FragAttacks: Recent Flaws in WPA2/3 and New Defenses

GISEC, 21 March ‘22, Dubai

Prof. dr. Mathy Vanhoef

NEW YORK UNIVERSITY

KU LEUVEN

DistriNet

Advancements in Wi-Fi security

- WPA3 is continously being updated
  - Preventing Dragonblood attack
  - Securing hotspots

Advancements in Wi-Fi security

- WPA3 is continously being updated
  - Preventing Dragonblood attack
  - Securing hotspots
- Operating channel validation
- Beacon protection
- KRACK patches proven secure

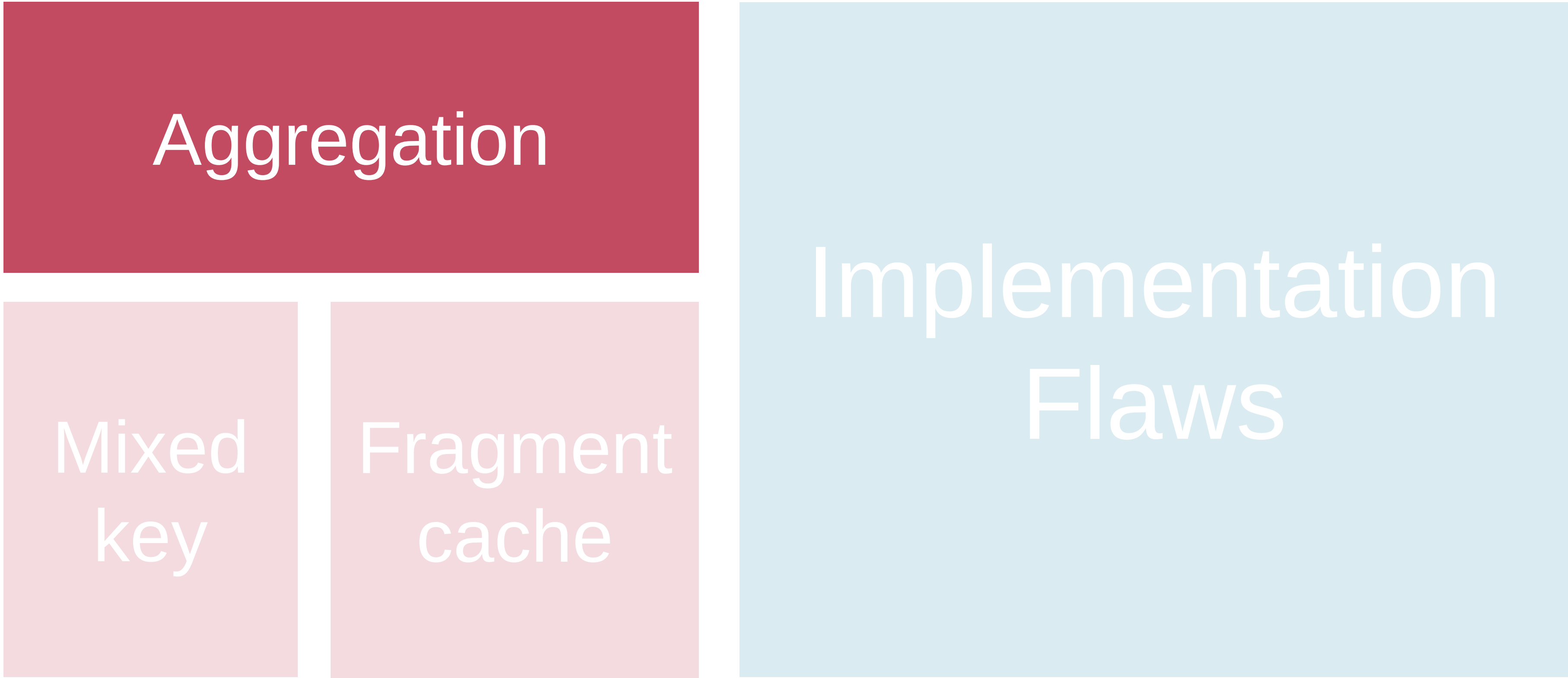
Despite these major advacements, we found **flaws in all Wi-Fi networks** (incl. WPA2/3)

Design flaws

Implementation Flaws

Design flaws

Implementation Flaws



Aggregation

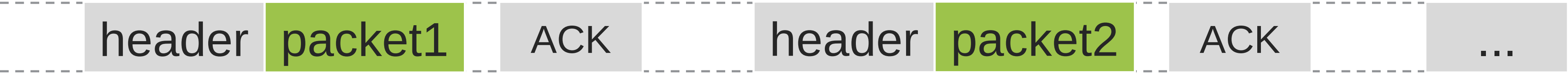
Mixed key

Fragment cache

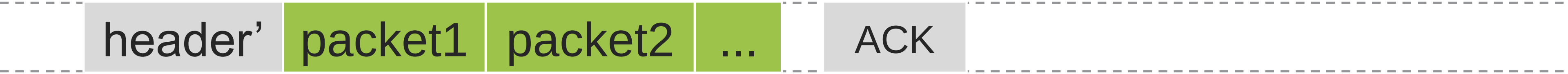
Implementation Flaws

Background

Sending small frames causes high overhead:

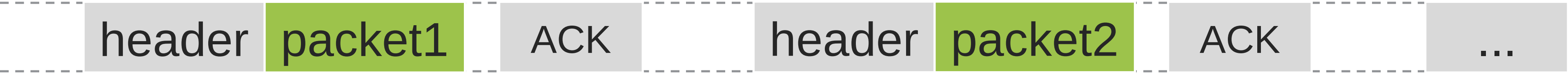


This can be avoided by **aggregating frames**:

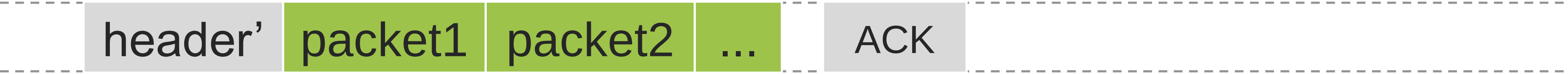


Background

Sending small frames causes high overhead:

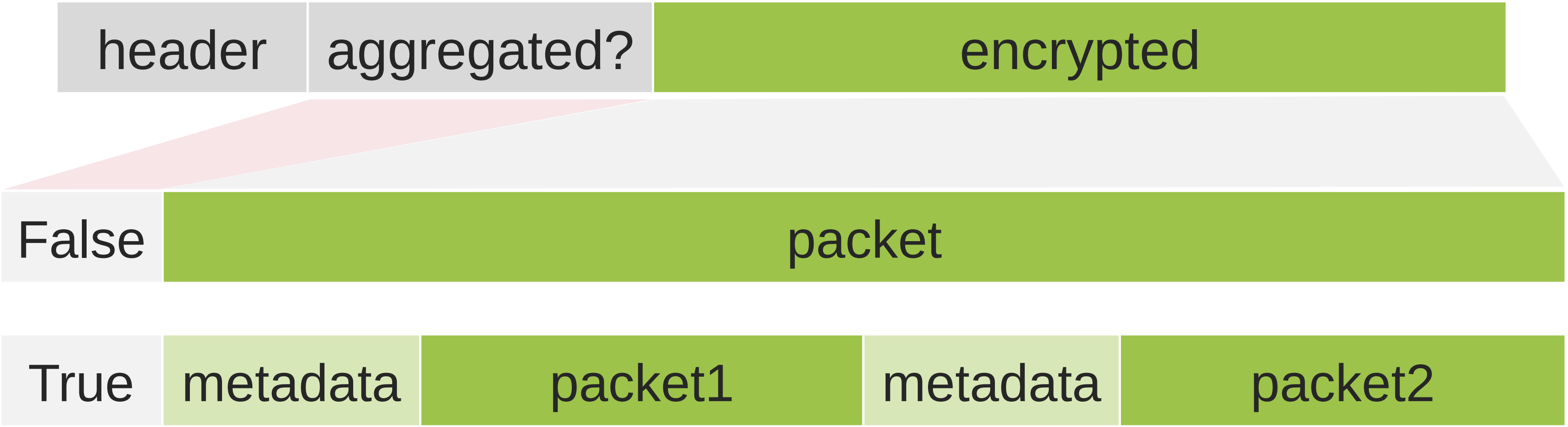


This can be avoided by **aggregating frames**:



Problem: how to recognize aggregated frames?

Aggregation design flaw

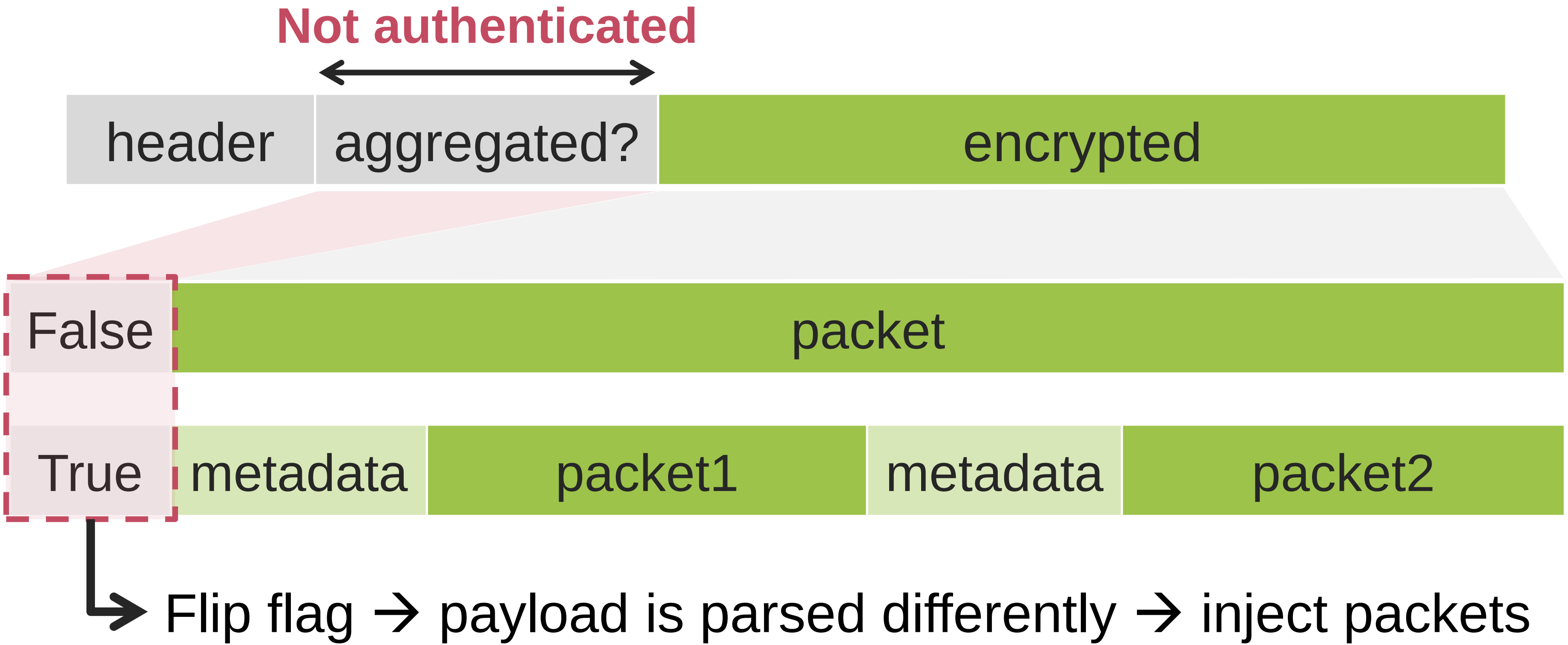


header | aggregated? | encrypted

False | packet

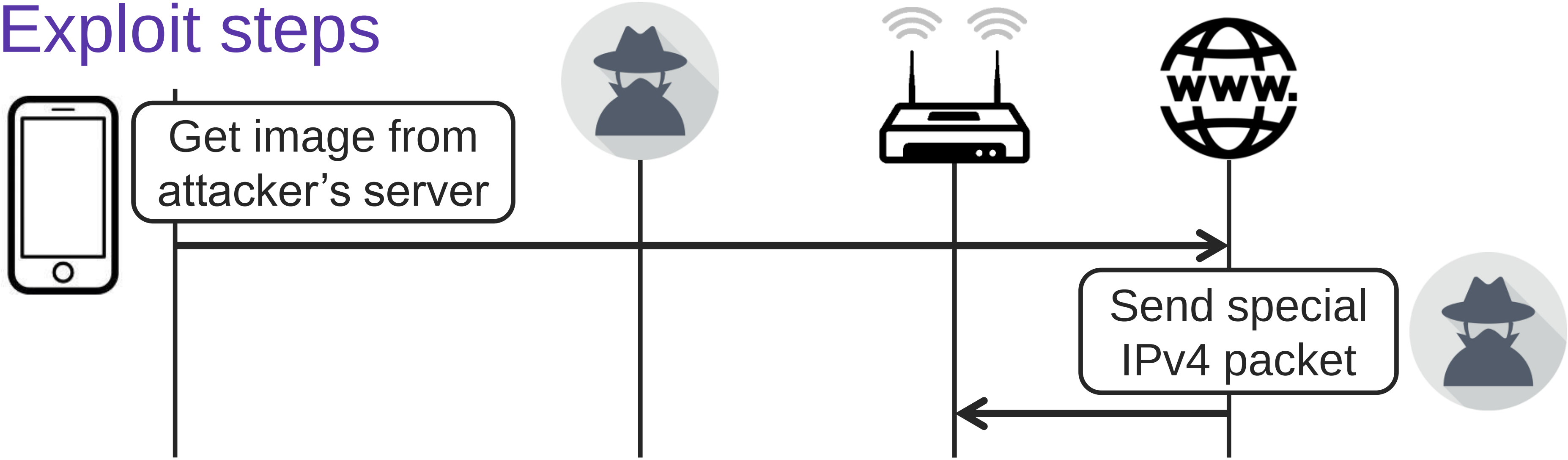
True | metadata | packet1 | metadata | packet2

Aggregation design flaw



Flip flag → payload is parsed differently → inject packets

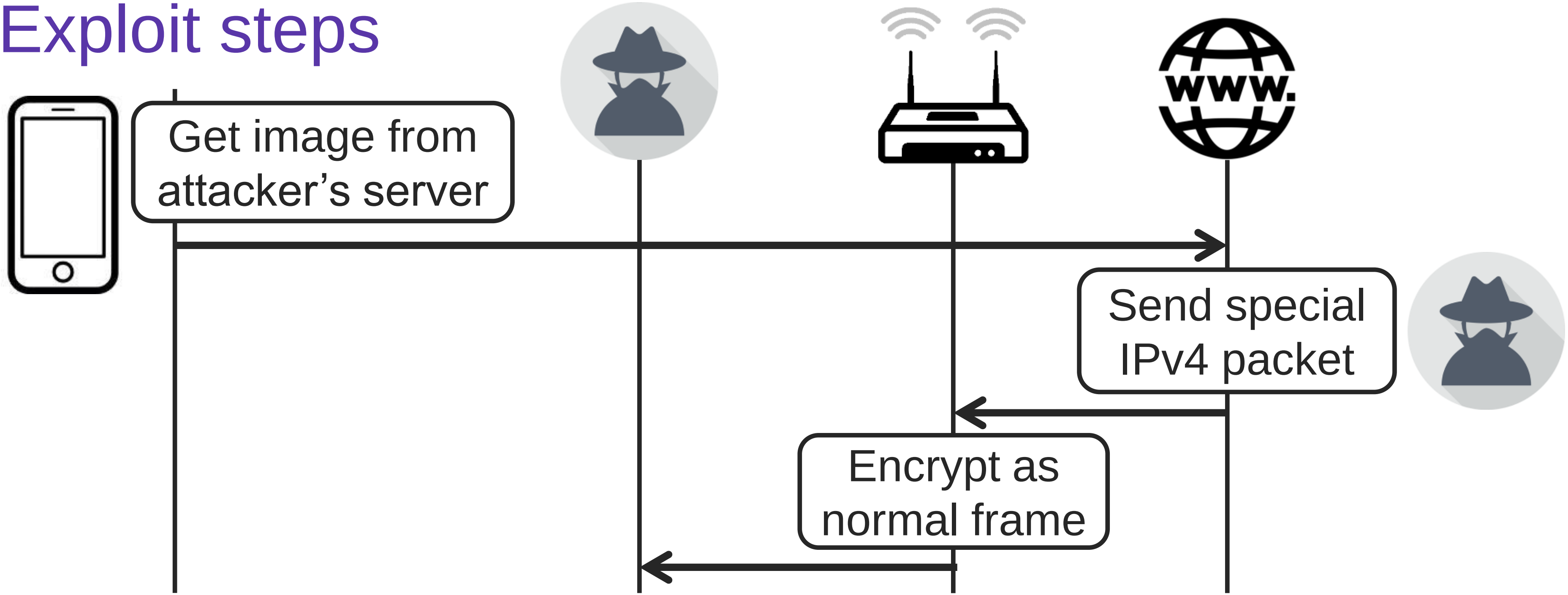
Exploit steps



Get image from attacker’s server

Send special IPv4 packet

Exploit steps

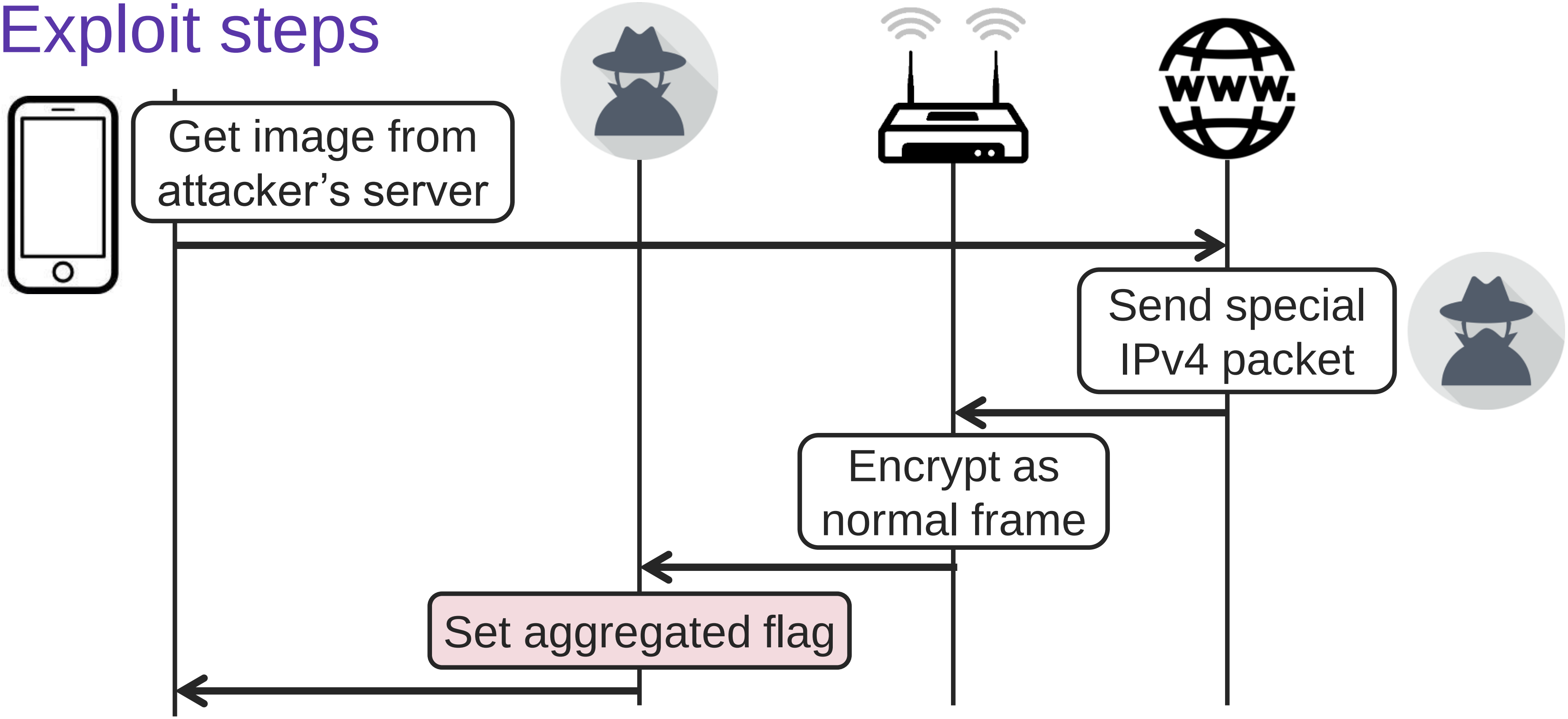


Get image from attacker’s server

Send special IPv4 packet

Encrypt as normal frame

Exploit steps



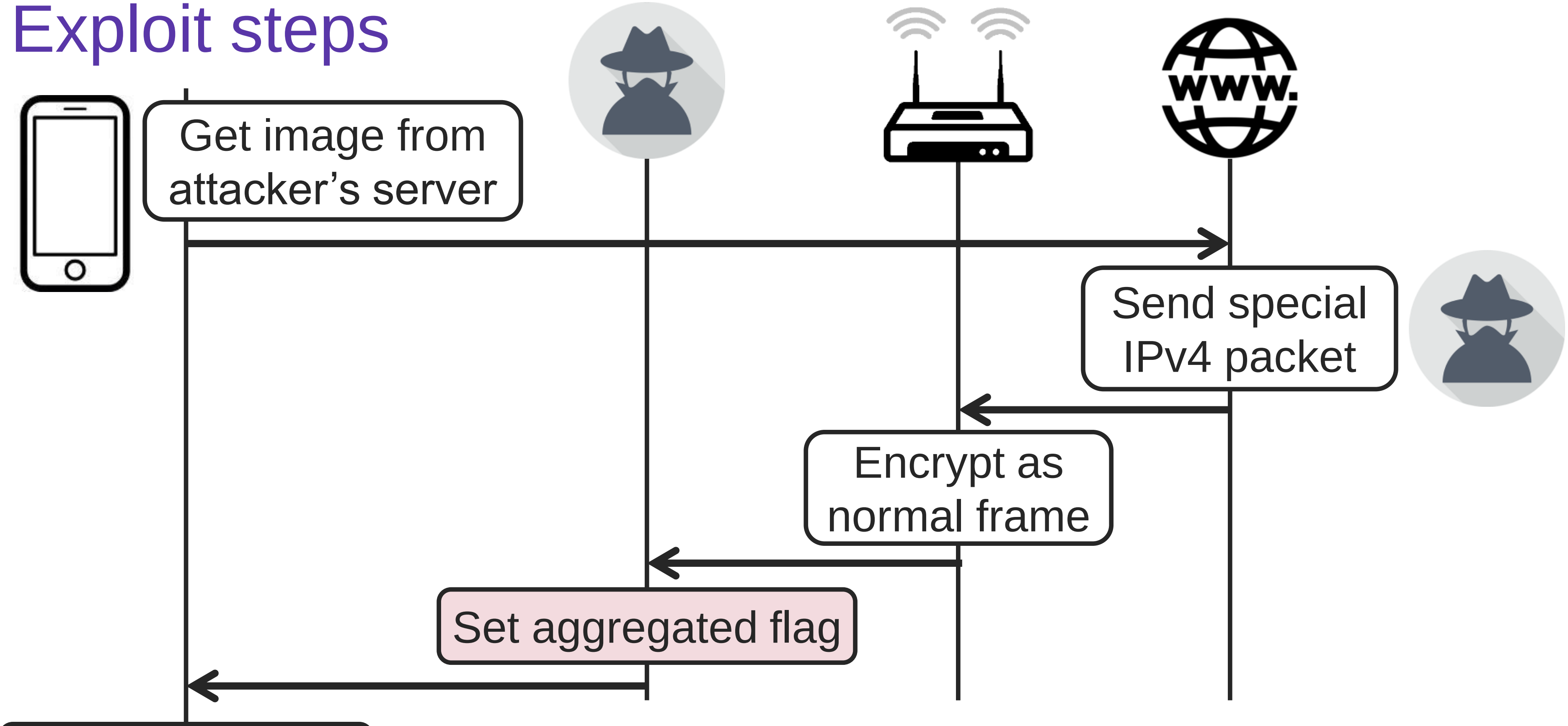
Get image from attacker's server

Send special IPv4 packet

Encrypt as normal frame

Set aggregated flag

Exploit steps



Get image from attacker's server

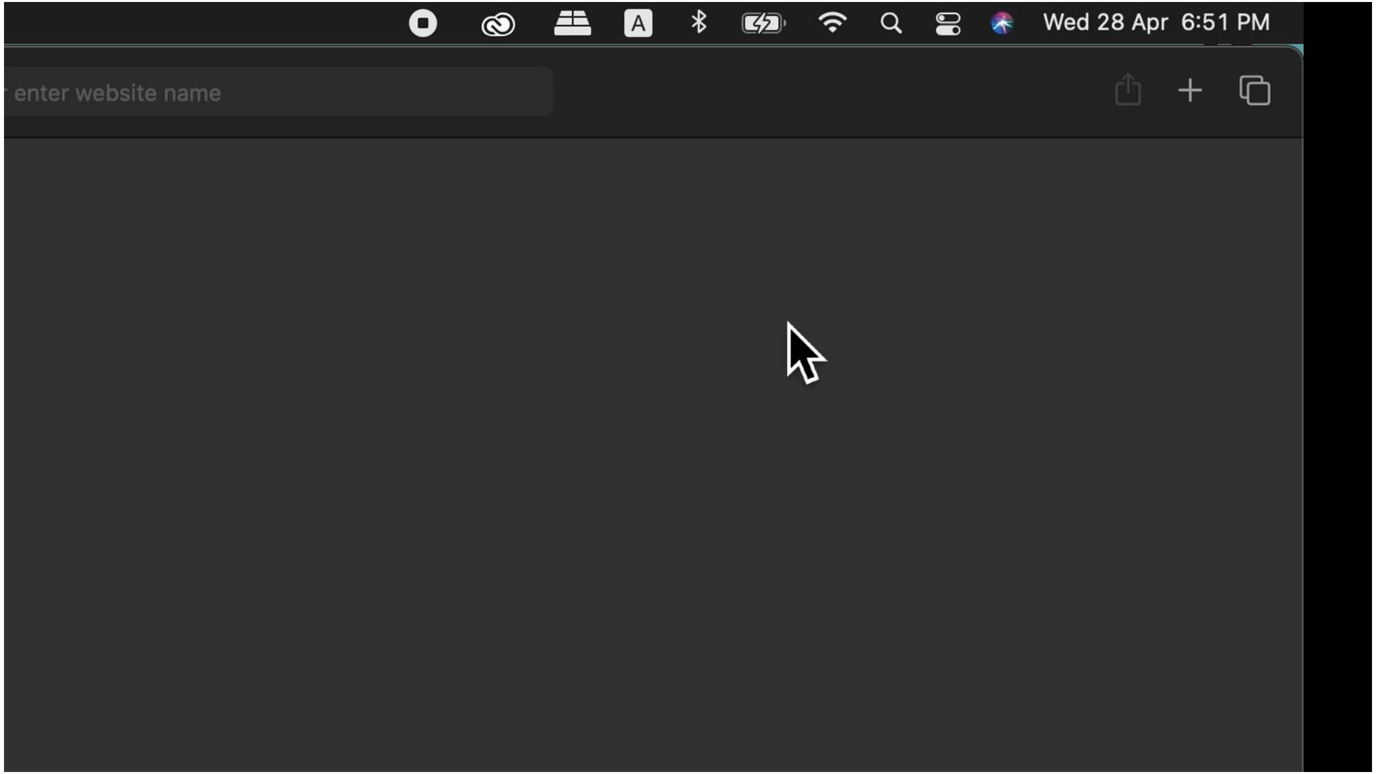
Send special IPv4 packet

Encrypt as normal frame

Set aggregated flag

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

DEMO



Design flaws

Implementation Flaws

Design flaws

Plaintext frames

Mixed fragments

Out of order frag

Broadcast fragments

EAPOL forwarding

Cloacked A-MSDUs

Out of order fragments

Trivial frame injection

Plaintext frames wrongly accepted:

- Depending if **fragmented**, **broadcasted,** or while **connecting**

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**

DEMO

Design flaws

Plaintext frames

Mixed fragments

Out of order frag

Broadcast fragments

EAPOL forwarding

Cloacked A-MSDUs

No fragmentation support

No fragmentation support

Some devices don’t support fragmentation

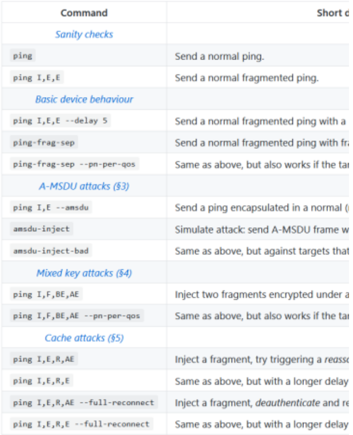
- But they **treat fragmented frames as full frames**
- Examples: OpenBSD and Espressif chips

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

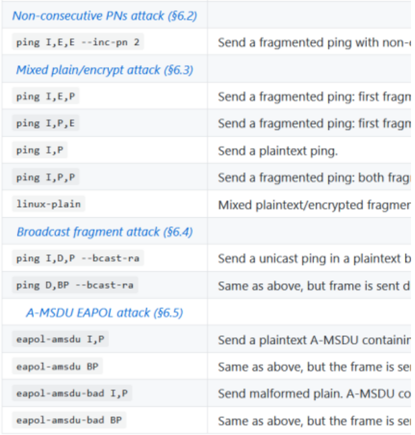
→ **All devices are vulnerable** to one or more flaws

Created tool to test devices

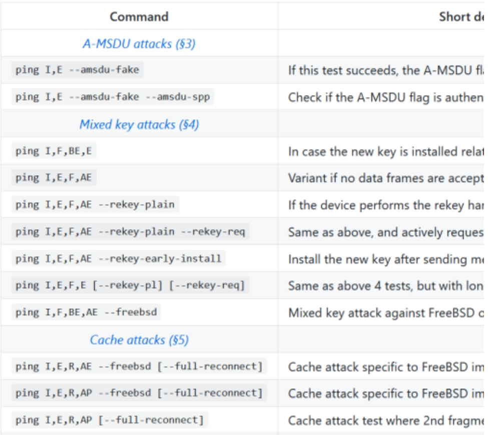
Has **45+ test cases** for both **clients and APs**:



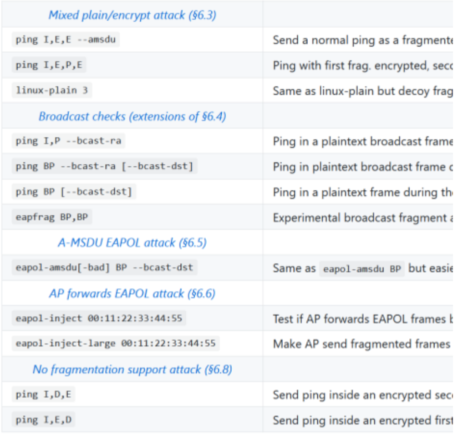
Command	Short d
Sanity checks	
`ping`	Send a normal ping.
`ping I,E,E`	Send a normal fragmented ping.
Basic device behaviour	
`ping I,E,E --delay 5`	Send a normal fragmented ping with a
`ping-frag-sep`	Send a normal fragmented ping with fr
`ping-frag-sep --pn-per-qos`	Same as above, but also works if the tar
A-MSDU attacks (§3)	
`ping I,E --amsdu`	Send a ping encapsulated in a normal (
`amsdu-inject`	Simulate attack: send A-MSDU frame w
`amsdu-inject-bad`	Same as above, but against targets that
Mixed key attacks (§4)	
`ping I,F,BE,AE`	Inject two fragments encrypted under a
`ping I,F,BE,AE --pn-per-qos`	Same as above, but also works if the tar
Cache attacks (§5)	
`ping I,E,R,AE`	Inject a fragment, try triggering a *reass*
`ping I,E,R,E`	Same as above, but with a longer delay
`ping I,E,R,AE --full-reconnect`	Inject a fragment, *deauthenticate* and re
`ping I,E,R,E --full-reconnect`	Same as above, but with a longer delay



Non-consecutive PNs attack (§6.2)	
`ping I,E,E --inc-pn 2`	Send a fragmented ping with non-
Mixed plain/encrypt attack (§6.3)	
`ping I,E,P`	Send a fragmented ping: first fragn
`ping I,P,E`	Send a fragmented ping: first fragn
`ping I,P`	Send a plaintext ping.
`ping I,P,P`	Send a fragmented ping: both frag
`linux-plain`	Mixed plaintext/encrypted fragmer
Broadcast fragment attack (§6.4)	
`ping I,D,P --bcast-ra`	Send a unicast ping in a plaintext b
`ping D,BP --bcast-ra`	Same as above, but frame is sent d
A-MSDU EAPOL attack (§6.5)	
`eapol-amsdu I,P`	Send a plaintext A-MSDU containir
`eapol-amsdu BP`	Same as above, but the frame is sei
`eapol-amsdu-bad I,P`	Send malformed plain. A-MSDU co
`eapol-amsdu-bad BP`	Same as above, but the frame is sei



Command	Short de
A-MSDU attacks (§3)	
`ping I,E --amsdu-fake`	If this test succeeds, the A-MSDU fl
`ping I,E --amsdu-fake --amsdu-spp`	Check if the A-MSDU flag is authen
Mixed key attacks (§4)	
`ping I,F,BE,E`	In case the new key is installed relat
`ping I,E,F,AE`	Variant if no data frames are accept
`ping I,E,F,AE --rekey-plain`	If the device performs the rekey har
`ping I,E,F,AE --rekey-plain --rekey-req`	Same as above, and actively reques
`ping I,E,F,AE --rekey-early-install`	Install the new key after sending me
`ping I,E,F,E [--rekey-pl] [--rekey-req]`	Same as above 4 tests, but with lon
`ping I,F,BE,AE --freebsd`	Mixed key attack against FreeBSD c
Cache attacks (§5)	
`ping I,E,R,AE --freebsd [--full-reconnect]`	Cache attack specific to FreeBSD im
`ping I,E,R,AP --freebsd [--full-reconnect]`	Cache attack specific to FreeBSD im
`ping I,E,R,AP [--full-reconnect]`	Cache attack test where 2nd fragme

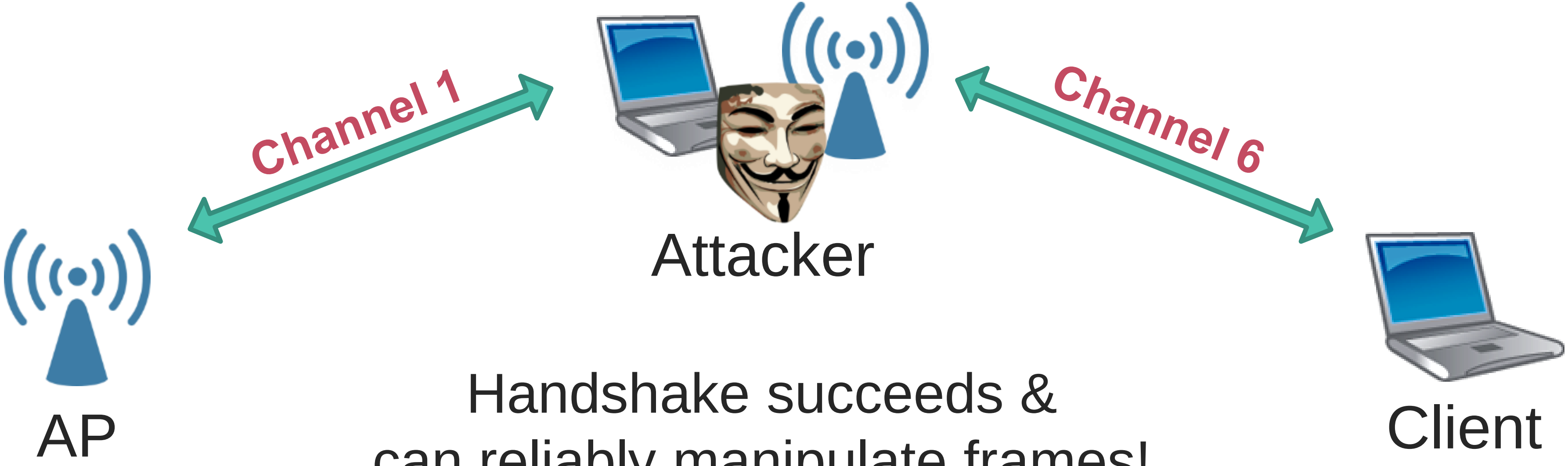


Mixed plain/encrypt attack (§6.3)	
`ping I,E,E --amsdu`	Send a normal ping as a fragmente
`ping I,E,P,E`	Ping with first frag. encrypted, secc
`linux-plain 3`	Same as linux-plain but decoy frag
Broadcast checks (extensions of §6.4)	
`ping I,P --bcast-ra`	Ping in a plaintext broadcast frame
`ping BP --bcast-ra [--bcast-dst]`	Ping in plaintext broadcast frame c
`ping BP [--bcast-dst]`	Ping in a plaintext frame during th
`eapfrag BP,BP`	Experimental broadcast fragment a
A-MSDU EAPOL attack (§6.5)	
`eapol-amsdu[-bad] BP --bcast-dst`	Same as `eapol-amsdu BP` but easie
AP forwards EAPOL attack (§6.6)	
`eapol-inject 00:11:22:33:44:55`	Test if AP forwards EAPOL frames b
`eapol-inject-large 00:11:22:33:44:55`	Make AP send fragmented frames
No fragmentation support attack (§6.8)	
`ping I,D,E`	Send ping inside an encrypted sec
`ping I,E,D`	Send ping inside an encrypted first

→ Available at https://github.com/vanhoefm/fragattack

Abusing design flaws requires multi-channel MitM

AP is cloned on **different channel**



Handshake succeeds &
can reliably manipulate frames!

Channel validation

Verify operating channel when connecting to a network

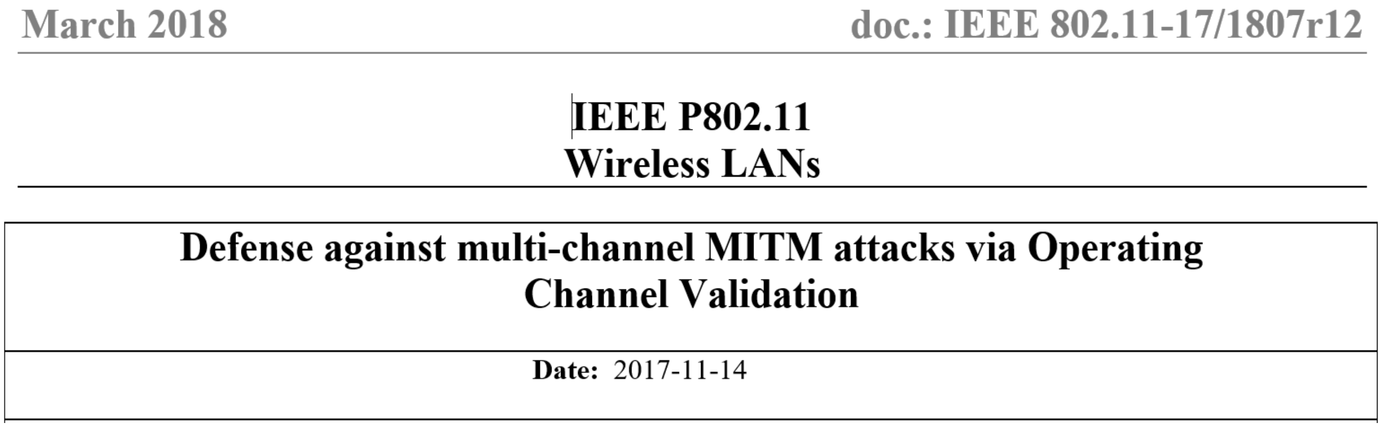
Also need to handle some edge cases:

- After the clients wakes up from sleep mode
- When the network switches channel due to radar detection

→ Implemented on **Linux & Android**

Channel validation

- Collaborated with industry (Broadcom and Intel) to standardize the defense
- **Now part of the latest update to the IEEE 802.11 standard**



March 2018 doc.: IEEE 802.11-17/1807r12

IEEE P802.11
Wireless LANs

Defense against multi-channel MITM attacks via Operating Channel Validation

Date: 2017-11-14

Channel validation

- Collaborated with industry (Broadcom and Intel) to standardize the defense
- **Now part of the latest update to the IEEE 802.11 standard**

- Recognized as an **optional feature of WPA3**
- Good initial step, hopefully becomes mandatory in future

Other defenses for Wi-Fi networks

Channel validation

Mitigates prerequisite of several recent attacks

Beacon protection

Authenticate beacons to prevent denial of service

- Both implemented on Linux and Android
- Now part of the **IEEE 802.11 standard**
- Wi-Fi Alliance is encouraging its adoption

Conclusion

- Discovered three **design flaws**
- Multiple **implementation flaws**
- Several flaws are **trivial to exploit**
- More info: www.fragattacks.com