Breaking & Disrupting WPA2/3 Networks by Abusing Sleep Mode

D. Schepers, A. Ranganathan, and **M. Vanhoef**

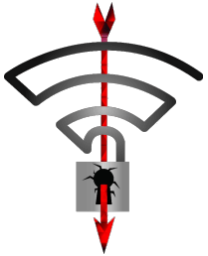
BruCON 2023, Belgium

Based on our USENIX Security ’23 paper

KU LEUVEN DistriNet Northeastern University

Quick Introduction

- Research: network & Wi-Fi security
- Previously discovered **KRACK**, FragAttacks, Dragonblood, …



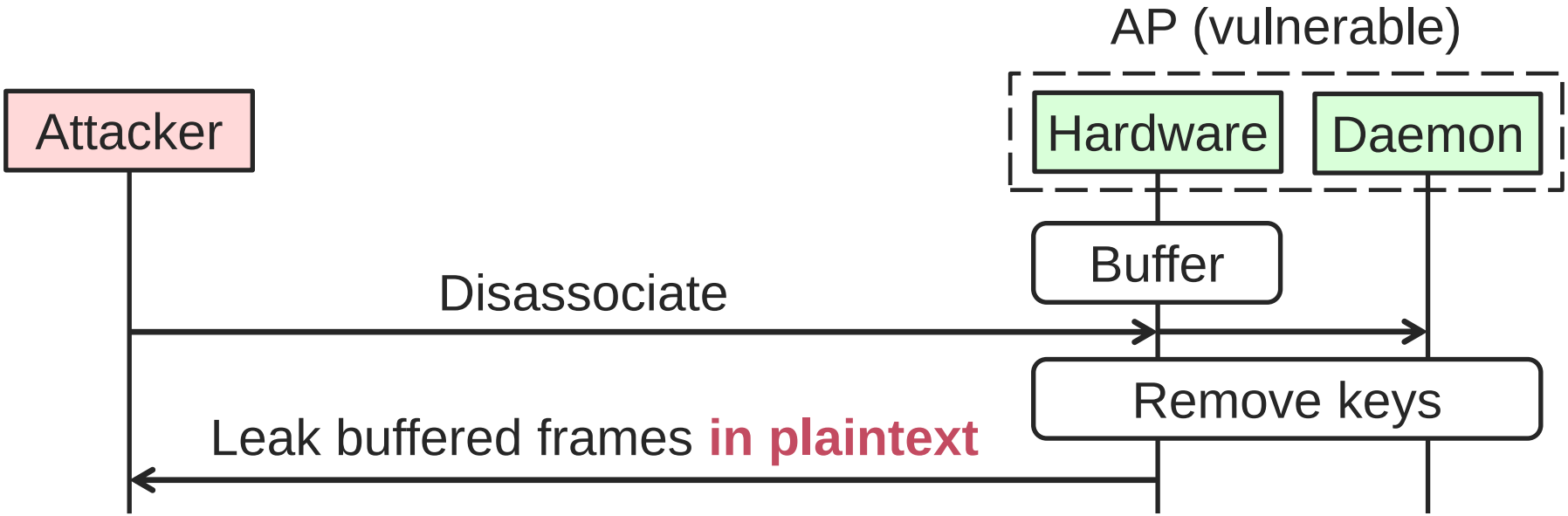
- Helped design Operating Channel Validation and **Beacon protection** (mandatory in Wi-Fi 7)
- Recently found flaws in 2/3rds of VPN clients

We collaborate with industry! ☺

History of Wi-Fi

- WEP (1999): quickly broken [FMS01]
- WPA1/2 (~2003)
  - Offline password brute-force
  - **KRACK** & **Kraken** [VP17,VP18]
- WPA3 (2018):
  - **Dragonblood** side-channels [VR20]

Background: Kr00k implementation flaw



Question: **how are “security contexts” managed**?

The Security Context

- Negotiated protocol suites, encryption keys, packet counters
- All information needed to securely communicate

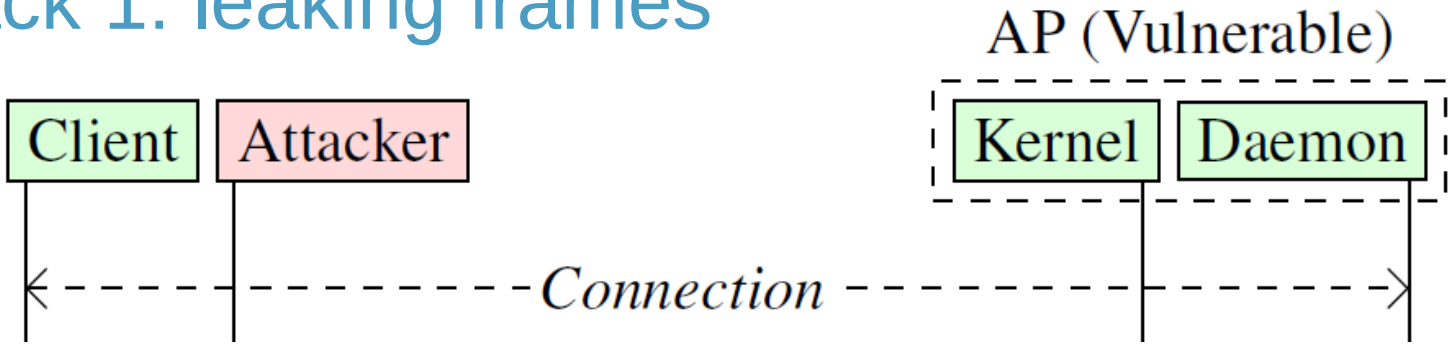
Relation between security context & sleep mode?

- When client wakes up **the security context might have changed** → what happens to queued frames?

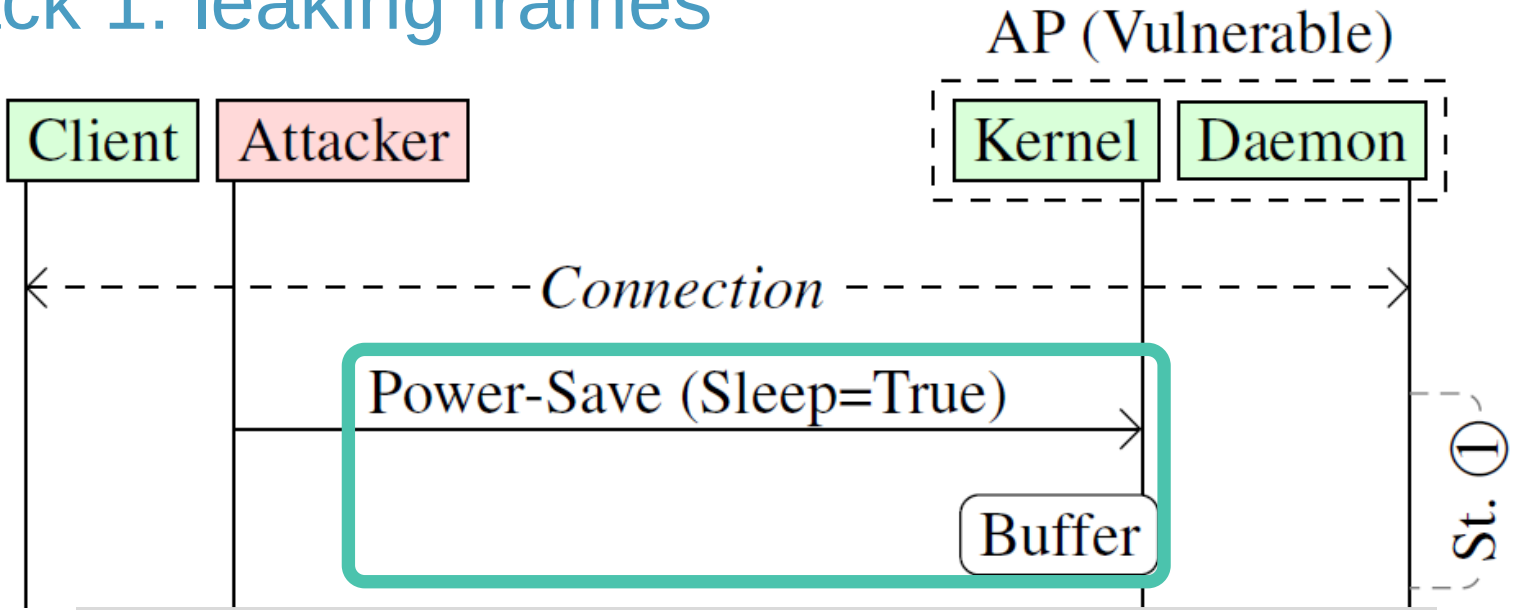
New attack 1: leaking frames

Attack 1: leaking frames

Attack 1: leaking frames

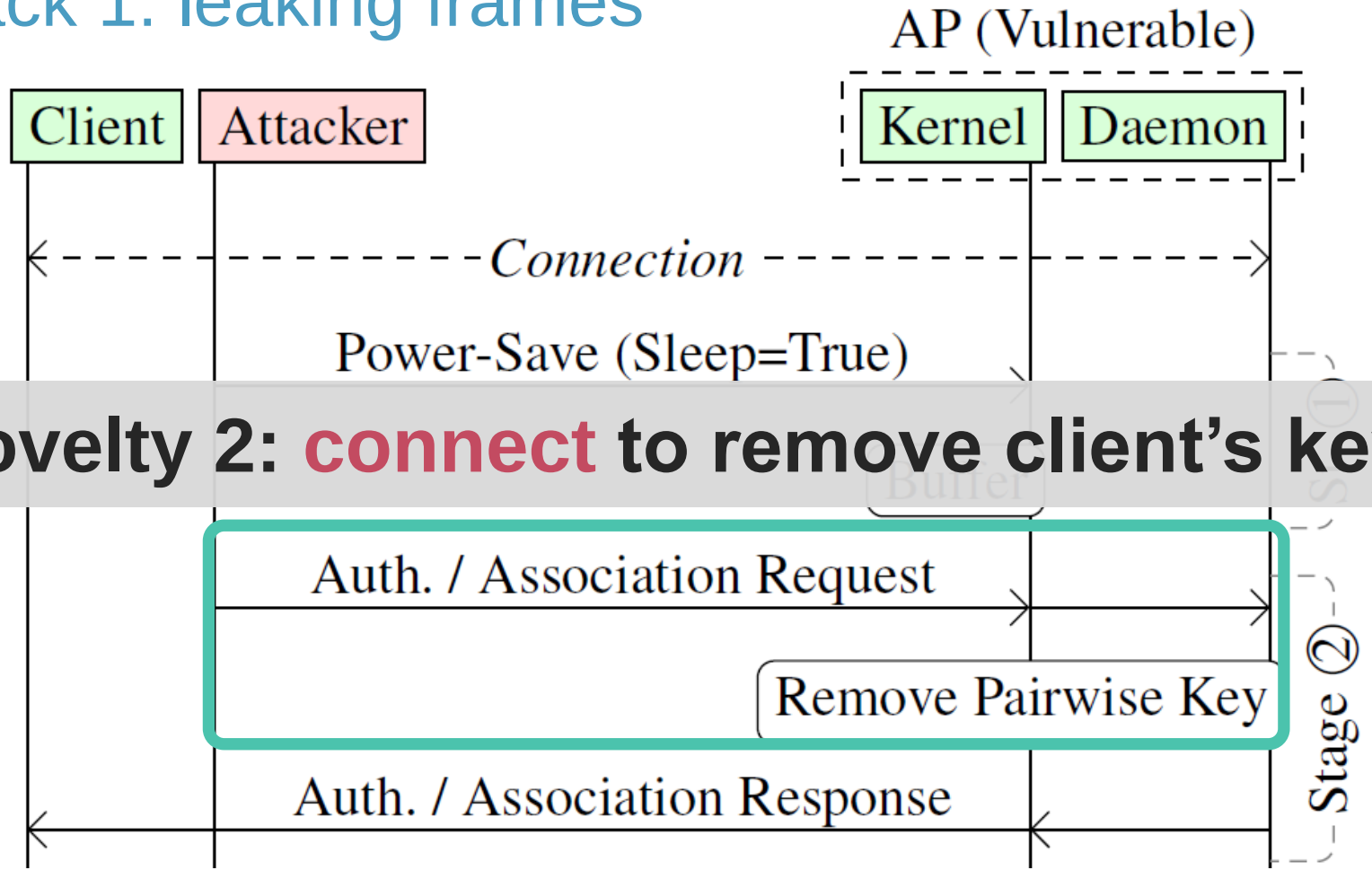


Attack 1: leaking frames



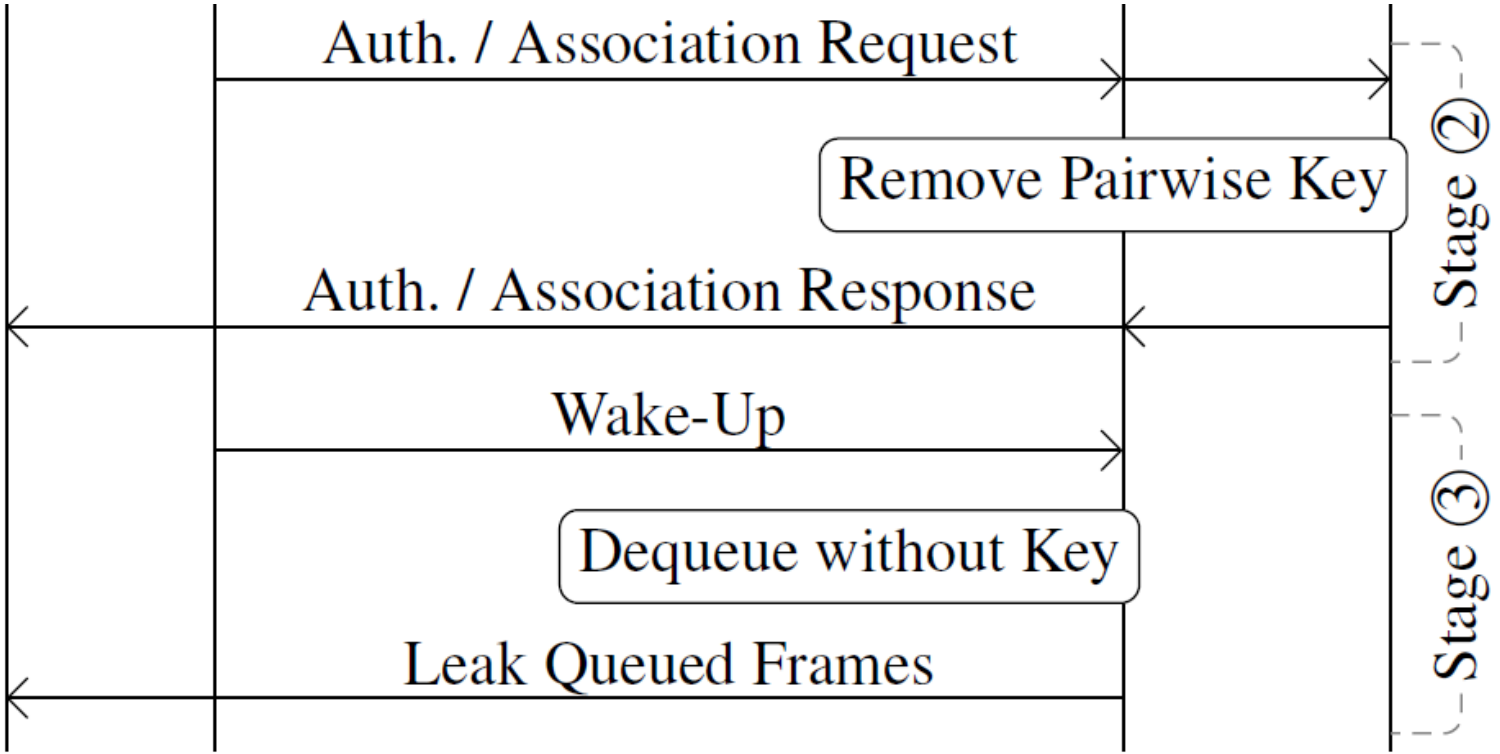
Novelty 1: controlled buffering

Attack 1: leaking frames

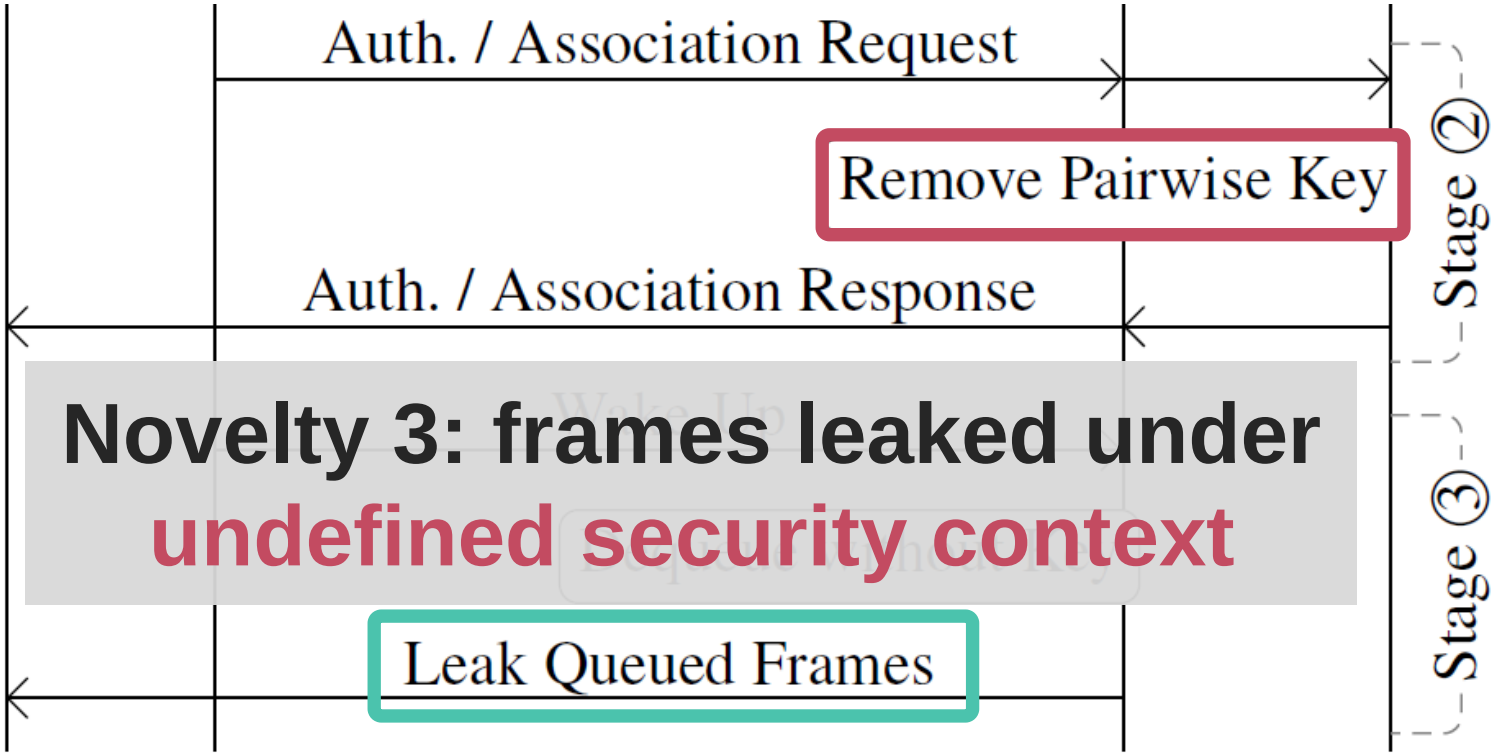


Novelty 2: connect to remove client's keys

Attack 1: leaking frames



Attack 1: leaking frames



Novelty 3: frames leaked under undefined security context

Undefined security context: FreeBSD example

How the frame is leaked depends on kernel version & driver:

Version	driver (vendor)	Leakage
13.0	run (Ralink)	Plaintext
13.1	run (Ralink)	WEP with all-zero key
13.1	rum (Ralink)	CCMP with group key
13.1	rtwn (Realtek)	CCMP with group key

- Malicious insiders know the group key!
- Linux, NetBSD, open Atheros firmware also affected

Root cause

Standard isn’t explicit on how to manage buffered frames

- Should drop buffered frames when refreshing/deleting keys

Frames are buffered in plaintext

- Alternative: encrypt frames *before* buffering them (like TLS)

New attack 2:

Network Disruptions

Background: DoS attacks

Well-known DoS attacks:

- Deauthentication: spoof “disconnect” frames
- Association: spoof “I want to connect” frames

Both remove connection state of the victim

Defense:

- **M**anagement **F**rame **P**rotection (**MFP** = 802.11w)
- This defense is required in WPA3

Management Frame Protection

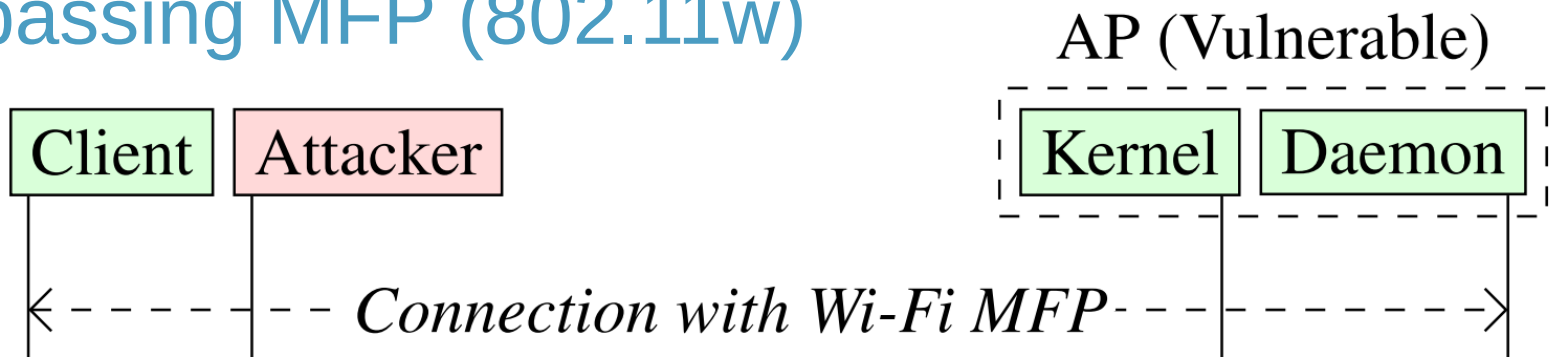
Wi-Fi has three frame types:

1. **Management**: network scanning, disconnecting,…
2. **Control**: acknowledgements, request to send,…
3. **Data**: transporting higher-layer data

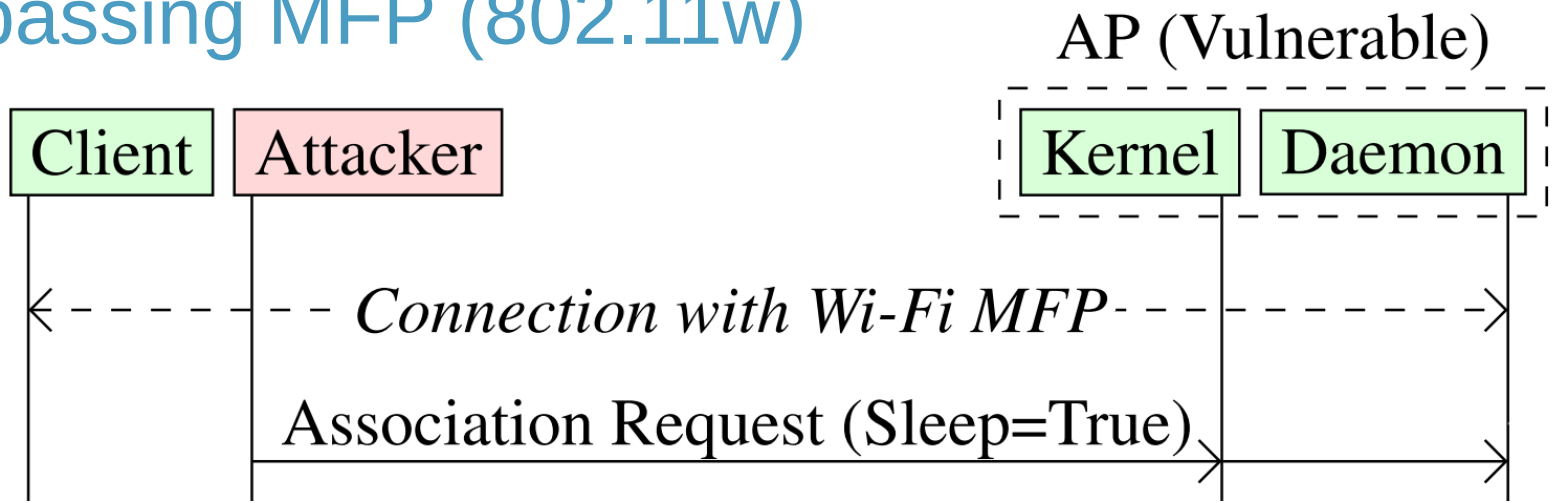
New Wi-Fi-certified devices must support MFP

- Can no longer trivially **deauthenticate** (disconnect) clients
- Late 2021: close to 5% of networks supported MFP [SRV21]

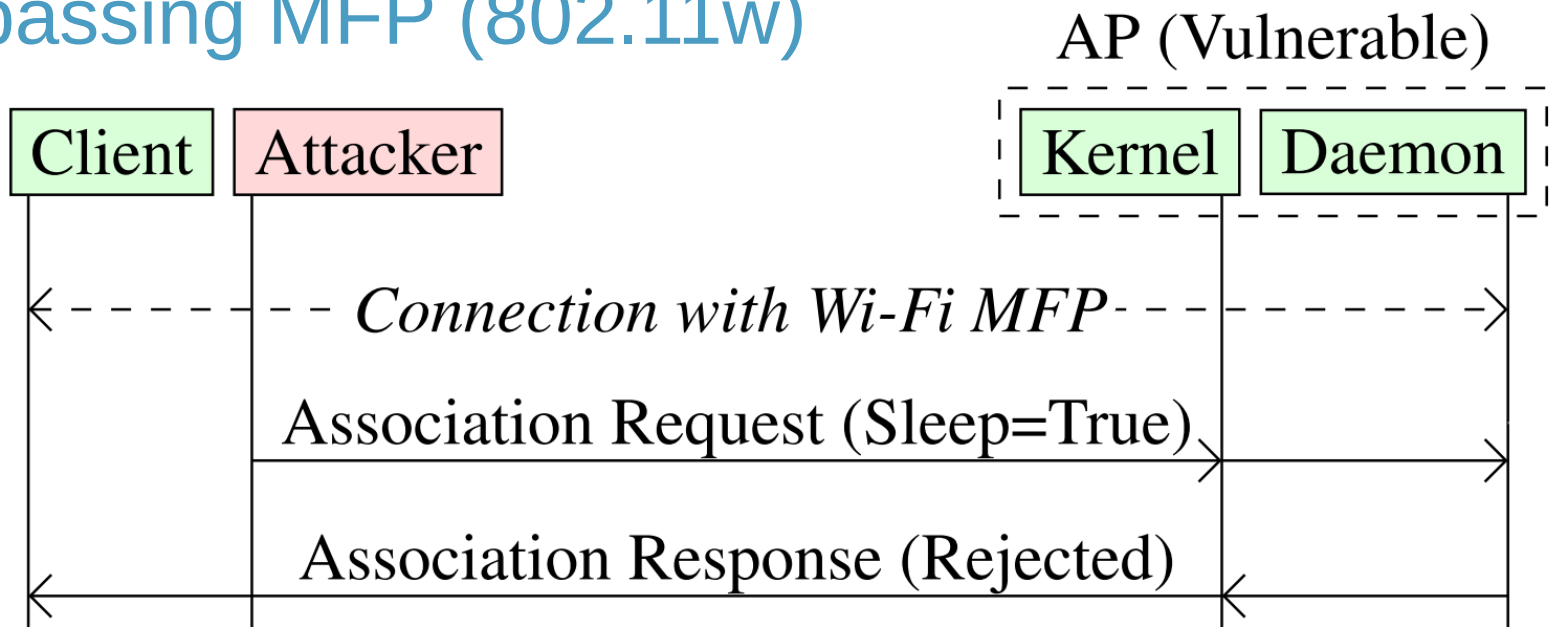
Bypassing MFP (802.11w)



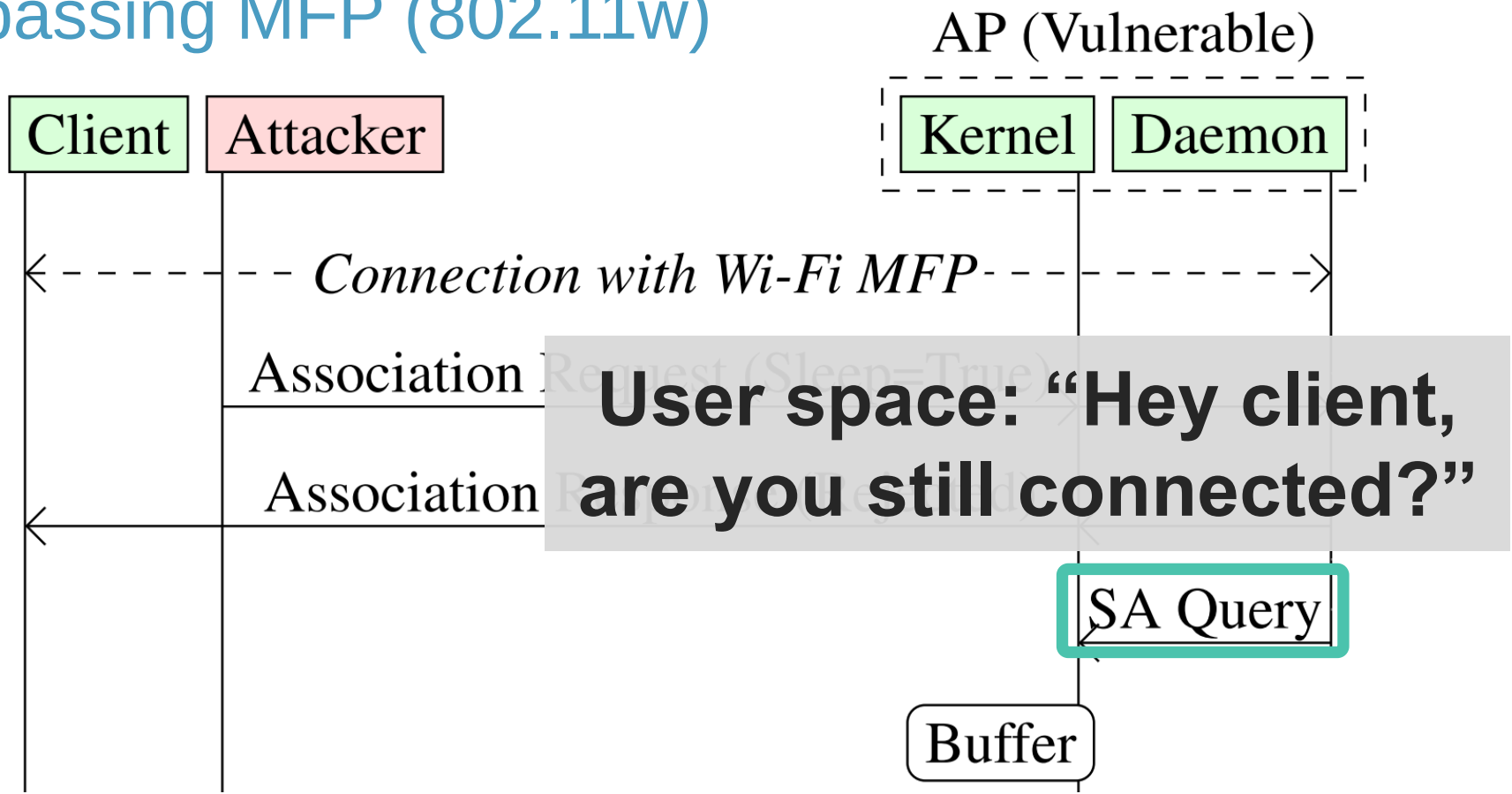
Bypassing MFP (802.11w)



Bypassing MFP (802.11w)

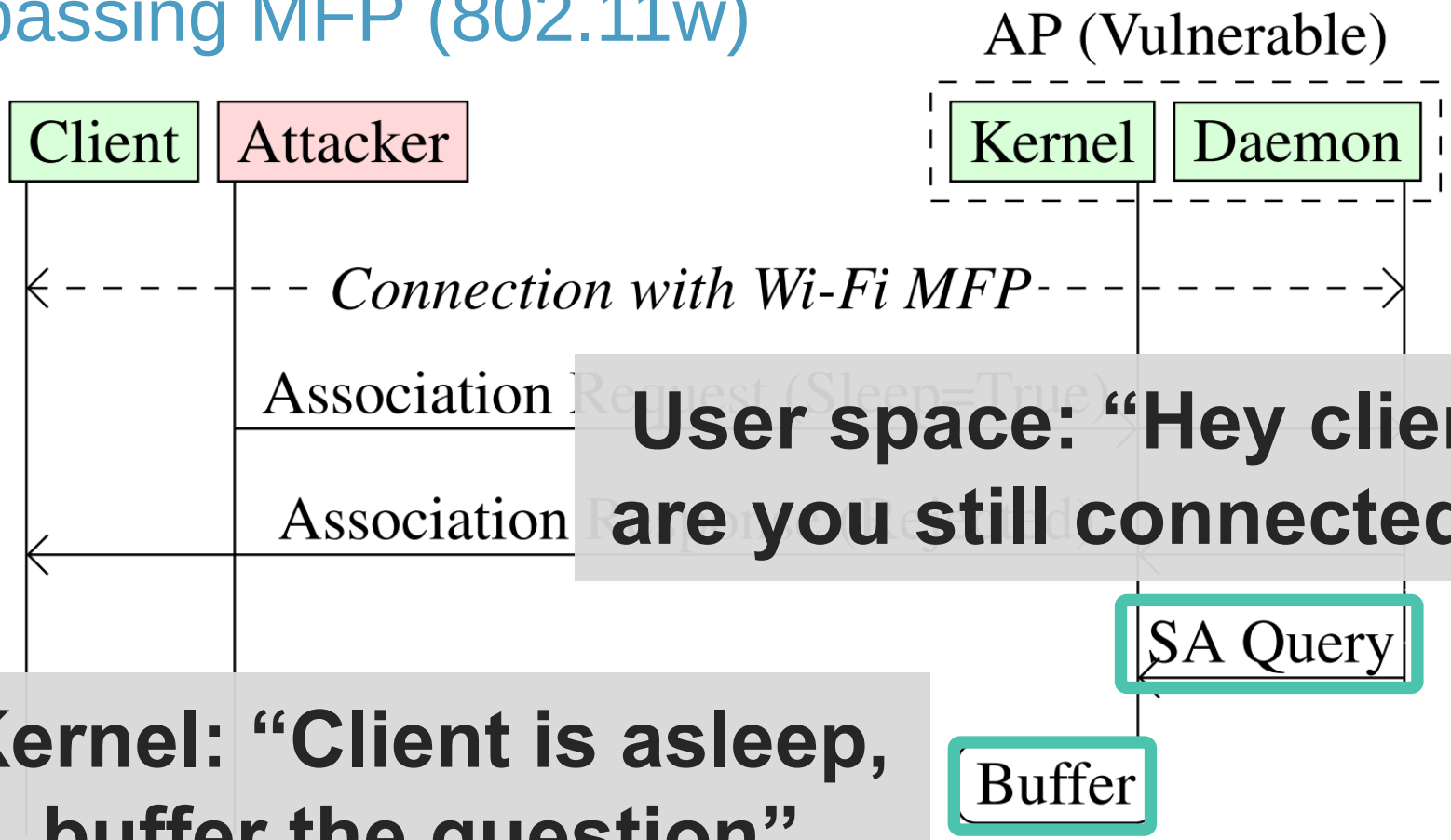


Bypassing MFP (802.11w)



User space: "Hey client, are you still connected?"

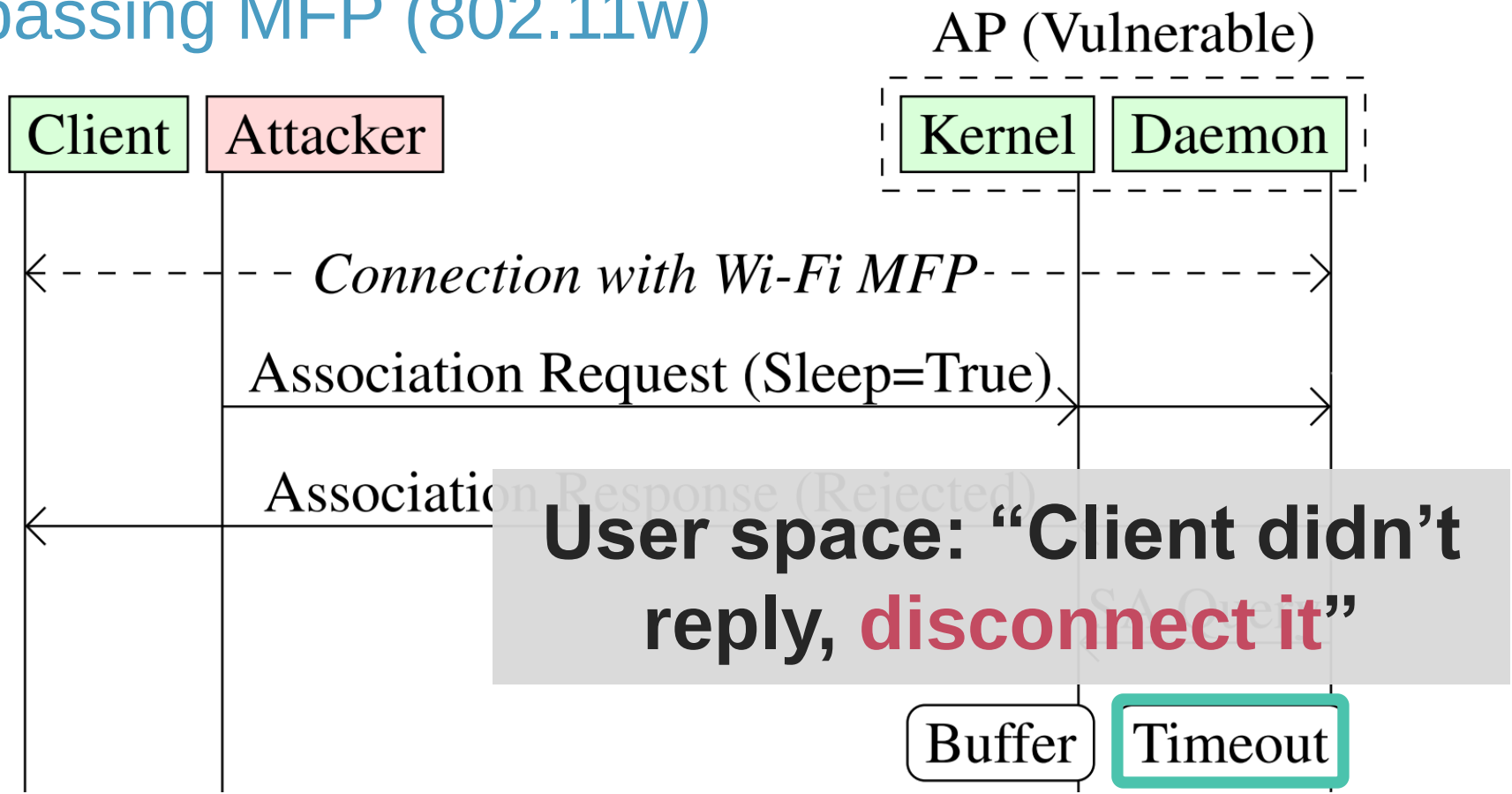
Bypassing MFP (802.11w)



User space: "Hey client, are you still connected?"

Kernel: "Client is asleep, buffer the question"

Bypassing MFP (802.11w)



User space: “Client didn’t reply, disconnect it”

Other Attacks & Defenses

Can also **force buffering of Fine Timing Measurements** frames

- Used to measure distance to AP and localize device
- For details, see our USENIX Security ‘23 paper “Framing Frames”

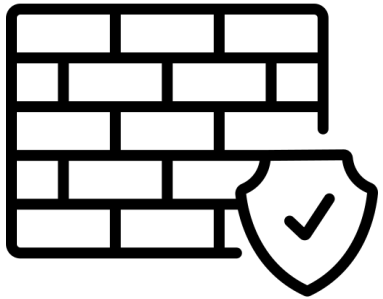
Defenses:

- Never buffer “are you still connected?” frames
- Authenticate the sleep bit in the header of Wi-Fi frames

New attack 3:

Bypassing client isolation

What is client isolation?



Blocks traffic between clients:

- Clients **cannot attack each other**
- ARP spoofing is not possible

All clients have unique encryption keys:

- Prevents “Hole 196” attack (Black Hat ’10)

→ **Defends against malicious insiders**

Attack 2: bypassing Wi-Fi client isolation

Attack targets networks that use **client isolation**:

- Defense against malicious or compromised internal clients
- Used by networks on large organizations, universities, hotspots,…

→ Attacker can connect to the network. But can't communicate with, or attack, others…

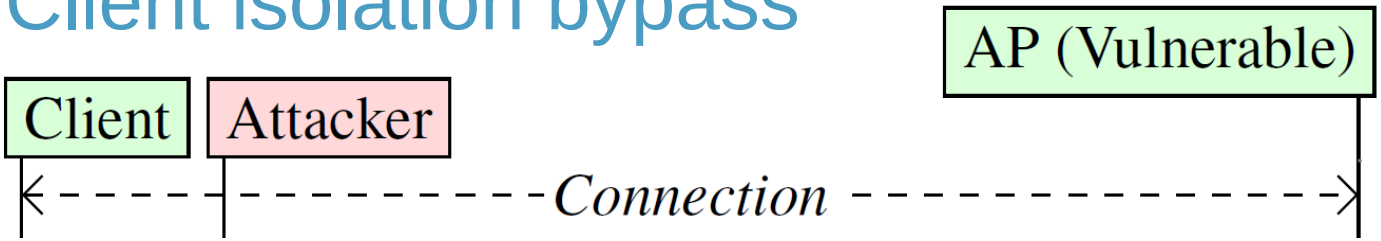
Attack 2: bypassing Wi-Fi client isolation

Attack targets networks that use **client isolation**:

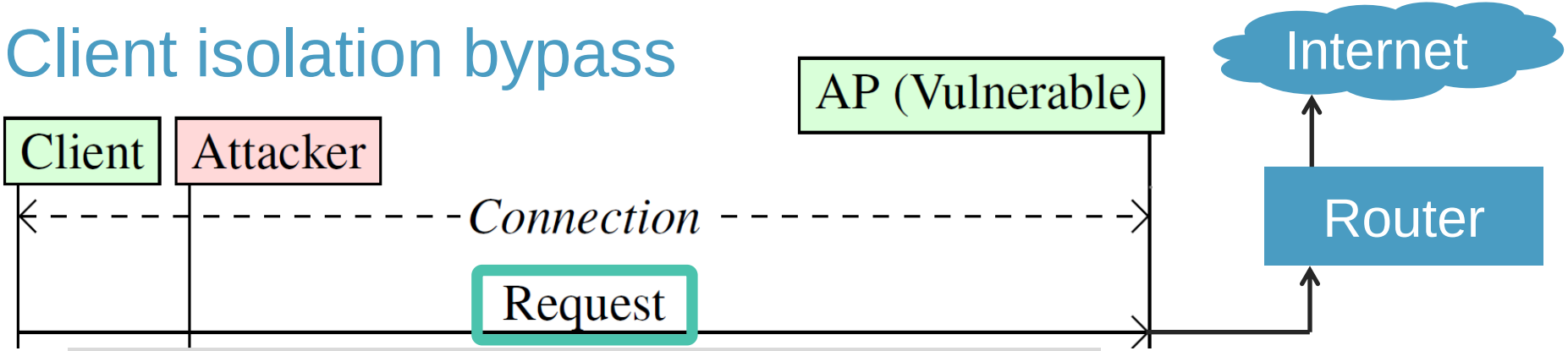
- Defense against malicious or compromised internal clients
- Used by networks on large organizations, universities, hotspots,…

→ Attacker can connect to the network. But can't communicate with, or attack, others… unless we manipulate the security context!

Client isolation bypass

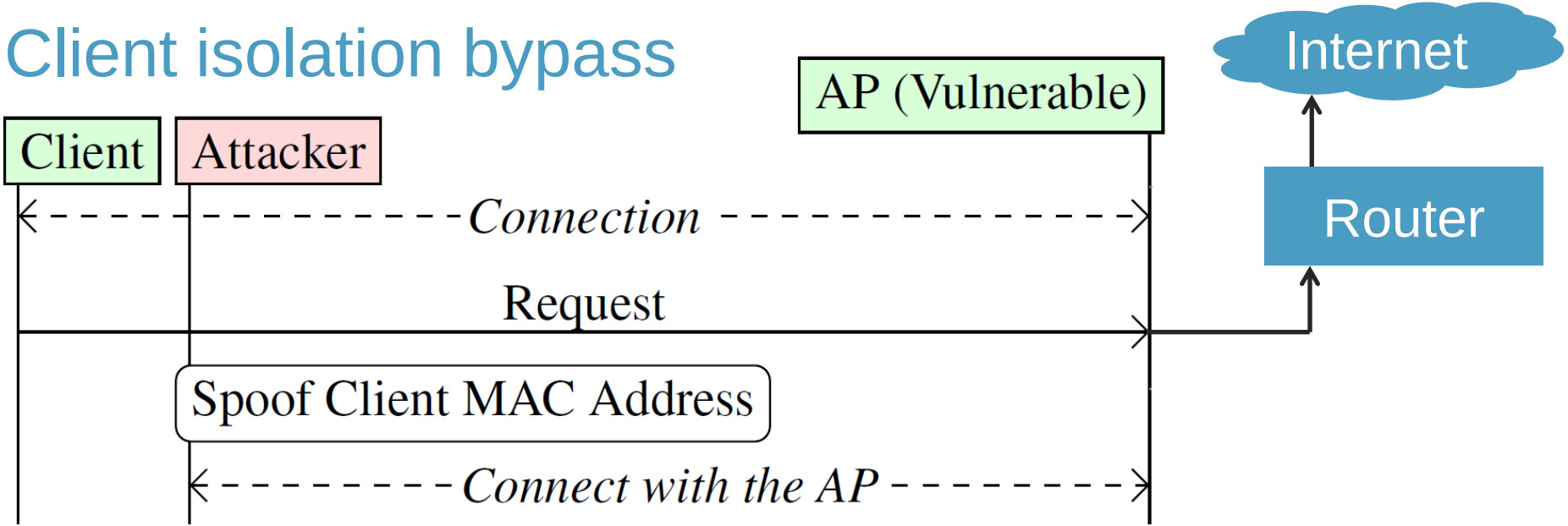


Client isolation bypass

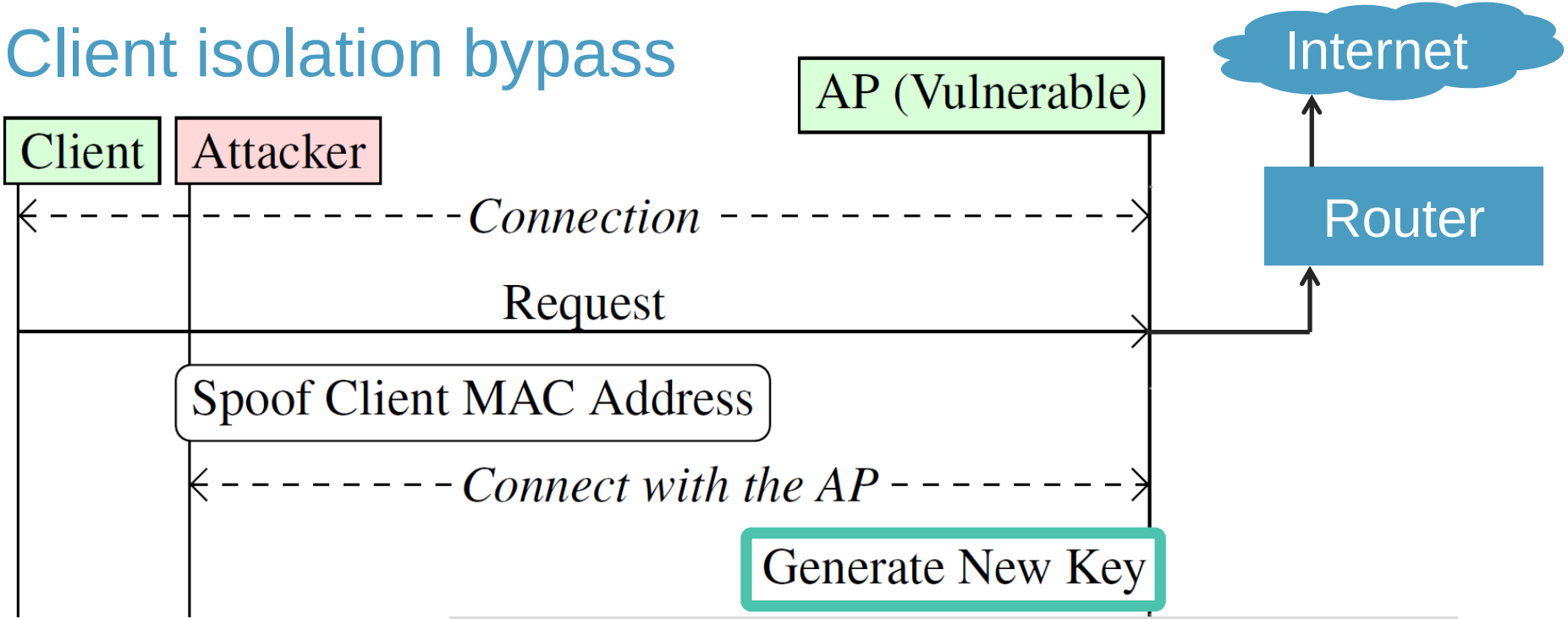


E.g., DNS or HTTP request

Client isolation bypass

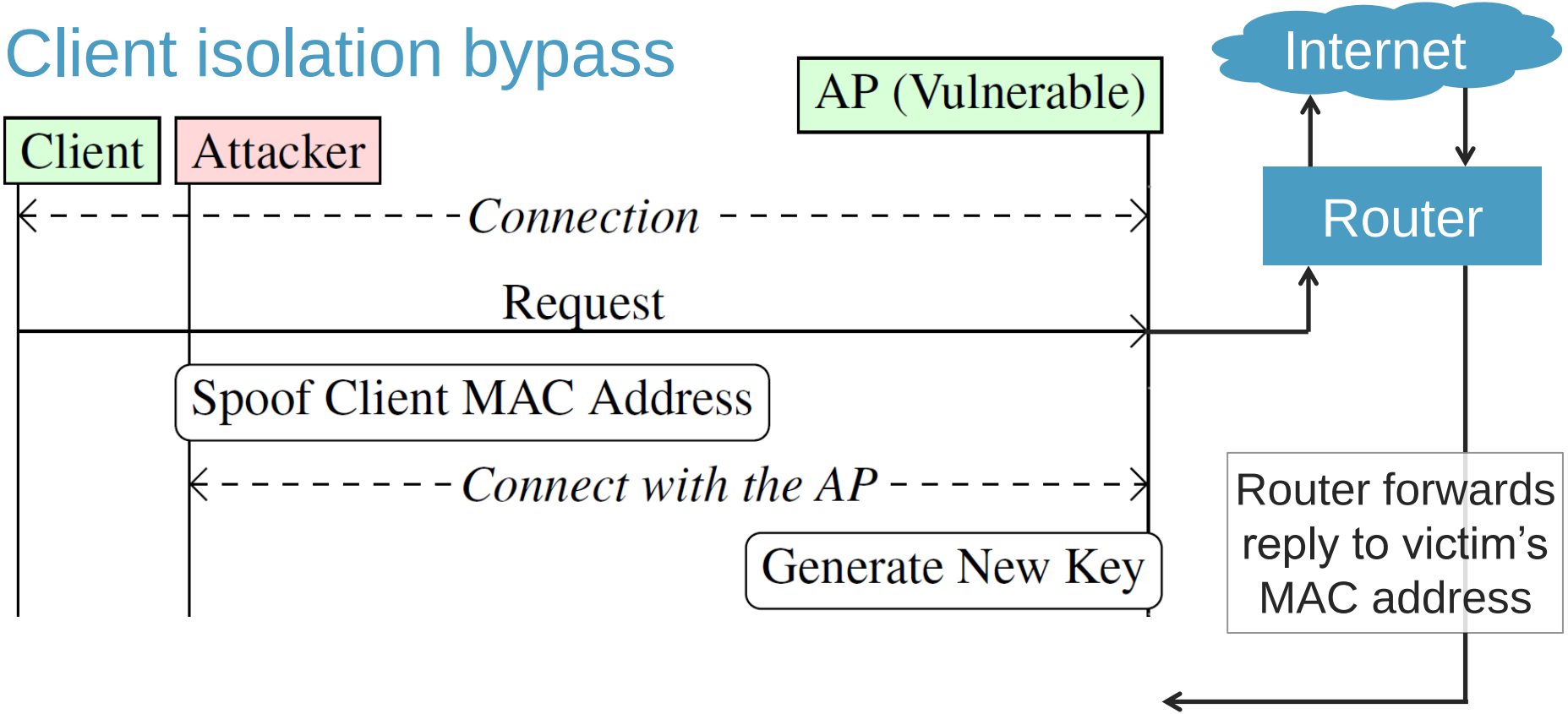


Client isolation bypass

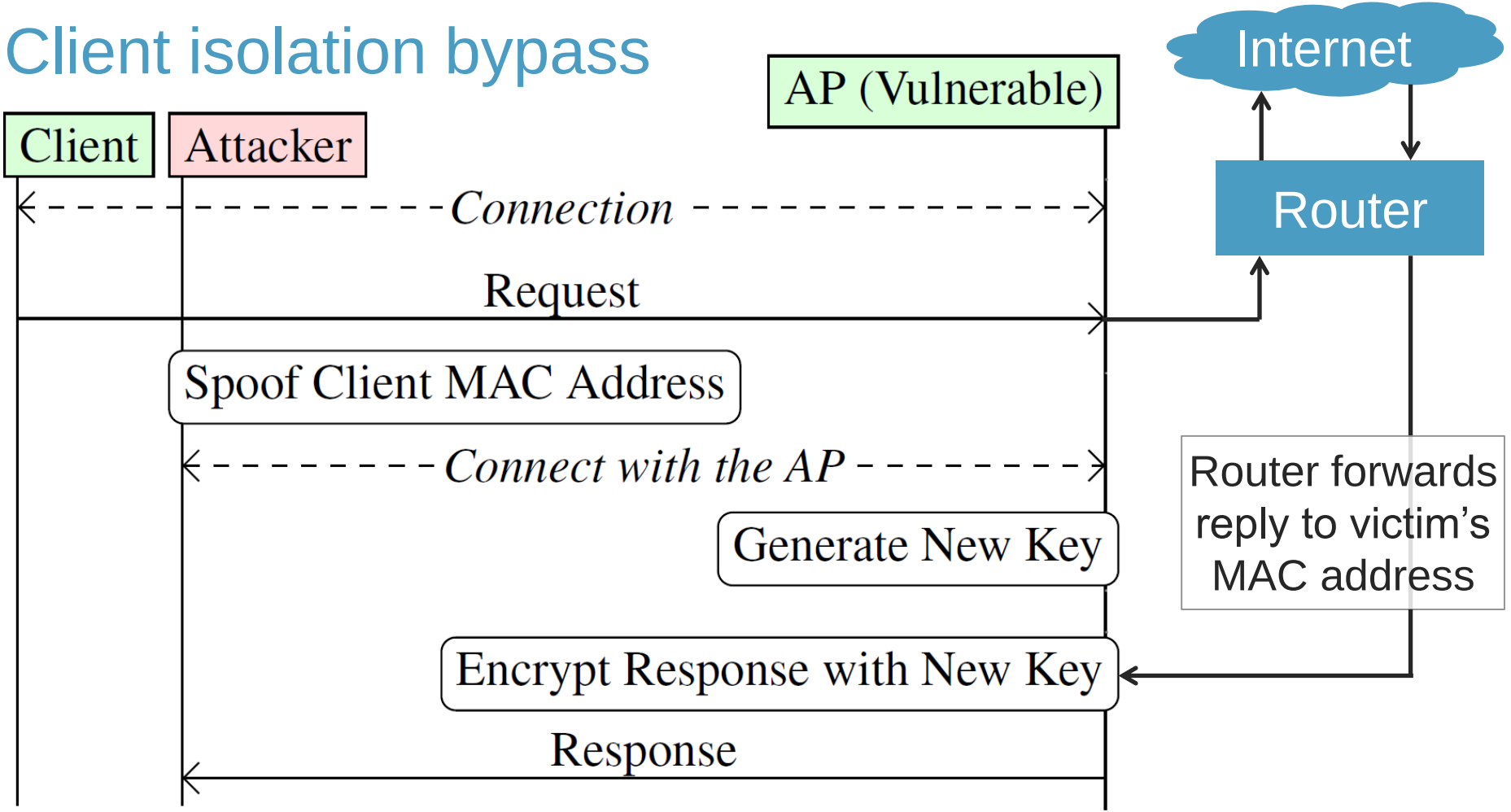


New key is associated with the victim's MAC address

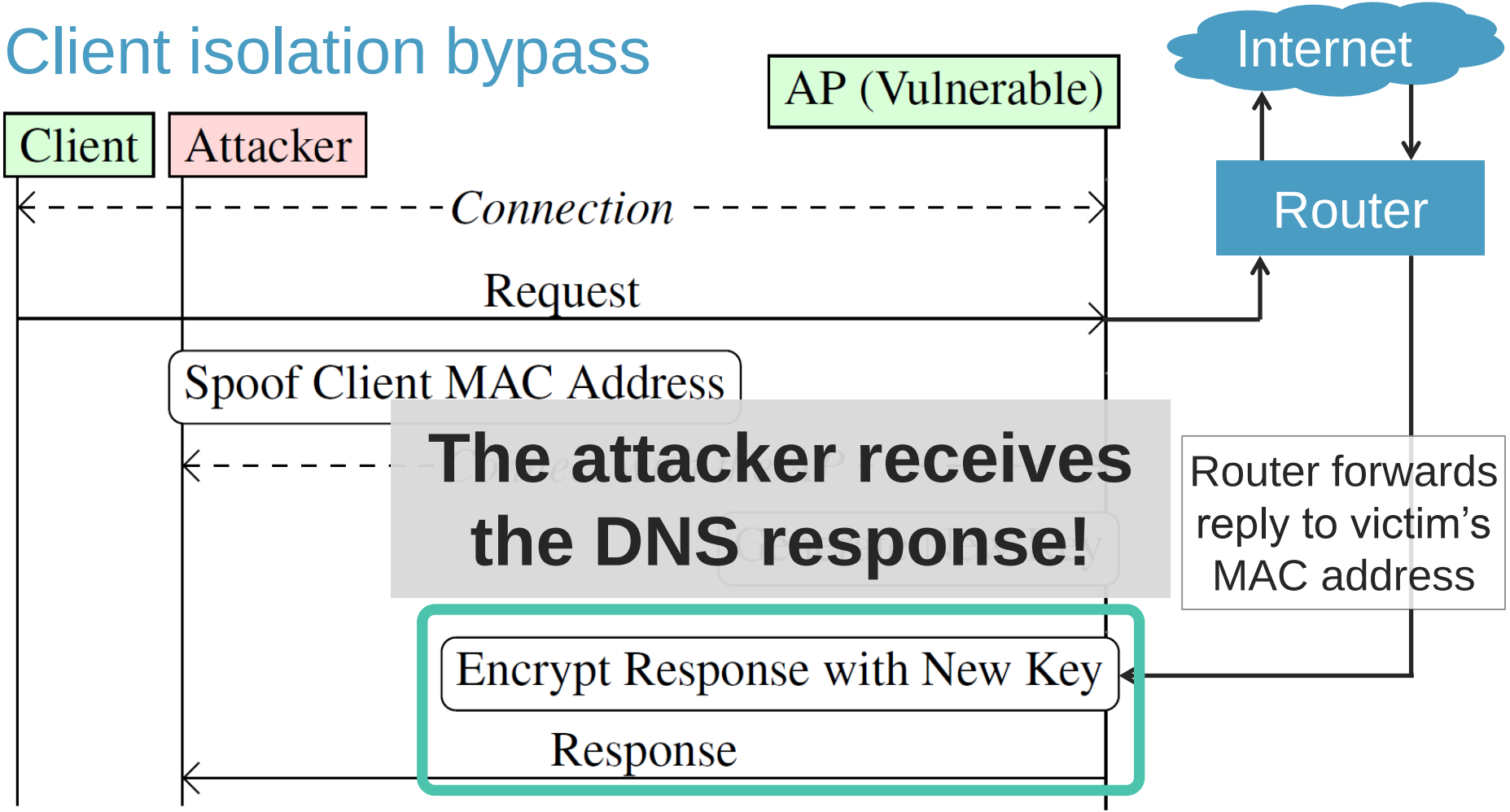
Client isolation bypass



Client isolation bypass



Client isolation bypass



Tool to test devices: MacStealer

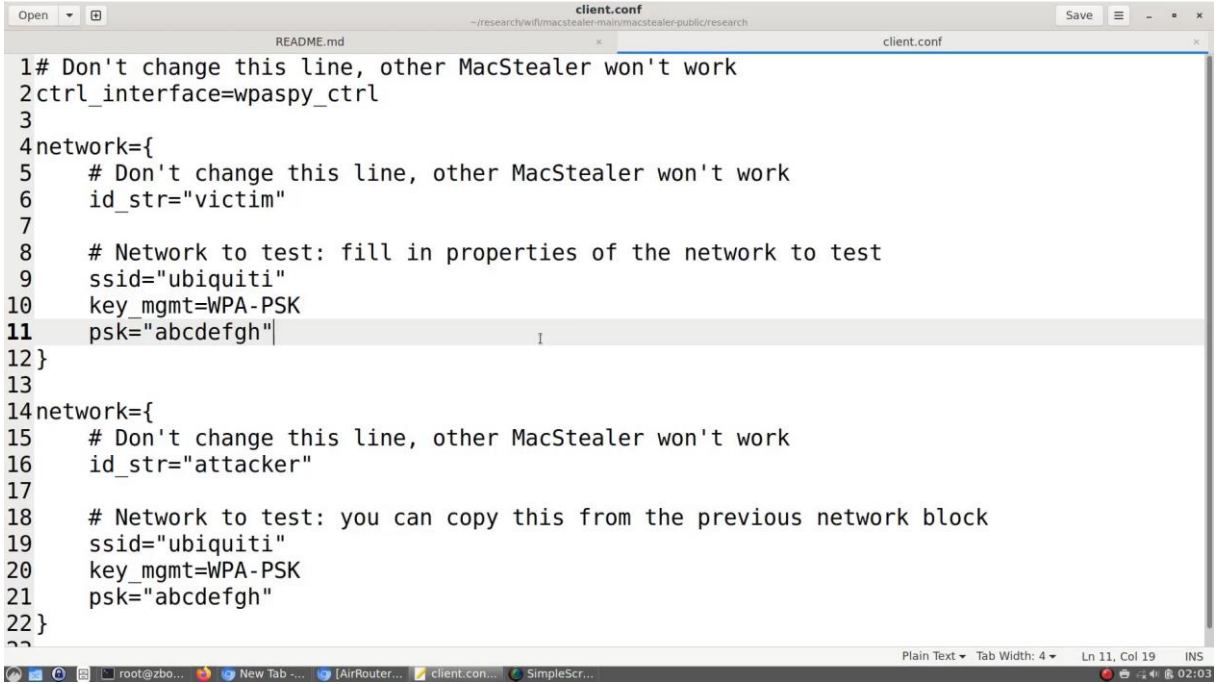
Command	Short description
Sanity checks	
`./macstealer.py wlan0 --ping`	Connect as victim & test server's retransmission behavior.
`./macstealer.py wlan0 --ping --flip`	Connect as attacker & test server's retransmission behavior.
Vulnerability tests	
`./macstealer.py wlan0`	Test the default variant of the MAC address stealing attack.
`./macstealer.py wlan0 --other-bss`	Let the attacker connect with a different AP than the victim.
Client isolation: Ethernet layer	
`./macstealer.py wlan0 --c2c wlan1`	Test client isolation on the Ethernet layer.
`./macstealer.py wlan0 --c2c-eth wlan1`	Test client isolation on the Ethernet layer (DNS).

Sanity checks

Vulnerability tests

Does the network use client isolation?

MacStealer demo



→ Ubuiqiti is one of the few vendors that implemented a mitigation!

Experiments

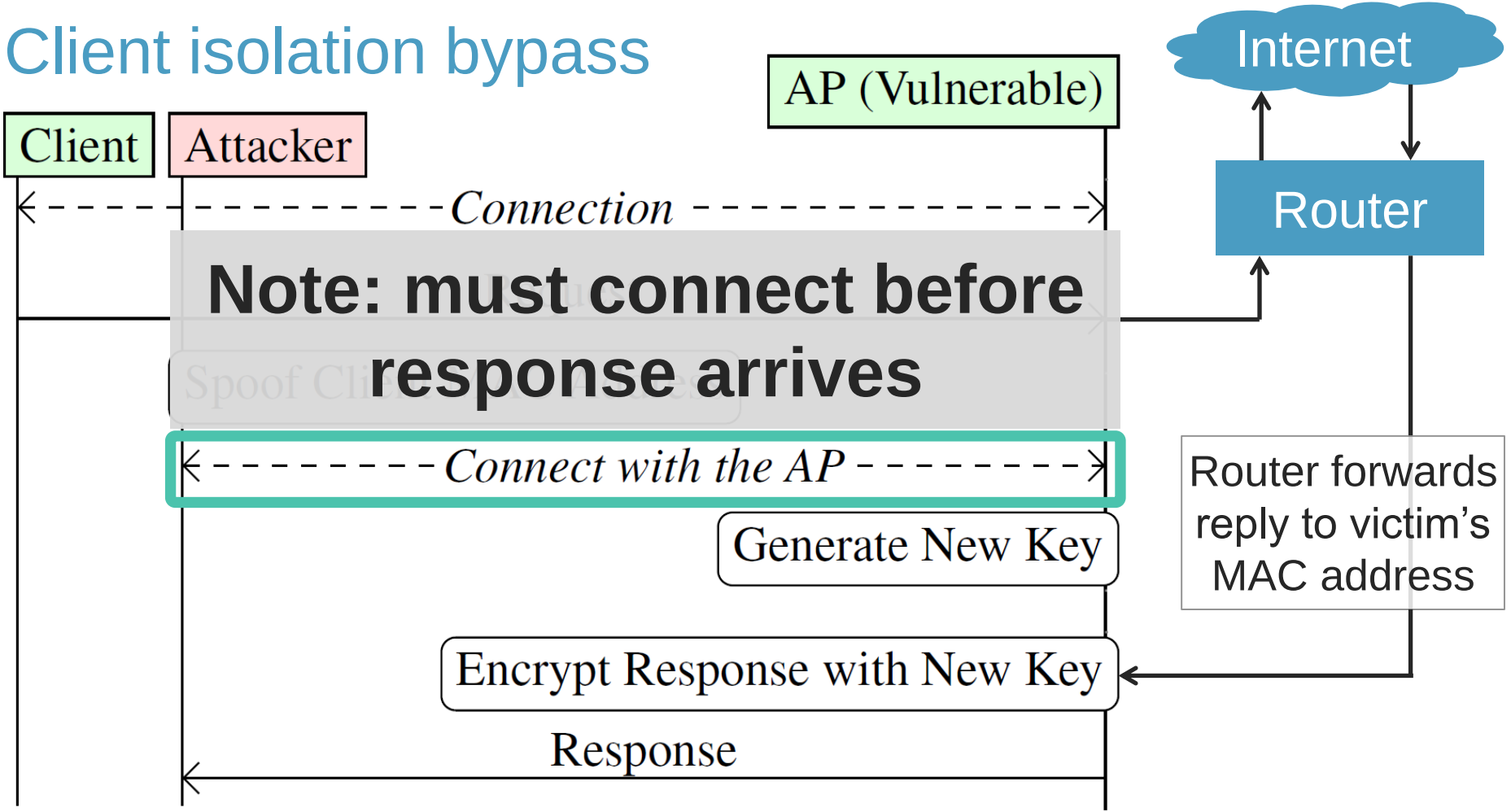
All tested professional & home APs were vulnerable

→ **Design flaw** in Wi-Fi client isolation!

→ Useful test for auditors

github.com/vanhoefm/macstealer

Client isolation bypass



Fast security context override

Technique to quickly reconnect. Experiments:

- Minimum reconnect time: ~12 ms
- Average UDP response time: [Verizon]
  - Transatlantic connections: ~70 ms
  - Connections within Europe: ~13 ms
- TCP responses are retransmitted → trivial to intercept
- Adversary can spoof MAC address of the default gateway

[Verizon] Verizon IP latency statistics

Root cause

Client identities are not bound to each other:

- Authenticated Wi-Fi 802.1X identity (username)
- But spoofable IP/MAC addresses

  → Wi-Fi attacker can spoof victim's identity on other layers

Other observation: client isolation was "bolted on" by vendors

- Not part of IEEE 802.11 standard → less studied

Fixing client isolation

One defense is disallowing recently-used MAC address, unless:

- Certain amount of time has passed (incomplete defense)
- We know it's the same user as before (complete defense)
  - Based on 802.1X identity or cached keys (not always available)

→ These aren't ideal fixes: impractical, incomplete, unreliable,…

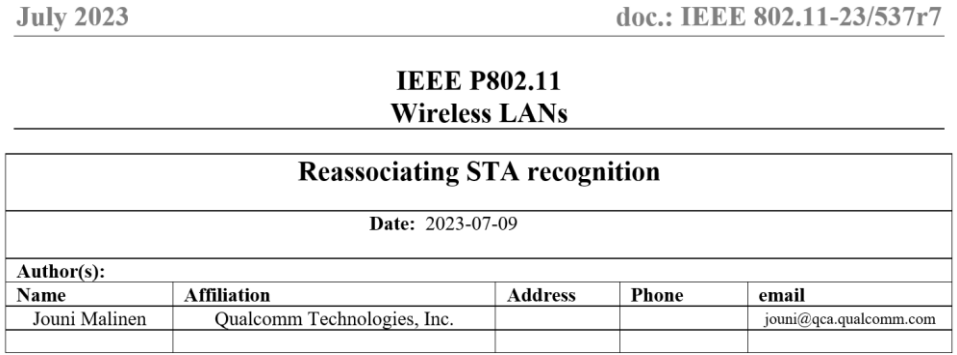
Current situation in practice

Currently few vendors implemented a defense or mitigation

- **Don’t rely on client isolation for security**
- Alternative: use VLANs to isolate groups

Accepted standard update:

- Recognize returning client
- Further updates being discussed in 802.11bh



July 2023 doc.: IEEE 802.11-23/537r7

IEEE P802.11
Wireless LANs

Reassociating STA recognition

Date: 2023-07-09

Author(s):

Name	Affiliation	Address	Phone	email
Jouni Malinen	Qualcomm Technologies, Inc.			jouni@qca.qualcomm.com

Conclusion

Standard is vague on how to manage buffered frames

- Can **leak frames** under different security context
- Important to **model/define transmit queues**

Can partially **bypass client isolation**

- All devices vulnerable → **design flaw**
- Hard to fully prevent