Injection attacks on 802.11n MAC frame aggregation

Pieter Robyns

universiteit hasselt | EDM
EXPERTISECENTRUM VOOR DIGITALE MEDIA | iMinds
CONNECT.INNOVATE.CREATE

Introduction and motivation

- Vulnerability that allows attacker to remotely inject raw MAC frames into open 802.11n networks
- Based on the packet-in-packet principle
- Allows an attacker to interact with services on the internal network
- Injection on the MAC layer → data frames, control frames or management frames

Goodspeed's packet-in-packet

- Technique introduced at Usenix WOOT '11 by Travis Goodspeed et al. [1]
- Embed complete radio frame (includes PHY) within the body of another frame
- Interference or noise → embedded frame interpreted by receiver
- First applied to 802.15.4 (ZigBee)

Outer frame	Hex	Embedded frame
Preamble	00 00 00 00	
Sync (SFD)	**a7**	
Length	**19**	
Data	01 08 82 ca fe ba be	
	00 00 00 00	Preamble
	a7	Sync (SFD)
	0a 01 08 82 ff ff ff ff	Data
	c9 d1	CRC
CRC	15 e8	

[1] 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.

Goodspeed's packet-in-packet

- Authors indicate several complications [1]:
  - Header and payload need to have the same symbol set
  - Header data rate must be compensated for
  - Example:

Modulation	Hex	Binary
2 Mbps 2-FSK	C0000003	**11**000000000000000000000000000011
1 Mbps 2-FSK	8001	**1** 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

  - Whitening: pseudo-random bits XOR'ed with frame
  - Differential signaling
  - In case of TDMA: inject into correct timeslot

[1] 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.

Goodspeed's packet-in-packet

- 802.11 investigated later [2]:
  - Data rate can change mid-packet: PLCP Preamble data rate vs. frame payload data rate
  - 127 bit PSDU scrambler
- Despite the above, PIP can still be performed for 802.11b at 1 Mbps and 2 Mbps
- Data rates > 2 Mbps are problematic
- Nowadays: 802.11n, 802.11ac, 802.11ad, etc.

[2] 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.

Our contribution

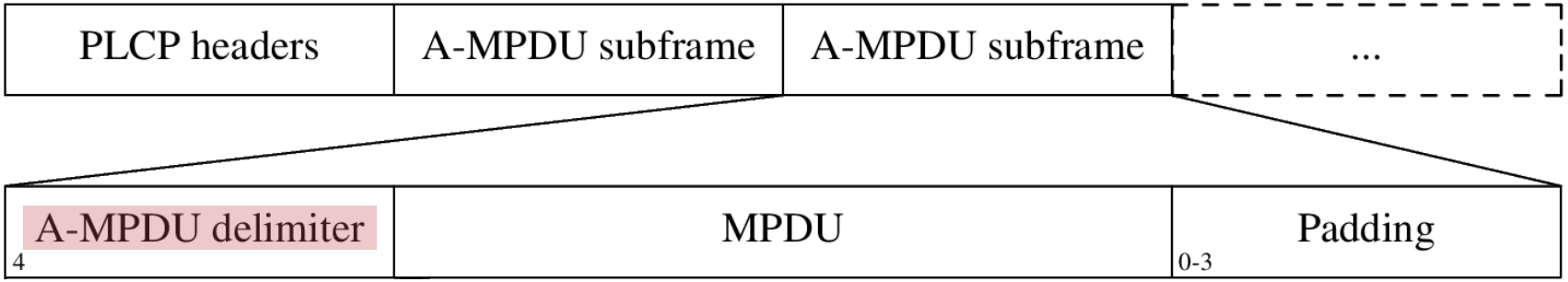
- Discovered a new vulnerability that allows us to perform PIP on the MAC layer instead of the PHY layer
  - All of the aforementioned complications are mitigated
  - Standard compliant; many devices are vulnerable
  - No wireless NIC required
  - No proximity to the 802.11 network required
- Approximation of success rate
- Defensive measure proposals

What is MAC frame aggregation?

- Starting from 802.11n, new features were added to both the PHY the MAC layer
- Goal was to increase throughput
- One if the new features is MAC frame aggregation
- Comes in two flavors:
  - A-MSDU
  - A-MPDU

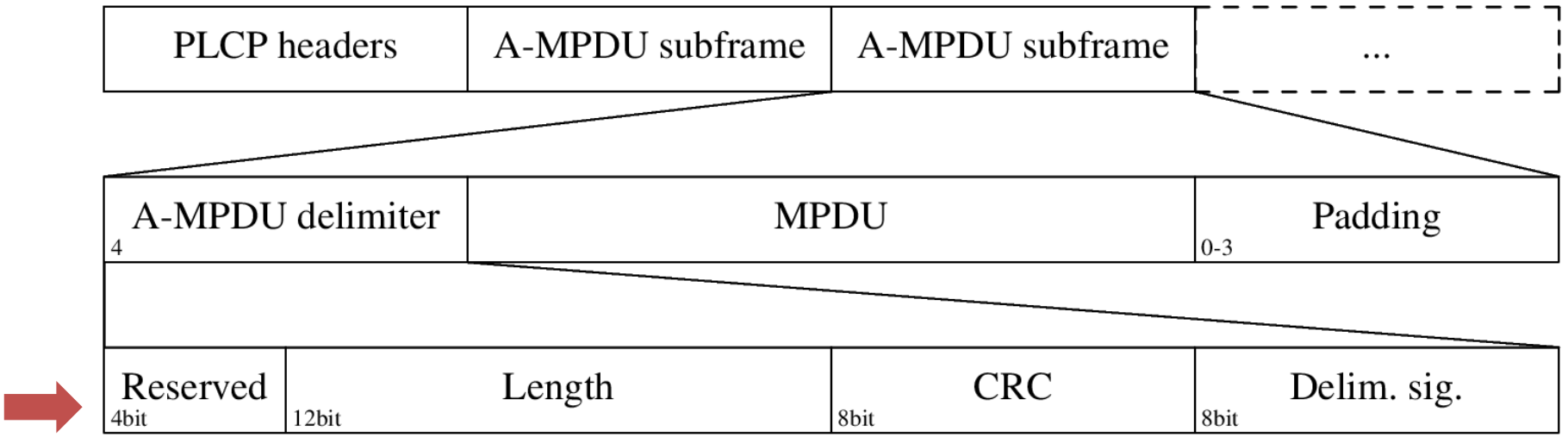
Aggregated MPDU (A-MPDU)

- Aggregates MPDUs from MAC sublayer
- Subframe boundaries defined by A-MPDU delimiter
  - Reserved: unused bits
  - Length: length of the subframe in bytes
  - CRC: 8-bit CRC of Length and Reserved fields
  - Delim. Sig.: the literal ASCII value for 'N'
- One CRC per **subframe (inside MPDU)**



Aggregated MPDU (cont.)

- A closer look at the delimiter itself



Aggregated MPDU (cont.)

- A-MPDU (de)aggregation is performed on the internal chip of the Wi-Fi device
  - Example: Atheros AR9271
- Therefore, completely transparent to the user
- Reason: most likely performance
- Some devices exist that do aggregation on the driver instead of the hardware

A-MPDU deaggregation vulnerability

- Algorithm for deaggregation is specified in the 802.11n standard
- In essence:
  1) Scan for delimiter signature on 4-byte boundary

```
00 00 20 00 00 00 20 00  00 00 20 00 00 00 20 4e  |.. ... ... ... N|
80 04 bb 4e 88 02 00 00  ff ff ff ff ff ff 4c 5e  |...N..........L^|
0c 9e 82 19 4c 5e 0c 9e  82 19 60 30 00 00 aa aa  |....L^....`0....|
03 00 00 00 08 00 45 00  00 22 00 01 00 00 40 01  |......E.."....@.|
57 38 0a 00 00 01 c0 a8  58 f9 08 00 ed f8 00 00  |W8......X.......|
00 fe 58 58 58 58 58 58  13 30 f0 e9 00 00 20 4e  |..XXXXXX.0.... N|
```

A-MPDU deaggregation vulnerability

- Algorithm for deaggregation is specified in the 802.11n standard
- In essence:
  1) Scan for delimiter signature on 4-byte boundary
  2) Check delimiter validity based on 8-bit CRC

```
00 00 20 00 00 00 20 00  00 00 20 00 00 00 20 4e  |.. ... ... ... N|
80 04 bb 4e 88 02 00 00  ff ff ff ff ff ff 4c 5e  |...N..........L^|
0c 9e 82 19 4c 5e 0c 9e  82 19 60 30 00 00 aa aa  |....L^....`0....|
03 00 00 00 08 00 45 00  00 22 00 01 00 00 40 01  |......E.."....@.|
57 38 0a 00 00 01 c0 a8  58 f9 08 00 ed f8 00 00  |W8......X.......|
00 fe 58 58 58 58 58 58  13 30 f0 e9 00 00 20 4e  |..XXXXXX.0.... N|
```

A-MPDU deaggregation vulnerability

- Algorithm for deaggregation is specified in the 802.11n standard
- In essence:
  1) Scan for delimiter signature on 4-byte boundary
  2) Check delimiter validity based on 8-bit CRC
  3) Send "Length" bytes to device driver (here 72 bytes)

```
00 00 20 00 00 00 20 00   00 00 20 00 00 00 20 4e   |.. ... ... ... N|
80 04 bb 4e 88 02 00 00   ff ff ff ff ff ff 4c 5e   |...N..........L^|
0c 9e 82 19 4c 5e 0c 9e   82 19 60 30 00 00 aa aa   |....L^....`0....|
03 00 00 00 08 00 45 00   00 22 00 01 00 00 40 01   |......E.."....@.|
57 38 0a 00 00 01 c0 a8   58 f9 08 00 ed f8 00 00   |W8......X.......|
00 fe 58 58 58 58 58 58   13 30 f0 e9 00 00 20 4e   |..XXXXXX.0.... N|
```

A-MPDU deaggregation vulnerability

- Algorithm for deaggregation is specified in the 802.11n standard
- In essence:
  1) Scan for delimiter signature on 4-byte boundary
  2) Check delimiter validity based on 8-bit CRC
  3) Send "Length" bytes to device driver (here 72 bytes)
  4) Discard padding if needed and repeat

```
00 00 20 00 00 00 20 00  00 00 20 00 00 00 20 4e  |.. ... ... ... N|
80 04 bb 4e 88 02 00 00  ff ff ff ff ff ff 4c 5e  |...N..........L^|
0c 9e 82 19 4c 5e 0c 9e  82 19 60 30 00 00 aa aa  |....L^....`0....|
03 00 00 00 08 00 45 00  00 22 00 01 00 00 40 01  |......E.."....@.|
57 38 0a 00 00 01 c0 a8  58 f9 08 00 ed f8 00 00  |W8......X.......|
00 fe 58 58 58 58 58 58  13 30 f0 e9 00 00 20 4e  |..XXXXXX.0.... N|
```

A-MPDU deaggregation vulnerability

- Algorithm for deaggregation is specified in the 802.11n standard
- In essence:
  1) Scan for delimiter signature on 4-byte boundary
  2) Check delimiter validity based on 8-bit CRC
  3) Send "Length" bytes to device driver (here 72 bytes)
  4) Discard padding if needed and repeat

```
00 00 20 00 00 00 20 00  00 00 20 00 00 00 20 4e  |.. ... ... ... N|
80 04 bb 4e 88 02 00 00  ff ff ff ff ff ff 4c 5e  |...N..........L^|
0c 9e 82 19 4c 5e 0c 9e  82 19 60 30 00 00 aa aa  |....L^....`0....|
03 00 00 00 08 00 45 00  00 22 00 01 00 00 40 01  |......E.."....@.|
57 38 0a 00 00 01 c0 a8  58 f9 08 00 ed f8 00 00  |W8......X.......|
00 fe 58 58 58 58 58 58  13 30 f0 e9 00 00 20 4e  |..XXXXXX.0.... N|
```

- However, what happens if the A-MPDU delimiter is corrupted by noise / interference?

A-MPDU deaggregation vulnerability

- In case of corruption, assumptions about the delimiter context break

Before: the deaggregator skips 72 bytes before searching next delimiter

```
00 00 20 00 00 00 20 00  00 00 20 00 00 00 20 4e  |.. ... ... ... N|
80 04 bb 4e 88 02 00 00  ff ff ff ff ff ff 4c 5e  |...N..........L^|
0c 9e 82 19 4c 5e 0c 9e  82 19 60 30 00 00 aa aa  |....L^....`0....|
03 00 00 00 08 00 45 00  00 22 00 01 00 00 40 01  |......E.."....@.|
57 38 0a 00 00 01 c0 a8  58 f9 08 00 ed f8 00 00  |W8......X.......|
00 fe 58 58 58 58 58 58  13 30 f0 e9 00 00 20 4e  |..XXXXXX.0.... N|
```

A-MPDU deaggregation vulnerability

- In case of corruption, assumptions about the delimiter context break

Before: the deaggregator skips 72 bytes before searching next delimiter

```
00 00 20 00 00 00 20 00   00 00 20 00 00 00 20 4e   |.. ... ... ... N|
80 04 bb 4e 88 02 00 00   ff ff ff ff ff ff 4c 5e   |...N..........L^|
0c 9e 82 19 4c 5e 0c 9e   82 19 60 30 00 00 aa aa   |....L^....`0....|
03 00 00 00 08 00 45 00   00 22 00 01 00 00 40 01   |......E.."....@.|
57 38 0a 00 00 01 c0 a8   58 f9 08 00 ed f8 00 00   |W8......X.......|
00 fe 58 58 58 58 58 58   13 30 f0 e9 00 00 20 4e   |..XXXXXX.0.... N|
```

After: the deaggregator searches next delimiter in the higher level payload!

```
00 00 20 00 00 00 20 00   00 00 20 00 00 00 20 4e   |.. ... ... ... N|
11 11 11 4e 88 02 00 00   ff ff ff ff ff ff 4c 5e   |...N..........L^|
0c 9e 82 19 4c 5e 0c 9e   82 19 60 30 00 00 aa aa   |....L^....`0....|
03 00 00 00 08 00 45 00   00 22 00 01 00 00 40 01   |......E.."....@.|
57 38 0a 00 00 01 c0 a8   58 f9 08 00 ed f8 00 00   |W8......X.......|
00 fe 58 58 58 58 58 58   13 30 f0 e9 00 00 20 4e   |..XXXXXX.0.... N|
```

A-MPDU deaggregation vulnerability

- In case of corruption, assumptions about the delimiter context break

Before: the deaggregator skips 72 bytes before searching next delimiter

```
00 00 20 00 00 00 20 00  00 00 20 00 00 00 20 4e  |.. ... ... ... N|
80 04 bb 4e 88 02 00 00  ff ff ff ff ff ff 4c 5e  |...N..........L^|
0c 9e 82 19 4c 5e 0c 9e  82 19 60 30 00 00 aa aa  |....L^....`0....|
03 00 00 00 08 00 45 00  00 22 00 01 00 00 40 01  |......E.."....@.|
57 38 0a 00 00 01 c0 a8  58 f9 08 00 ed f8 00 00  |W8......X.......|
00 fe 58 58 58 58 58 58  13 30 f0 e9 00 00 20 4e  |..XXXXXX.0.... N|
```

After: the deaggregator searches next delimiter in the higher level payload!

```
00 00 20 00 00 00 20 00  00 00 20 00 00 00 20 4e  |.. ... ... ... N|
11 11 11 4e 88 02 00 00  ff ff ff ff ff ff 4c 5e  |...N..........L^|
0c 9e 82 19 4c 5e 0c 9e  82 19 60 30 00 00 aa aa  |....L^....`0....|
03 00 00 00 08 00 45 00  00 22 00 01 00 00 40 01  |......E.."....@.|
57 38 0a 00 00 01 c0 a8  58 f9 08 00 ed f8 00 00  |W8......X.......|
00 fe 58 58 58 58 58 58  13 30 f0 e9 00 00 20 4e  |..XXXXXX.0.... N|
```

A-MPDU deaggregation vulnerability

- In case of corruption, assumptions about the delimiter context break

Before: the deaggregator skips 72 bytes before searching next delimiter

```
00 00 20 00 00 00 20 00  00 00 20 00 00 00 20 4e  |.. ... ... ... N|
80 04 bb 4e 88 02 00 00  ff ff ff ff ff ff 4c 5e  |...N..........L^|
0c 9e 82 19 4c 5e 0c 9e  82 19 60 30 00 00 aa aa  |....L^....`0....|
03 00 00 00 08 00 45 00  00 22 00 01 00 00 40 01  |......E.."....@.|
57 38 0a 00 00 01 c0 a8  58 f9 08 00 ed f8 00 00  |W8......X.......|
00 fe 58 58 58 58 58 58  13 30 f0 e9 00 00 20 4e  |..XXXXXX.0.... N|
```

After: the deaggregator searches next delimiter in the higher level payload!

```
00 00 20 00 00 00 20 00  00 00 20 00 00 00 20 4e  |.. ... ... ... N|
11 11 11 4e 88 02 00 00  ff ff ff ff ff ff 4c 5e  |...N..........L^|
0c 9e 82 19 4c 5e 0c 9e  82 19 60 30 00 00 aa aa  |....L^....`0....|
03 00 00 00 08 00 45 00  00 22 00 01 00 00 40 01  |......E.."....@.|
57 38 0a 00 00 01 c0 a8  58 f9 08 00 ed f8 00 00  |W8......X.......|
00 fe 58 58 58 58 58 58  13 30 f0 e9 00 00 20 4e  |..XXXXXX.0.... N|
```

A-MPDU deaggregation vulnerability

- In case of corruption, assumptions about the delimiter context break

Before: the deaggregator skips 72 bytes before searching next delimiter

```
00 00 20 00 00 00 20 00   00 00 20 00 00 00 20 4e   |.. ... ... ... N|
80 04 bb 4e 88 02 00 00   ff ff ff ff ff ff 4c 5e   |...N..........L^|
0c 9e 82 19 4c 5e 0c 9e   82 19 60 30 00 00 aa aa   |....L^....`0....|
03 00 00 00 08 00 45 00   00 22 00 01 00 00 40 01   |......E.."....@.|
57 38 0a 00 00 01 c0 a8   58 f9 08 00 ed f8 00 00   |W8......X.......|
00 fe 58 58 58 58 58 58   13 30 f0 e9 00 00 20 4e   |..XXXXXX.0.... N|
```

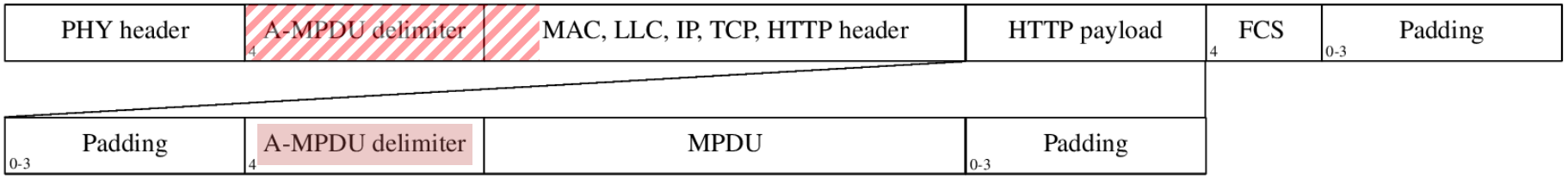
After: the deaggregator searches next delimiter in the higher level payload!

```
00 00 20 00 00 00 20 00   00 00 20 00 00 00 20 4e   |.. ... ... ... N|
11 11 11 4e 88 02 00 00   ff ff ff ff ff ff 4c 5e   |...N..........L^|
0c 9e 82 19 4c 5e 0c 9    82 19 60 30 00 00 aa a    |....L^....`0....|
03 00 00 00 08 00 45 0    00 22 00 01 00 00 40 1    |......E.."....@.|
57 38 0a 00 00 01 c0 a    58 f9 08 00 ed f8 00 0    |W8......X.......|
00 fe 58 58 58 58 58 5    13 30 f0 e9 00 00 20 e    |..XXXXXX.0.... N|
```

Payload crafted by the attacker can contain a valid delimiter

A-MPDU deaggregation vulnerability

- Results in packet-in-packet style injection of arbitrary frames
- Vulnerability is triggered if one or more bytes of the delimiter are damaged
- Correct length and CRC need to be calculated by attacker
- HTTP example:

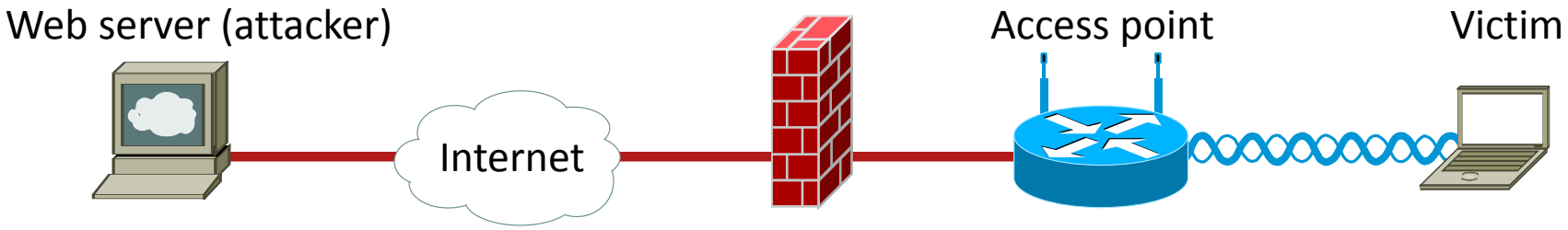


Implementation of the attack

- Replicate aggregation algorithm in software based on standard
- Involves calculating delimiter, correct offset, padding and MAC CRC
- Embed resulting payload in higher layer packet
- Available on Github: https://github.com/rpp0/aggr-inject

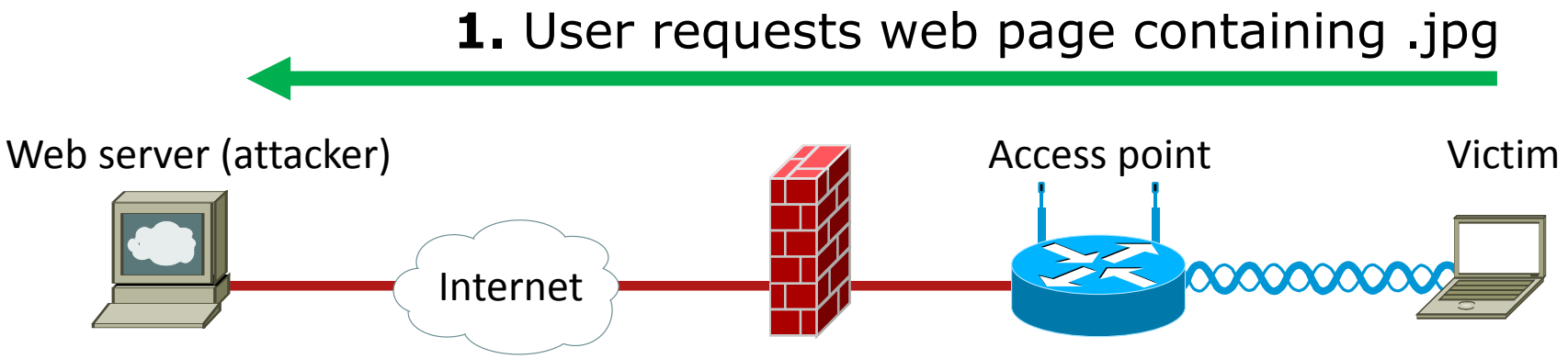
Proof-of-concept

- "Malicious download attack"
- Web server hosts .jpg containing valid MPDU subframes (Beacon frames)
- Could be any type of frame / protocol



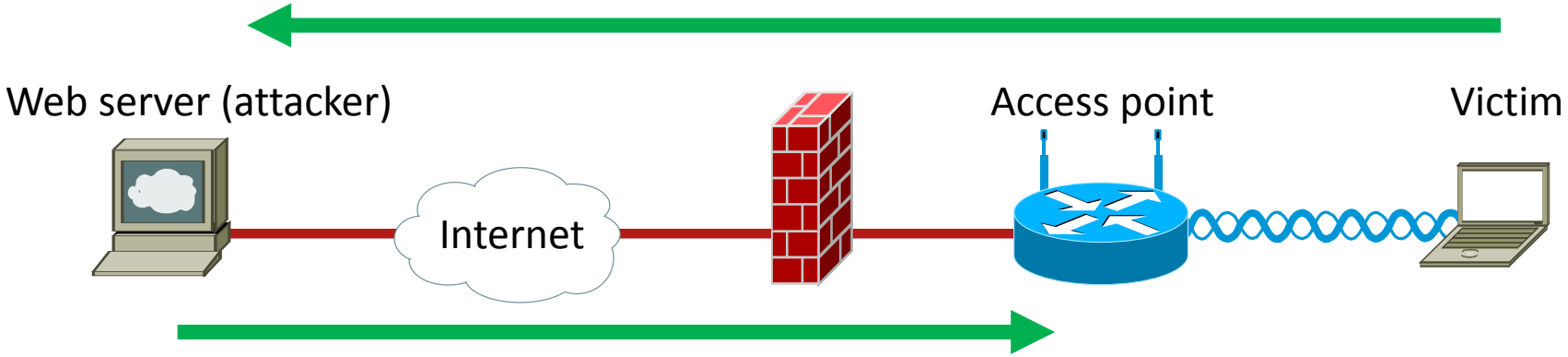
Proof-of-concept

- "Malicious download attack"
- Web server hosts .jpg containing valid MPDU subframes (Beacon frames)
- Could be any type of frame / protocol



Proof-of-concept

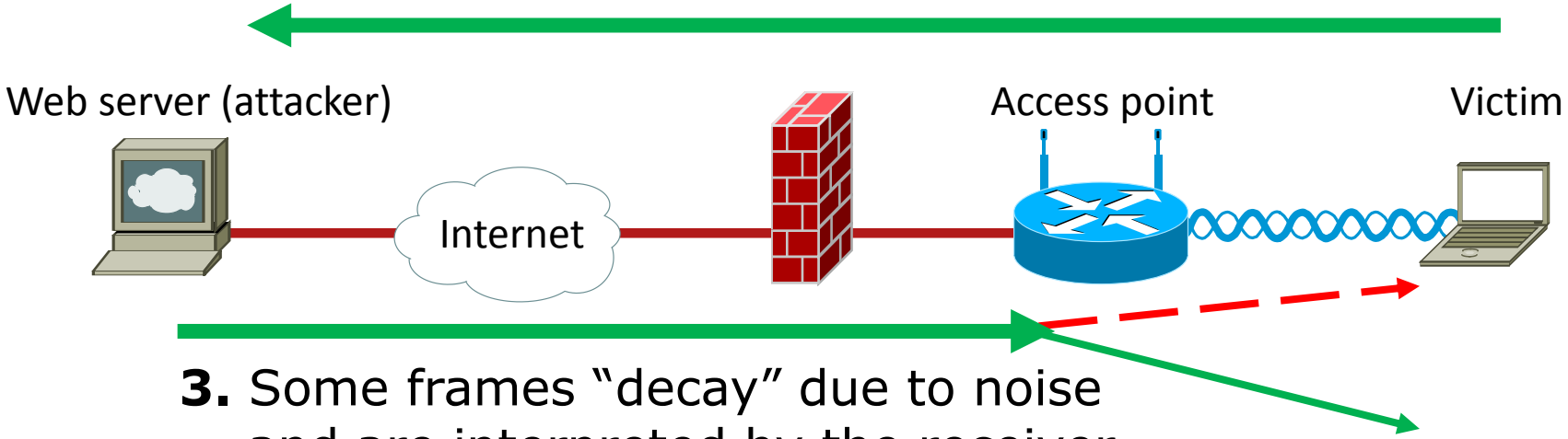
- "Malicious download attack"
- Web server hosts .jpg containing valid MPDU subframes (Beacon frames)
- Could be any type of frame / protocol



2. Server replies with .jpg containing malicious subframes

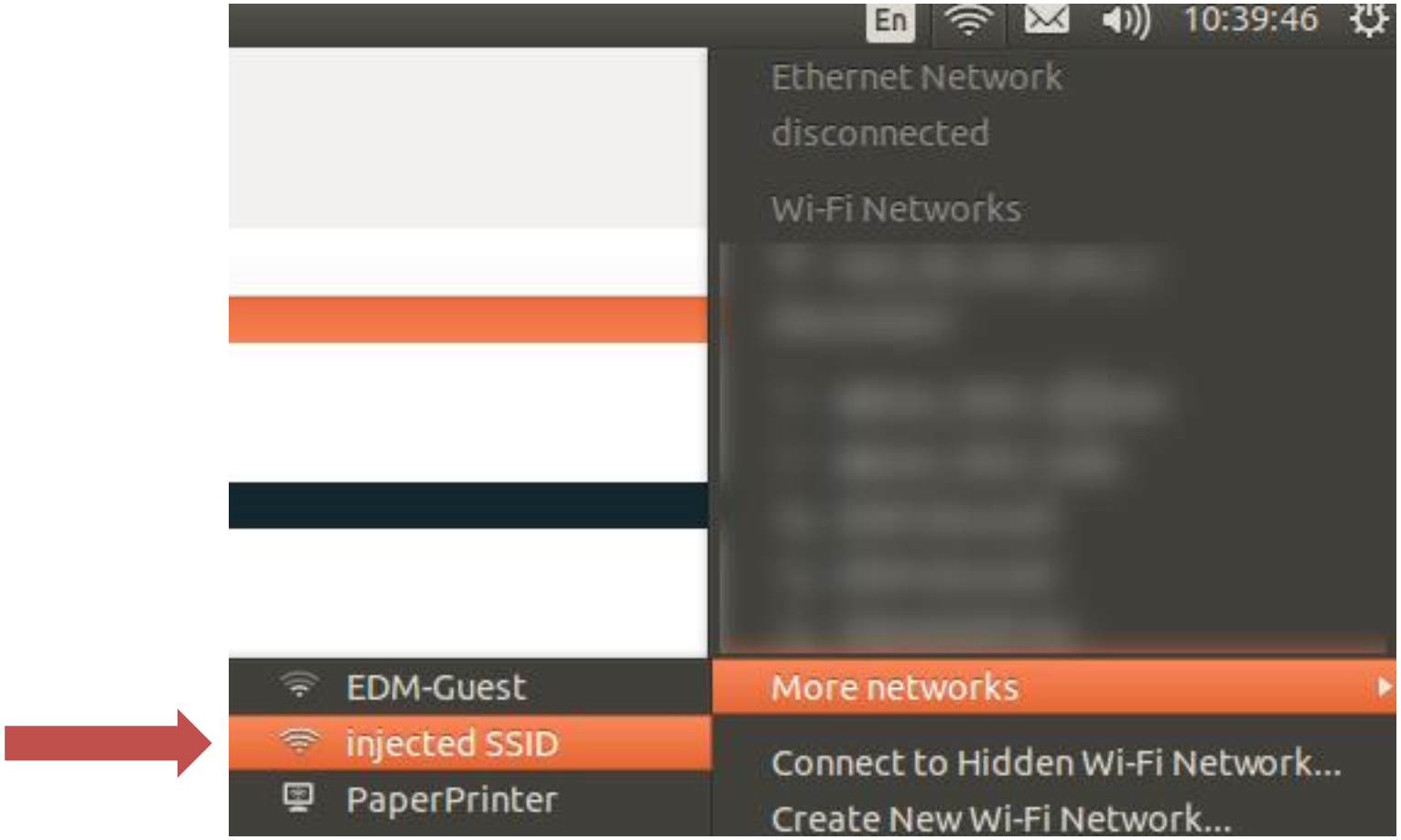
Proof-of-concept

- "Malicious download attack"
- Web server hosts .jpg containing valid MPDU subframes (Beacon frames)
- Could be any type of frame / protocol



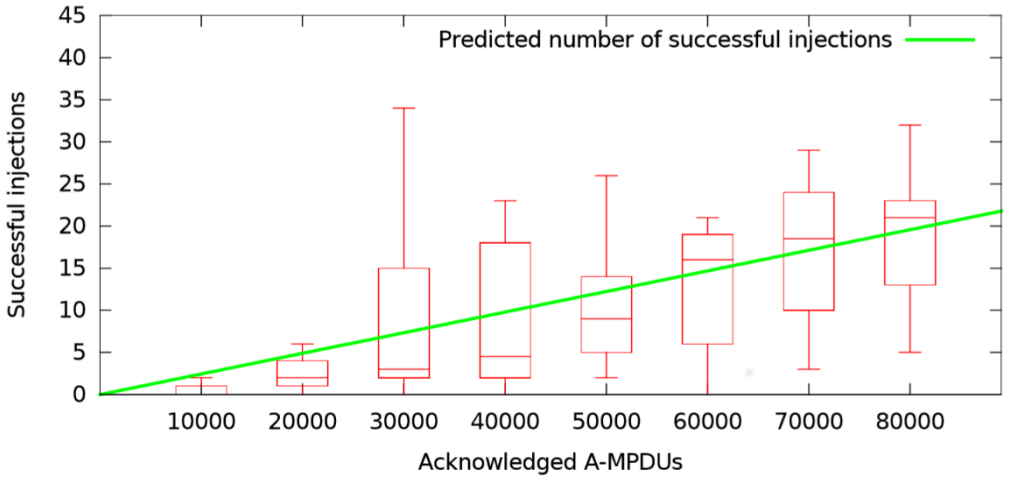
3. Some frames "decay" due to noise and are interpreted by the receiver

Proof-of-concept



Success rate

- Depends on two factors:
  - Aggregation rate: how often does the remote AP perform A-MPDU aggregation?
  - Corruption probability: how likely is a frame to become corrupted when transmitted from AP to victim?
- Our lab network: one injection per 4095 A-MPDUs
  - ~ seconds / minutes depending on data rate



- A single injection count may include multiple MPDU subframes

Vulnerable devices

- All devices that were able to exchange aggregated frames with the AP were vulnerable
- These are the devices we tested
- Probably many more

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

Mitigations

- WPA2-AES
  - Attacker cannot determine a plaintext payload which would produce ciphertext with a valid delimiter

- Disable A-MPDU aggregation
- Drop corrupted A-MPDUs (similar to A-MSDU)
- LangSec style approaches:
  - Different modulation for header and payload (different symbol set)
  - ICBLBCs or: different code words for header and payload
- Deep packet inspection

Future work

- Implementation of the mitigation strategies in order to determine their effectiveness
- Use technique in other wireless protocols where aggregation is performed similarly
- Fingerprinting possible based on sensitivity to attack?

Questions?

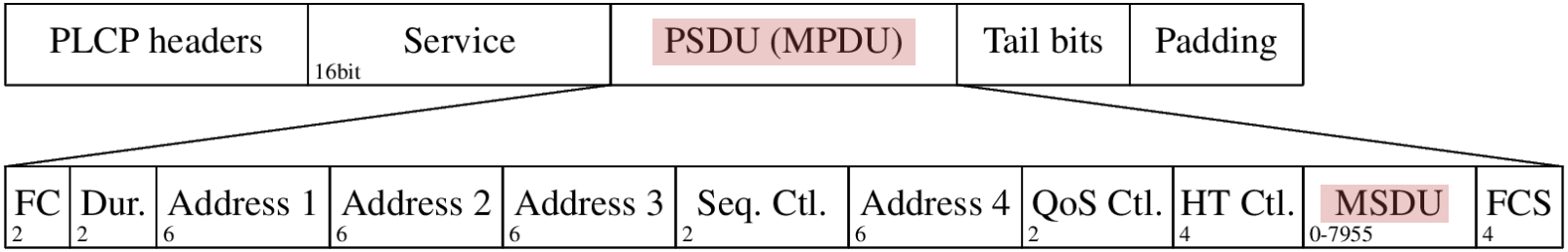
Pieter Robyns

E-mail: pieter.robyns@uhasselt.be
Github: https://github.com/rpp0

Backup slides

MSDU / MPDU?

- Terminology to denote a certain section of the 802.11 frame
- MPDU or MAC Protocol Data Unit (above PHY / PLCP layer)
- MSDU or MAC Service Data Unit (above MAC layer)



Determining the delimiter length

- Convert to little endian (see spec.)

```
80 04 bb 4e 88 02 00 00  ff ff ff ff ff ff 4c 5e  |...N.........L^|
```

```
04 80
```

- Extract 12 bit length

```
LSB                    MSB
   0000 000100100000
```

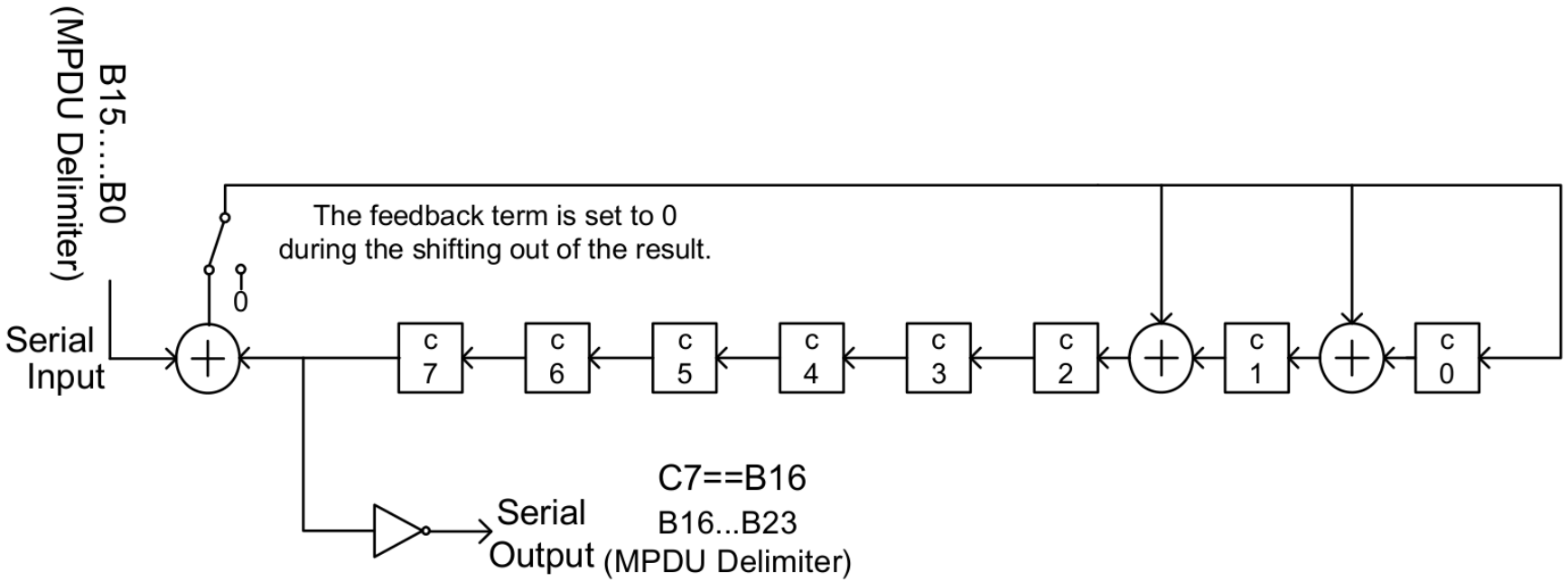
- Convert to decimal

```
000100100000
```

```
72 (64 + 8)
```

Calculation of the delimiter CRC

- In hardware



- In Python

```
crc_fun = crcmod.mkCrcFun(0b100000111, rev=True, initCrc=0x00, xorOut=0xFF)
crc = crc_fun(struct.pack('<H', mpdu_len))
```

Analytical approximation

$$p_i \approx p_a \cdot \left(p_c \cdot \frac{4}{L_s}\right) \cdot \left(1 - \frac{p_c \cdot {}^{6}/_{L_a}}{{}^{L_a}/_{L_s}}\right) \cdot (1 - p_c)$$

Frame is aggregate

Frame delimiter is corrupted

The PLCP header of the aggregate is not corrupted

The embedded subframe is not corrupted

p_i : Probability of a successful injection (**per MPDU**)

p_a : Probability of aggregation (per MPDU)

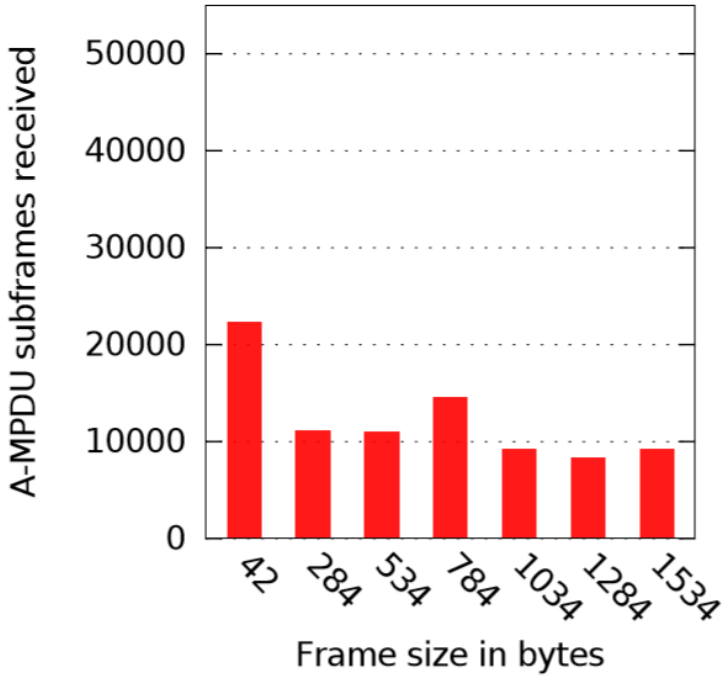
p_c : Probability of frame corruption (per MPDU)

L_a : A-MPDU length in bytes

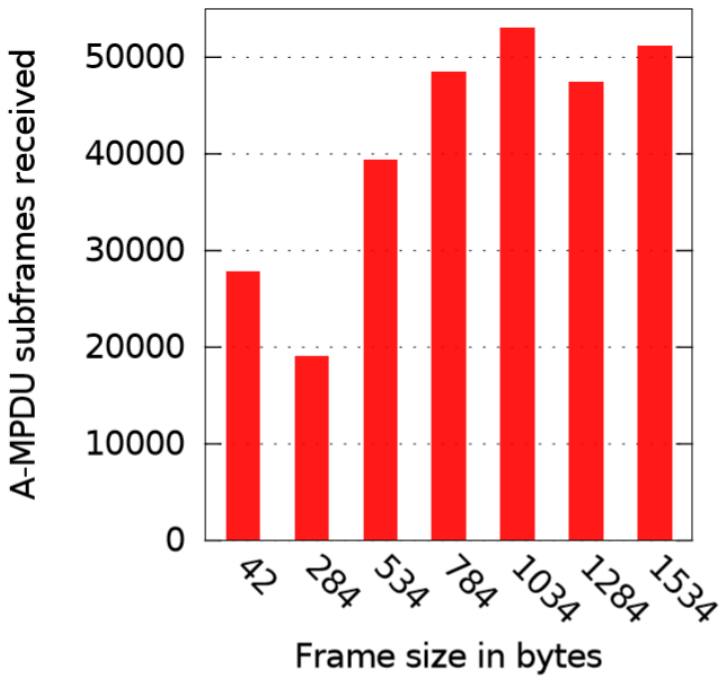
L_s : MPDU subframe length in bytes

Detailed received subframes per size

MikroTik

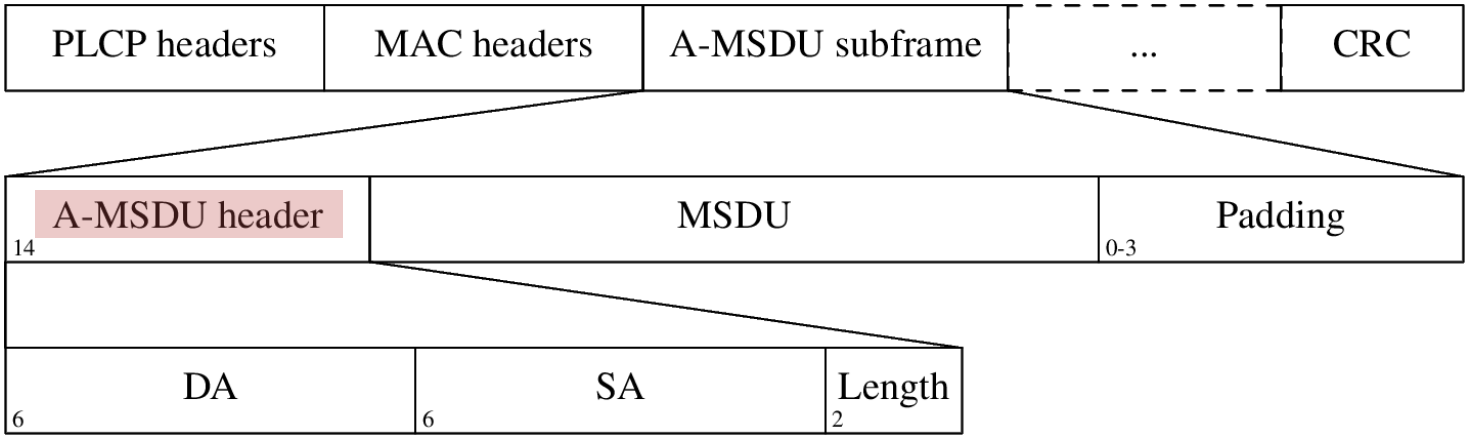


hostapd



Why A-MSDU is not vulnerable

- Aggregates MSDUs from LLC sublayer
- Subframes are delimited with A-MSDU headers
- A-MSDU header is structurally equivalent to 802.3 header (DA, SA, Length)
- **One CRC per aggregate**



Cascading injected subframes

- Deaggregator aligns to attacker's 4-byte boundary
- One corruption → multiple injected subframes
- Not accounted for in success rate analysis