Protecting Wi-Fi Beacons from Outsider Forgeries

Mathy Vanhoef
New York University Abu Dhabi
mathy.vanhoef@nyu.edu

Prasant Adhikari
New York University
prasant.adhikari@nyu.edu

Christina Pöpper
New York University Abu Dhabi
christina.poepper@nyu.edu

ABSTRACT

All Wi-Fi networks periodically broadcast beacons to announce their presence to nearby clients. These beacons contain various properties of the network, including dynamic information to manage the behavior of clients. We first show that an adversary can forge beacons to carry out various known as well as novel attacks. Motivated by these attacks, we propose a scheme to authenticate beacon frames that is efficient and has low bandwidth overhead. We evaluate the security properties of this scheme, and discuss its current implementation in Linux. By collaborating with industry partners, our scheme also got incorporated into the draft 802.11 standard, increasing the chance of it being implemented by vendors.

CCS CONCEPTS

• **Networks → Mobile and wireless security**.

KEYWORDS

pre-authentication, 802.11, beacon, spoofing, denial-of-service

ACM Reference Format:
Mathy Vanhoef, Prasant Adhikari, and Christina Pöpper. 2020. Protecting Wi-Fi Beacons from Outsider Forgeries. In *13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '20), July 8–10, 2020, Linz (Virtual Event), Austria.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3395351.3399442

1 INTRODUCTION

Securing Wi-Fi has become increasingly important due to its wide adoption in both mundane and critical settings. However, recent attacks, such as key reinstallations (KRACKs) against WPA2 and vulnerabilities in the new WPA3 standard, show that securing Wi-Fi continues to be a challenge [15, 16]. In this paper, we explore a remaining weak spot in Wi-Fi, namely pre-authentication traffic, and focus on the risk of unprotected beacons which are transmitted before (and after) a security association is established between the client and access point. More importantly, we present an extension to the 802.11 standard that prevents our newly discovered attacks.

In Wi-Fi networks, beacons are periodically transmitted by all Access Points (APs) to announce the presence of the network to nearby clients. Unfortunately, these frames are unprotected and can be spoofed by an adversary. In this paper, we survey related work for attacks that rely on spoofed beacons, and we systematically analyze both the standard and implementations for new attacks.

The new attacks we discovered include implementation-specific ones and attacks that work against all Wi-Fi clients. First, we found several techniques to reduce the throughput of clients. For example, we demonstrate an attack where an adversary spoofs beacon frames that cause the target to lower their transmission power. This makes the network connection unusable for the target. Additionally, we discuss a way to force a target into delaying its transmissions when other stations are using the channel. This reduces the throughput of the targeted station to almost zero, giving other stations an unfair share of the available airtime. We also discuss attacks such as battery depletion and attacks that prevent a target from reliably receiving broadcast and multicast frames.

To defend against our presented attacks, we design a scheme that prevents outsiders from forging beacon frames. Our proposed solution extends the Broadcast Integrity Protocol (BIP) that is used to protect robust management frames. Simplified, we authenticate beacons using a symmetric key that is generated by the AP, and this key is securely transported to clients when connecting to the network. Once connected, all beacons can be verified. To defend against cases where a client receives spoofed beacons before connecting, the client has to store a single reference beacon received before connecting such that it can be verified when the key is received.

As our defense does not enable clients to verify beacons before being connected, we also present a method to report rogue APs. Specifically, authenticated clients can send a notification frame to the real AP when they detect forged beacons. Network administrators can then take appropriate actions if a rogue AP is detected, e. g., they can use trilateration to determine the position of the rogue AP and physically remove it.

To summarize, our main contributions are:

- We study the risk of unprotected beacon frames and how they can be abused by an adversary (Section 3).
- We propose a backwards-compatible extension to the 802.11 standard to authenticate beacon frames, including a mechanism to let clients inform the AP of rouge APs (Section 4).
- We implement our proposed extension and evaluate its security properties (Section 5).

Additionally, we introduce the 802.11 standard in Section 2, discuss related work in Section 6, and finally conclude in Section 7.

2 BACKGROUND

In this section we introduce the 802.11 standard and cover essential concepts used throughout the paper.

2.1 Beacons and Information Elements (IEs)

Beacon frames start with a header that contains, among other things, the sender address of the frame. The header is followed by fixed parameters (see Figure 1 on page 4), the most important for us being

WiSec '20, July 8–10, 2020, Linz (Virtual Event), Austria

ACM ISBN 978-1-4503-8006-5/20/07…$15.00
https://doi.org/10.1145/3395351.3399442

the timestamp and interval. The timestamp contains the precise transmission time of the beacon, and the interval defines when the next beacon will be sent. The beacon then contains a dynamic number of Information Elements (IEs). An IE starts with its type (i. e., element ID), followed by its length and content. If a receiver does not recognize the type of an IE, it will ignore the IE, and proceed with processing the next IE.

2.2 Channel Access Mechanism

To determine when to transmit, the 802.11 standard relies on Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA). This means before transmitting, stations sense whether the medium is idle. Such waiting periods are called Interframe Spacings (IFSs). If the medium is idle, the device can transmit instantly. Otherwise, it must wait until the medium is idle, after which it picks a random number within the backoff window, and waits a corresponding amount of time. If, during the backoff window, no one else started transmitting, the device itself can transmit. In 802.11, the backoff window is defined by the interval $[\mathrm{CW}_{\mathrm{min}}, \mathrm{CW}_{\mathrm{max}}]$.

2.3 Power-Save (PS) Mode

To lower energy usage, clients can enter a sleep state where they do not transmit or receive frames. Before doing so, the client informs the AP it will go to sleep such that the AP will buffer unicast frames for the client. Sleeping clients periodically wake up to receive beacon frames to check if frames are buffered. In particular, each beacon frame contains a Traffic Indication Map (TIM) that lists for which clients unicast frames are buffered. To request buffered frames, the client can poll the AP. Group-addressed frames, i. e., broadcast or multicast frames, are sent at every Delivery TIM (DTIM) interval immediately after the beacon frame. Therefore, to receive group-addressed frames, clients must wake up at every DTIM interval.

2.4 Protected Management Frames (PMF)

With WPA3 the usage of Protected Management Frames (PMF) is mandatory. This feature encrypts and authenticates management frames. For example, it encrypts and authenticates deauthentication frames, which prevents trivial Denial-of-Service (DoS) attacks. Additionally, group-addressed robust management frames are protected using the Broadcast Integrity Protocol (BIP) protocol. Unfortunately, beacons are not considered robust frames and are unprotected.

2.5 Operating Channel Validation (OCV)

The draft 802.11 standard adopted operating channel validation to prevent channel-based Man-in-the-Middle (MitM) attacks [3, 9]. In this attack, the adversary clones the network on another channel by spoofing beacons, forces the victim into connecting to the network on this different channel, and then forwards frames to the real AP. This results in a MitM position where the adversary cannot decrypt traffic, but can reliably manipulate, delay, or block frames. In practice, operating channel validation can be enabled at build time in hostap since version 2.8, which was released in April 2019.

3 ABUSING UNPROTECTED BEACONS

In this section we present several existing and novel attacks that are performed by forging Wi-Fi beacons.

3.1 Methodology

3.1.1 Attack Discovery. To evaluate how an adversary can abuse unprotected beacon frames, we first surveyed related work for known attacks. Then we investigated all information elements that can be included in beacons according to the 802.11 standard and studied whether they can be abused in practice. Finally, we audited the Linux kernel for new attacks, and we inspected the widely-used user-space hostap daemon. These code audits assure we also detect the impact of vendor-specific extensions to the standard. Overall, this revealed practical and theoretic DoS attacks.

3.1.2 Beacon Injection. Our attacks are performed by forging beacon frames. Since beacons are by default broadcasted, all associated clients are attacked simultaneously. However, we discovered that it is also possible to send beacons with a unicast receiver address, meaning we can target specific clients. This technique worked against all tested devices and operating systems. Moreover, this technique increases the chance that the victim receives the forged beacon frame. This is because unicast are automatically retransmitted by our Wi-Fi dongle until the receiver acknowledges the frame. These automatic retransmissions are important if the target device is in sleep mode and may otherwise miss injected frames.

To further increase the chance that devices in sleep mode receive the injected beacon, we extended ModWifi with the ability to inject frames immediately after a legitimate beacon [14]. This is accomplished by disabling backoff and reducing interframe spacings of Atheros dongles, detecting the transmission of the real beacon while it is still in the air [14, §4.2], to then immediately transmit the forged beacon after the real one. We found that in practice our tested device are indeed still awake immediately after the real beacon, and hence properly receive the injected beacon.

3.1.3 Tested Devices. All attacks were tested against Windows, Linux, Android, macOS, and iOS. We tested Linux 5.4.6 and Windows 10 using the following Wi-Fi chips: Intel AC8265, TP-Link TL-WN722N, ZyXEL NWD6505, and an Alfa AWUS051NH. We also tested a Nexus 5X, Samsung i9305, iPad Pro, iPhone Xr, and MacBook Pro, using their default builtin Wi-Fi chips.

3.2 Silencing Stations

We found two techniques that can be used to silence stations. The first is the known quiet attack of Konings et al., which abuses forged beacons to make clients (temporarily) pause transmissions [7]. The second attack is a novel technique that we discovered while auditing the standard and the Linux kernel, and which can be used to make clients lower their transmission power. To test the impact of these attacks on connectivity, we used iperf3 to measure the throughput between the victim and the router.

3.2.1 Quiet Attack. The quiet information element of 802.11h can be used to request clients to stop transmitting for a specified amount of time [7]. This feature was defined so 5 GHz APs can temporarily silence stations to more accurately detect nearby weather radars. Konings et al. found that several devices were vulnerable to this attack. However, none of the devices we tested were vulnerable. Against the ZyXEL dongle on Windows 10 injecting a quiet element caused the connection to become unstable, but the client kept transmitting frames. Our observations match the recent experiments

of Brakel, who tested 6 clients in the 2.4 GHz band, and found that the impact of the quiet attack is negligible [12]. Therefore, we conjecture that nowadays most vendors defend against this attack.

3.2.2 Power Constraints. The 802.11 standard contains a mechanism that allows the AP to inform clients about the maximum transmit power. First, APs can use the country IE to inform clients about the regulatory maximum transmit power of the current channel. Second, APs can use the power constraint IE to further reduce the transmit power below the regulatory maximum. We will refer to both these elements as 802.11 power elements. Concretely, the transmit power used by the client is the maximum transmit power defined in the country element minus the value contained in the power constraint element. On Linux, if the result of this subtraction is a negative number, the transmission limit is set to zero decibels.

The Linux kernel also supports the vendor-specific Dynamic Transmit Power Control information element of Cisco. This element contains a signed byte that defines the maximum transmission power in decibels. In contrast to the power constraint element of 802.11, this vendor-specific element can be used to force a Linux client into using a negative transmission limit (in decibels).

None of the above IEs affected our Android, iPhone, and Windows devices when connected to either a 2.4 or 5 GHz network. However, against our iPad and MacBook, spoofing the 802.11 power elements caused the target to limit their transmission power. Depending on their distance with the AP, the AP will no longer receive frames sent by the target. Against Linux, all IEs had an effect. The 802.11 power elements caused the target to lower transmission power, while spoofing the Cisco element could even be used to kill the connection with the AP. Interestingly, in all other cases vulnerable clients remain connected to the AP, since they still receive beacons from the AP. However, to the user it seems as if there is no connection, because the client's frames no longer reach the AP.

3.3 Targeted Unfairness

The Wi-Fi Multimedia (WMM) certification of the Wi-Fi Alliance is a subset of 802.11e with as core feature Enhanced Distributed Channel Access (EDCA). Summarized, EDCA enables prioritized access to the wireless medium, for example, it can be used to prioritize voice traffic over background internet traffic. This is accomplished by letting IFS times depend on type of data frame (recall Section 2.2). This frame-dependent IFS value is called the Arbitration IFS (AIFS) in EDCA. Additionally, the initial and maximum backoff window $[\mathrm{CW}_{\min}, \mathrm{CW}_{\max}]$ also depend on the type of frame. In practice, APs inform clients about the IFS and backoff window parameters using the WMM element in beacon frames. This enables APs to change traffic prioritization rules to adapt to changes in traffic load.

An adversary can forge the WMM element to make clients use malicious AIFS and backoff window values. In this novel attack, we make the victim use maximal AIFS and backoff values, meaning AIFS equals 15, and $\mathrm{CW}_{\min} = \mathrm{CW}_{\max} = 2^{15} - 1$. After forging a beacon, the adversary has to assure that a beacon from the legitimate AP does not reset the values back to normal. To accomplish this, we abuse the update field in the WMM element. This field is incremented by the AP whenever the EDCA parameters change, allowing clients to quickly check if any parameters have changed. In our attack we capture the current update count of the real AP. Then we inject our malicious EDCA parameters a first time using an incremented count, and then a second time using the original update count of the AP. This assures that when the client receives real beacon frames again, it will not reset the EDCA parameters back to normal, since the update count was the same as before.

Our experiments show that Linux is vulnerable with all the Wi-Fi dongles we tested. Similarly, our MacBook Pro, iPhone, and iPad were also vulnerable. Windows was affected when using our Alfa and TP-Link dongle, but not when using the Intel and ZyXel chips. Finally, our Nexus 5X was not affected, while our older Samsung i9305 was. This shows that the Wi-Fi chip being used influences whether a device is affected. The precise impact of the attack depends on the target, and the number of nearby clients. In general, if the target is the only active client, its throughput as measured by iperf3 is significantly lowered. If other clients are active, the throughput of a vulnerable target can drop to almost zero.

Finally, we remark that the Multi-User (MU) EDCA element of 802.11ax is similar to 802.11e's EDCA element. We conjecture that against new devices this element can be abused in similar ways.

3.4 Battery Depletion Attacks

In another new attack we discovered, the adversary forges beacons with a malicious TIM element to make all clients believe the AP is buffering unicast data frames for them. As a result, clients will poll the AP for buffered frames. However, the AP has no data to send, meaning the clients needlessly transmit a poll frame, and stay awake longer than necessary. Since sending the frame and staying aware longer takes energy, repeatedly performing this novel attack will drain the battery of resource-constrained devices.

To confirm the practicality of our new attack, we use our extended ModWifi framework to inject a forged beacon immediately after the legitimate one. The forged beacon contains a modified TIM element that indicates the AP is buffering unicast frames for all associated clients. We found that with all tested devices, this causes the target to poll the AP for buffered frames. This confirms that spoofed beacons can induce victims into transmitting unnecessary frames, which can be abused to drain the battery of devices.

3.5 Preventing Frame Delivery

In 2003, Bellardo et al. argued that an adversary can spoof timestamp values and Traffic Indication Map (TIM) elements inside beacons to hinder the delivery of broadcast and unicast and broadcast frames [2]. These two known attacks work as follows:

3.5.1 Timing Information Forgery. An adversary can spoof the timestamp parameter, which is used by clients to determine when the next beacon frame will be sent. By modifying this value, clients in sleep mode will wake up at the wrong time and will no longer receive the legitimate beacons [2]. Additionally, because the AP sends group-addressed frames immediately after sending a beacon, targeted clients will also no longer reliably receive broadcast and multicast frames.

3.5.2 TIM Forgery. An adversary can forge TIM elements to make it appear that the AP never has any buffered data frames for clients. This prevents clients from receiving data when they are in sleep mode, since they will no longer poll the AP for buffered frames [2].

3.6 Channel Switch Announcements (CSAs)

Previous works have shown that channel switch announcements can be abused as a DoS and as a technique to obtain a channel-based man-in-the-middle position [7, 13]. A channel switch announcement is an IE that is included in beacons, probe responses, and action frames, to indicate that the AP is about to change channels. For example, when the AP detects a radar pulse, it is required to switch to another channel to avoid interference with weather radars. Clients are informed of this channel switch using a CSA. An adversary can abuse this to forge a beacon frame with a CSA element, causing associated clients to switch to another channel. This causes the targeted client to lose the connection with the AP. Even if the client eventually switches back to the old channel when the AP is not found on the new channel, continuously forging channel switch announcements effectively disconnects the client.

When replicating this known attack, our Linux and Apple devices were vulnerable. Against Windows, the attack only worked when the target used an Intel AC8265 radio. Both our Android phones were not affected. Additionally, we discovered that against Linux including an invalid channel number in the CSA element causes it to instantly deauthenticate from the AP. Finally, against Linux it also works against clients on the 2.4 GHz band.

3.7 Partial Man-in-the-Middle Attacks

In a MitM attack, an adversary intercepts frames before they arrive at their destination, allowing the adversary to delay, modify, or block frames before forwarding them to their destination. In Wi-Fi, a known instantiation of this attack is a channel-based MitM position. Although this attack can be prevented by using Operating Channel Validation (OCV) [13], we found it is still possible to obtain a partial MitM even when using OCV. In our attack where we spoof beacons, we can only intercept frames that are sent by a client to an AP, and those frames must be sent using physical-layer features that the AP does not support. More concretely, by inspecting the Linux kernel, we discovered the following two partial MitM techniques:

3.7.1 Bandwidth Changes. The Linux kernel tracks the maximum bandwidth currently supported by an AP by inspecting the High Throughput (HT), Very HT (VHT), High Efficiency (HE), and Operating Mode Notification elements of beacons. By spoofing these elements, an adversary can make clients transmit frames using a bandwidth that the AP does not support. The adversary can then intercept such frames without the AP receiving them, subsequently manipulate these frames, and finally forward them to the real AP by transmitting them using a lower bandwidth.

Interestingly, OCV securely verifies the supported bandwidth when connecting to the network. However, because the client's user space daemon that implements OCV is not aware of bandwidth changes made by the kernel, our partial MitM will not be detected.

Finally, during our tests to confirm that spoofed beacons influence the maximum bandwidth a Linux client may use, we discovered that sending a HT or VHT element with invalid information causes the client to disconnect from the network. This can be abused to forcibly disconnect clients, even when PMF is enabled.

3.7.2 Preamble Support. We conjecture that the ERP element of 802.11g can also be abused. It contains a field indicating whether the AP supports receiving frames using a short Barker preamble. By spoofing this field, we can make clients use a short preamble even though old APs may not yet support it. We conjecture that other physical-layer parameters, such as the number of supported spatial streams, can be abused in similar ways.

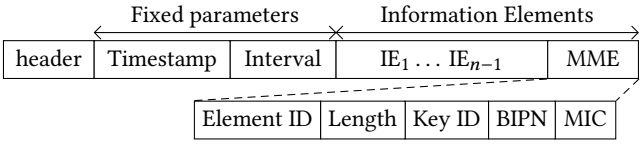

Figure 1: Beacon with a Management MIC Element (MME).

4 BEACON PROTECTION

We now present a scheme to protect beacons against outsider forgeries, and devise a method to let clients inform the real AP if spoofed beacons, i. e., a rogue AP, has been detected. Together with industry partners we created a specification of this scheme [4], which has been adopted by the IEEE into the draft of the 802.11 standard [10].

4.1 Design Requirements

Our defense affects the transmission and reception of all beacons, and these frames are by default sent every 102.4 milliseconds at low bitrates. Hence the cryptographic operations must be efficient, especially considering the resource-constrained nature of certain Wi-Fi devices. The resulting data overhead must be low to minimize the increased airtime usage of beacons. Ideally, our solution is also straightforward to implement such that vendors are more likely to adopt it. With this in mind, we chose to use symmetric cryptography and leverage cryptographic primitives that are already implemented in most Wi-Fi devices. Finally, in our solution we focus on infrastructure networks, and we require that the defense is backward compatible with existing Wi-Fi deployments.

4.2 Beacon Protection Defense

To protect beacon frames, we include an extra IE in each beacon that can be used to verify its authenticity. Using an IE provides backward compatibility since old clients ignore unknown elements (recall Section 2.1). Instead of defining a new IE, we reuse the Management Message integrity code Element (MME) of the Broadcast/multicast Integrity Protocol (BIP). The BIP protocol is part of the Protected Management Frames (PMF) amendment that is used to authenticate robust management frames such as deauthentication and disassociation frames. As a result, our defense requires that PMF is supported and enabled. This is a realistic assumption since nowadays all WPA2 and WPA3 devices are required to support PMF.

The layout of the MME element is shown in Figure 1. The Message Integrity Code (MIC) field is an authentication tag calculated by the AP, and depending on the negotiated cipher suite its size is either 8 or 16 bytes and calculated using the CMAC or GMAC algorithm of BIP. The MME also includes a 48-bit counter called the Beacon Integrity Packet Number (BIPN) that is initialized to zero and incremented after every transmitted beacon and hence is unique. When using GMAC, the unique nonce required by this algorithm equals the BIPN. To calculate the MIC, we introduce a

new key called the Beacon Integrity Group Temporal Key (BIGTK) that is randomly generated when the AP is started. This key is distributed to clients when they are connecting and authenticating with the network. As a result, in our solution the authenticity of beacon frames can only be validated after connecting to the AP. Beacon contents received prior to connecting with the AP should be validated after the AP has been successfully authenticated. As a result, before connecting, all beacon content should only be regarded as hints.

The MME appears after all fields that it protects, so it must be the last element in the beacon frame [1, §12.5.4.2]. We exclude the timestamp from the MIC by masking it to zero, because this field is added by the hardware right before sending the frame, meaning its value is not known in advance. Further, even if the timestamp was included in the MIC, an adversary could still use physical-layer attacks to delay the reception of the frame, causing the timestamp to be incorrect at the time of reception. Note that the liveness of beacon frames is still guaranteed by the incremental BIPN inside the MME element. That is, the receiver uses the BIPN to detect replayed beacons, which are silently ignored. Once the client has received the beacon protection key, any beacons without an MME element must also be discarded. Finally, the KeyID field in the MME is used when refreshing the beacon protection key (see Section 4.4).

4.3 Reporting Rogue APs

In our proposed defense, clients can only authenticate beacons after connecting to the AP. This means that initially they must make decisions based on unauthenticated beacon contents. Clients can limit risks associated to this by verifying (old) beacon content upon receiving the beacon protection key. To further limit the impact of an adversary trying to attack clients before connecting to the AP, we also devised a mechanism that allows connected clients to report rogue APs to the legitimate AP. More concretely, if a client detects a beacon without an MME, or with an invalid MME, it can report this to the legitimate AP using a WNM Notification frame. The network administrator can then take appropriate actions to mitigate and, e. g., localize and remove the rouge AP.

4.4 Distributing and Updating Keys

When clients connect to a network, the AP distributes the BIGTK as well as the current BIPN number and key id of the BIGTK. By including the IPN, replays of old beacon frames are prevented after connecting to the AP. The key ID is used by a receiver to determine which of two BIGTK keys must be used to validate received beacon frames. This allows the AP to generate a new BIGTK and distribute it to clients, while using the old BIGTK and associated key id until all clients successfully received the new BIGTK. Once all clients have received the new key, the AP can then start using the new key, and will indicate this change by using the new key id value in the MME (see Figure 1). This enables frequent updates of the beacon protection key, for example whenever a client disconnects, which prevents old clients from abusing the beacon protection key.

In practice the BIGTK and its associated info is included in message 3 of the 4-way handshake, in the FT and FILS handshake, in the group key handshake, and in the WNM Sleep response frame [4]. The AP can refresh the BIGTK using the group key handshake.

4.5 Multiple BSSID Beacon Protection

Special care is needed for beacons that announce the presence of multiple networks. These beacons contain a Multiple BSSID element, introduced by the 802.11v amendment in 2011, that enables an AP to use a single beacon to announce the presence of multiple (virtual) networks. This can for example be used to let a single physical AP advertise both an employee and guest network. In this setup, these different networks use a single beacon frame that is protected by a shared beacon protection key.

In practice, when using the multiple BSSID element to advertise both a private and public network, it may be undesirable to distribute the beacon protection key to clients in the untrusted network. Our defense can differentiate between trusted and non-trusted clients by only sending the beacon protection key to trusted clients. When doing this, only trusted clients will be able to verify the authenticity of beacons. However, this has the advantage that the untrusted clients will not be able to spoof beacons, since they do not possess the beacon protection key. Note that clients in the public network indirectly still receive some protection because the trusted clients can use the WNM notification feature to report spoofed beacons, reducing the risk of attacks to untrusted clients.

4.6 Backward Compatibility

Backward compatibility is provided by using an information element to authenticate beacons. This element will be ignored by old clients. Additionally, when connecting to a network, the AP transports the beacon protection key to the client inside a key descriptor element. Old clients will ignore this extra element, while new clients will recognize this element and treat this as an indication that the AP supports beacon protection.

When the AP supports beacon protection, it will set the beacon protection flag in the extended capabilities element of all beacons and probe responses. This information can be used by clients to decide to which AP to connect. However, when actually connecting to the AP, this field should only be regarded as a hint. This is because before being connected, an adversary can manipulate this flag. A client securely determines whether the AP actually supports beacon protection by seeing if a BIGTK is included in the handshake. Since the field that transports the BIGTK is authenticated in all handshakes, an adversary cannot manipulate this field, and hence is unable to perform a downgrade attack.

5 EVALUATION

In this section we analyze security aspects of our design, present a proof-of-concept implementation, and discuss future work.

5.1 Security Considerations

One security risk in our scheme is that beacons can be forged by insiders. Although this can be avoided by using public key encryption, this would require a public key infrastructure, would mean that vendors need to implement new cryptographic protocols instead of reusing BIP, and more importantly would increase the overhead per beacon. Additionally, we remark that the existing MFP protection of Wi-Fi also does not protect against insider forgeries. To reduce the risk of clients that were previously connected from abusing the beacon protection key, APs can periodically renew this key using

the group key handshake. If desirable, the beacon protection key can even be renewed whenever a client leaves the network.

We excluded the timestamp field from the MIC calculation, because including it introduces strict requirements for hardware implementations. This is because the timestamp field contains the precise time of when the frame was sent by the hardware. Unfortunately, this means an adversary can potentially modify its value. However, because the MME field contains a packet number, an adversary cannot replay old beacons. Additionally, even when we would have protected the timestamp field, a powerful physical-layer adversary would still be able to delay the frame from arriving, meaning the timestamp value would no longer be accurate.

Finally, before connecting, a client must consider all info in beacons as hints that have to be verified later. In practice a client can store a single beacon from which it extracts all information about the network. When receiving the BIGTK, the client can then verify the authenticity of this stored beacon. If its authenticity cannot be verified, the client should abort connecting to the network.

5.2 Implementation

In February 2020, the Linux kernel and user-space hostap daemon started to implement beacon protection.[1] We focus on these implementations, instead of our own, since they will be part of future Linux releases. First, the kernel is extended such that drivers can inform the kernel whether they support beacon protection. Additionally, hostapd has been extended to transport the BIGTK to the client when connecting to a network, and the client was modified to enable beacon protection when a BIGTK is received while connecting to an AP. To use beacon protection, hardware (or firmware) updates to the network card are also needed. This is because beacons are not generated by the operating system, but by the network card. To nevertheless test beacon protection, and confirm it prevents our attacks, we used the mac80211_hwsim driver which provides virtual Wi-Fi interfaces to simulate Wi-Fi networks.

5.3 Future Work

An open question is whether it is also possible to efficiently defend against insider forgeries. A public-key scheme would introduce a larger data overhead to beacons, and also requires more costly computations on the radio chip. An interesting research direction would be to investigate techniques to reduce this overhead.

6 RELATED WORK

Closest to our work is the paper of Martínez et al., who reviewed known attacks relying on spoofed beacons, and presented a method to detect spoofed beacons based on their transmission time [8]. In contrast, we perform a more systematic analysis of how spoofed beacons can be abused, we test attacks in practice, and we discovered several novel practical attacks. Additionally, their proposed defense can be bypassed by controlling when spoofed beacons are sent, while our defense relies on authenticating beacons using shared keys and well-known cryptographic primitives.

Techniques to detect spoofed data frames can often also be used to detect spoofed beacons. For example, directional fingerprints can be used to detect spoofed frames [17], and deviations in the physical signal caused by device-specific hardware imperfections can be used to fingerprint a transmitter and detect spoofed frames [5]. Unfortunately, such defenses require specialized hardware, and its security guarantees against more powerful adversaries are unclear. Other works analyze sequence numbers of frames to detect spoofed ones [6], or analyze the received signal strength [11]. However, an adversary can trivially bypass such defenses if she is aware of them.

[1] See https://w1.fi/cgit/hostap/commit/?id=2d4c78aef718 and the git parent commits for hostap, and https://patchwork.kernel.org/patch/11398093/ for the Linux kernel.

7 CONCLUSION

We discovered several novel attacks that can be performed by forging beacon frames, and presented an extension to the 802.11 standard that prevents outsiders from forging beacons. Our proposed extension to 802.11 is designed to be efficient and keep implementation complexity low, which is achieved by reusing existing functionality of the BIP protocol. This approach played an essential role in getting our defense incorporated into a draft of the 802.11 standard, and in motivating vendors to implement it. We are hopeful that this approach will accelerate the adoption of our defense.

ACKNOWLEDGMENTS

This work is partially supported by the Center for Cyber Security at New York University Abu Dhabi (NYUAD). Mathy Vanhoef holds a Postdoctoral fellowship from the Research Foundation Flanders.

REFERENCES

[1] IEEE Std 802.11. 2016. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Spec.*
[2] John Bellardo and Stefan Savage. 2003. 802.11 denial-of-service attacks: real vulnerabilities and practical solutions. In *USENIX Security*.
[3] Nehru Bhandaru, Thomas Derham, Mathy Vanhoef, and Ido Ouzieli. 2018. Defense against multi-channel MITM attacks via Operating Channel Validation. Retrieved 25 February 2020 from https://mentor.ieee.org/802.11/dcn/17/11-17-1807-12-000m-defense-against-multi-channel-mitm-attacks-via-operating-channel-validation.docx
[4] Nehru Bhandaru, Thomas Derham, Mathy Vanhoef, and Ido Ouzieli. 2019. Beacon Protection - for CID 2116 and CID 2673. Retrieved 16 May 2020 from https://mentor.ieee.org/802.11/dcn/11-19-0314-02-000m-beacon-protection.doc.
[5] Vladimir Brik, Suman Banerjee, Marco Gruteser, and Sangho Oh. 2008. Wireless device identification with radiometric signatures. In *MobiCom*.
[6] Fanglu Guo and Tzi-cker Chiueh. 2005. Sequence number-based MAC address spoof detection. In *Workshop on Recent Advances in Intrusion Detection*.
[7] Bastian Könings, Florian Schaub, Frank Kargl, and Stefan Dietzel. 2009. Channel switch and quiet attack: New DoS attacks exploiting the 802.11 standard. In *LCN*.
[8] Asier Martínez, Urko Zurutuza, Roberto Uribeetxeberria, Miguel Fernández, Jesus Lizarraga, Ainhoa Serna, and Iñaki Vélez. 2008. Beacon frame spoofing attack detection in IEEE 802.11 networks. In *ARES*.
[9] Jon Rosdahl and Mark Hamilton. 2018. Minutes for REVmd - July 2018 - San Diego. Retrieved 30 January 2020 from https://mentor.ieee.org/802.11/
[10] Jon Rosdahl, Michael Montemurro, and Mark Hamilton. 2019. Minutes for REVmd - March 2019 - Vancouver. Retrieved 2 February 2020 from https://mentor.ieee.org/802.11/
[11] Yong Sheng, Keren Tan, Guanling Chen, David Kotz, and Andrew Campbell. 2008. Detecting 802.11 MAC layer spoofing using received signal strength. In *INFOCOM*.
[12] Jakel van Brakel. 2019. Availability analysis of SURFwireless.
[13] Mathy Vanhoef, Nehru Bhandaru, Thomas Derham, Ido Ouzieli, and Frank Piessens. 2018. Operating Channel Validation: Preventing Multi-Channel Man-in-the-Middle Attacks Against Protected Wi-Fi Networks. In *WiSec*.
[14] Mathy Vanhoef and Frank Piessens. 2014. Advanced Wi-Fi attacks using commodity hardware. In *ACSAC*.
[15] Mathy Vanhoef and Frank Piessens. 2017. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. In *CCS*.
[16] Mathy Vanhoef and Eyal Ronen. 2020. Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. In *IEEE Security & Privacy (SP)*. IEEE.
[17] Jie Xiong and Kyle Jamieson. 2010. SecureAngle: improving wireless security using angle-of-arrival information. In *SIGCOMM*.