



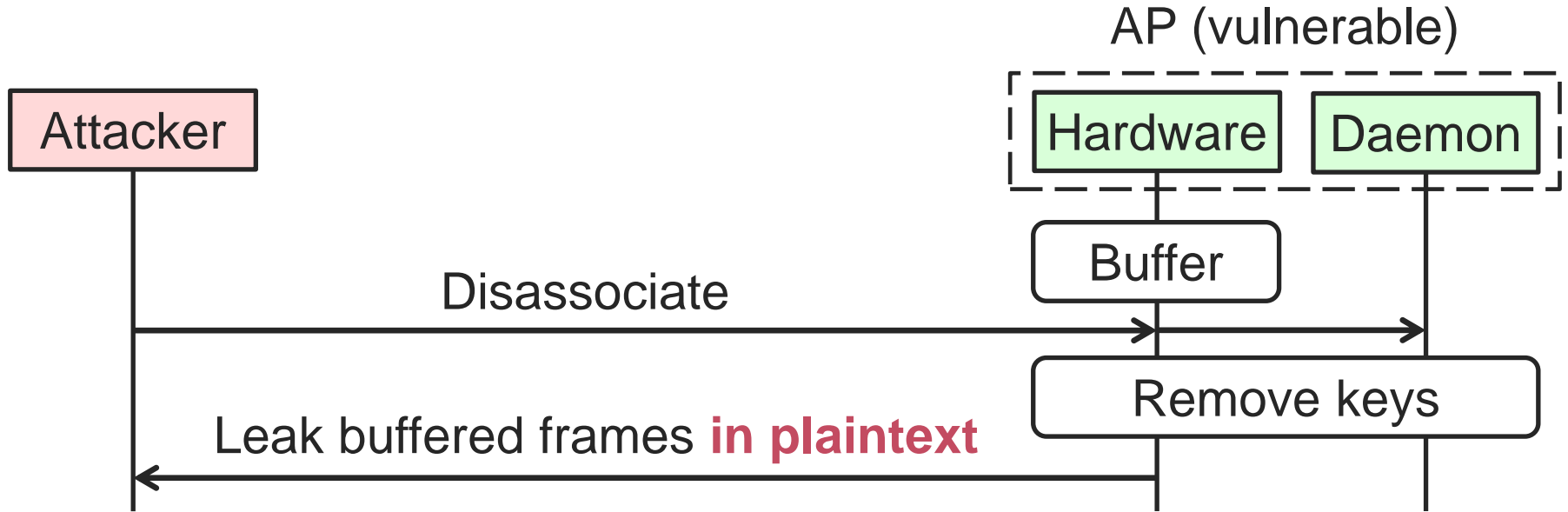
*Sweet Dreams:*  
**Abusing Sleep Mode to Break Wi-Fi  
Encryption & Disrupt WPA2/3 Networks**

Mathy Vanhoef, Domien Schepers,  
and Aanjhan Ranganathan

# 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]

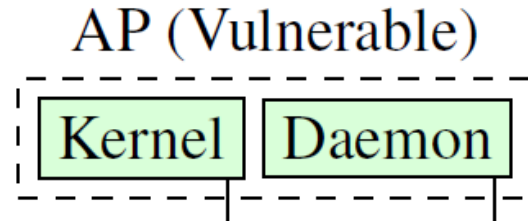
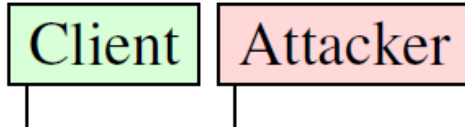
# Background: Kr00k implementation flaw



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

# New attack 1: leaking frames

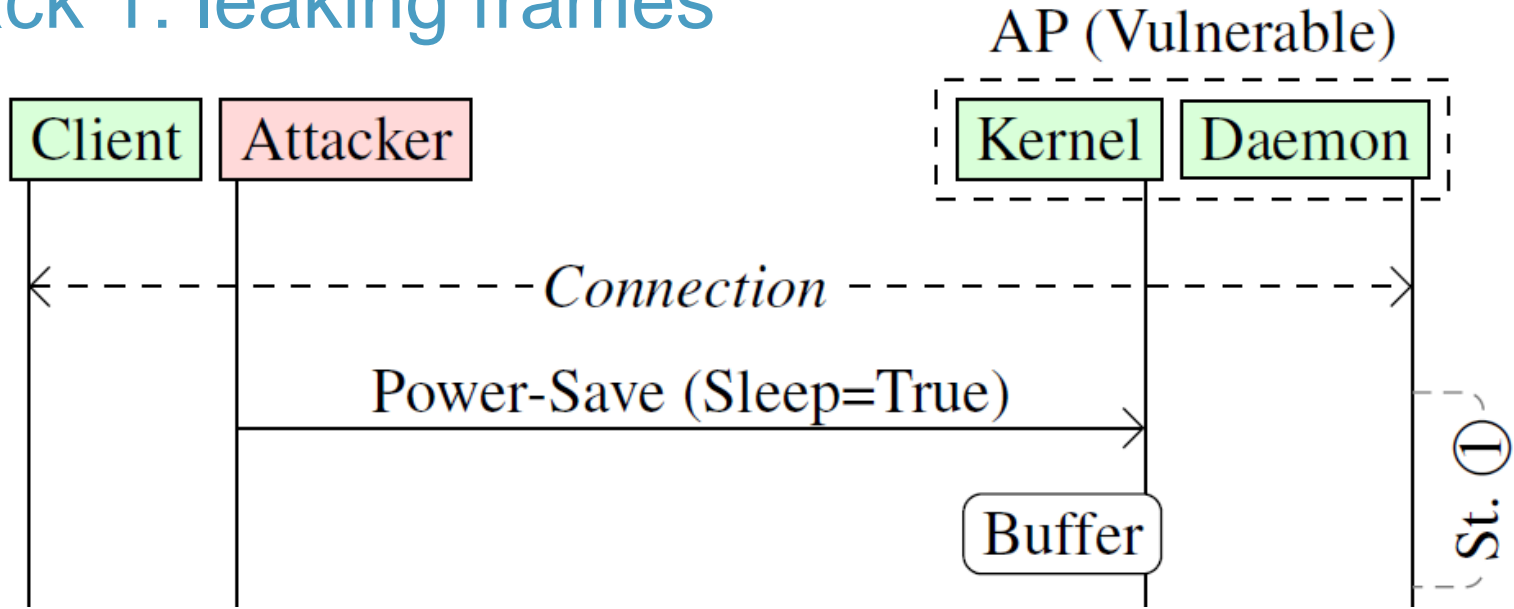
# Attack 1: leaking frames



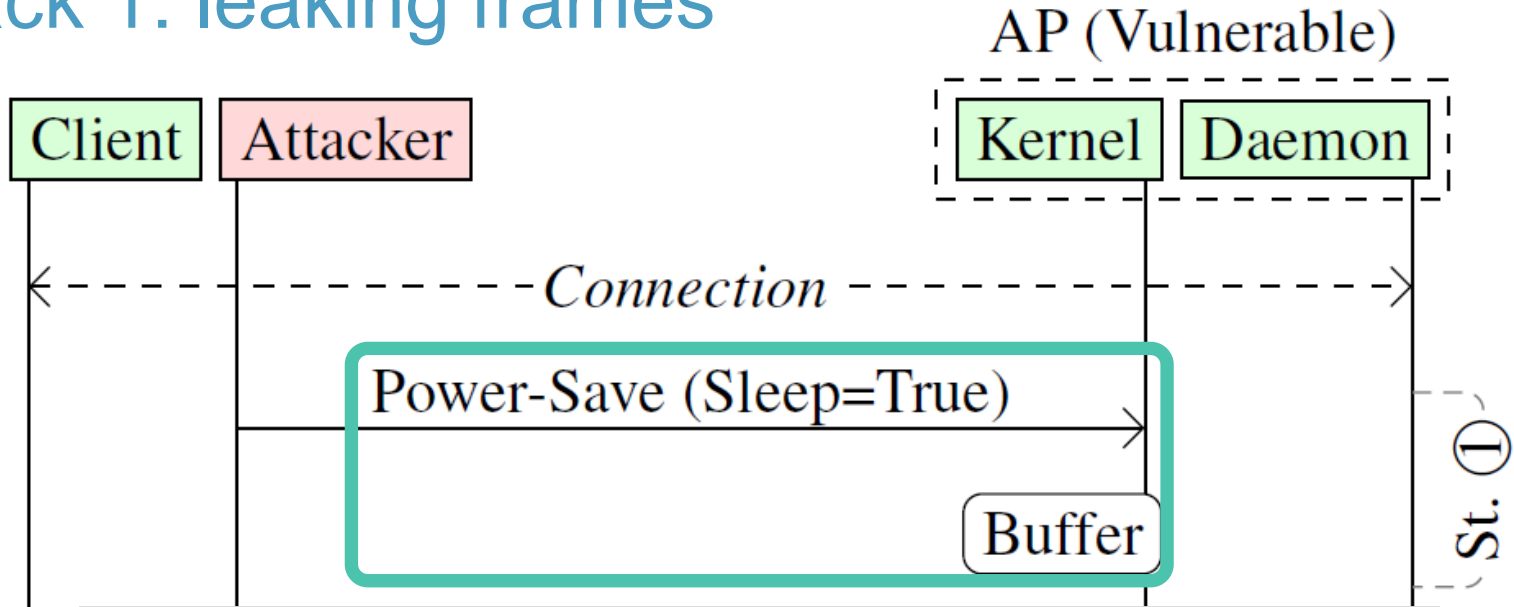
# Attack 1: leaking frames



# Attack 1: leaking frames

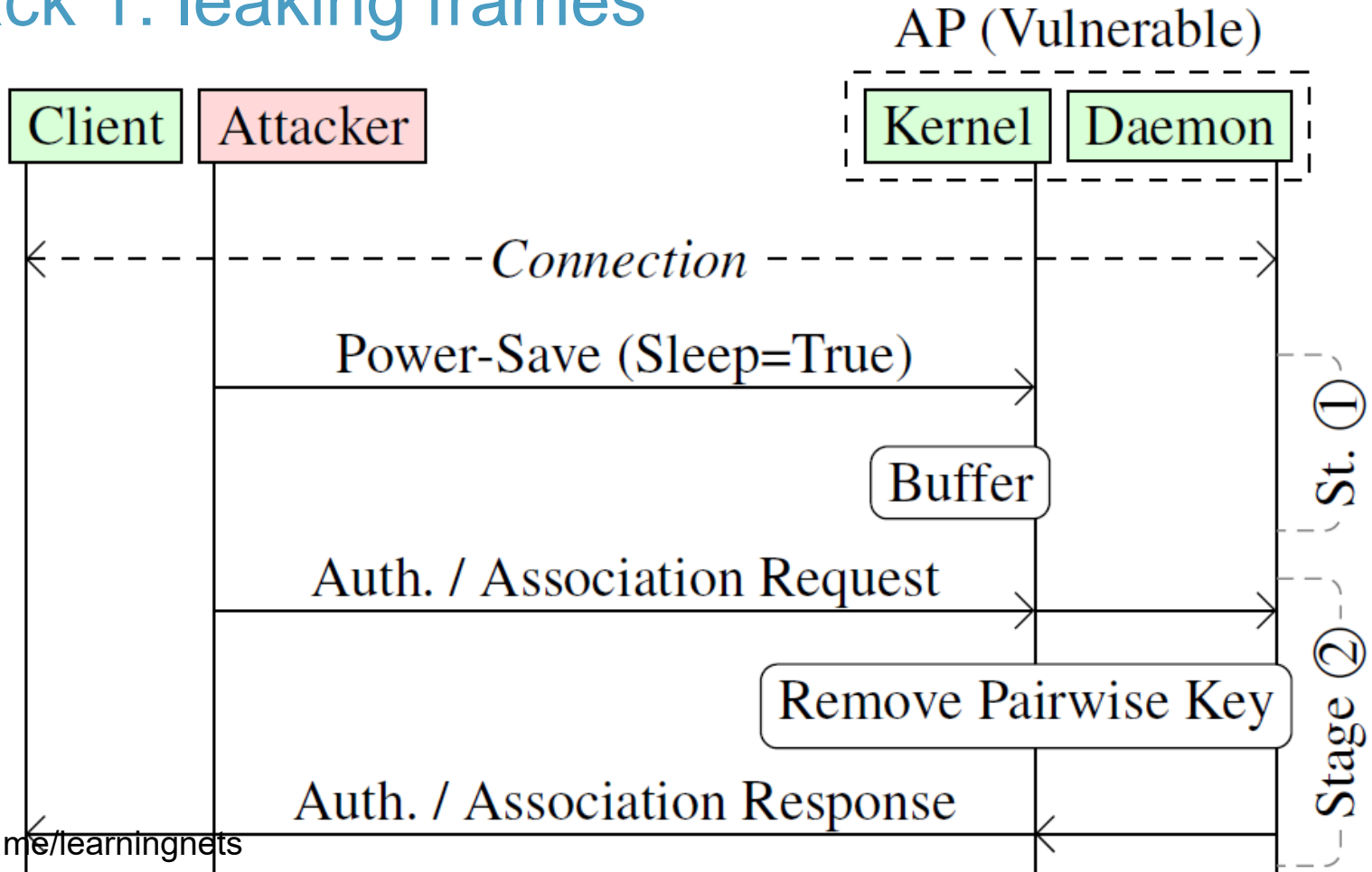


# Attack 1: leaking frames

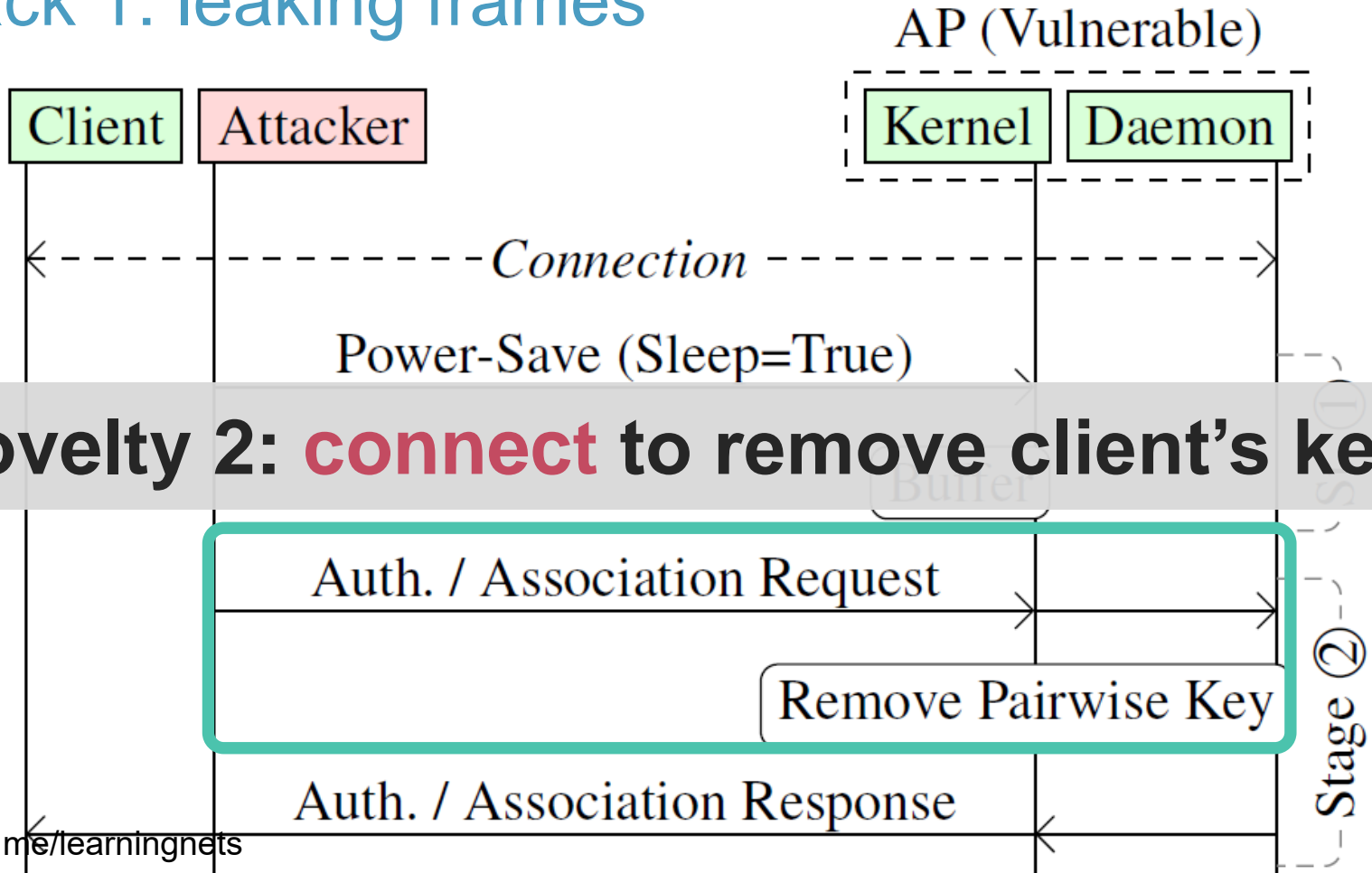


**Novelty 1: controlled buffering**

# Attack 1: leaking frames

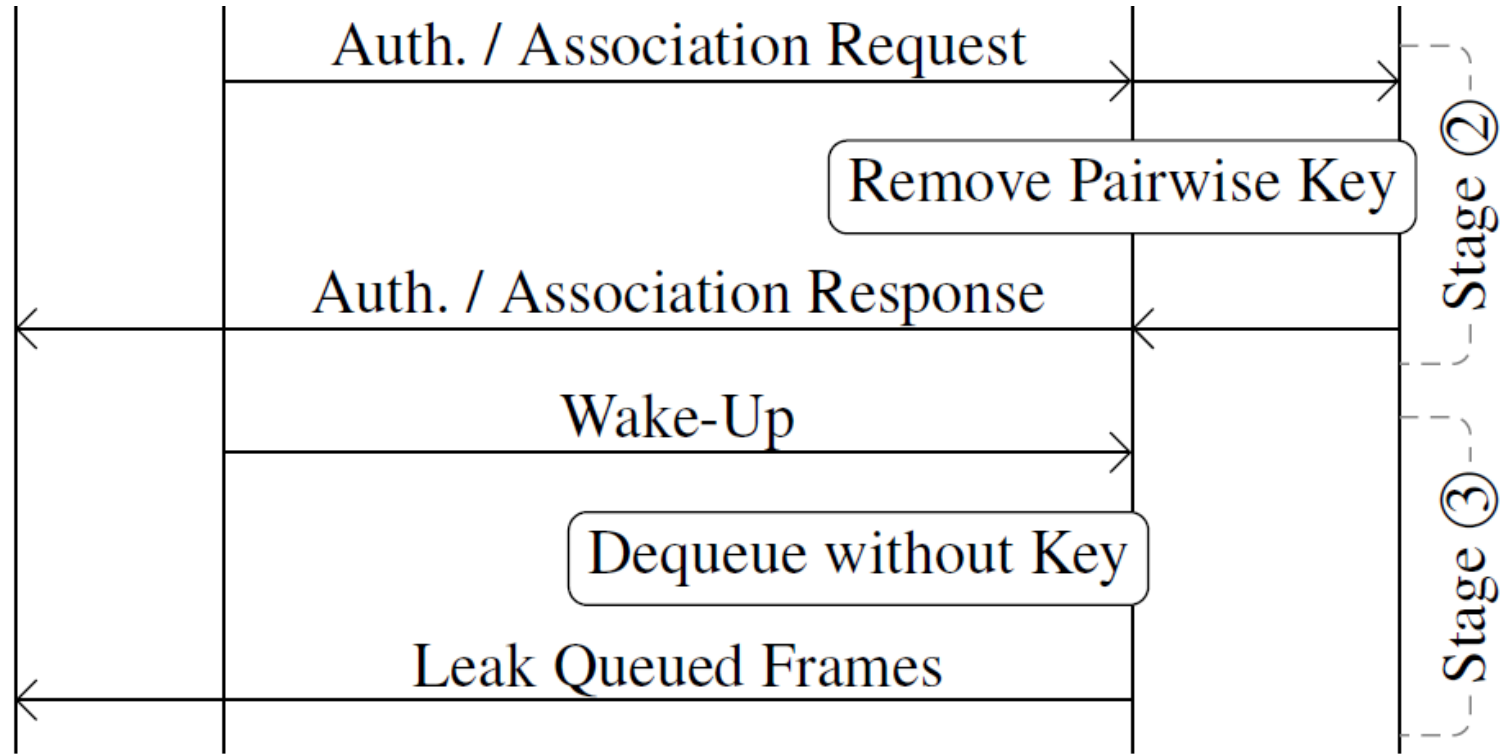


# Attack 1: leaking frames

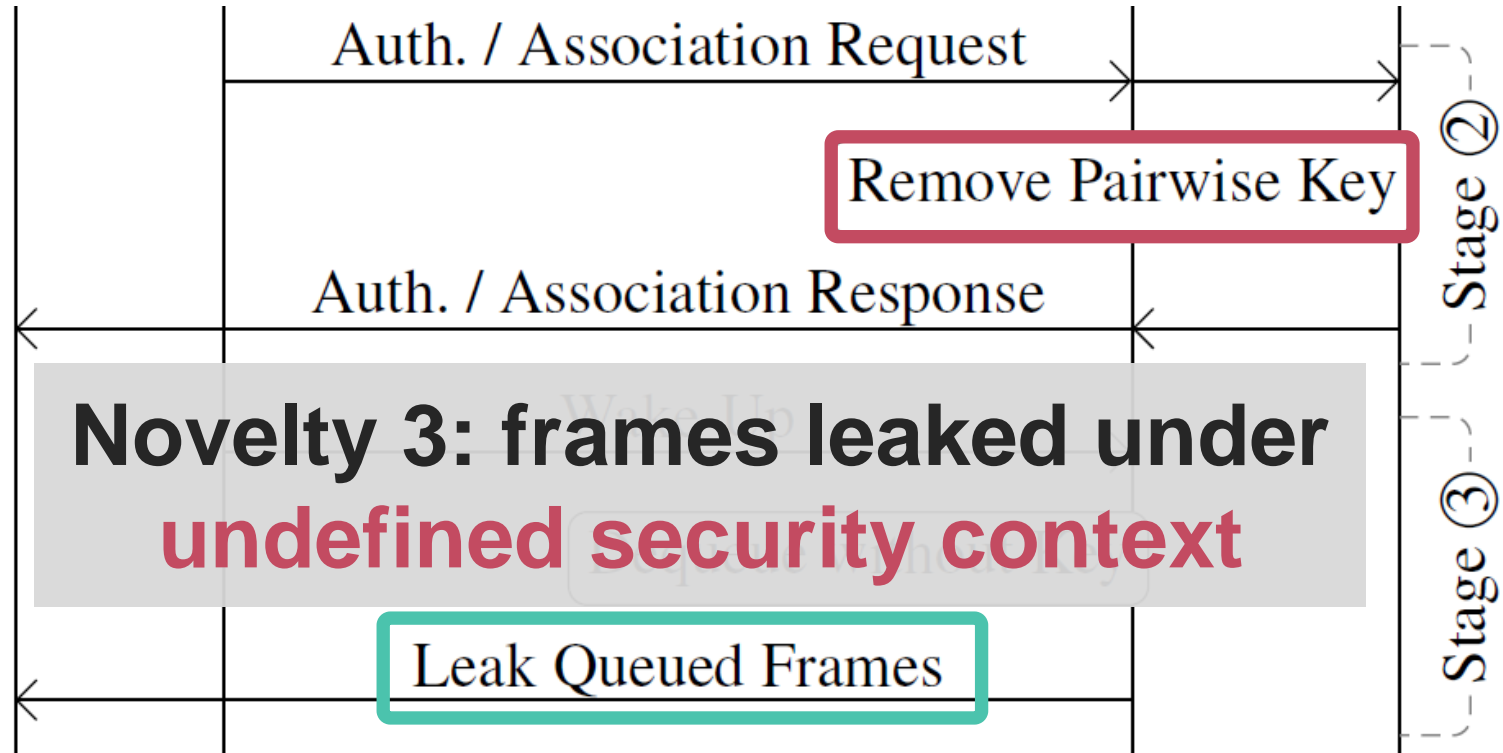


**Novelty 2: connect to remove client's keys**

# Attack 1: leaking frames



# Attack 1: leaking frames



# 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

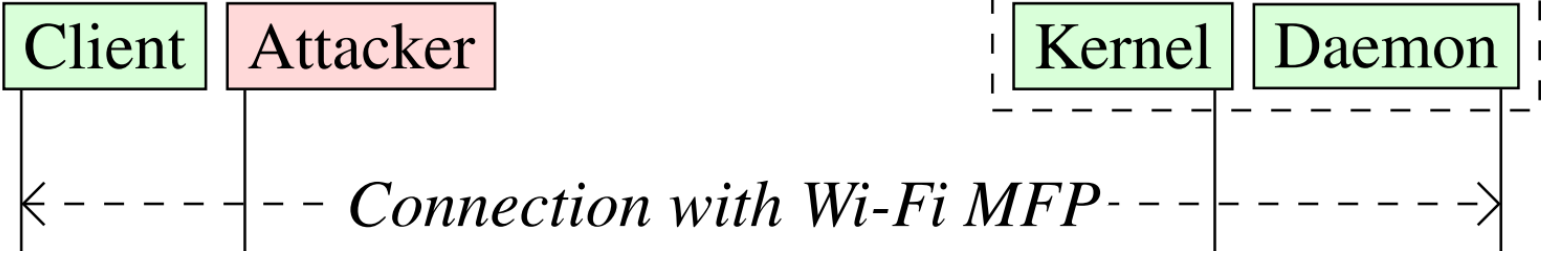
Both remove connection state of the victim



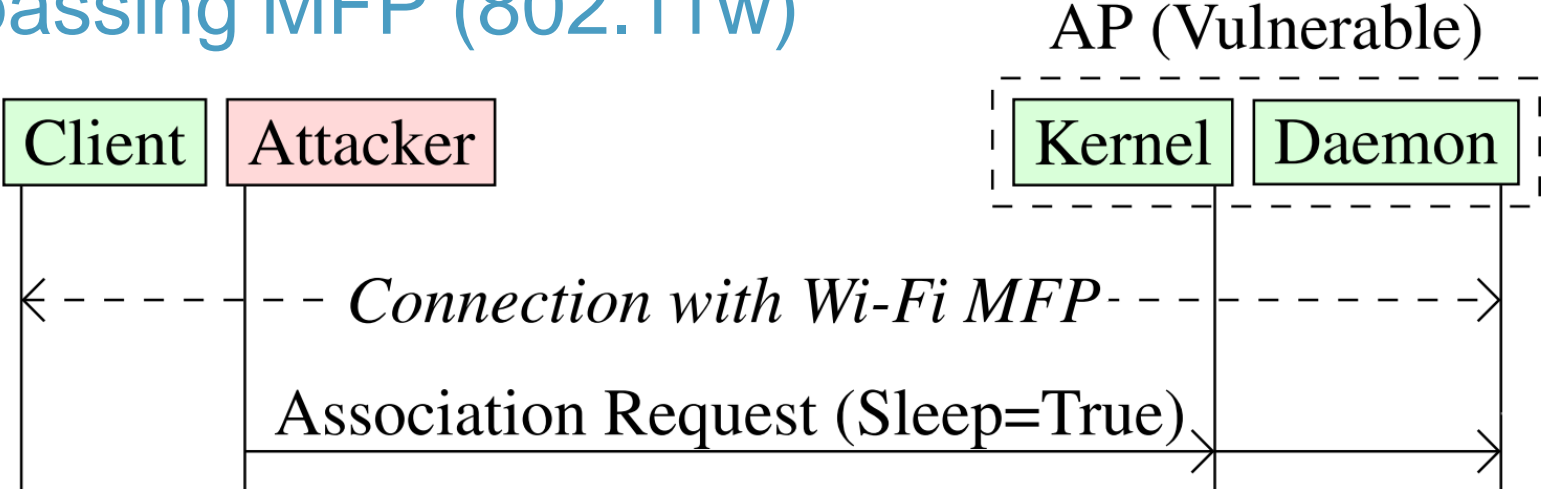
Defense:

- › Management Frame Protection (**MFP** = 802.11w)
- › This defense is required in WPA3

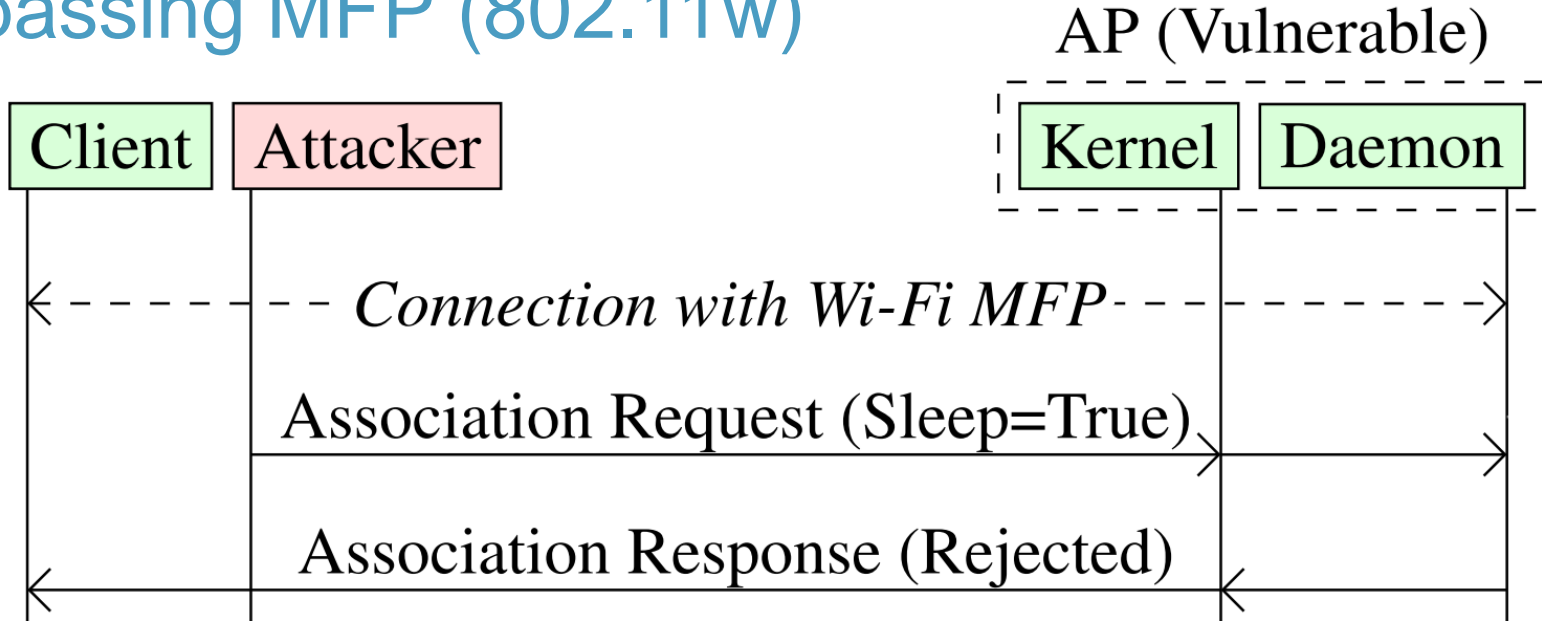
# Bypassing MFP (802.11w)



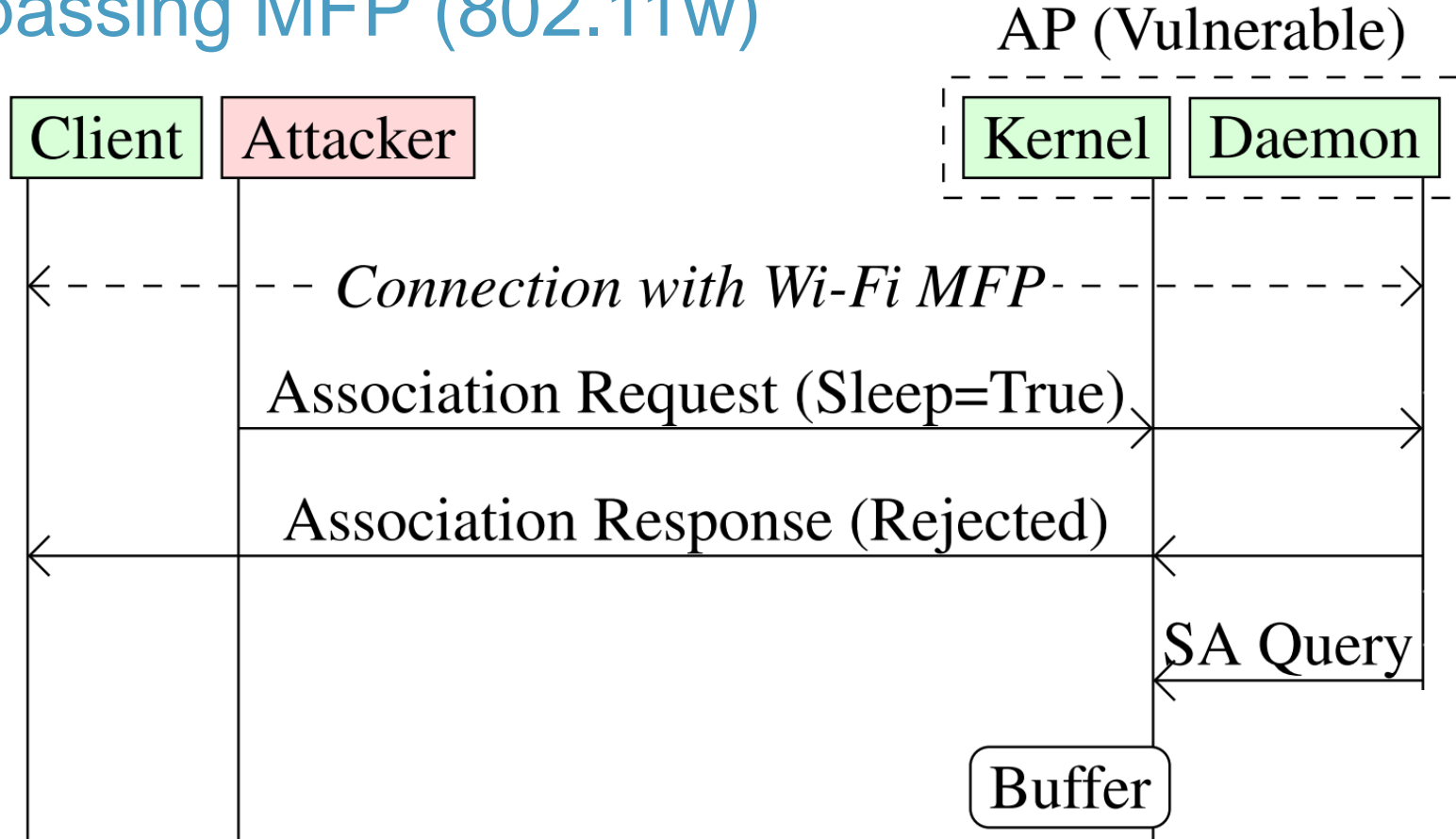
# Bypassing MFP (802.11w)



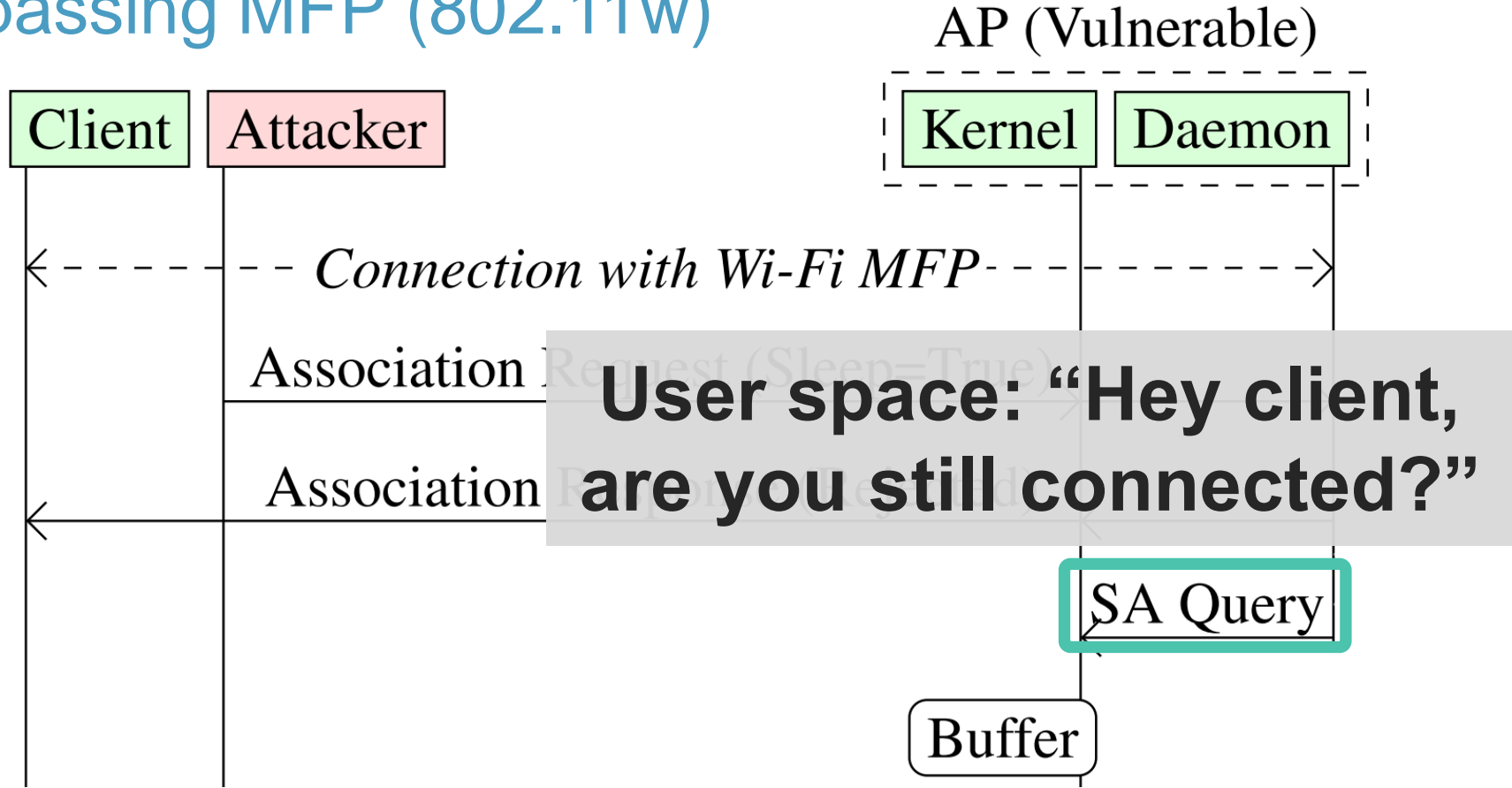
# Bypassing MFP (802.11w)



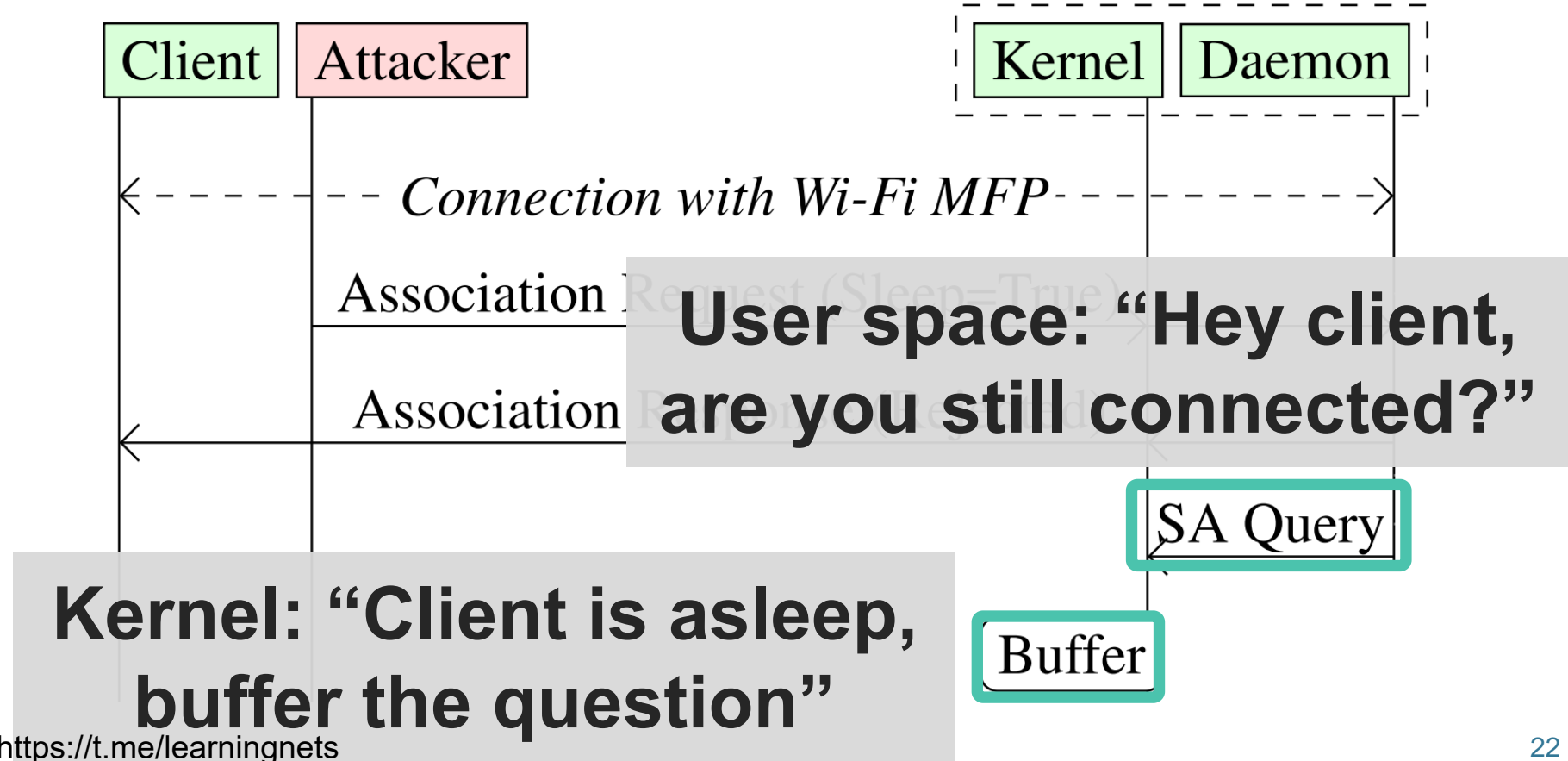
# Bypassing MFP (802.11w)



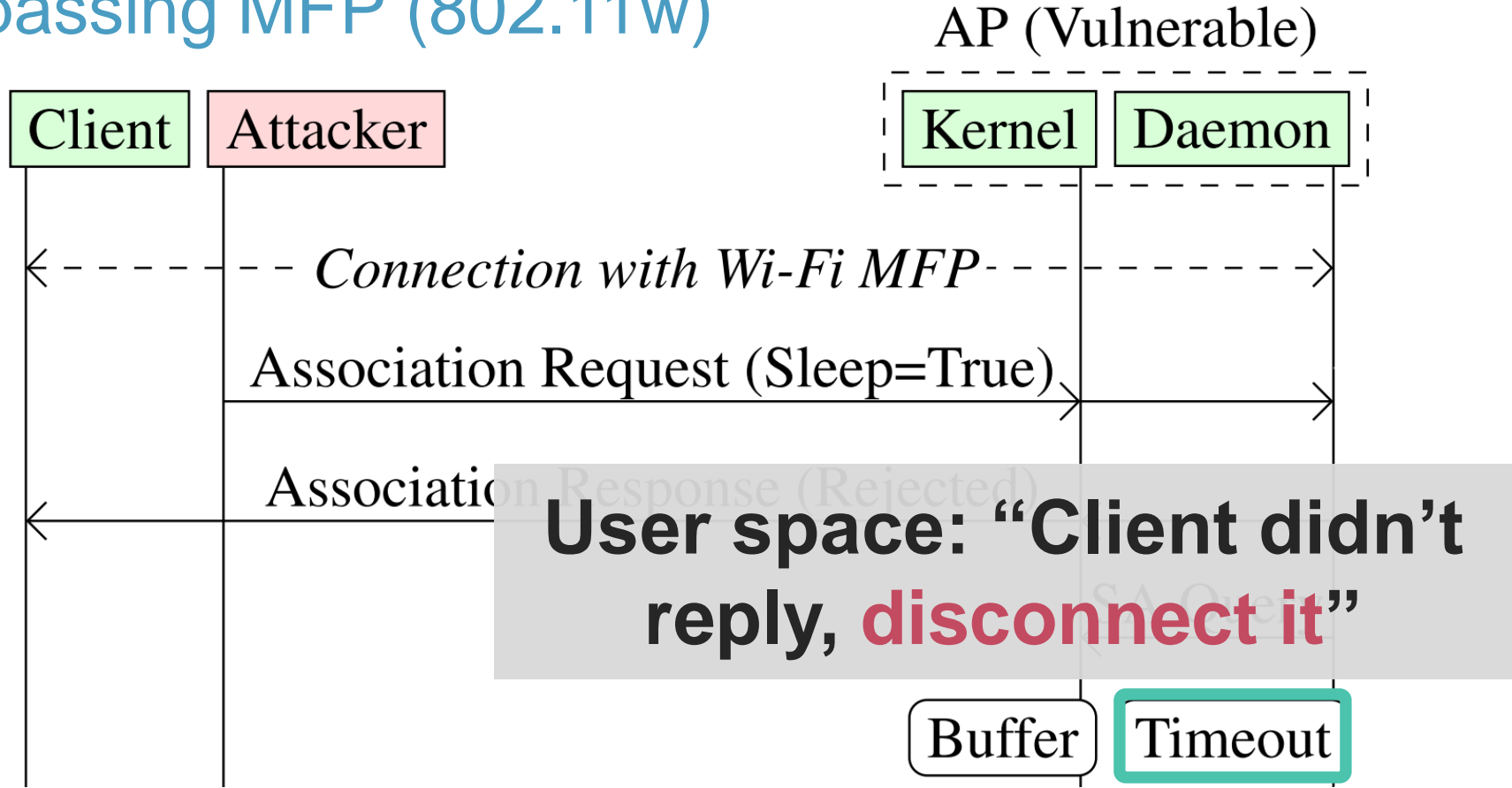
# Bypassing MFP (802.11w)



# Bypassing MFP (802.11w)



# Bypassing MFP (802.11w)



# 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 paper “Framing Frames: Bypassing Wi-Fi Encryption by Manipulating Transmit Queues” (USENIX Security)

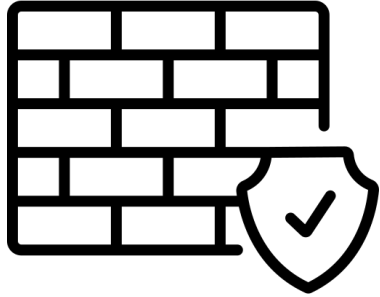
Defenses:

- › Never buffer “are you still connected?” frames
- › Authenticate the sleep bit in the header of Wi-Fi frames
- › **Standard should be updated** with one of these defenses

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

Target is networks that use **client isolation**. Examples:

- › Company network with malicious/compromised clients
- › Public hotspots that require authentication

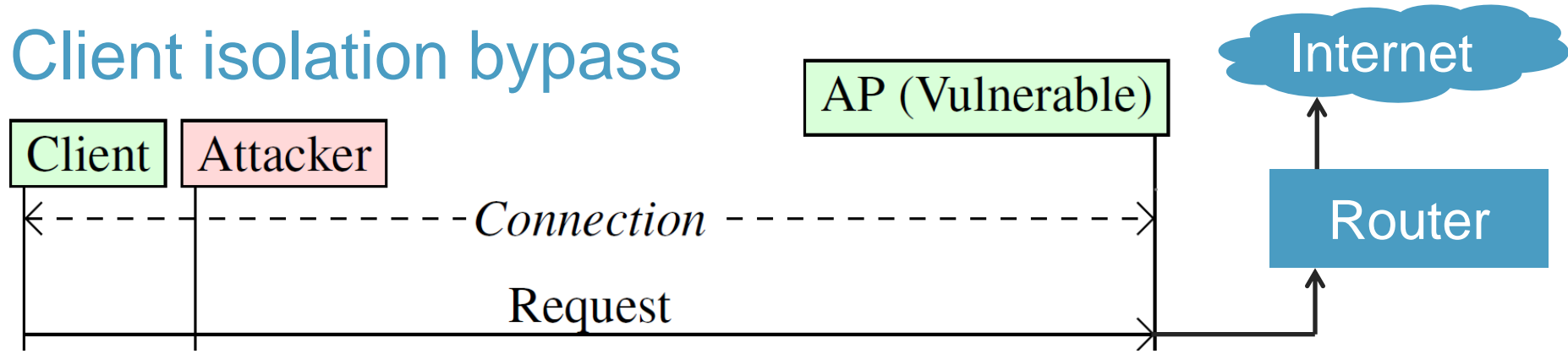


→ Adversary can connect to the network, but can't attack others

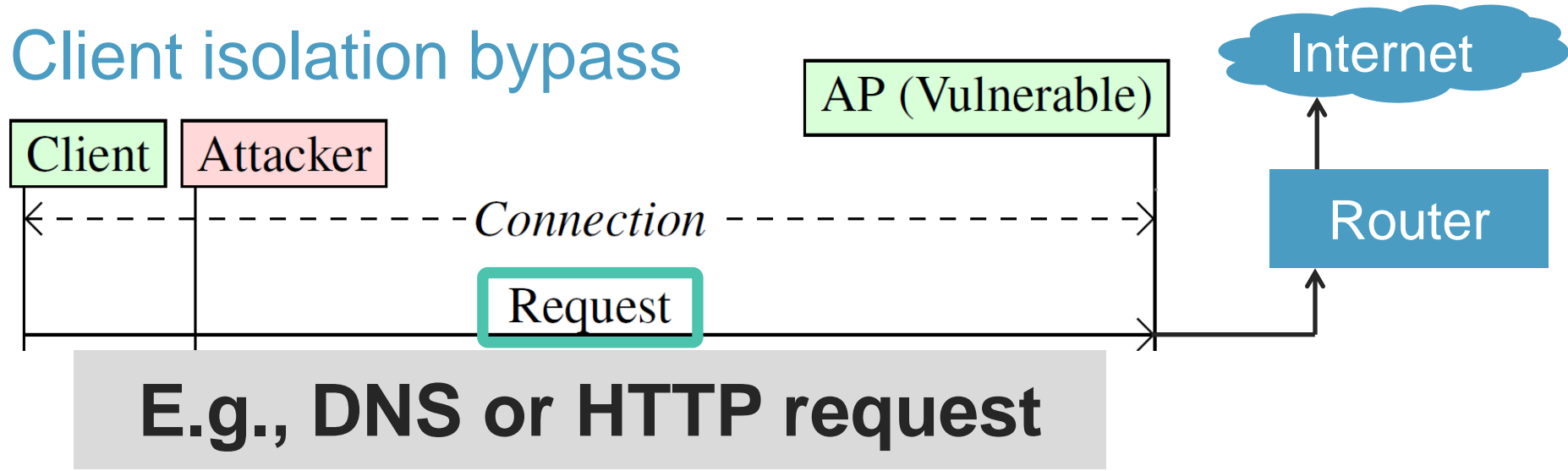
# Client isolation bypass



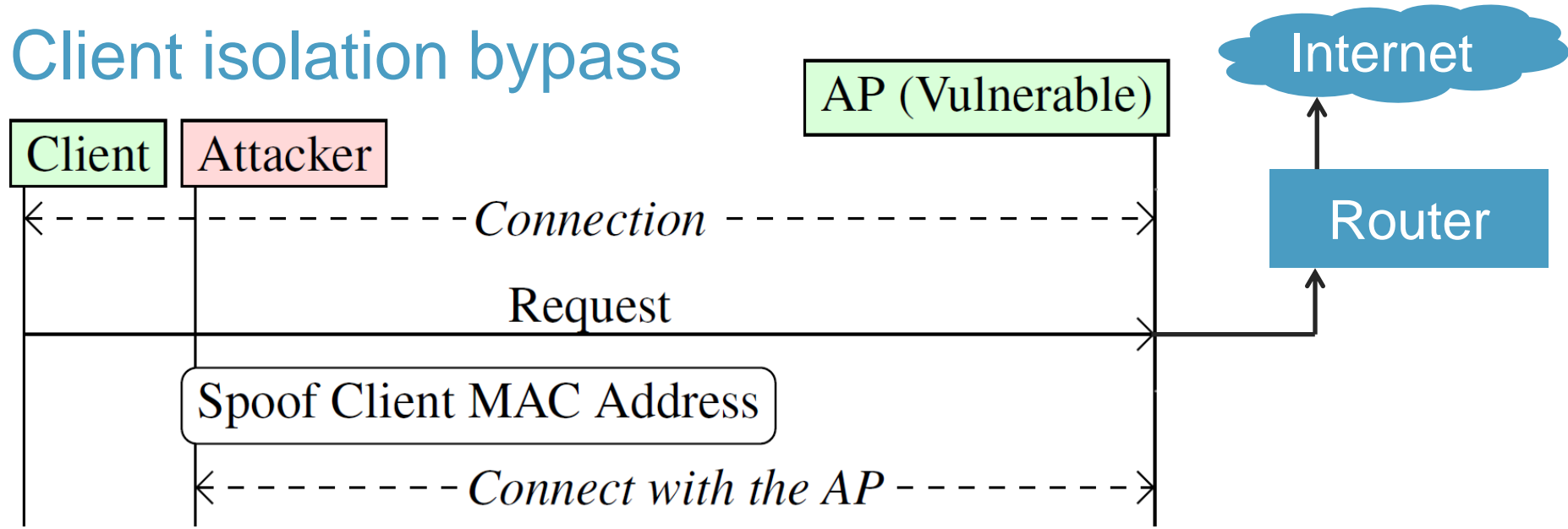
# Client isolation bypass



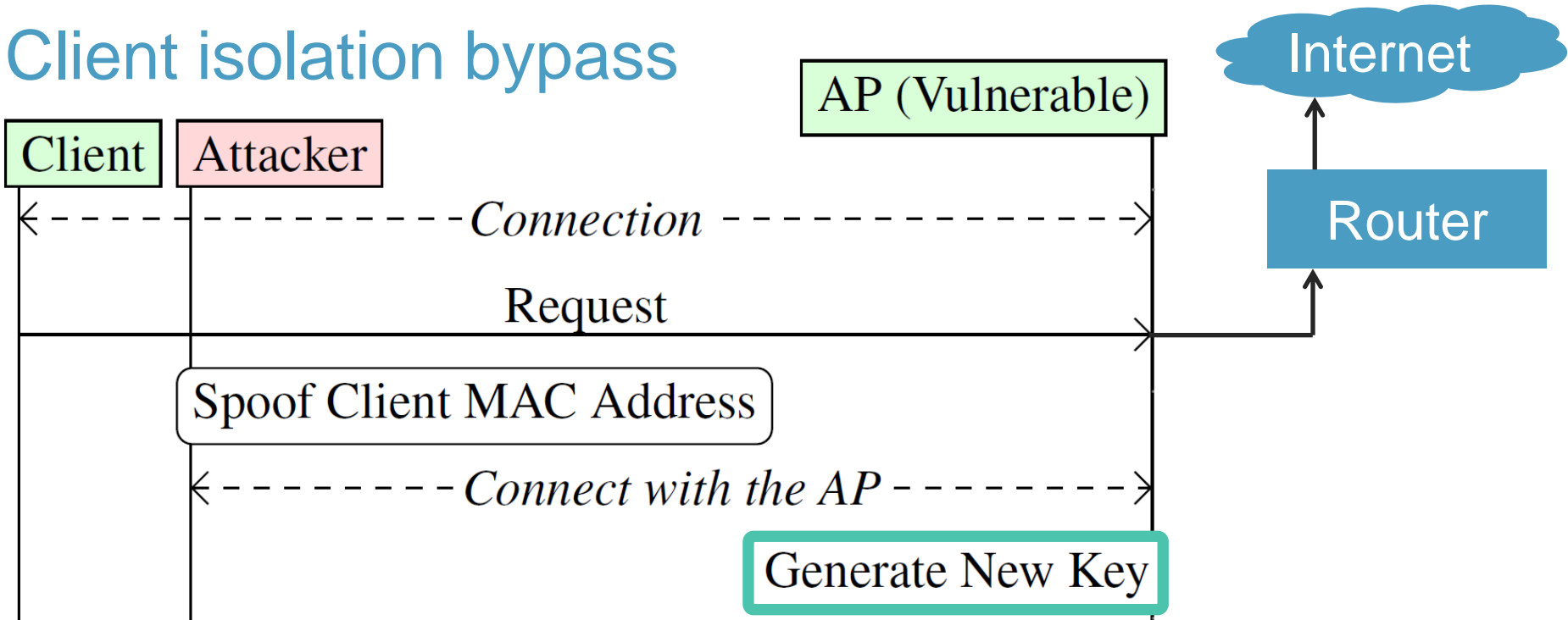
# Client isolation bypass



# Client isolation bypass

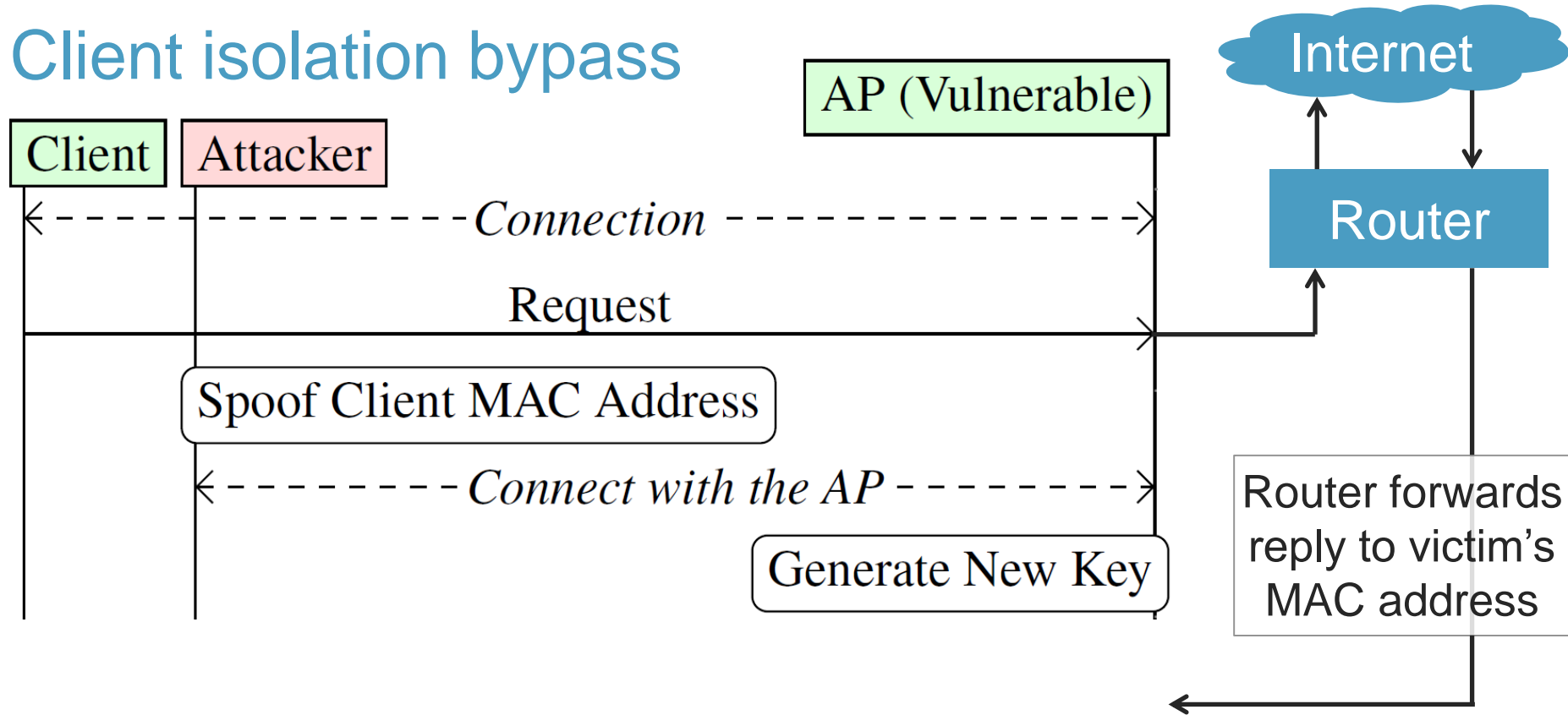


# Client isolation bypass

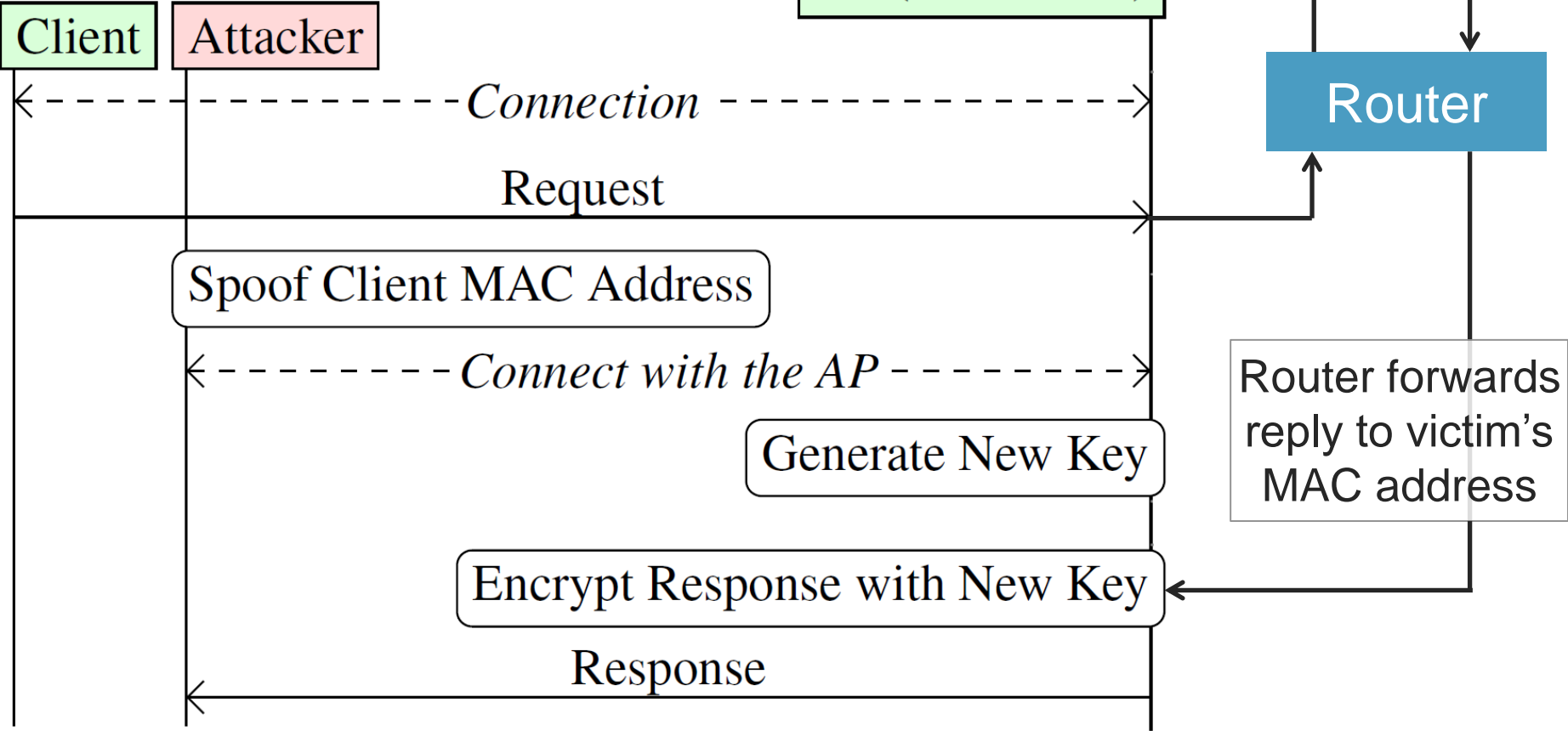


**New key is associated with the victim's MAC address**

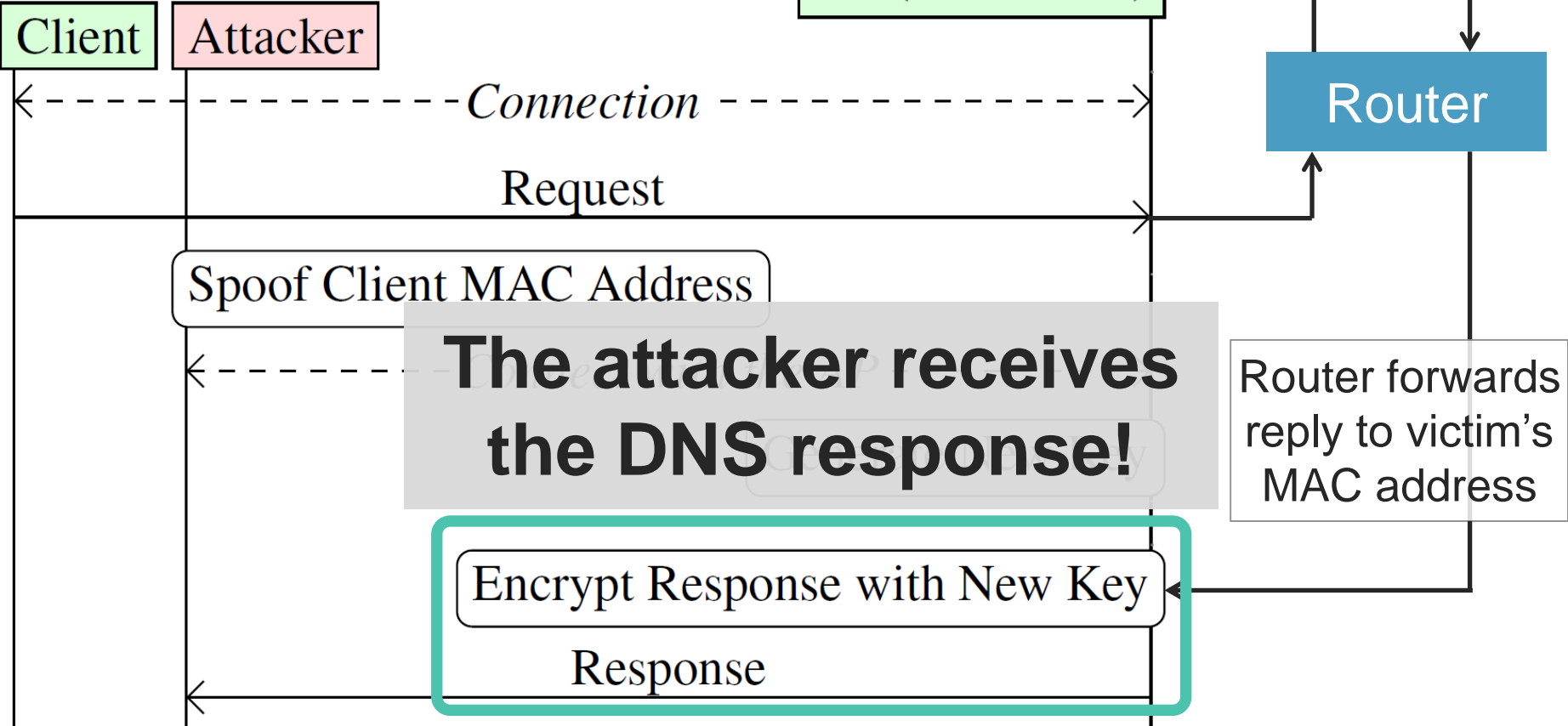
# Client isolation bypass



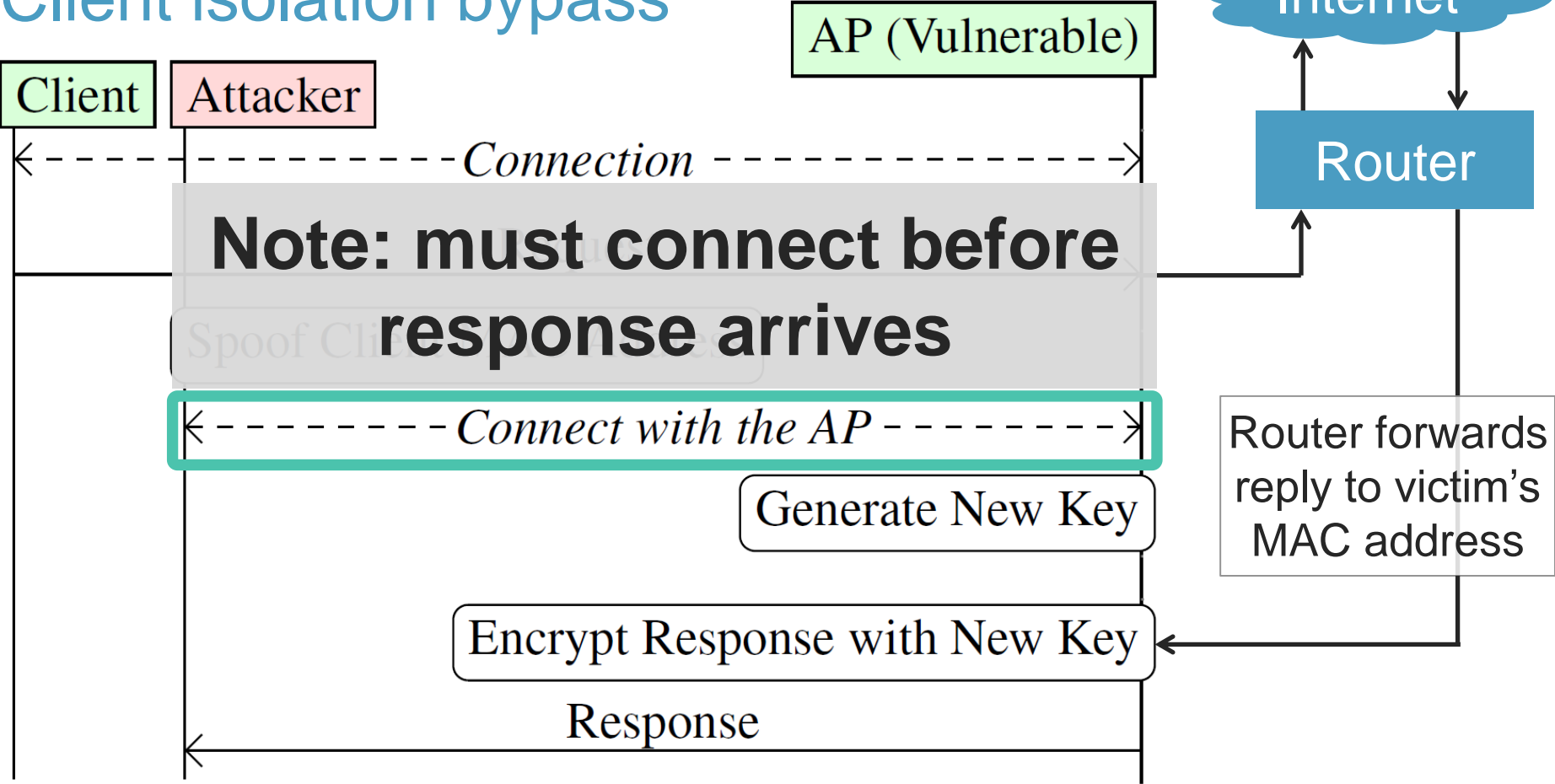
# Client isolation bypass



# Client isolation bypass



# Client isolation bypass



# Fixing client isolation

**Disallow recently-used MAC address** unless:

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

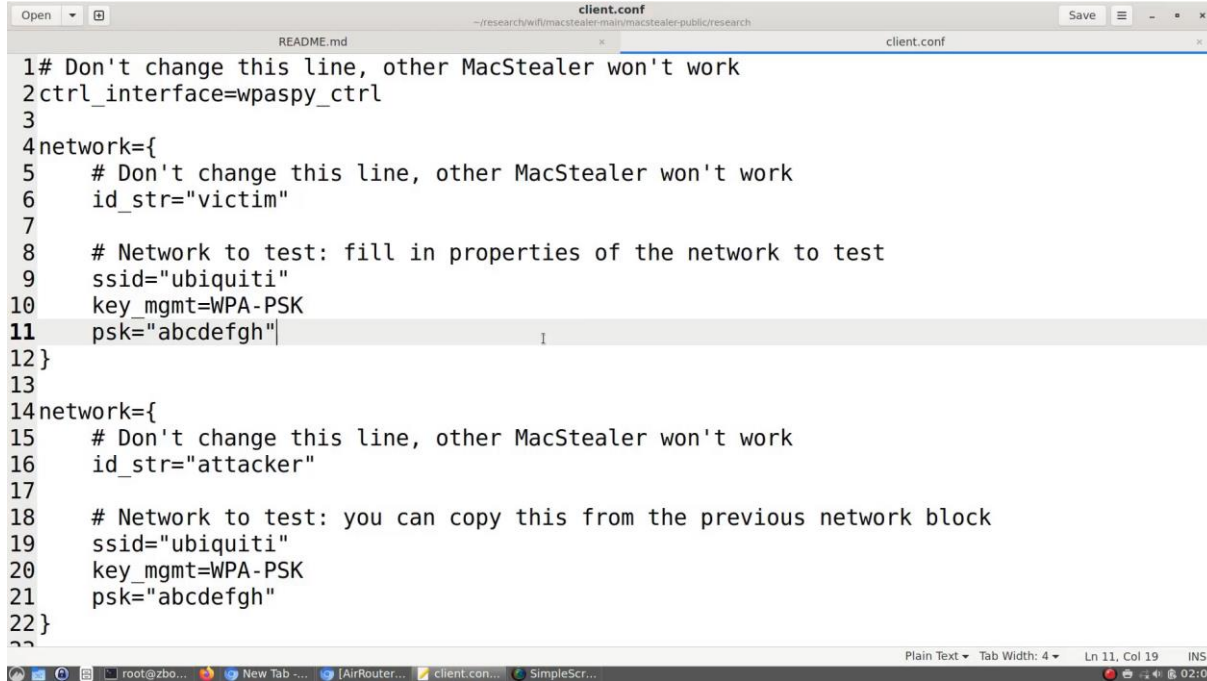
Currently few vendors implemented a defense or mitigation

- › Client isolation is flawed but still useful
- › Alternative: use VLANs to isolate groups

# Tool to test devices: MacStealer

Command	Short description
<i>Sanity checks</i>	
<code>./macstealer.py wlan0 --ping</code>	<b>Sanity checks</b>
<code>./macstealer.py wlan0 --ping --flip</code>	
<i>Vulnerability tests</i>	
<code>./macstealer.py wlan0</code>	<b>Vulnerability tests</b>
<code>./macstealer.py wlan0 --other-bss</code>	
<i>Client isolation: Ethernet layer</i>	
<code>./macstealer.py wlan0 --c2c wlan1</code>	<b>Does the network use client isolation?</b>
<code>./macstealer.py wlan0 --c2c-eth wlan1</code>	

# MacStealer demo



```
client.conf
~/research/wifumacstealer-main/macstealer-public/research
client.conf
1# Don't change this line, other MacStealer won't work
2ctrl_interface=wpa_supplicant
3
4network={
5    # Don't change this line, other MacStealer won't work
6    id_str="victim"
7
8    # Network to test: fill in properties of the network to test
9    ssid="ubiquiti"
10   key_mgmt=WPA-PSK
11   psk="abcdefgh"
12}
13
14network={
15   # Don't change this line, other MacStealer won't work
16   id_str="attacker"
17
18   # Network to test: you can copy this from the previous network block
19   ssid="ubiquiti"
20   key_mgmt=WPA-PSK
21   psk="abcdefgh"
22}
??
```

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

<https://t.me/learningnets>

# Experiments

All tested professional & home APs were vulnerable

- **Design flaw** in Wi-Fi client isolation!
- Useful test for auditors



[github.com/vanhoefm/macstealer](https://github.com/vanhoefm/macstealer)

# 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

# Backup slide: root cause

Client identity not authenticated across the network stack:

- › Wi-Fi security: 802.1X identity (username)
  - › Packet routing: IP/MAC addresses
- } Not bound to each other
- Wi-Fi attacker can spoof client's identity on other layers

Other observation: client isolation was “bolted on” by vendors

- › Not part of IEEE 802.11 standard → less studied

# Backup slide: 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