Fragment and Forge: 
Breaking Wi-Fi Through Frame Aggregation and Fragmentation

USENIX Security '21 (video made on April 20, 2021)

Mathy Vanhoef
Advancements in Wi-Fi security

› WPA3 is continuously being updated
  › Preventing recent Dragonblood [VR20] attack
  › Securing hotspots using asymmetric crypto

› Operating channel validation [VBDOP18]

› Beacon protection [VAP20]

› KRACK patches proven secure [CKM20]

Despite these major advancements, found **flaws in all networks** (incl. WPA2/3)
Design flaws

Implementation Flaws
Background

Sending small frames causes high overhead:

header packet1 ACK header packet2 ACK ...

This can be avoided by **aggregating frames**:

header’ packet1 packet2 ... ACK

**Problem: how to recognize** aggregated frames?
### Aggregation design flaw

<table>
<thead>
<tr>
<th>header</th>
<th>aggregated?</th>
<th>encrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>packet</td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>metadata</td>
<td>packet1</td>
</tr>
<tr>
<td></td>
<td>metadata</td>
<td>packet2</td>
</tr>
</tbody>
</table>
Aggregation design flaw

Not authenticated

<table>
<thead>
<tr>
<th>header</th>
<th>aggregated?</th>
<th>encrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>metadata</td>
<td>packet1</td>
</tr>
</tbody>
</table>

Flip flag → payload is parsed differently → inject packets
Exploit steps

Get image from attacker’s server

Send special IPv4 packet
Exploit steps

1. Get image from attacker’s server
2. Encrypt as normal frame
3. Send special IPv4 packet
4. Encrypt as normal frame
Exploit steps

1. Get image from attacker’s server
2. Set aggregated flag
3. Encrypt as normal frame
4. Send special IPv4 packet
Exploit steps

1. Get image from attacker’s server
2. Encrypt as normal frame
3. Set aggregated flag
4. Inject any packet

→ **Inject** ICMPv6 RA with malicious DNS server

Send special IPv4 packet

Encrypt as normal frame

Set aggregated flag

Inject any packet
Exploit steps

1. Inject any packet
   → Inject ICMPv6 RA with malicious DNS server

2. Set aggregated flag

3. Inject special EAPOL frame

Bug in AP → do attack w/o user interaction
(affected 2/4 of home APs)
Implementation Flaws

- Aggregation
- Mixed key
- Fragment cache
Background

Large frames have a high chance of being corrupted:

Avoid by **fragmenting** & only retransmitting lost fragments:

→ Protected header info defines place in original frame
Fragment cache design flaw

Fragments aren’t removed after disconnecting:

$$Enc_k(Frag_0)$$

Store fragment
Fragment cache design flaw

Fragments aren’t removed after disconnecting:

\[ Enc_k(Frag_0) \]

\[ Enc_m(Frag_1) \]

\[ Enc_m(Frag_1) \]

\[ Enc_m(Frag_1) \]

\[ Enc_m(Frag_1) \]

\[ Enc_m(Frag_1) \]

Client connects

\[ Enc_k(Frag_0) \]

Disconnect

Store fragment

Attacker’s \( Frag_0 \) and client’s \( Frag_1 \) is reassembled
Summary of impact

Abuse to **exfiltrate or inject packets** assuming:
1. Hotspot-like network where users distrust each other
2. Client sends fragmented frames (rare unless Wi-Fi 6)

Even the ancient **WEP protocol is affected**!

› WEP is also affected by the mixed key design flaw

→ Design flaws have been **part of Wi-Fi since 1997**
Implementation Flaws

Aggregation

Mixed key

Fragment cache
Mixed key design flaw

Fragments decrypted with different keys are reassembled:

\[ \text{Enc}_k(Frag_0), \text{Enc}_k(Frag_1) \quad \Rightarrow \quad \text{Enc}_k(Frag_0) \]
Mixed key design flaw

Fragments decrypted with different keys are reassembled:

\[ \text{Enc}_k(\text{Frag}_0), \text{Enc}_k(\text{Frag}_1) \quad \text{Enc}_k(\text{Frag}_0) \]

Refresh session key from \( k \) to \( m \)
Mixed key design flaw

Fragments decrypted with **different keys are reassembled**:

\[ \text{Enc}_k(\text{Frag}_0), \text{Enc}_k(\text{Frag}_1) \]

\[ \text{Enc}_m(\text{Frag}_0), \text{Enc}_m(\text{Frag}_1) \]
Mixed key design flaw

Fragments decrypted with different keys are reassembled:

$\text{Enc}_k(Frag_0), \text{Enc}_k(Frag_1) \rightarrow \text{Enc}_k(Frag_0)$

Refresh session key from $k$ to $m$

$\text{Enc}_m(Frag_0), \text{Enc}_m(Frag_1) \rightarrow \text{Enc}_m(Frag_1)$

→ Can mix fragments of different frames
Summary of impact

Abuse to **exfiltrate data** assuming:

1. Someone sends fragmented frames (rare unless Wi-Fi 6)
2. Victim will connect to server of attacker
3. Network periodically refreshes the session key
Summary of impact

Abuse to **exfiltrate data** assuming:

1. Someone sends fragmented frames (rare unless Wi-Fi 6)
2. Victim will connect to server of attacker
3. Network periodically refreshes the session key

» **Combine with implementation flaw** to avoid this condition
Design flaws

Implementation Flaws
Design flaws

- Plaintext frames
- Broadcast fragments
- EAPOL forwarding
- Cloacked A-MSDUs
- Mixed fragments
- Out of order fragments
Trivial frame injection

Plaintext frames wrongly accepted:

› Depending if fragmented, broadcasted, or while connecting
› Sometimes frames that resemble a handshake message
› Examples: Apple and some Android devices, some Windows dongles, home and professional APs, and many others!

→ Can trivially inject frames
Design flaws

- Plaintext frames
- Broadcast fragments
- Cloacked A-MSDUs
- Mixed fragments
- EAPOL forwarding
- Out of order frag
- No fragmentation support
No fragmentation support

Some devices don’t support fragmentation

› But they **treat fragmented frames as full frames**

› Examples: OpenBSD and Espressif chips

  ➔ Abuse to **inject frames** under right conditions

  ➔ **All devices are vulnerable** to one or more flaws
Created tool to test devices

Has **45+ test cases** for both clients and APs:

→ Available at [https://github.com/vanhoefm/fragattack](https://github.com/vanhoefm/fragattack)
Discussion

Design flaws took two decades to discover

› Without modified drivers some attacks will fail
› Fragmentation & aggregation wasn’t considered important
Discussion

Design flaws took two decades to discover
› Without modified drivers some attacks will fail
› Fragmentation & aggregation wasn’t considered important

Long-term lessons:
› **Adopt defences early** even if concerns are theoretic
› Isolate **security contexts** (data decrypted with different keys)
› **Keep fuzzing** devices. Wi-Fi Alliance can help here!
Conclusion

› Discovered three **design flaws**
› Multiple **implementation flaws**
› Several flaws are **trivial to exploit**
› More info: [www.fragattacks.com](http://www.fragattacks.com)