Abusing Wi-Fi Beacons and Detecting & Preventing Attacks

Mathy Vanhoef, Prasant Adhikari, and Christina Pöpper.
With special thanks to various IEEE members.

Black Hat Webcast, 17 September 2020.
Background: beacons

- Wi-Fi networks use beacons to announce their presence.
- They are sent every ~100 ms by an Access Point.

Contains properties of the network:
- Name of the network
- Supported bitrates (e.g. 11n or 11ac)
- Regulatory constraints (e.g. transmission power)
  - ...

Problem: beacons can be forged by an adversary!
Our contributions

Novel **attacks** abusing beacons

**Defense** to prevent outsider forgeries

**Standardized** as part of 802.11

Defense is being **implemented** by Linux and might become **part of WPA3**
Taking a step back: Wi-Fi security

Focus was protecting data, not beacons:

› WEP, WPA1/2: only includes data frame protection
› WPA3: includes management frame protection
› Operating channel validation: verifies channel info

→ In all cases **beacons remain unprotected**
Beacons are not protected

- Tag: SSID parameter set: cisco
- Tag: Supported Rates 1(B), 2(B), 5.5(B), 11(B), 6, 9, 12, 18, [Mbit/sec]
- Tag: DS Parameter set: Current Channel: 1
- Tag: Traffic Indication Map (TIM): DTIM 0 of 0 bitmap
- Tag: Country Information: Country Code GB, Environment Unknown (0x04)
- Tag: Power Constraint: 3
- Tag: ERP Information
- Tag: Extended Supported Rates 24, 36, 48, 54, [Mbit/sec]
- Tag: QBSS Load Element 802.11e CCA Version
- Tag: RM Enabled Capabilities (5 octets)
- Tag: HT Capabilities (802.11n D1.10)

- Tag: RSN Information
- Tag: Mobility Domain
- Tag: HT Information (802.11n D1.10)
- Tag: Extended Capabilities (10 octets)
- Ext Tag: HE Capabilities (IEEE Std 802.11ax/D3.0)
- Ext Tag: HE Operation (IEEE Std 802.11ax/D3.0)
- Ext Tag: Spatial Reuse Parameter Set

- WPA version & channel: verified when connecting [WiSec’18]
- All other fields can be spoofed by an adversary
Novel Attacks
Power constraint attacks

Beacons contain the maximum allowed transmit power

- Country Info: First Channel Number: 1, Number First Channel Number: 1
  Number of Channels: 13
  Maximum Transmit Power Level: 20dBm
- Tag: Power Constraint: 3
  Tag Number: Power Constraint (32)
  Tag length: 1
  Local Power Constraint: 3

→ Adversary can lower transmission power of victim
Power constraint attacks

Beacons contain the maximum allowed transmit power

Experiments:

› iPad, MacBook, and Linux: lowers transmit power of device
› All other test devices not affected (unknown why)
Power constraint attacks

Beacons contain the maximum allowed transmit power

Vendor-specific power element of Cisco:
› Can also be exploited to lower transmit power of device
› Linux: can be abused to **forcibly disconnect a victim**
   » Normally we cannot set negative transmission limits
   » But with the Cisco power element we can
Power constraint attacks

DEMO!
Lowering a victim’s bandwidth

› Before transmission the medium must be idle:

In use
Lowering a victim’s bandwidth

› Before transmission the medium must be idle:

- In use
- SIFS
Lowering a victim’s bandwidth

› Before transmission the medium must be idle:

- **In use**
- SIFS
- AIFSN
Lowering a victim’s bandwidth

› Before transmission the medium must be idle:

<table>
<thead>
<tr>
<th>In use</th>
<th>SIFS</th>
<th>AIFSN</th>
<th>Backoff (CW)</th>
</tr>
</thead>
</table>
## Lowering a victim’s bandwidth

Before transmission the medium must be idle:

<table>
<thead>
<tr>
<th>In use</th>
<th>SIFS</th>
<th>AIFSN</th>
<th>Backoff (CW)</th>
<th>Packet 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lowering a victim’s bandwidth

› Before transmission the medium must be idle:

- In use
- SIFS
- AIFSN
- Backoff (CW)
- Packet 2

› Beacon contains the duration of these waiting periods:

- Ac Parameters ACI 0 (Best Effort), ACM no
- ACI / AIFSN Field: 0x03
- ECW: 0xa4
  1010 .... = ECW Max: 10
  .... 0100 = ECW Min: 4
- CW Max: 1023
- CW Min: 15
- TXOP Limit: 0
Lowering a victim’s bandwidth

› Before transmission the medium must be idle:

<table>
<thead>
<tr>
<th>In use</th>
<th>SIFS</th>
<th>AIFSN</th>
<th>Backoff (CW)</th>
<th>Packet 2</th>
</tr>
</thead>
</table>

› Spoofing this info causes clients to **delay transmissions**:

<table>
<thead>
<tr>
<th>In use</th>
<th>SIFS</th>
<th>AIFSN</th>
<th>Backoff (CW)</th>
</tr>
</thead>
</table>

› If another device transmits in the meantime, the victim restarts the waiting process & **possibly never transmits**
Lowering a victim’s bandwidth: experiments

Linux is affected with any network card we tested.

Apple devices are affected (Macbook Pro, iPhone, iPad).

Windows is affected depending on network card (e.g. Alfa and TP-Link cards are affected but not Intel ones).

Android is affected depending on the device: Nexus 5X was affected, but not our old Samsung i9305.
Targeted unfairness

DEMO!
MitM Attack
MitM Attack

Adversary forwards frames between both channels
MitM Attack

› Adversary forwards frames between both channels
› This MitM makes other attacks easier (e.g. KRACK)
Other attacks & findings

Partial machine-in-the-middle attack
› Bypasses channel operating validation in Linux

Battery depletion attacks
› Spoof beacons to **make clients stay awake**

**Send beacon as unicast frames** to target specific clients
› Worked against all tested devices
Practical attack considerations

Beacons are by default broadcasted to all clients

- This means we attack all clients simultaneously

  Receiver address: Broadcast (ff:ff:ff:ff:ff:ff:ff)
  Transmitter address: Cisco-Li_82:b2:55 (00:0c:41:82:b2:55)
  BSS Id: Cisco-Li_82:b2:55 (00:0c:41:82:b2:55)

We can also send them as unicast frames to a specific victim:

  Receiver address: Apple_82:36:3a (00:0d:93:82:36:3a)
  Transmitter address: Cisco-Li_82:b2:55 (00:0c:41:82:b2:55)
  BSS Id: Cisco-Li_82:b2:55 (00:0c:41:82:b2:55)
Our Defense
Design goals

Focus on **practicality & simplicity** to encourage adoption

› Cryptographic operations must be efficient
› Bandwidth overhead must be low
  » Beacons are sent at low bitrate and consume significant airtime

**Straightforward to implement**

› Ideally reuse existing crypto primitives of Wi-Fi
Design approach

To achieve our goals, we rely on **symmetric encryption**
› Reuse crypto primitives of management frame protection

We **defend against outsider attacks**
› Adversary doesn’t possess network credentials
› Similar to protection of broadcast Wi-Fi traffic
Beacon protection: new element

We add a **new type-length-value element** to beacons:

<table>
<thead>
<tr>
<th>Element ID</th>
<th>Length</th>
<th>Key ID</th>
<th>Nonce</th>
<th>MIC</th>
</tr>
</thead>
</table>

- Clients that do not recognize this element will ignore it
- Nonce: incremental number to **prevent replay attacks**
- Message Integrity Check: **CMAC or GMAC** over the beacon
  - Existing crypto primitive of management frame protection
  - All WPA3-capable devices already support it
Key management

Key used to generate/verify the authenticity tag?

› AP generates a fresh **beacon protection key** when booting
› **AP always sends the beacon key** when a client connects
  » Older clients will ignore this key
  » New clients will enable beacon protection

→ Adversary can’t manipulate handshake that transports the beacon key, **preventing downgrade attacks**.
Pre-authentication behavior

Periodic beacons

Client cannot verify beacon before connecting (no key!)
Pre-authentication behavior

Periodic beacons

Store 1 beacon as reference & extra info from it
Pre-authentication behavior

Periodic beacons

Store 1 beacon as reference & extra info from it

Connect to network and receive beacon protection key

Verify authenticity reference beacon (disconnect if invalid)
Pre-authentication behavior

1. **Periodic beacons**
2. **Store** 1 beacon as reference & extra info from it
3. **Connect to network and receive** beacon protection key
4. **Verify authenticity** reference beacon (disconnect if invalid)
5. **Send data**
Reporting forged beacons

› Clients can report forged beacons to the AP
› Can now **detect far away rogue APs**

Out of range
Reporting forged beacons

› Clients can report forged beacons to the AP
› Can now **detect far away rouge APs**
Reporting forged beacons

› Clients can report forged beacons to the AP
› Can now detect far away rogue APs
Practical Results
Specification

› Collaborated with industry to standardize our defense (Intel, Broadcom, Qualcomm and Huawei)

› Since March 2019 part of the (draft) IEEE 802.11 standard:

<table>
<thead>
<tr>
<th>March 2019</th>
<th>doc.: IEEE 802.11-19/0314r2</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE P802.11</td>
<td>Wireless LANs</td>
</tr>
<tr>
<td>802.11</td>
<td>Beacon Protection - for CID 2116 and CID 2673</td>
</tr>
<tr>
<td>Date: 2019-03-11</td>
<td></td>
</tr>
</tbody>
</table>
Specification

› Collaborated with industry to standardize our defense (Intel, Broadcom, Qualcomm and Huawei)

› Since March 2019 part of the (draft) IEEE 802.11 standard:

Might become part of WPA3 specification? 😊

In addition, Wi-Fi Alliance has identified the following potential security protocol updates and will review all comments received:

15. Hash-to-element password generation, Client Privacy Mechanisms, Operation Channel Validation, and Beacon protection in IEEE Draft

Source: https://www.wi-fi.org/security-development (July 2020)
Specification

› Collaborated with industry to standardize our defense (Intel, Broadcom, Qualcomm and Huawei)

› Since March 2019 part of the (draft) IEEE 802.11 standard:

Special thanks to:

› Nehru Bhandaru and Thomas Derham (Broadcom)
› Emily Qi and Ido Ouzueli (Intel)
› Jouni Malinen and Menzo Wentink (Qualcomm)
› Yunsong Yang (Huawei)
Implementation

Now being implemented by Linux:
› Kernel: generate and verify authentication tags
› Hostap: manages keys and enables beacon protection
Conclusion

› Prevent outsiders from forging beacons

› Our focus on practicality paid off:
  
  » Defense is now part of the 802.11 standard
  
  » Being implemented by Linux
  
  » Might become part of WPA3?