed SSID. Such malformed frames could trigger a buffer overflow vulnerability on the receiving host[3].

3.7 Success rate

To determine whether our attack would be feasible in practice, we would like to calculate the probability of successful injection. An injection is considered successful when at least one subframe crafted by the attacker is received without errors by our victim. Here we present both an analytical approximation and experimental measurement of this probability.

3.7.1 Analytical approximation

Observe that the event of a successful injection occurs when any previous A-MPDU subframe delimiter of the aggregate becomes corrupted due to interference, as this will trigger the A-MPDU parsing algorithm. We assume that both the probability of aggregation p_a and probability of a single frame corruption p_c are known constants. After all, a significant amount of related work for analytically determining the probability p_c already exists [3, 31, 17, 5], which is outside the scope of this paper.

Both of the input variables p_c and p_a can be modeled using Bernouilli trials. The probability that a received frame is corrupted by interference can be written as $p_c = 1-q_c$, where q_c is the probability that a single frame is received correctly. If the frame is part of an aggregate, and becomes corrupted after transmission, we can distinguish between three cases:

1. The PLCP header of the A-MPDU was corrupted. In this case the entire A-MPDU would be dropped and the receiver would interpret neither legitimate nor injected frames.

2. Any of the A-MPDU delimiter bytes were corrupted. This means that the `Length` field of the current subframe will be ignored, and that the A-MPDU parsing algorithm from Listing 1 will be performed to search for the next valid A-MPDU delimiter.

3. Part of the subframe MPDU was corrupted. Here, the subframe in question will be passed to the MAC layer, where it will be dropped because of an incorrect FCS.

Let us now assume for the sake of clarity that each byte in the A-MPDU of length L_a has the same probability of becoming corrupted. Given a corrupted frame and that the PLCP header is 6 bytes in length, the probability that the corruption occurred in the PLCP header is $\frac{6}{L_a}$. By applying Bayes' rule we get $p_c \cdot \frac{6}{L_a}$ as the unconditional probability. We would like to express this probability per subframe instead of per aggregate, so the probability should be divided by the number of subframes. This results in the following final probability $\frac{p_c \cdot \frac{6}{L_a}}{L_a/L_s}$.

In our second case, for any corrupt subframe of length L_s where $4 \leq L_s \leq L_a$, the probability that the A-MPDU subframe delimiter is corrupted is $\frac{4}{L_s}$. Analogous to our previous case, the unconditional probability becomes $p_c \cdot \frac{4}{L_s}$. For an entire aggregate, the probability that at least one delimiter becomes corrupted is $p_c \cdot \frac{L_a}{L_s} \cdot \frac{4}{L_s}$.

Finally, we do not need to consider the third case, since it does not matter whether part of the MPDU is corrupted or not: corruption of the A-MPDU delimiter is sufficient to trigger the subframe delimiter signature scanning algorithm and consequently, inject a subframe. For the victim to interpret the payload provided by an attacker, the injected subframe itself must not be corrupted, which has the probability $1 - p_c$.

Putting everything together, the probability of injecting a frame p_i and the number of successful injections per A-MPDU n_i become:

$$p_i \approx p_a \cdot (p_c \cdot \frac{4}{L_s}) \cdot (1 - \frac{p_c \cdot \frac{6}{L_a}}{L_a/L_s}) \cdot (1 - p_c) \qquad (1)$$

$$n_i \approx p_a \cdot (p_c \cdot \frac{4L_a}{L_s^2}) \cdot (1 - \frac{p_c \cdot \frac{6}{L_a}}{L_a/L_s}) \cdot (1 - p_c) \qquad (2)$$

A network administrator can use this model to approximate the success rate of our attack on their network. The

[3] An example is the heap-based buffer-overflow vulnerability found in NetGear WG311v1 wireless devices, which allows attackers to execute arbitrary code in kernel mode on the host [16].

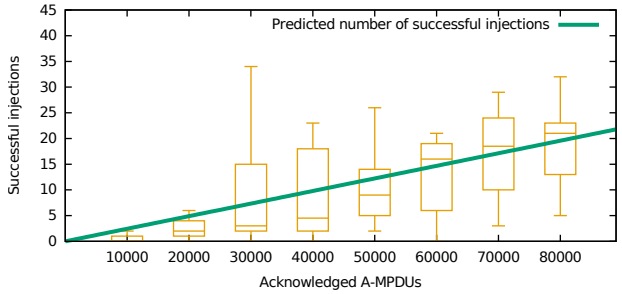


Figure 6: Repeated measurements of the malicious download attack with different file sizes in our experimental setup show that the number of successful injections versus the number of A-MPDUs acknowledged by the receiver is overestimated by our analytical model when using the parameters $p_c = 0.08$, $p_a = 0.03$, $L_a = 65535$, and $L_s = 1538$, as indicated by the green line.

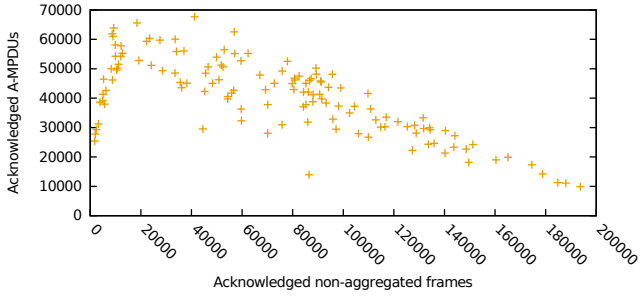


Figure 7: Experimental measurement of the number of acknowledged A-MPDUs versus the number of acknowledged regular frames during a malicious download attack. Each point in the data set represents a download of a 300 MB malicious `jpg` file.

accuracy of the model depends on the choice of the parameters p_a and p_c. As an example, these input variables will be approximated for our experimental setup in the following section.

3.7.2 Experimental measurement

To experimentally determine the success rate for our setup, we repeatedly performed our Beacon frame injection attack from Section 3.6.2 and compared the number of successful injections with the number of acknowledged A-MPDUs. Figure 6 shows the results of these measurements for 285 malicious `jpg` file downloads of sizes between 30 MB and 500 MB via the MikroTik AP.

In our analytical approach from Section 3.7.1, we defined the frame aggregation and corruption probability as two input variables for our model. If we would like to predict the number of successful injections in a realistic environment, these probabilities need to be measured, which is not trivial as they can vary significantly between different trials of the same experiment.

As an example, Figure 7 shows the variations of the aggregation rate over different trials, for 300 MB of transmitted data frames via the MikroTik AP. It is this variation in the aggregation rate which "smears out" the x-axis of our measurement of the number of successful injections from Figure 6. Thus, for a fixed number of transmitted data frames, the number of received A-MPDUs can vary greatly between different measurements.

The second input variable, the frame corruption probability, naturally varies between different trials due to external factors such as contending stations and other radio sources. Instances of poor channel conditions can lead to spikes in the number of successful injections, such as the outliers that can be seen in the boxplot of Figure 6 for 30,000 acknowledged A-MPDUs on the x-axis.

Despite the aforementioned random fluctuations, we can see that the median number of successful injections increases with the number of received A-MPDUs as expected. If we would like to predict the number of successful injections per A-MPDU, our analytical approximation can be used to fit the experimental data.

Suppose we want to estimate the number of A-MPDUs to transmit in order to have at least one successful injection in our experimental setup, given $L_a = 65535$ and $L_s = 1538$. The first step is then to determine the input variables for our model, p_c and p_a.

For determining p_c, it is easier to calculate $1 - q_c$ instead, where q_c is the probability of receiving a packet *without* errors. This probability can be determined by measuring the PDR at the receiver for a given time t. Previous work by Vlavianos et al. has shown that this is a reliable metric to measure the link quality of a wireless network [28]. De Couto et al. define this metric as:

$$\mathrm{PDR}(t) = \frac{\mathrm{count}(t - w, w)}{w/\tau} \tag{3}$$

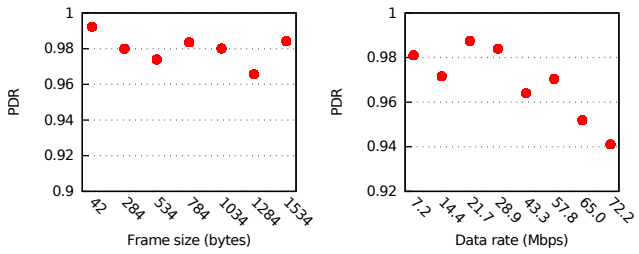
where $\mathrm{count}(t - w, w)$ is the number of correctly received frames, excluding retransmissions, at the receiver and w/τ the total number of transmitted packets [7].

The PDR depends on several variables such as the amount of interference, transmit power of the sender, distance to the receiver, the data rate, and the frame size. Since the attacker only has control over the frame size, we would like to measure the impact of this variable on the PDR.

Figure 8a shows the mean PDR for 50 repeated measurements per given frame size, with 50,000 packets sent to the receiver in each iteration and a distance of 1 meter between the AP (MikroTik) and the receiver.

From previous work, we know that decreasing the packet size increases the PDR [28]. In practice, the results can vary between measurements. For example, observe in Figure 8a that the measurements with a frame size of 1534 bytes have about the same mean PDR as the measurements with a frame size of 42. Such variations can be caused by the rate controller of the AP, which lowers the data rate in case of a suboptimal PDR. Figure 8b shows the impact of the rate controller. Here, the mean PDR was calculated based on 50 repeated measurements with 10,000 frames of size 1534, transmitted at a fixed data rate. Indeed, the PDR is higher when the data rate is low.

Since the mean PDR was equal to 0.92 with the rate controller enabled during our attack, we will use the value 0.08 as p_c in our model. Next, we need to determine the aggre-



(a) (b)

Figure 8: The mean PDR for our test setup generally decreases as the frame size increases (a), but displays some variations. Such variations can be introduced by the rate controller, which can improve the PDR by lowering the data rate (b).

gation probability p_a. We measured that this probability is 0.03 for $L_s = 1534$.

We now have all required inputs for our model. For an A-MPDU length of 65535 bytes, a subframe length of 1538 bytes including padding and A-MPDU delimiter, aggregation probability of 0.03 and PDR of 0.92, $n_i \approx 0.00025$. This means that on average, one injection will be successful per 4065 transmitted A-MPDUs. The prediction is plotted as a green line in Figure 6, and appears to slightly overestimate the actual measured number of successful injections; given our assumptions and selected parameters.

4. DEFENSIVE MEASURES

We have demonstrated in Section 3.6 how an attacker can use our injection methodology to perform several attacks. We will now suggest a number of defensive measures that network administrators and vendors can put in place in order to mitigate these kinds of injection attacks. Since not all of these measures are equally feasible, we will provide a comparison as well in Section 4.7.

4.1 Encryption

A simple solution for defending against our injection attack is to make use of MAC layer encryption such as WPA2-AES. This method does not require modifications to the firmware, driver or 802.11n standard, and is easy to implement.

When both encryption and A-MPDU aggregation are used, each MPDU of the aggregate frame will be encrypted seperately. Therefore, if A-MPDU delimiter corruption occurs, the delimiter signature scanning algorithm will only see the encrypted payload, which is evidently not equal to the payload created by the attacker. Note that even if the attacker knows the master encryption key, it will not be possible to inject frames from a remote location, since a unique key pair is derived from the master key for each individual session. This session key is not known to the attacker.

4.2 Disable A-MPDU frame aggregation

Another effective defensive measure that can be implemented by network administrators is disabling A-MPDU frame aggregation. However, there are several disadvantages that should be considered. Firstly, connected clients can no longer benefit from the increased data rate provided by A-MPDU frame aggregation. A-MSDU aggregation could be used as a less efficient [26, 9] alternative.

A second disadvantage is that, depending on the device, this method may require modifications to the firmware or driver of the NIC. For example, on Linux systems the `mac80211` protocol driver needs to be modified to clear the `IEEE80211_HW_AMPDU_AGGREGATION` flag. Performing such changes on all APs of a network is practically infeasible, especially if the drivers or firmware are proprietary.

4.3 Drop corrupted A-MPDUs

Some chipsets, such as the AR9271, set a certain register flag when a subframe delimiter error has occurred. The firmware or device driver can be modified to drop the entire A-MPDU in case this flag is set, similar to how an A-MSDU would be dropped in case of an error. However, this would cause an impact on performance that depends on the probability that an error will occur in any A-MPDU delimiter.

4.4 LangSec stacks

As mentioned by [10], LangSec stacks can provide a robust defence against PIP attacks. These network stacks essentially use formal language theoretic methods for input validation. Previous research by Sassaman et al. [25] shows that by treating inputs to network protocol stacks as simple to parse input languages, the security can be improved or even guaranteed. An example that is frequently referred to in this research is Structured Query Language (SQL) injection, where treating the user input as executable code can completely mitigate injection attacks.

In context of LangSec, we use the notation M for a message, $\mathcal{D}$ for a decoding function, and $\mathcal{E}$ for an encoding function. The 802.11n aggregation mechanism can now be described as $\mathcal{D}(\mathcal{E}(M))$, where M is a list of MPDUs, $\mathcal{E}$ is the aggregation process or the addition of the A-MPDU delimiters, and $\mathcal{D}$ is the deaggregation process or the removal of the A-MPDU delimiters. Naturally, the intended behaviour by the designers of the frame aggregation mechanism is that $\mathcal{D}(\mathcal{E}(M)) = M$, or in other words that list of MPDUs is not altered after the aggregation process. However, the introduction of random noise on the wireless channel then gives us the probability that $\mathcal{E}(M)$ is altered and hence, that $\mathcal{D}(\mathcal{E}(M)) \neq M$ which introduces a frame injection vulnerability as we saw earlier.

From a language theoretic point of view, an effective defensive measure against our injection method would be to design a recognizer that can parse a frame unambiguously with minimal computational overhead. For MAC frames sent over a wireless link, this is harder to accomplish compared to SQL query input, because if regarded a language, a frame has a much more complex grammar than a SQL query. Furthermore, SQL query input is usually already protected from transmission errors by underlying layers such as TCP. Conversely, MAC frames are sent over unreliable channels. In case of corruption, a frame header or delimiter can become indistinguishable from the frame data, since the same modulation and encoding is used for the entire aggregate, and since boundaries are defined by `Length` fields which may become corrupted as well. A valid MPDU M_2 can then exist so that $M_2 = \mathcal{E}(M_1)$, and consequently, $\mathcal{D}(M_2)$ becomes a valid operation.

One solution to this problem comes in the form of an encoding technique that facilitates unambiguous encapsulation, which was proposed by Ossmann et al. This technique, named Isolated Complementary Binary Linear Block Codes (ICBLBC), uses codes of a certain Hamming distance, for example (5,2,2) codes, with an additional unique "isolation" property. Such codes can be divided into two groups, where each codeword in one group is isolated from any codeword of the other group by a Hamming distance of 3. The first group of codewords can then be used for the header, and the second for the payload. In this example, the Hamming distance between the two groups ensures that up to two bit errors can exist without breaking the isolation between header and payload [10, 19, 18]. Though we believe this method is an effective defence, it is difficult to implement, as the hardware of the wireless NICs needs to be modified to implement the new coding scheme. Such modifications also need to be uniformly implemented by all 802.11 devices, and therefore a standard amendment is required as well. Finally, the overhead of the extra isolation bit in the code would lead to a slight impact on the performance.

4.5 Modulation switch

A different technique that can be used to achieve isolation between header and payload is switching between modulation schemes. Similar to how the modulation scheme switches from DBPSK in the PLCP preamble to Quadrature Amplitude Modulation (QAM) in the MPDU, the A-MPDU delimiter could be transmitted using a different modulation scheme than the subframe itself. Given that no symbols from the header modulation scheme can be created by using the payload modulation scheme, it will be impossible to inject payload data that can be interpreted as a header.

This approach is the least practical, since it would require changes in the standard and costly hardware modifications. Additionally, backwards compatibility with existing devices would be lost.

4.6 Deep packet inspection

Since a large number of packets typically need to be transmitted by the attacker before injection can be successful, a spike in network activity might be flagged by an Intrusion Detection System (IDS) as unusual, and the attacker could be blocked consequently. However, in our experience, transmitting a large number of frames is not *always* necessary for the attack to succeed. Moreover, in some networks, high loads of traffic might be commonplace.

A more effective approach would be to peform deep packet inspection. The IDS can look for payloads that resemble 802.11 headers and drop those packets. Disadvantages of this method are that expensive hardware is often required, and that the processing required to validate packets could introduce latency.

4.7 Comparison

Table 3 summarizes and compares all of the proposed defensive measures in terms of advantages. In this table, the filled portion of a circle denotes the presence of the advantage. A transparent circle means that the advantage is absent.

From the table we can derive that using encryption is the simplest and most effective measure. The only disadvantage is that users will have to provide some type of credential, for example a username / password combination or secret key, before the network can be joined. It should be mentioned however, that this disadvantage can be removed by deploying a technology such as Wi-Fi Passpoint [29].

	Standard compliant	No hw. mod. required	No driver or fw. mod. required	Negligible throughput loss	Low cost	Supports open networks	Backwards compatible
Encryption	●	●	●	●	●	○	●
Disable A-MPDU aggregation	●	●	○	○	●	●	●
Drop corrupted A-MPDUs	●	●	○	○	●	●	●
Langsec (ICBLBCs)	○	○	●	●	◕	●	○
Modulation switch	○	○	●	●	○	●	○
Deep packet inspection	●	●	●	●	○	●	●

Table 3: Comparison between the proposed defensive measures.

5. CONCLUSIONS AND FUTURE WORK

We have demonstrated a novel frame injection attack that can be remotely performed against networks which support the A-MPDU frame aggregation mechanism introduced in the 802.11n standard. Additionally, we have shown two example attack scenarios that use our injection method. We then demonstrated that the success rate of this attack depends on the frame corruption probability or link quality of the target network, and on the probability that a frame is aggregated between the last hop and the victim host. We have experimentally determined that for maximizing the probability that a frame becomes corrupted, large frames should be transmitted at a high data rate, whereas for maximizing the aggregation probability, small frames transmitted at a high data rate yield better results. Finally, several defensive measures that can be applied to mitigate the attack were described and compared.

This paper mainly focused on the general principles of injection attacks on aggregation mechanisms. In future work, the behavior of new devices besides the ones discussed can be analysed and checked for similar or new vulnerabilities. For example, notable differences between aggregation schemes and sensitivity to our injection attack could be used for fingerprinting devices from different vendors. Since the injection can be performed remotely, such tests can be done on a very large scale, for example by probing networks connected to the internet. Another useful application can be found in the domain of WIDS evasion: because a frame can be interpreted differently by two identical receivers upon injection, this technique can be used to bypass existing WIDS systems or even trigger a false alarm.

6. ACKNOWLEDGEMENTS

We would like to thank Arno Barzan, Bram Bonné, Oleksij Rempel, Michael Ossmann, Sujith Manoharan, and the anonymous reviewers for their insightful comments, suggestions, and discussions regarding this work.

7. REFERENCES

[1] ABI Research. 161 Million Consumer Wi-Fi Access Points Shipped in 2013; 802.11ac Sales Rapidly Accelerating, 2015 (accessed). https://www.abiresearch.com/press/1391-million-consumer-wi-fi-access-\points-shipped-.

[2] A. Barisani and D. Bianco. Fully arbitrary 802.3 packet injection: maximizing the Ethernet attack surface. *BlackHat USA, August*, 2013. http://dev.inversepath.com/download/802.3/whitepaper.txt.

[3] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *Selected Areas in Communications, IEEE Journal on*, 18(3):535–547, 2000.

[4] A. Chadd. Atheros ath9k transmit path documentation, 2015 (accessed). https://github.com/erikarn/ath9k-docs/blob/master/ath9k-xmit.txt.

[5] P. Chatzimisios, A. C. Boucouvalas, and V. Vitsas. Performance analysis of IEEE 802.11 DCF in presence of transmission errors. In *Communications, 2004 IEEE International Conference on*, volume 7, pages 3854–3858. IEEE, 2004.

[6] A. Dabrowski, K. Krombholz, J. Ullrich, and E. R. Weippl. QR Inception: Barcode-in-Barcode Attacks. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, pages 3–10. ACM, 2014.

[7] D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4):419–434, 2005.

[8] M. S. Gast. *802.11n: A Survival Guide*. O'Reilly, 2012.

[9] B. Ginzburg and A. Kesselman. Performance analysis of A-MPDU and A-MSDU aggregation in IEEE 802.11 n. In *Sarnoff symposium*, pages 1–5. IEEE, 2007.

[10] T. Goodspeed. Phantom Boundaries and Cross-layer Illusions in 802.15. 4 Digital Radio. 2014.

[11] T. Goodspeed and S. Bratus. 802.11 Packets in Packets, A Standard Compliant Exploit of Layer 1. In *28th Chaos Communications Congress*, pages 1–60, 2011.

[12] T. Goodspeed, S. Bratus, R. Melgares, R. Shapiro, and R. Speers. Packets in Packets: Orson Welles' In-Band Signaling Attacks for Modern Radios. In *WOOT*, pages 54–61, 2011.

[13] G. R. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X. P. Costa, and B. Walke. The IEEE 802.11 universe. *Communications Magazine, IEEE*, 48(1):62–70, 2010.

[14] IEEE Computer Society. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE 802.11 Standard, IEEE, September 2012.

[15] I. R. Jenkins, R. Shapiro, S. Bratus, T. Goodspeed, R. Speers, and D. Dowd. Speaking the Local Dialect: Exploiting differences between IEEE 802.15. 4 Receivers with Commodity Radios for fingerprinting, targeted attacks, and WIDS evasion. In *Proc. of the ACM Conf. on Security and Privacy in Wireless and Mobile Networks (WiSec 2014)*, pages 63–68, 2014.

[16] Laurent Butti. NetGear WG311v1 Wireless Driver 2.3.1 - 10 SSID Heap Buffer Overflow Vulnerability, 2015 (accessed). http://www.exploit-db.com/exploits/29167/.

[17] Y. Lin and V. W. Wong. WSN01-1: frame aggregation and optimal frame size adaptation for IEEE 802.11 n WLANs. In *Global telecommunications conference, 2006.*, pages 1–6. IEEE, 2006.

[18] M. Ossmann. Unambiguous Encapsulation, 2013. https://www.mail-archive.com/langsec-discuss@mail.langsec.org/msg00000.html.

[19] M. Ossmann and D. Spill. Unambiguous Encapsulation - Separating Data and Signaling. *Great Scott Gadgets Technical Report 2014-03-1*, 2014.

[20] E. Perahia. IEEE 802.11 n development: history, process, and technology. *Communications Magazine, IEEE*, 46(7):48–55, 2008.

[21] Pieter Robyns. MAC frame aggregation injection implementation, April 2015. https://github.com/rpp0/aggr-inject.

[22] Pieter Robyns. Packet trace of the Beacon injection experiment, April 2015. http://research.edm.uhasselt.be/~probyns/traces/beacon_inj.tar.gz.

[23] Pieter Robyns. Packet trace of the remote host scan experiment, April 2015. http://research.edm.uhasselt.be/~probyns/traces/inj_host_scan.tar.gz.tar.gz.

[24] Qualcomm Atheros. Atheros ath9k_htc transmit path documentation, 2015 (accessed). https://github.com/qca/open-ath9k-htc-firmware/blob/master/target_firmware/wlan/if_owl.c#L1343.

[25] L. Sassaman, M. L. Patterson, S. Bratus, and M. E. Locasto. Security applications of formal language theory. *Systems Journal, IEEE*, 7(3):489–500, 2013.

[26] D. Skordoulis, Q. Ni, H.-H. Chen, A. P. Stephens, C. Liu, and A. Jamalipour. IEEE 802.11 n MAC frame aggregation mechanisms for next-generation high-throughput WLANs. *Wireless Communications, IEEE*, 15(1):40–47, 2008.

[27] The Linux Foundation. TX A-MPDU aggregation, 2015 (accessed). https://www.kernel.org/doc/htmldocs/80211/aggregation.html.

[28] A. Vlavianos, L. K. Law, I. Broustis, S. V. Krishnamurthy, and M. Faloutsos. Assessing link quality in IEEE 802.11 wireless networks: Which is the right metric? In *Personal, Indoor and Mobile Radio Communications*, pages 1–6, 2008.

[29] Wi-Fi Alliance. Wi-Fi Certified Passpoint, 2015 (accessed). http://www.wi-fi.org/discover-wi-fi/wi-fi-certified-passpoint.

[30] Wireless Geographic Logging Database. Wi-Fi network statistics, 2015 (accessed). https://wigle.net/stats.

[31] J. Yeo and A. Agrawala. Packet error model for the IEEE 802.11 MAC protocol. In *Personal, Indoor and Mobile Radio Communications, 2003.*, volume 2, pages 1722–1726. IEEE, 2003.