When Network Protocols Meet New Threat Models

Mathy Vanhoef

SecAppDev, June 5, 2024.

KU LEUVEN DistriNet

Introduction

Goal of this talk:

› Explain some interesting network attacks + demos ☺

› Common theme: attacks are enabled by novel threat model

I will use the word “threat model” rather informally:

› In some attacks, the adversary is given extra capabilities

› In other attacks, the focus is more on new attack techniques

Agenda

- Attacks that introduced new threat models:
  - **The BEAST and HEIST attack (TLS/HTTPS)**
  - The Multi-Channel MitM (KRACK)
  - Outbound Connections (FragAttacks)
  - Client Isolation (Framing Frames)
  - DNS Spoofing & VPNs (TunnelCrack)
- Conclusion

The BEAST attack against SSL/TLS

- Phillip Rogaway (‘95): CBC encryption can be attacked when the Initialization Vectors (IVs) are predictable
- Fixed in TLS1.1, but TLS1.0 was still very common
  - “It’s hard to abuse, so not important to fix”
- Duong & Rizzo (‘11): attacked CBC in practice by assuming **malicious JavaScript in the browser + network MitM**
  - And extended attack to achieve full plaintext recovery
  - Sudden scramble to update implementations

Reference: One Bad Apple: Backwards Compatibility Attacks on State-of-the-Art Cryptography

The BEAST Threat Model

- Arguably most influential contribution was the threat model:
  - Attack can execute JavaScript in the victim's browser
  - And attacker can intercept (encrypted) network traffic
- *This **new threat model** completely broke TLS 1.0*
- The "BEAST threat model" was (and is) used in many works
  - In many attacks against RC4, including our RC4 NOMORE attack
  - Many TLS attacks (Lucky13, Bleichenbacher attacks, DROWN)
  - In the **CRIME and BREACH attack** to abuse compression

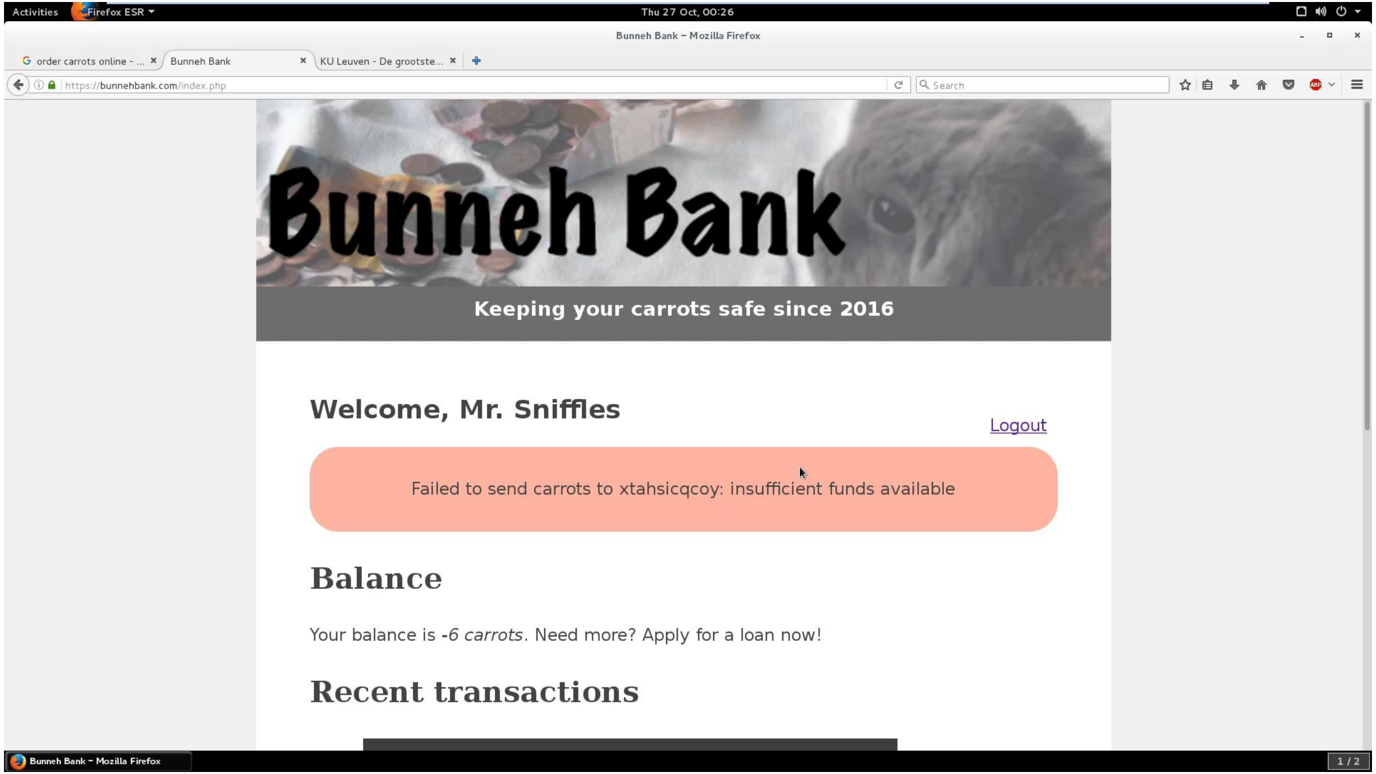
Abusing compression: CRIME and BREACH

- Abuse compression in TLS/HTTPS to leak data in response
- Idea is to make a page reflect the guessed **CRSF token**
  - Correct guess results in smaller response due to efficient compression
- Like BEAST, relied on malicious JavaScript + network MitM
  - Network MitM was used to measure length of response

HEIST attack: also abuses compression to recover CRF token

- But uses **timing side-channels instead of needing MitM**

DEMO: HEIST Attack



Reflection

- The new “BEAST threat model” enabled various follow-up works to construct more practical attacks
- Some attacks were further improved to reduce the required capabilities of the attacker

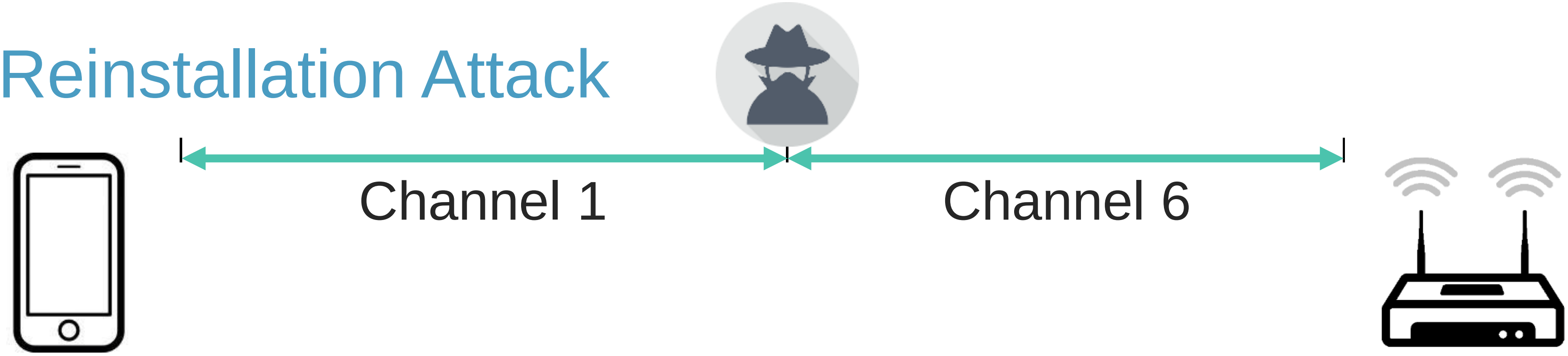
“Attacks only get better, they never get worse.”

— Bruce Schneier

Agenda

- Attacks that introduced new threat models:
  - The BEAST attack (TLS)
  - **The Multi-Channel MitM (KRACK)**
  - Outbound Connections (FragAttacks)
  - Client Isolation (Framing Frames)
  - DNS Spoofing & VPNs (TunnelCrack)
- Conclusion

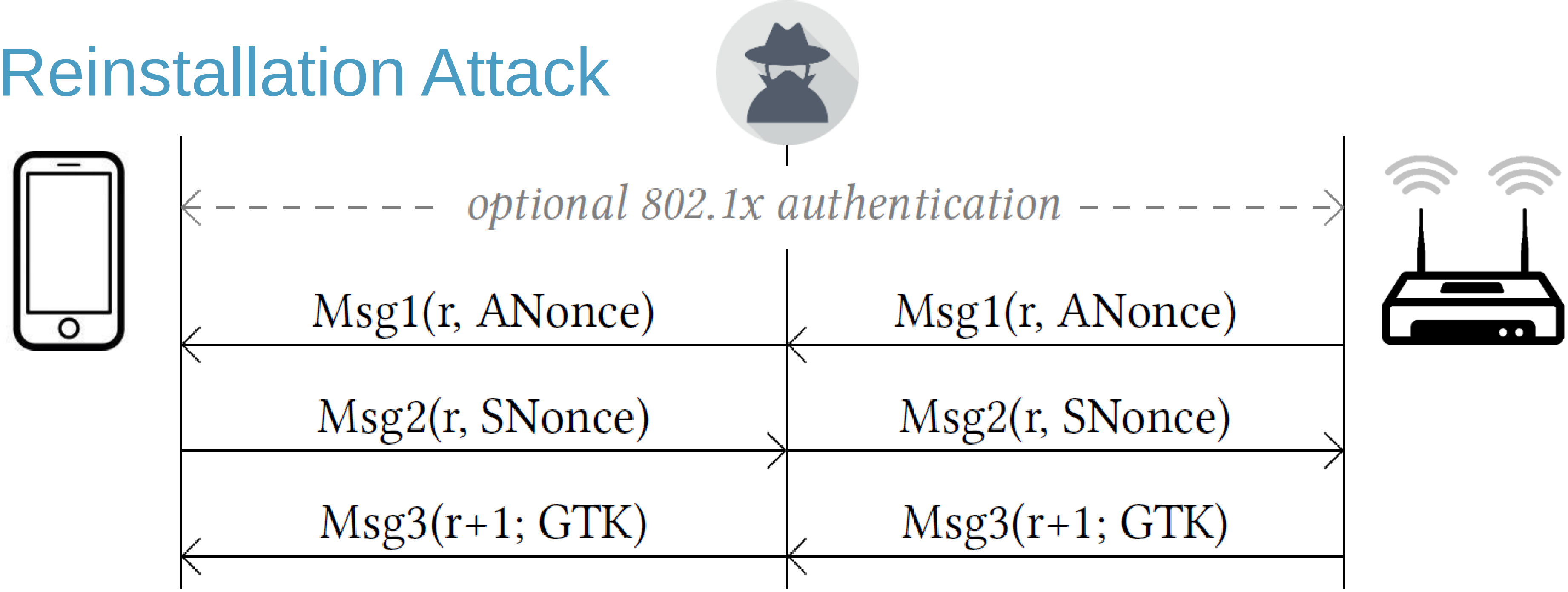
Reinstallation Attack



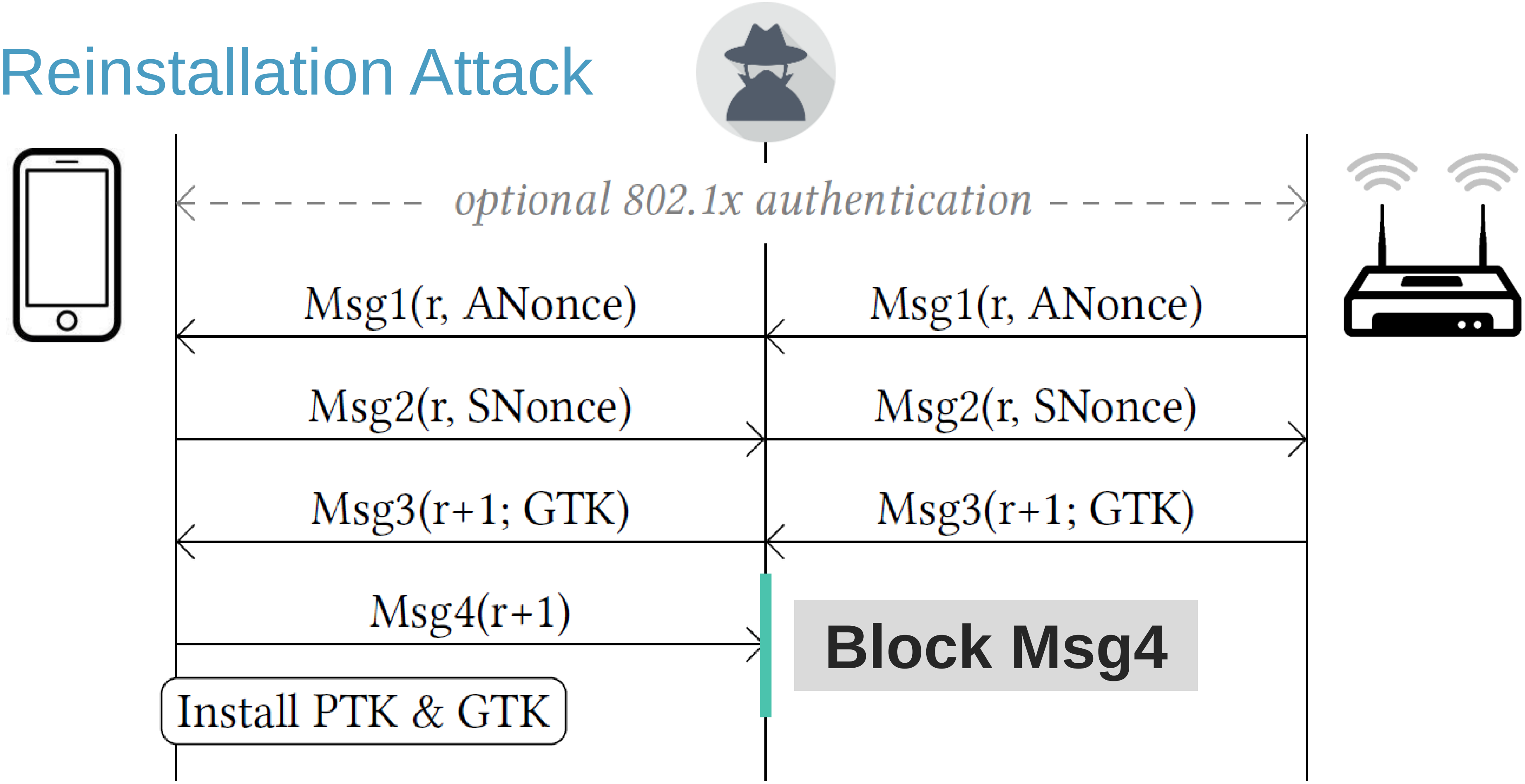
➔ Called a “Multi-Channel MitM” (MC-MitM)

Reinstallation Attack

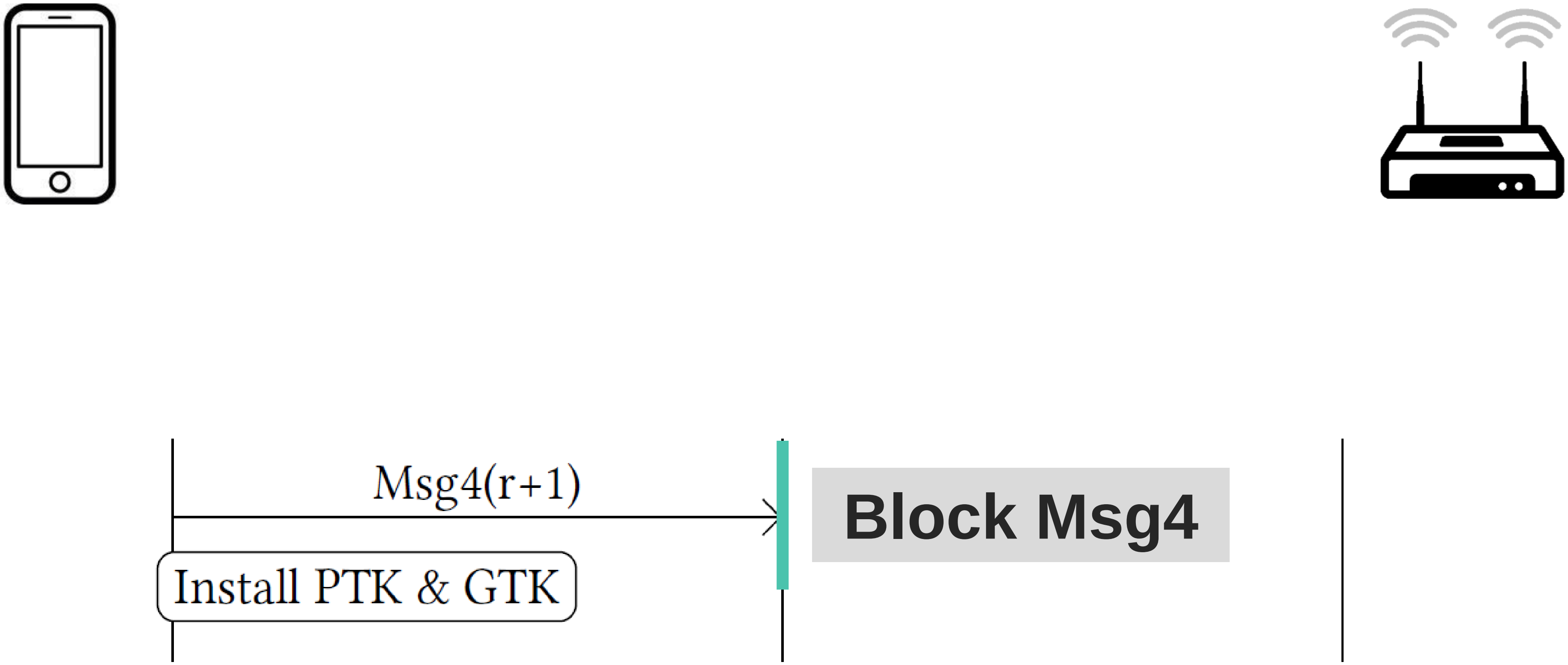
Reinstallation Attack



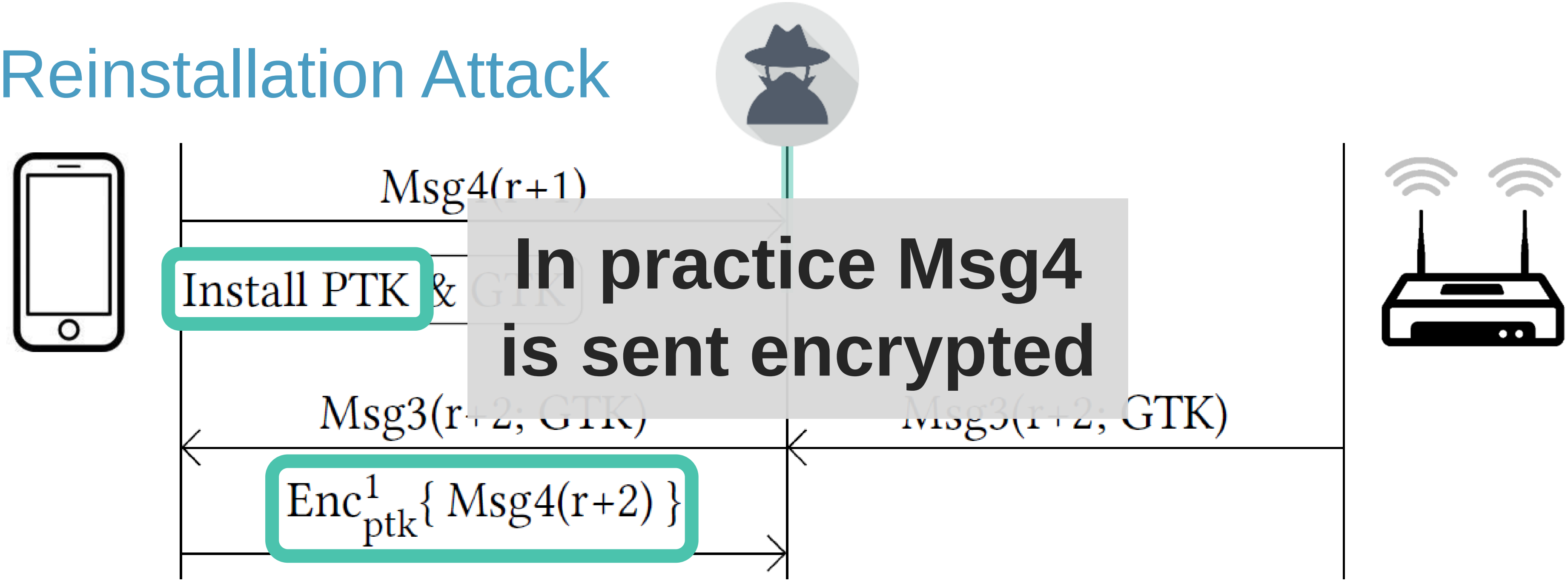
Reinstallation Attack



Reinstallation Attack

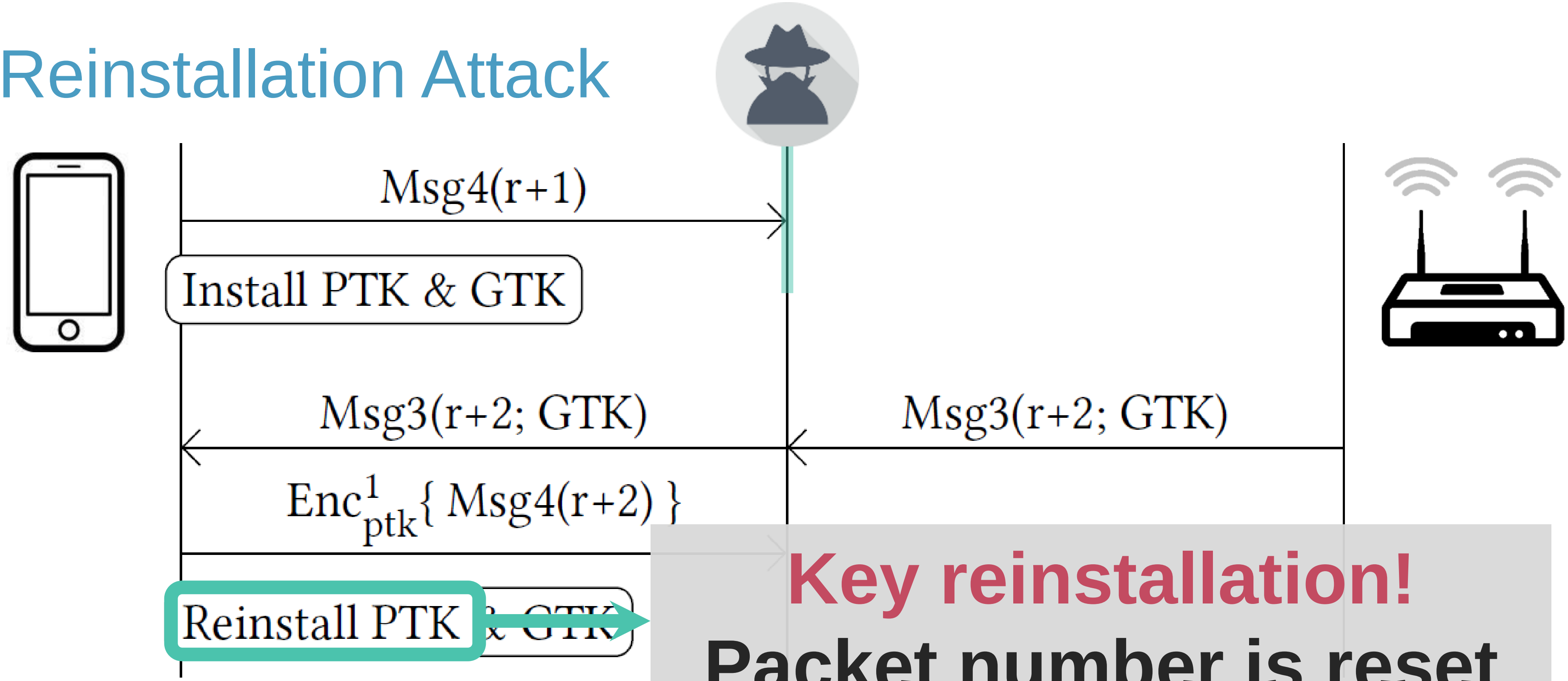


Reinstallation Attack



In practice Msg4 is sent encrypted

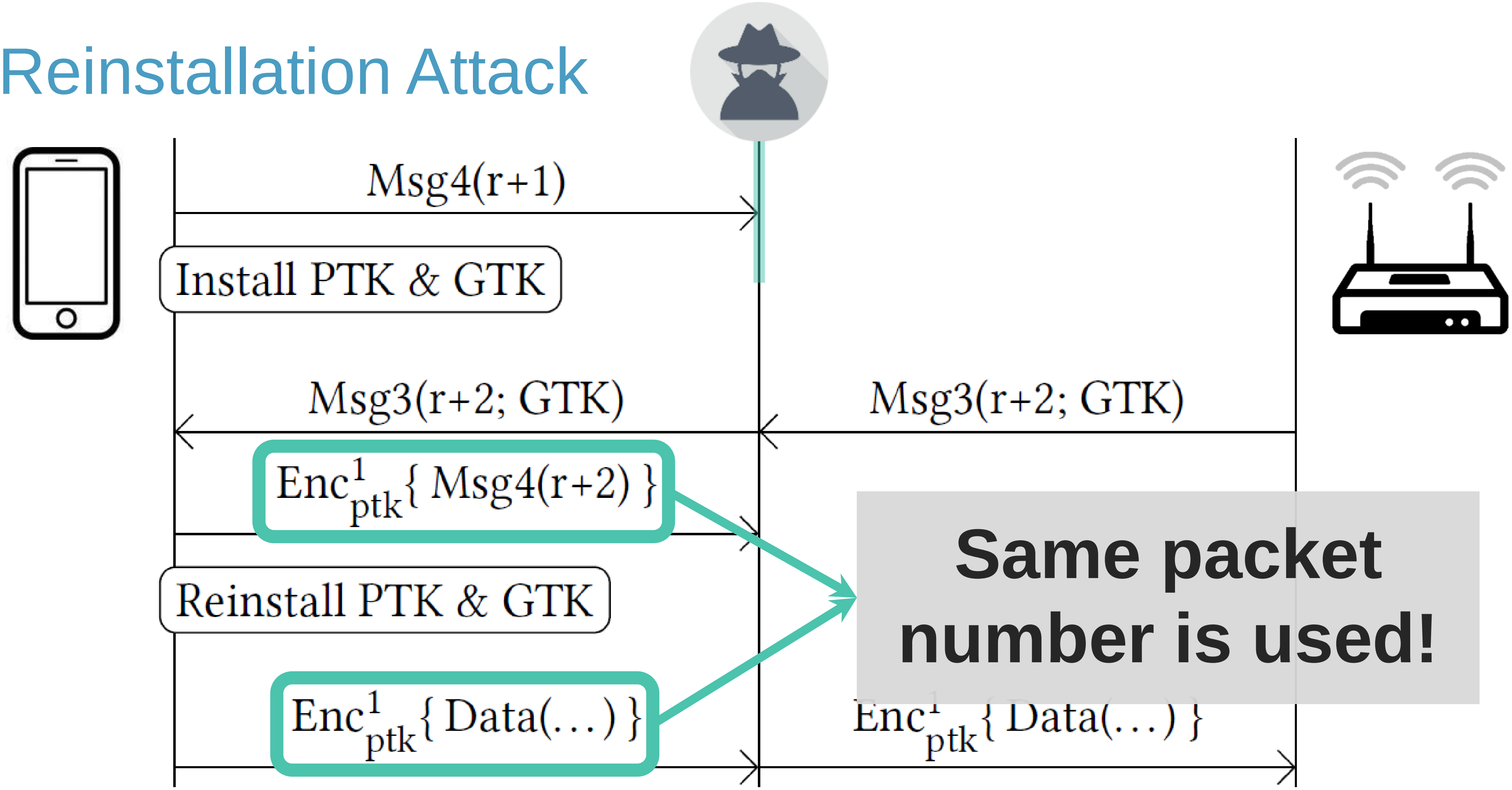
Reinstallation Attack



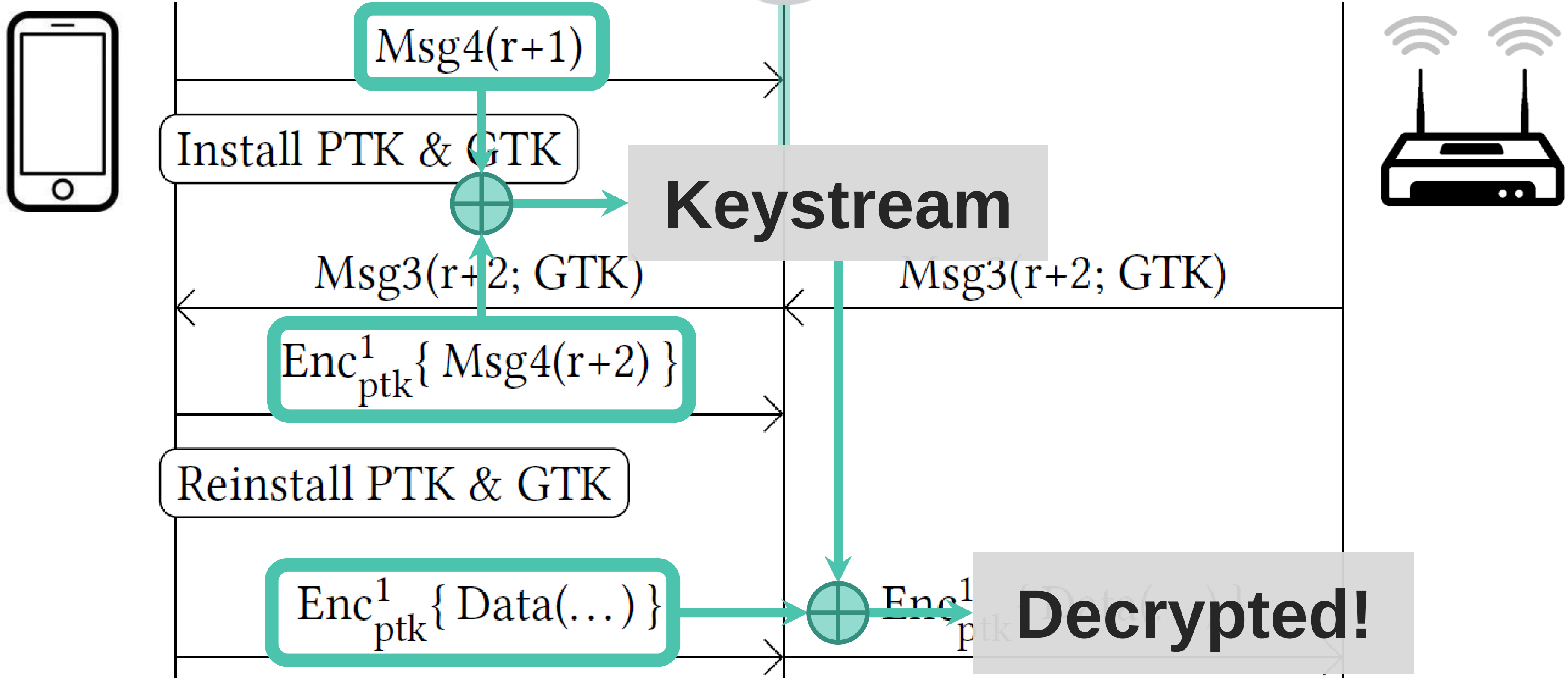
Key reinstallation!

Packet number is reset

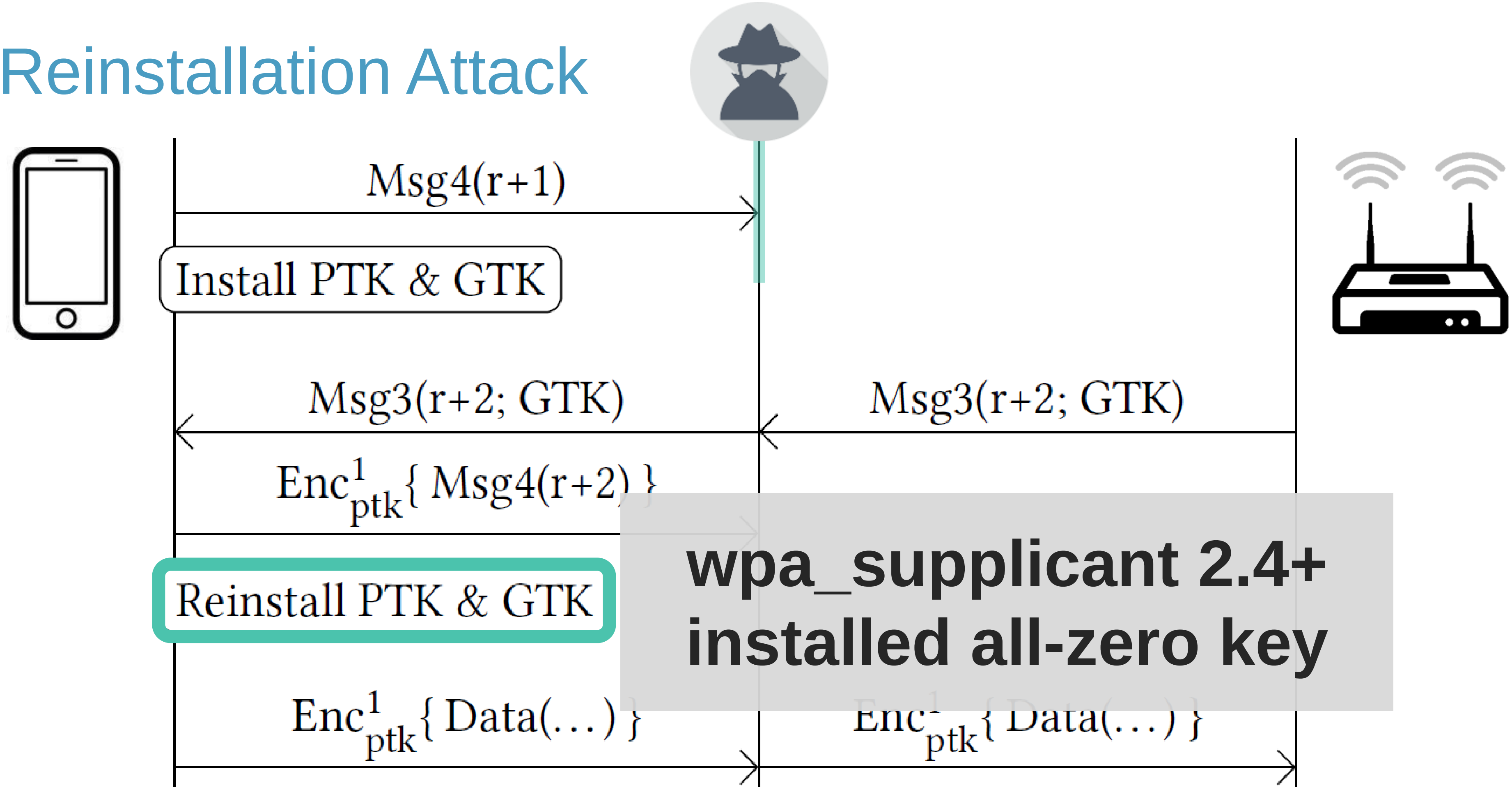
Reinstallation Attack



Reinstallation Attack

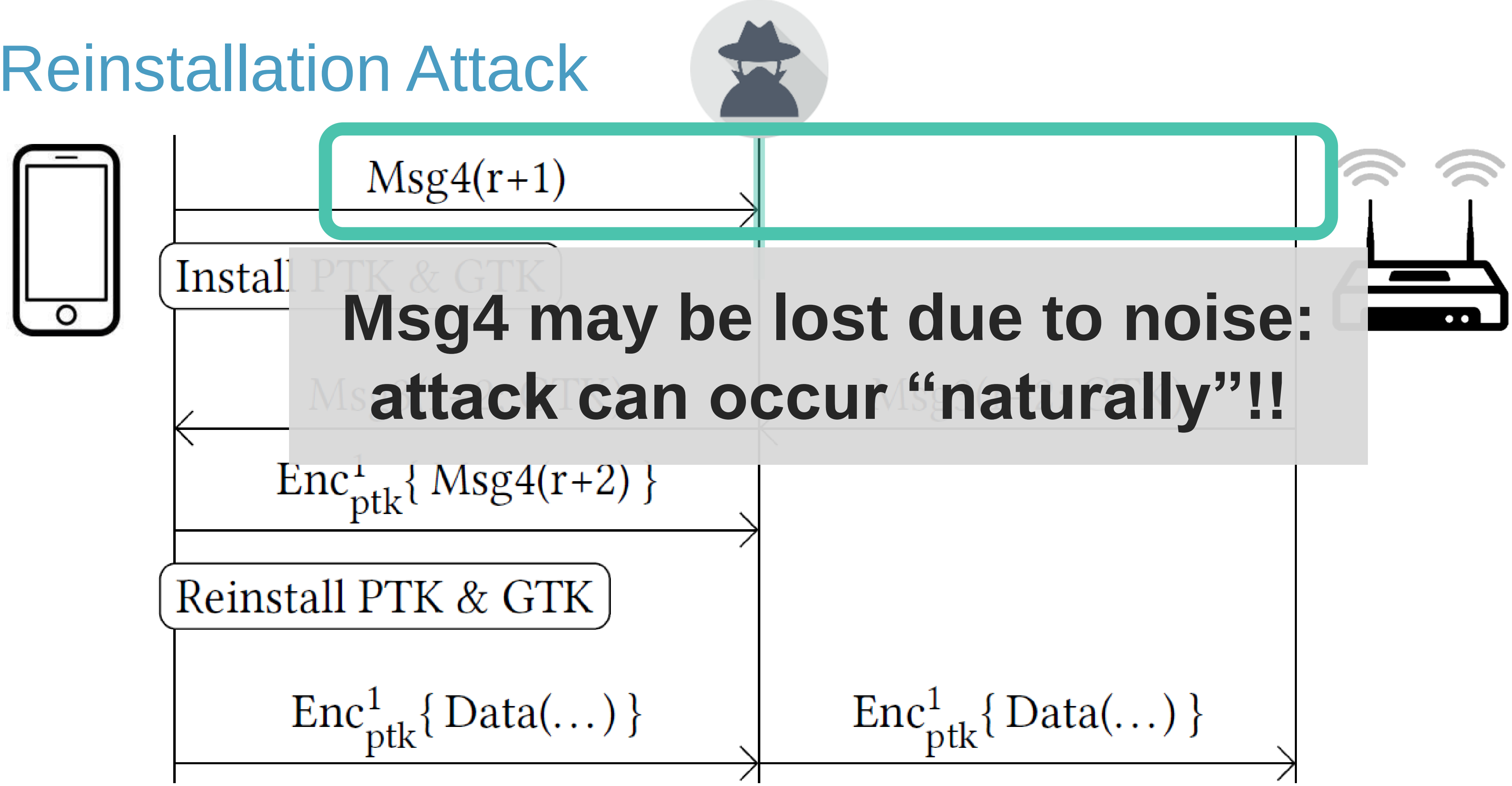


Reinstallation Attack



wpa_supplicant 2.4+ installed all-zero key

Reinstallation Attack



Msg4 may be lost due to noise: attack can occur “naturally”!!

Installation of all-zero key was detected (!!)

Bug report on Linux's hostap mailing list:

"While testing with supplicant 2.4 we observed [..]:

4. *We send M4 and install PTK*
5. *We received M3 again*
6. *We send M4 and install PTK*

… we install it as 0 again in step (6)"

[1] An issue with supplicant receiving retranmitted M3 (Atul Joshi)
[2] An issue with supplicant receiving retranmitted M3 (Jouni Malinen)
[3] Fix TK configuration to the driver in EAPOL-Key 3/4 retry case

This bug was then fixed

- “[..] possibility of the authenticator having to retry EAPOL-Key message 3/4 in case the first EAPOL-Key message 4/4 response is lost. That case **ended up trying to reinstall the same TK to the driver**, but the key was not available”
- They didn’t realize an adversary can force this situation
- The MC-MitM threat model allows us to do this reliably!

[1] An issue with supplicant receiving retranmitted M3 (Atul Joshi)
[2] An issue with supplicant receiving retranmitted M3 (Jouni Malinen)
[3] Fix TK configuration to the driver in EAPOL-Key 3/4 retry case

The MC-MitM is used in several works now

- The MC-MitM was originally used by us to break WPA-TKIP
- Was used to infer resource sizes in combination with malicious JavaScript, i.e., in a BEAST-like attack
- To exploit an implementation flaw in Broadcom code
- In our “framing frames” attack
- Also used in the FragAttacks research

References:

- Advanced WiFi Attacks Using Commodity Hardware (ACSAC’14)
- Request and Conquer: Exposing Cross-Origin Resource Size (USENIX Sec ’16)
- Discovering Logical Vulnerabilities in the Wi-Fi Handshake Using Model-Based Testing (Asia CCS ’17)
- Framing Frames: Bypassing Wi-Fi Encryption by Manipulating Transmit Queues (USENIX Sec ’23)

Agenda

- Attacks that introduced new threat models:
  - The BEAST and HEIST attack (TLS/HTTPS)
  - The Multi-Channel MitM (KRACK)
  - **Outbound Connections (frAgAttacks)**
  - Client Isolation (Framing Frames)
  - DNS Spoofing & VPNs (TunnelCrack)
- Conclusion

Background

Sending small frames causes high overhead:

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

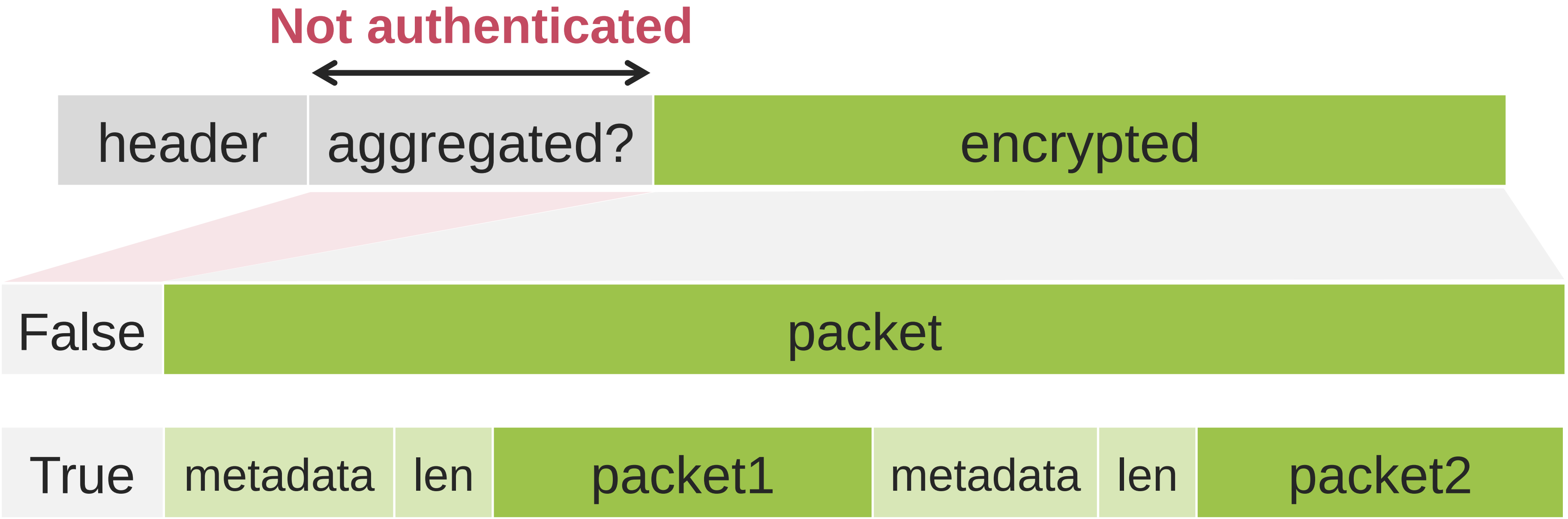
Background

Sending small frames causes high overhead:

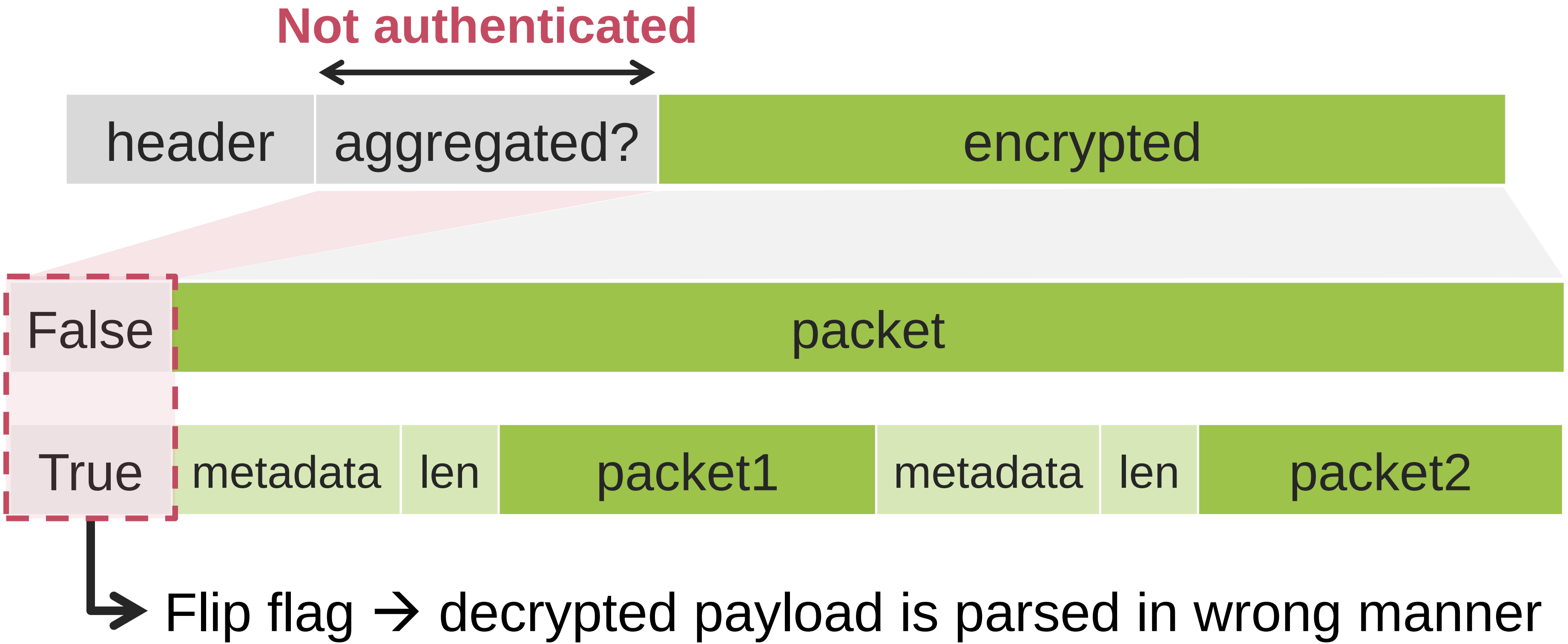
This can be avoided by **aggregating frames**:

Problem: how to recognize aggregated frames?

Aggregation design flaw



Aggregation design flaw



Flip flag → decrypted payload is parsed in wrong manner

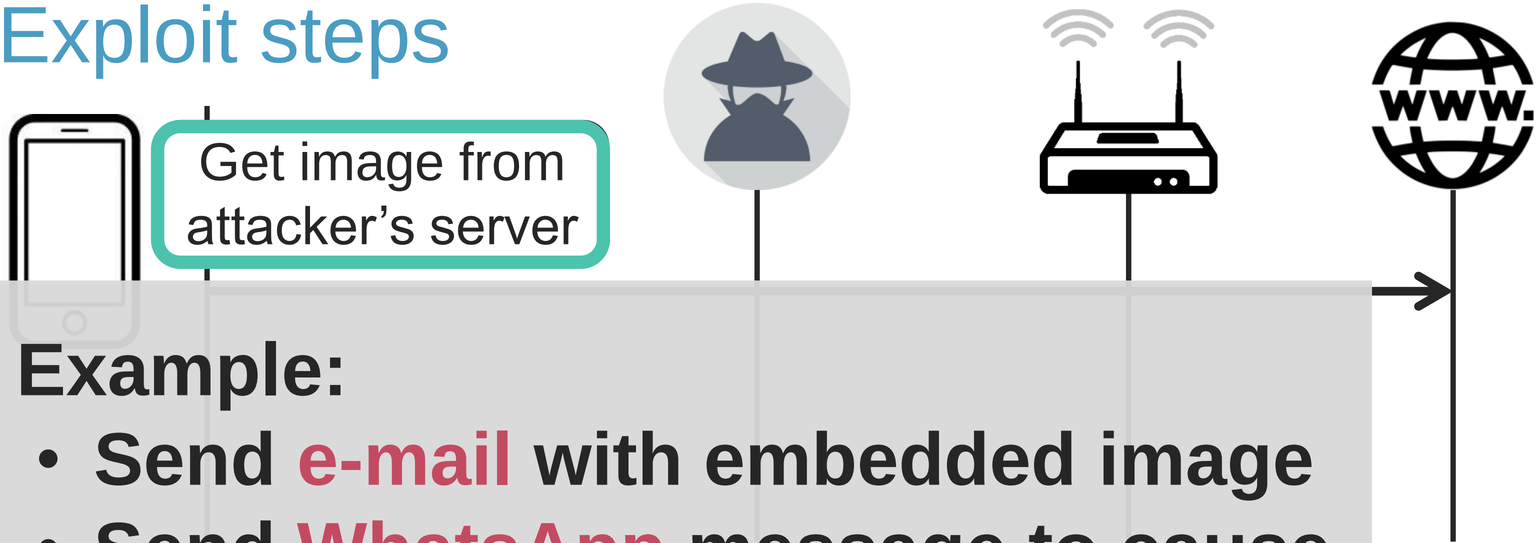
A-MSDU

- Flaw was noticed while 802.11n was being standardized, but implementations based on the draft already existed (2007)
- *“QoS bit 7 should be protected to guard against attack that at minimum leads to a flood of traffic”*
- *“While it is* ***hard to see how this can be exploited****, it is clearly a flaw that is capable of being fixed.”*

→ Exploit by using new threat model ☺ (2021)

[1] Msdu Protection by Nancy Cam-Winget et al. (2007)
[2] Why did nobody notice the aggregation design flaw before?

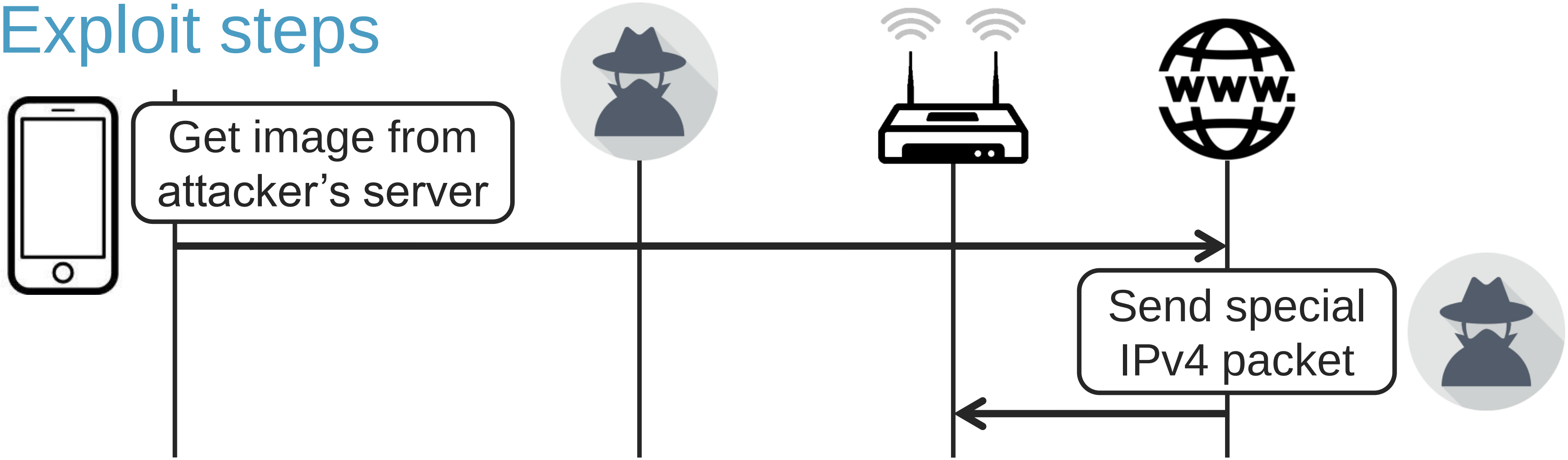
Exploit steps



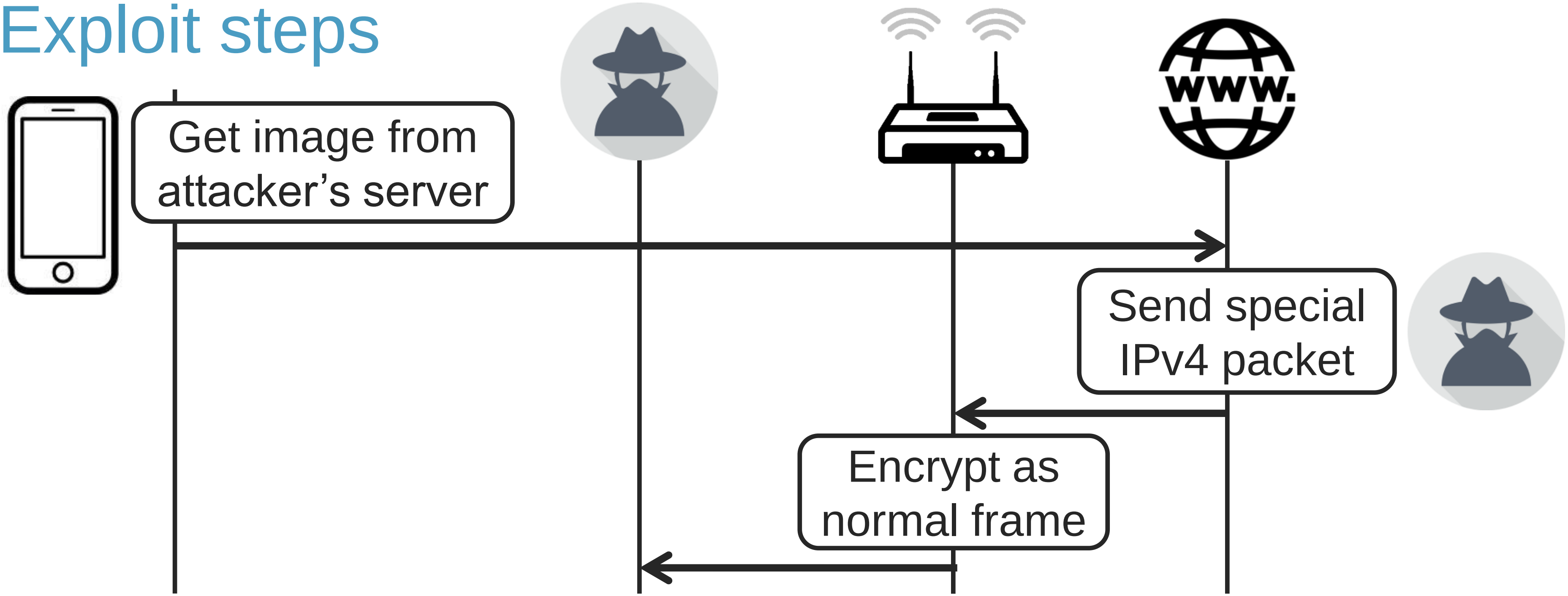
Example:

- **Send e-mail with embedded image**
- **Send WhatsApp message to cause link/image preview**

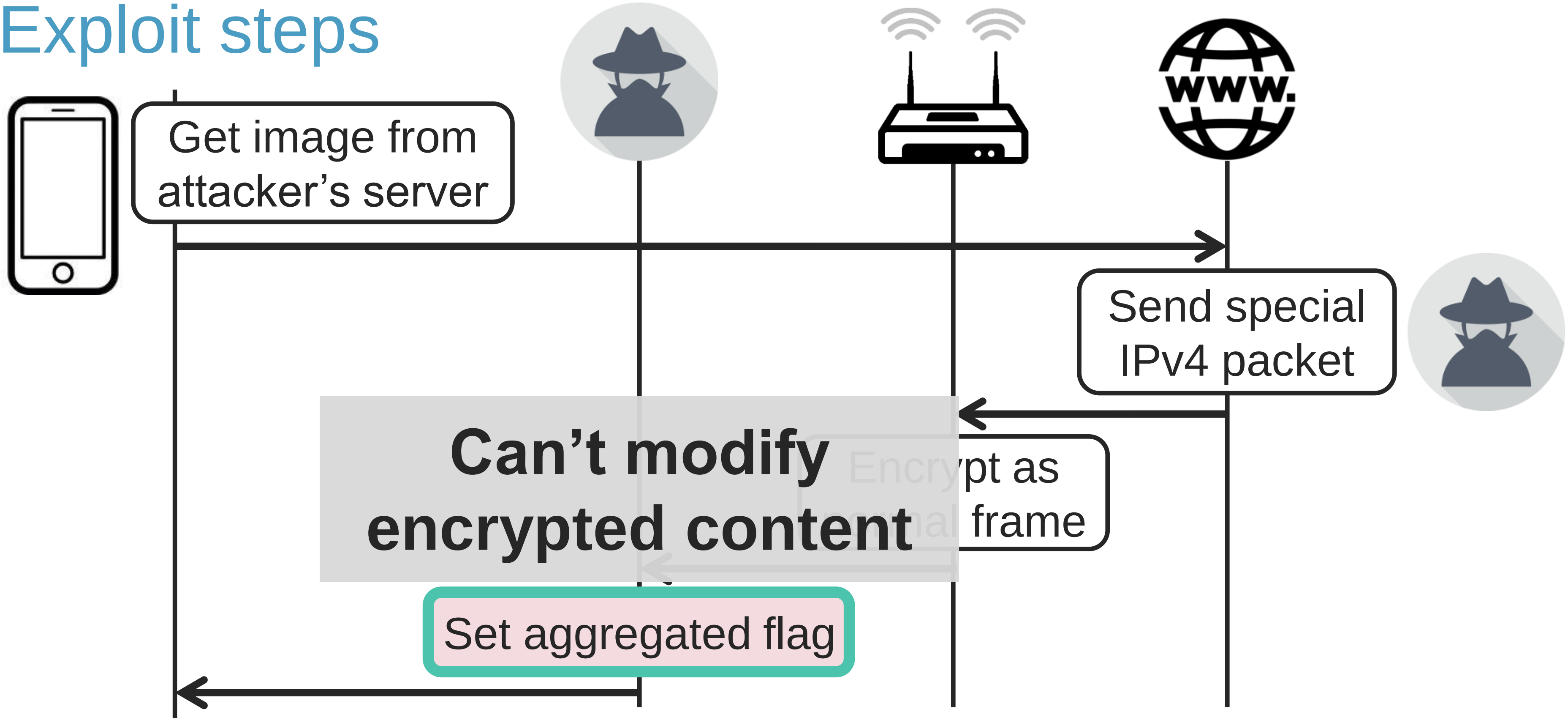
Exploit steps



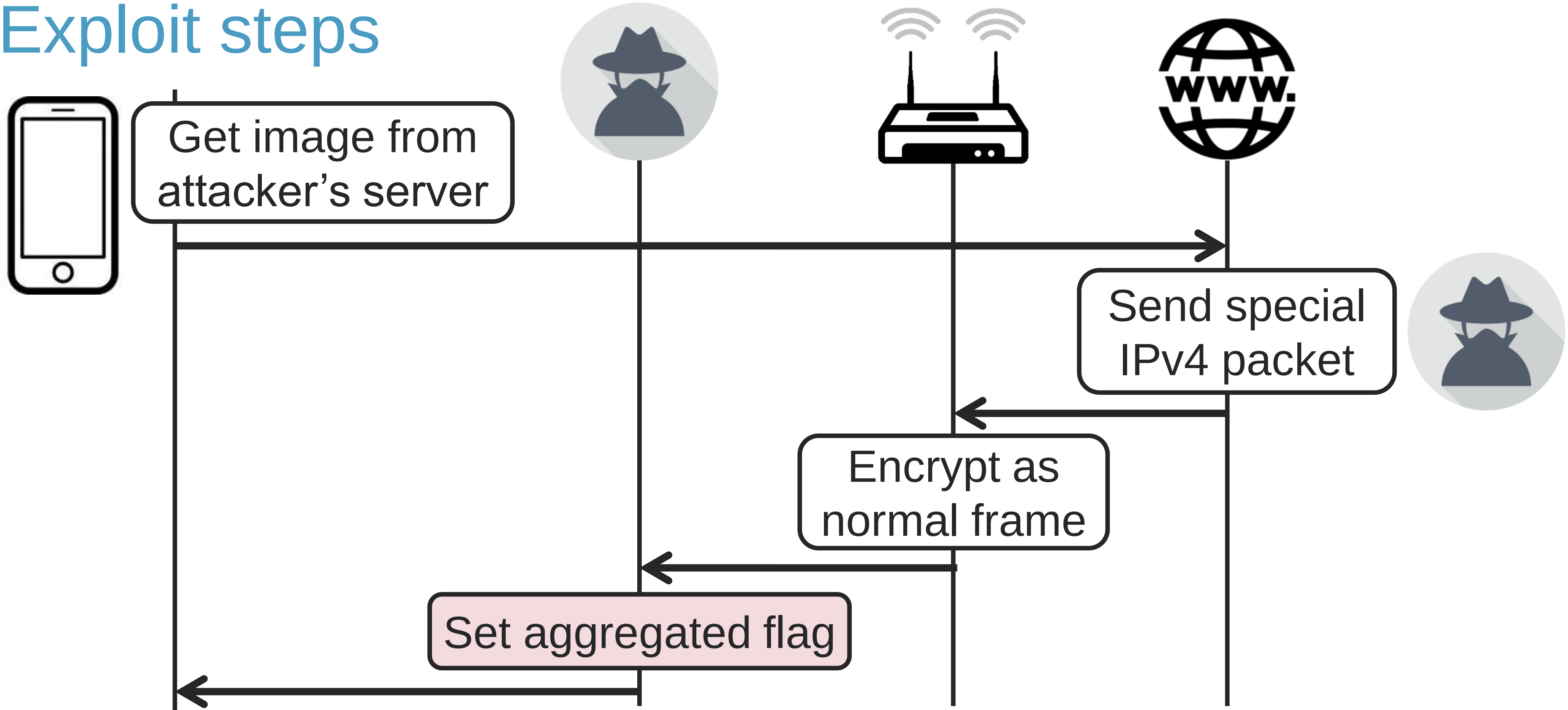
Exploit steps



Exploit steps

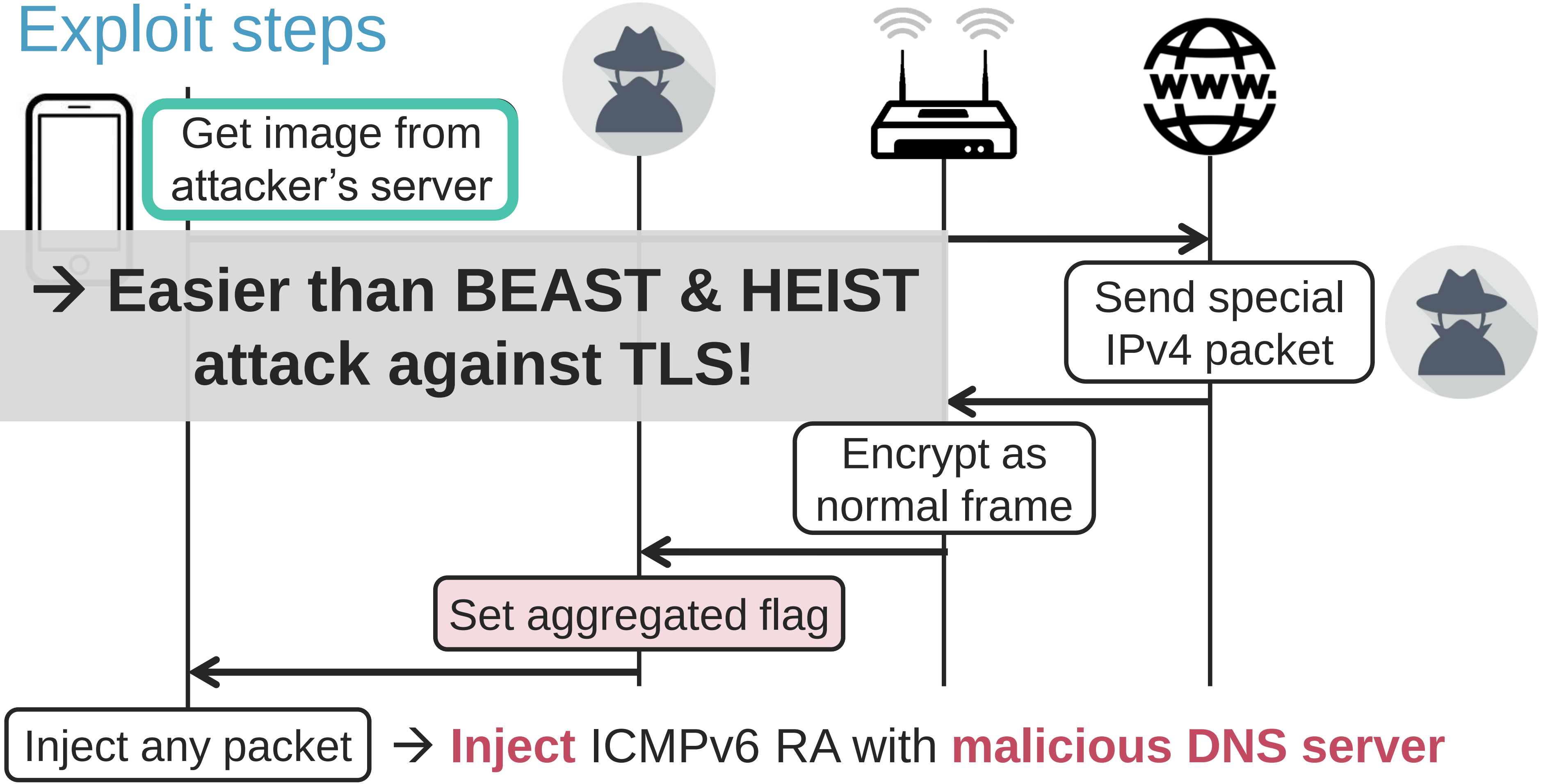


Exploit steps



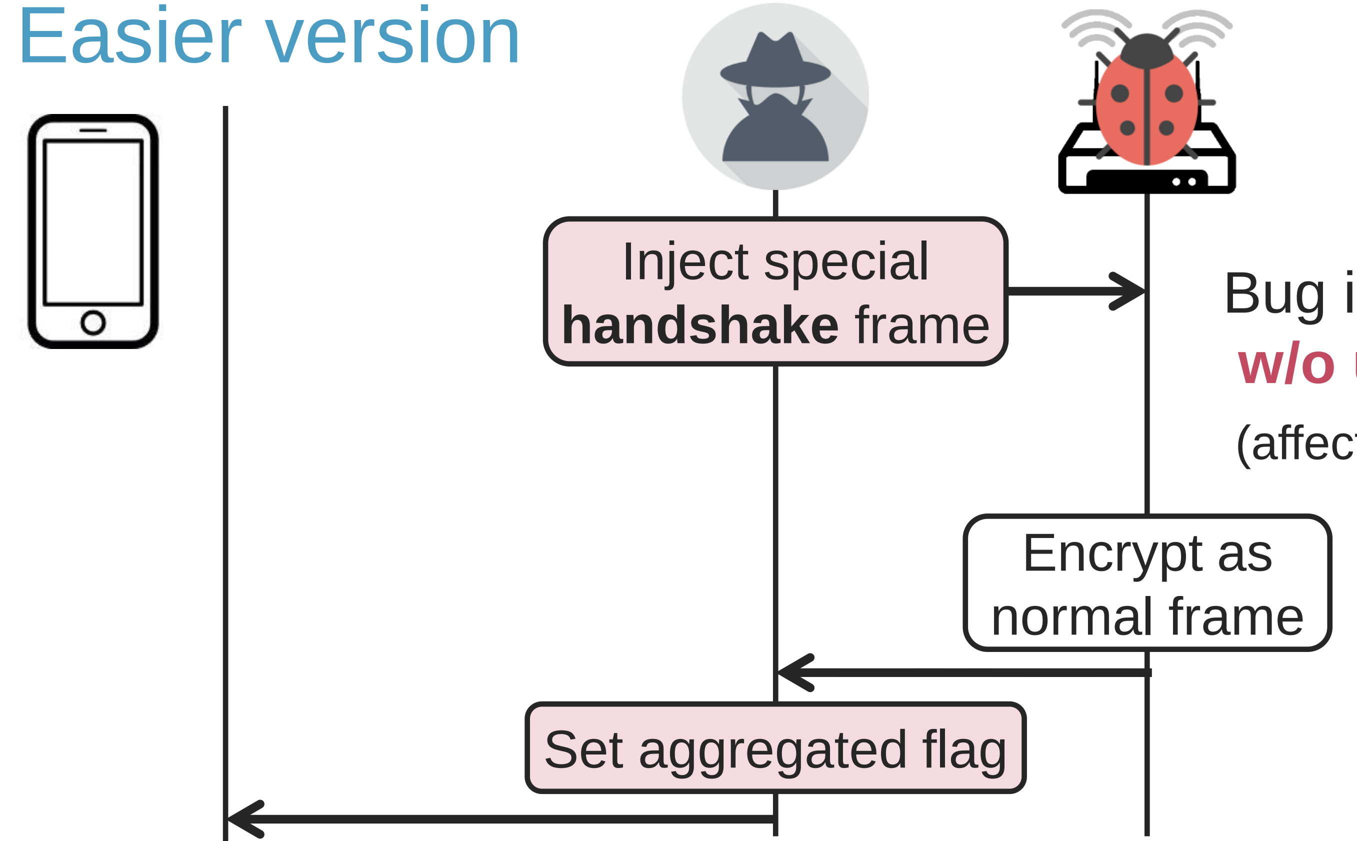
Inject any packet → **Inject** ICMPv6 RA with **malicious DNS server**

Exploit steps



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

Easier version

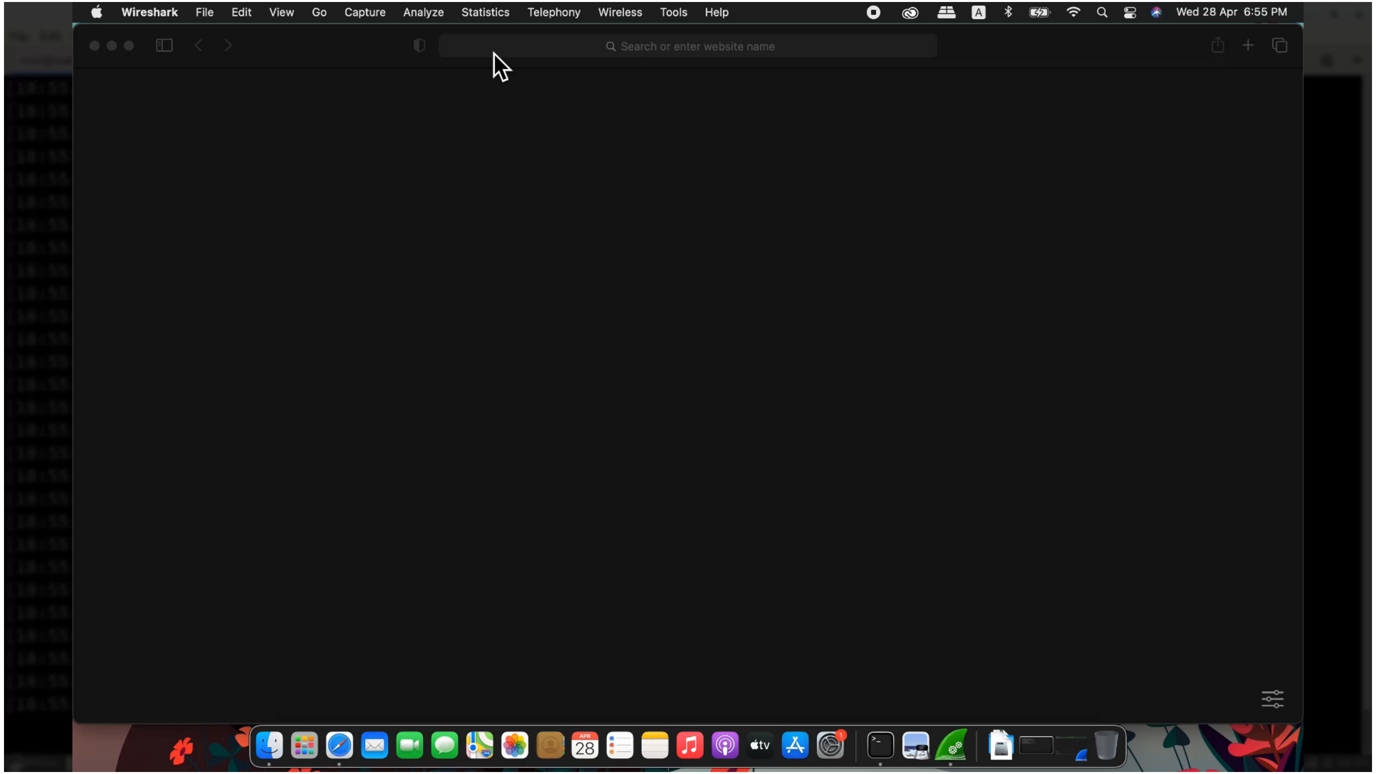


Bug in AP → do attack **w/o user interaction**

(affected $^2/_4$ of home APs)

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

DEMO: FragAttacks A-MSDU Flaw



Agenda

- Attacks that introduced new threat models:
  - The BEAST and HEIST attack (TLS/HTTPS)
  - The Multi-Channel MitM (KRACK)
  - Outbound Connections (FragAttacks)
  - **Client Isolation (Framing Frames)**
  - DNS Spoofing & VPNs (TunnelCrack)
- Conclusion

Bypassing Wi-Fi client isolation

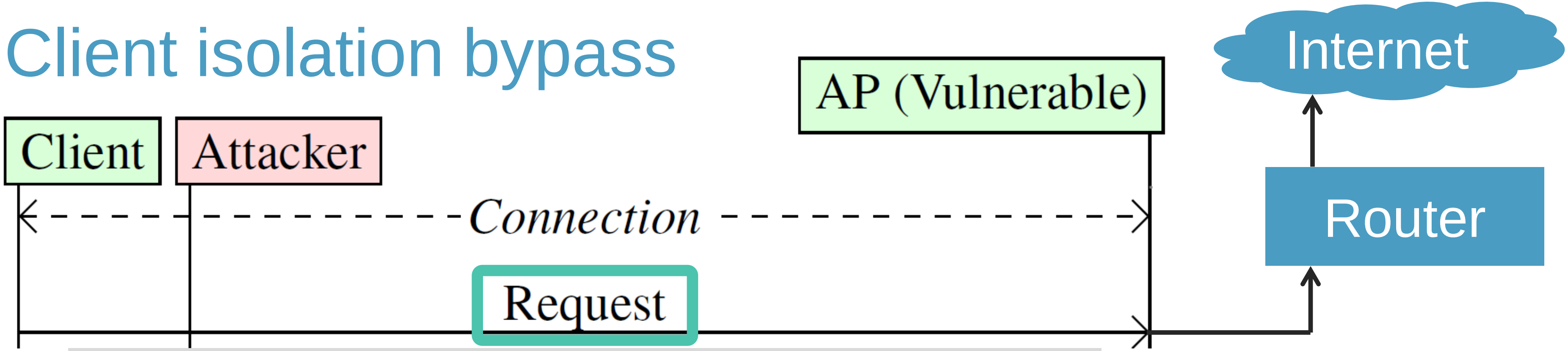
Many networks use **client isolation**. Examples:

- Company network to contain malicious/compromised clients
- Protected hotspots to prevent users attacking each other

- Client isolation is not part of IEEE 802.11 nor WPA*
- “bolted on” by vendors → Wi-Fi meets a new thread model ☺

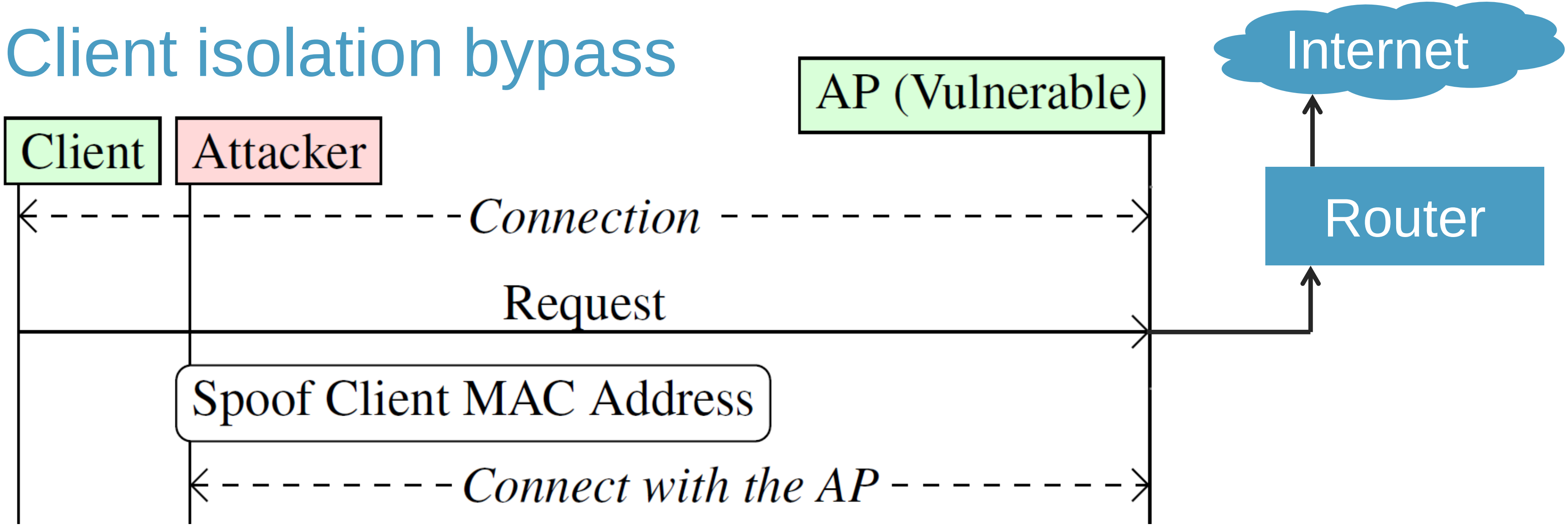
Client isolation bypass

Client isolation bypass

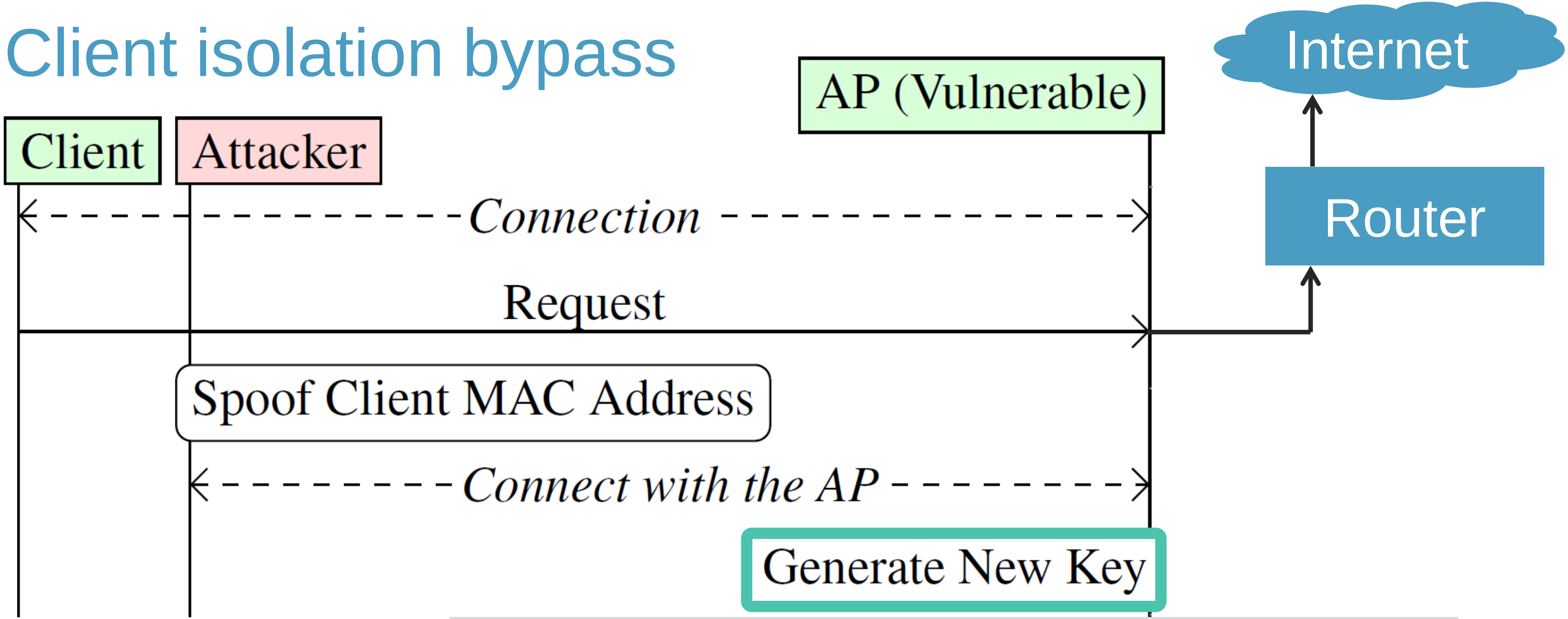


E.g., DNS or HTTP request

Client isolation bypass

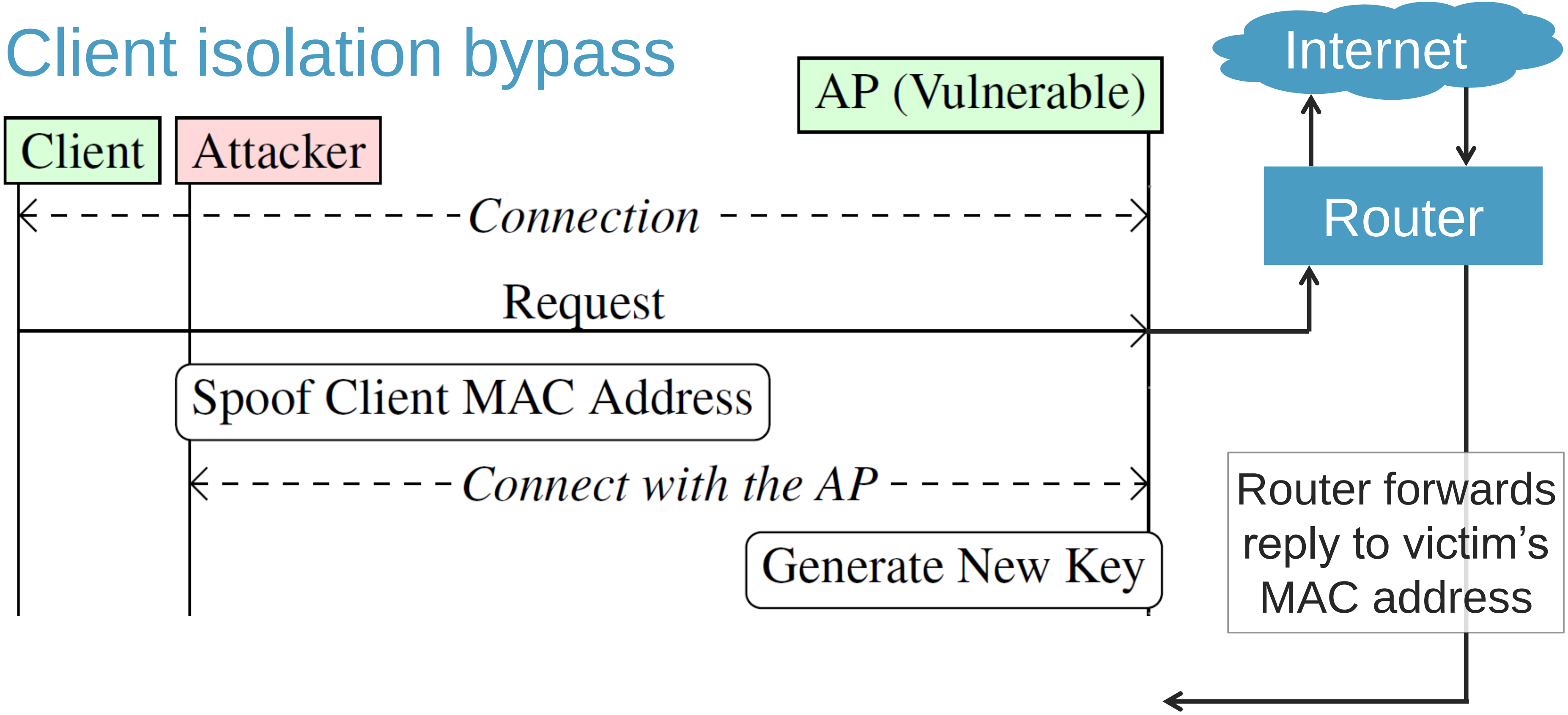


Client isolation bypass

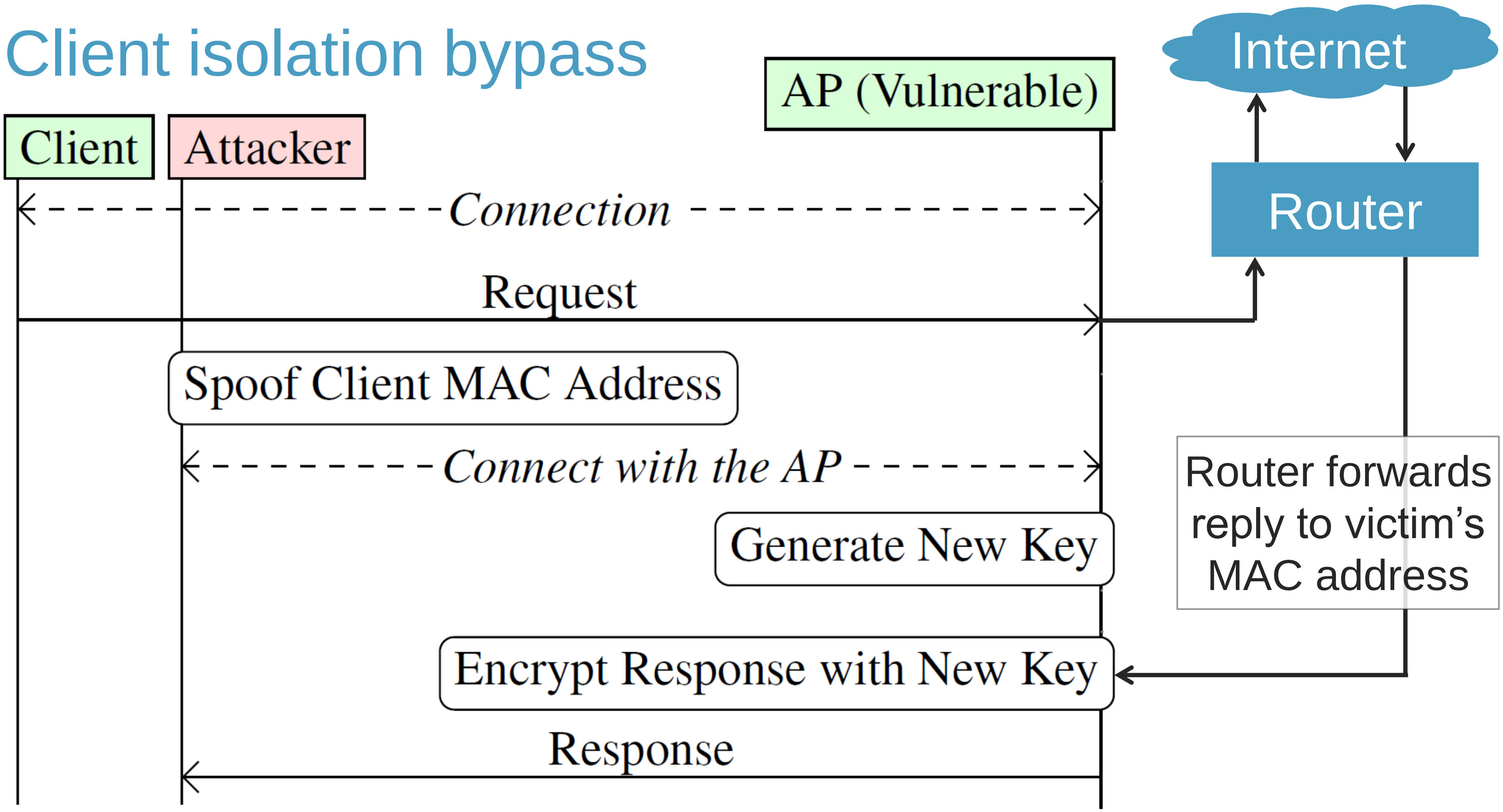


New key is associated with the victim's MAC address

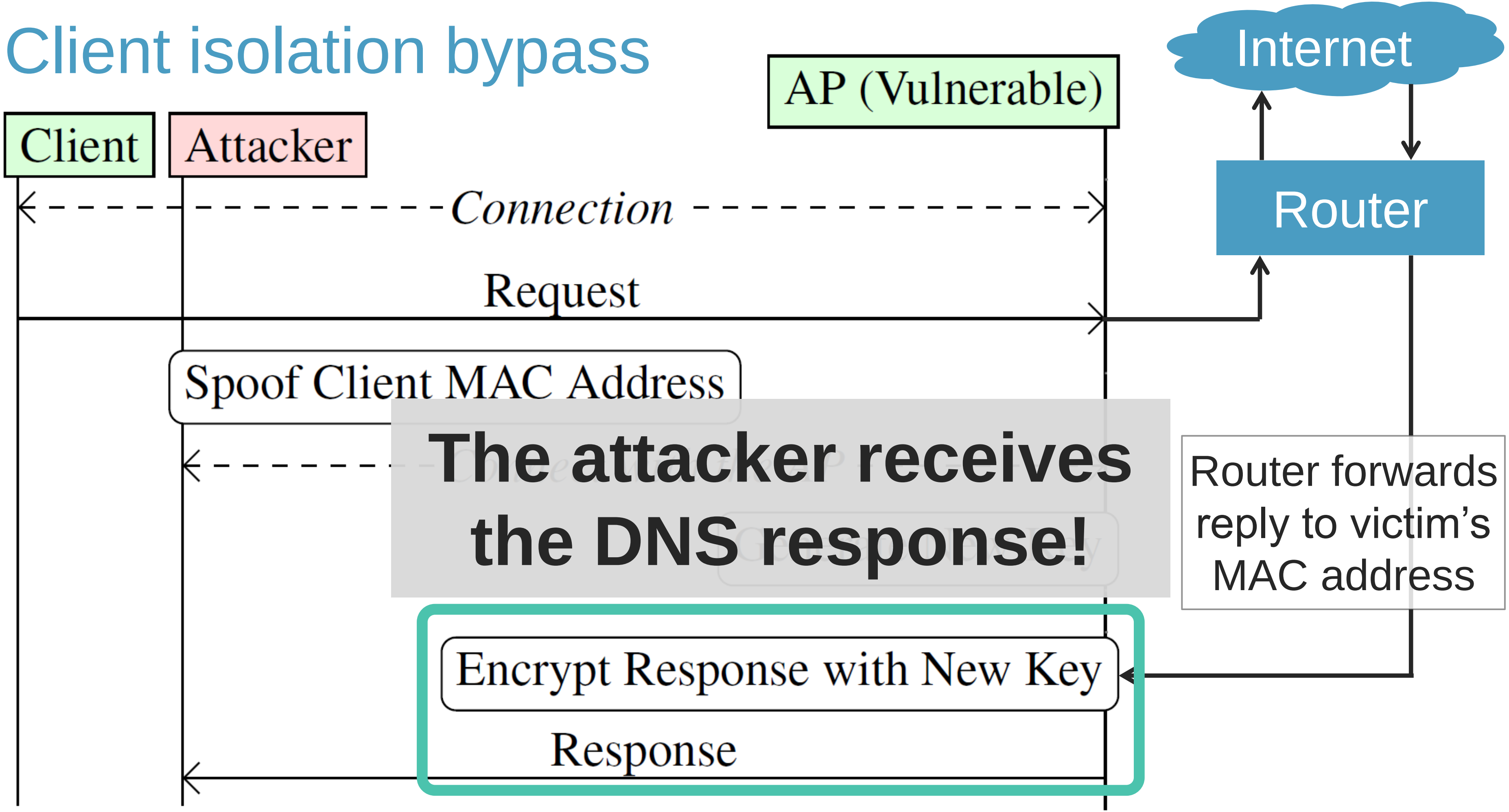
Client isolation bypass



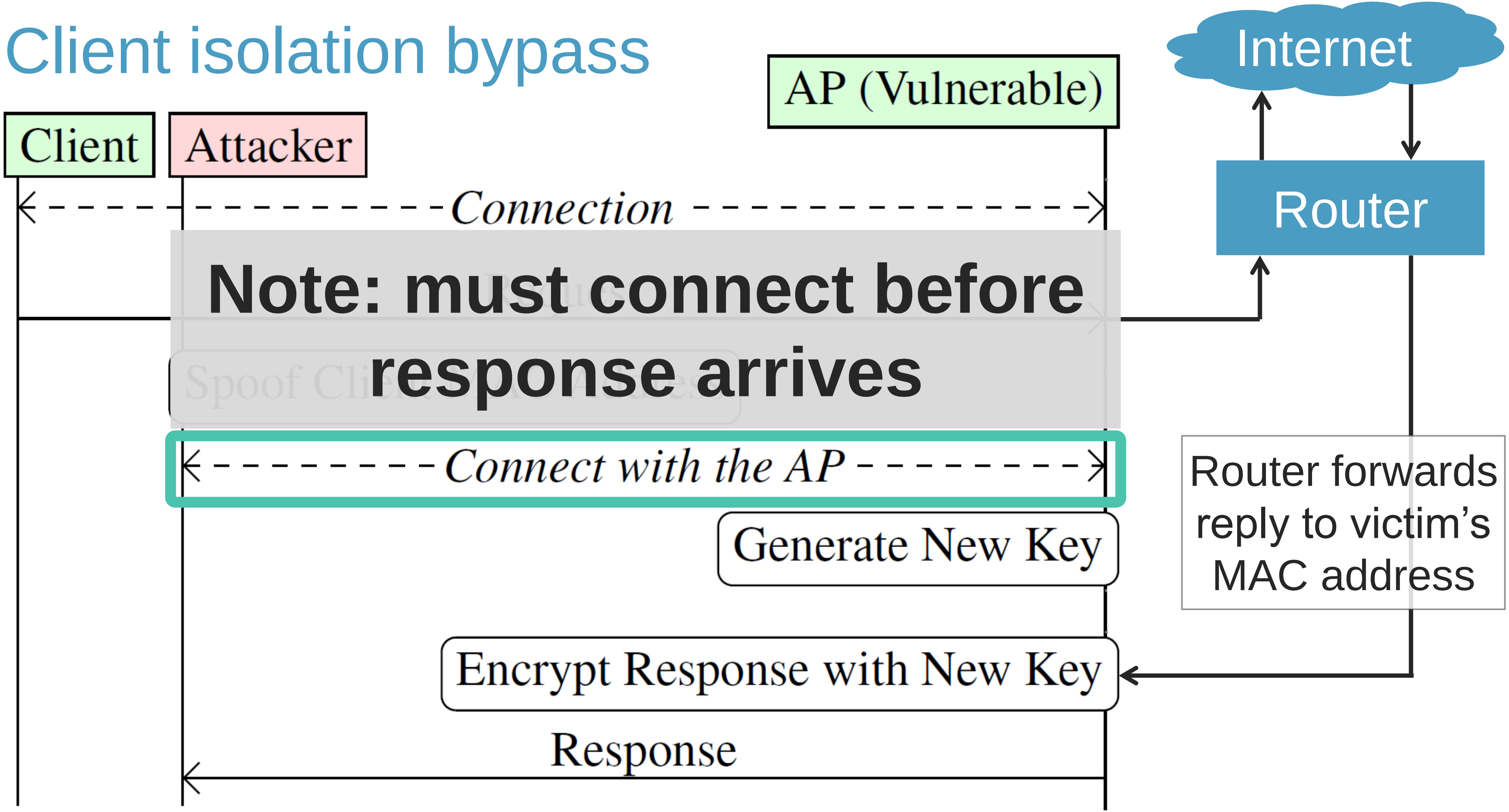
Client isolation bypass



Client isolation bypass



Client isolation bypass



Agenda

› Attacks that introduced new threat models:
  - The BEAST and HEIST attack (TLS/HTTPS)
  - The Multi-Channel MitM (KRACK)
  - Outbound Connections (FragAttacks)
  - Client Isolation (Framing Frames)
  - **DNS Spoofing & VPNs (TunnelCrack)**

› Conclusion

Background: VPN client routing table


```
  $ ip route
1 default via tun0
```

1. By default, send packets over tun0 = over the VPN tunnel

Background: VPN client routing table


```
  $ ip route
1 default via tun0
2 2.2.2.2 via eth0
```

1. By default, send packets over tun0 = over the VPN tunnel
2. **ServerIP exception**: avoid re-encryption of VPN packets

We assume secure DNS behavior


```
$ cat /etc/resolv.conf
nameserver 6.6.6.6
```

Can’t trust the network’s DNS server

We assume secure DNS behavior

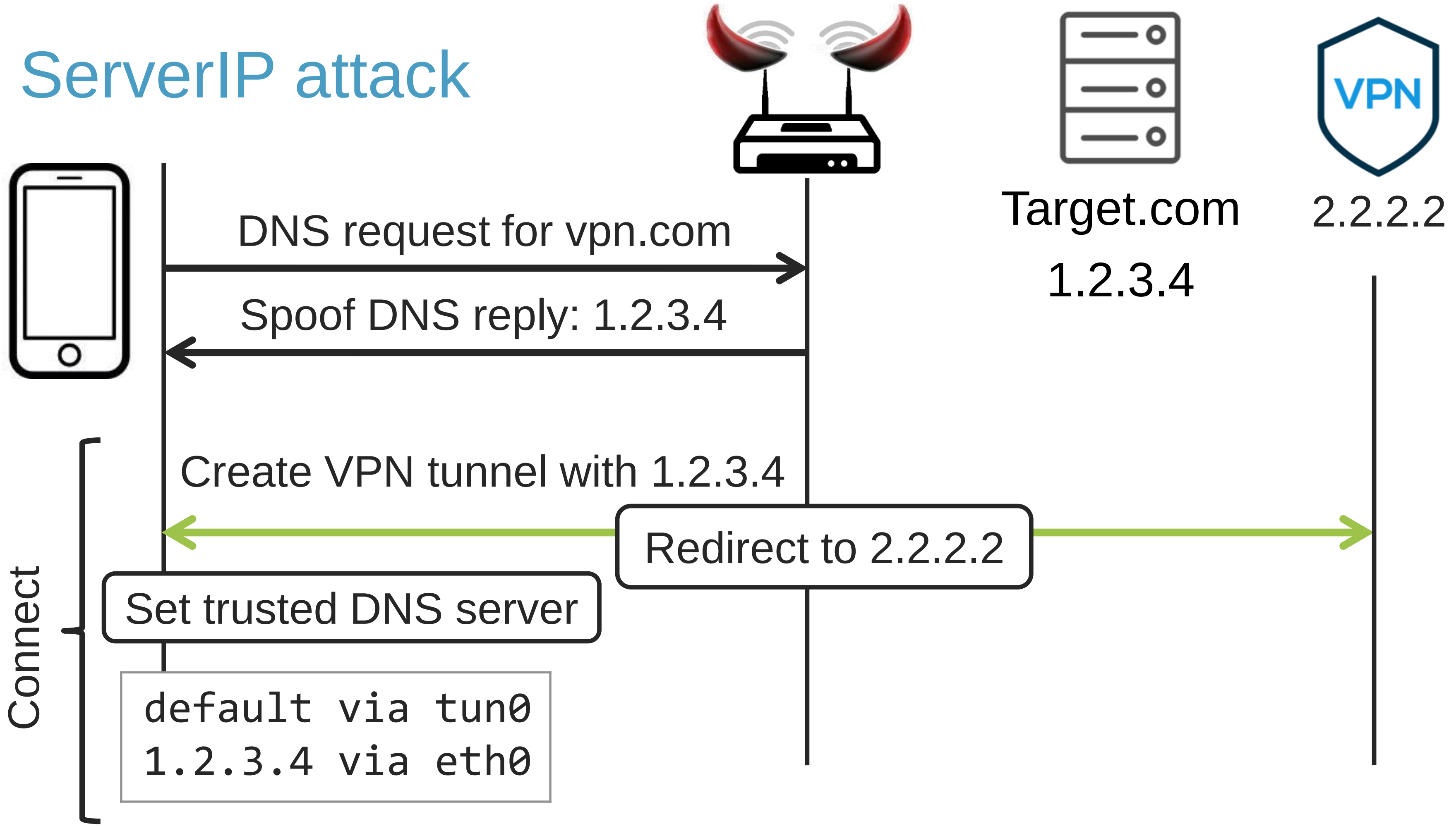
```
$ cat /etc/resolv.conf
nameserver 2.2.2.3
```

Can’t trust the network’s DNS server. Once connected:

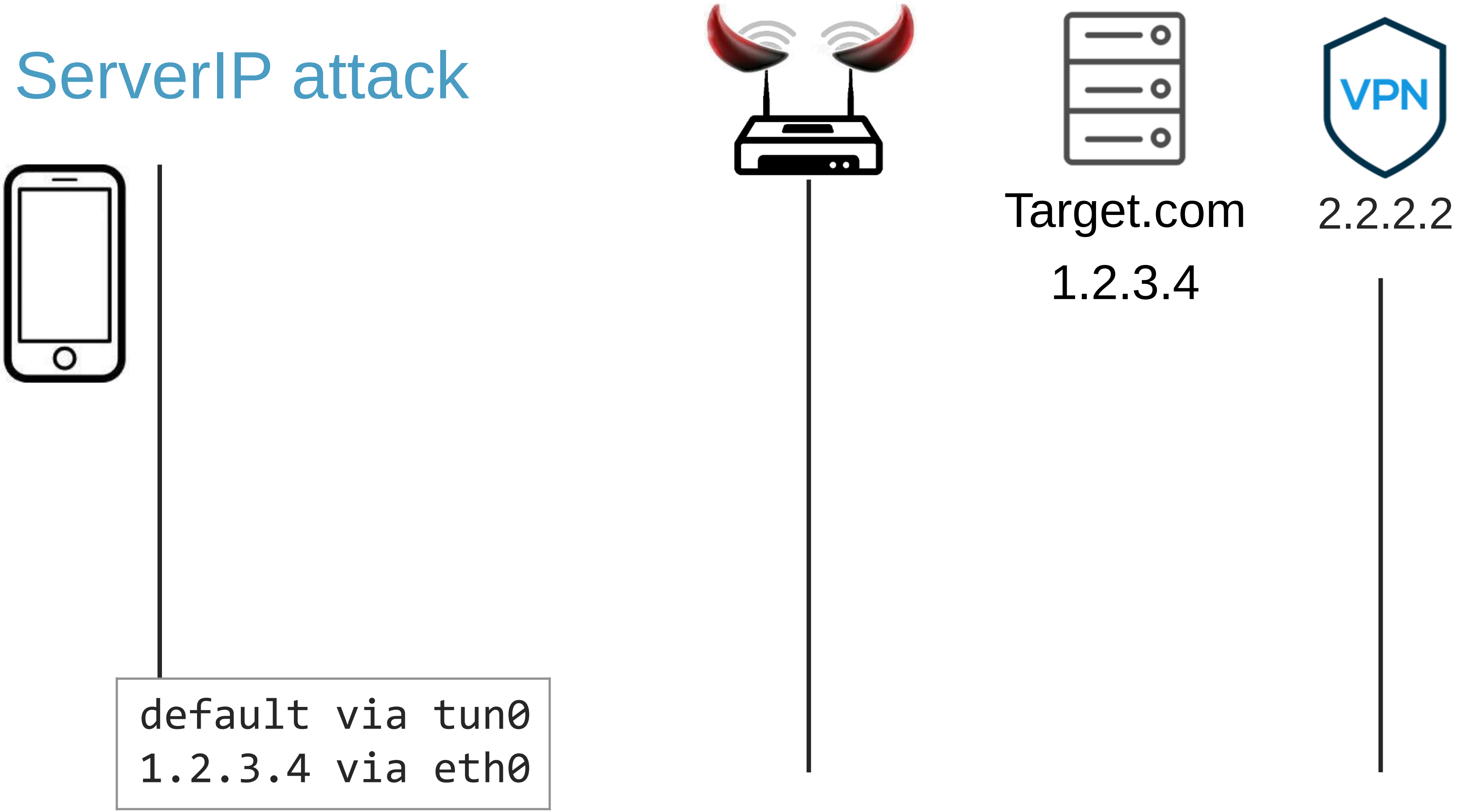
1. The VPN client sets a **trusted DNS server**
2. DNS is sent **through the VPN tunnel**

   + we assume other routing-based attacks are prevented

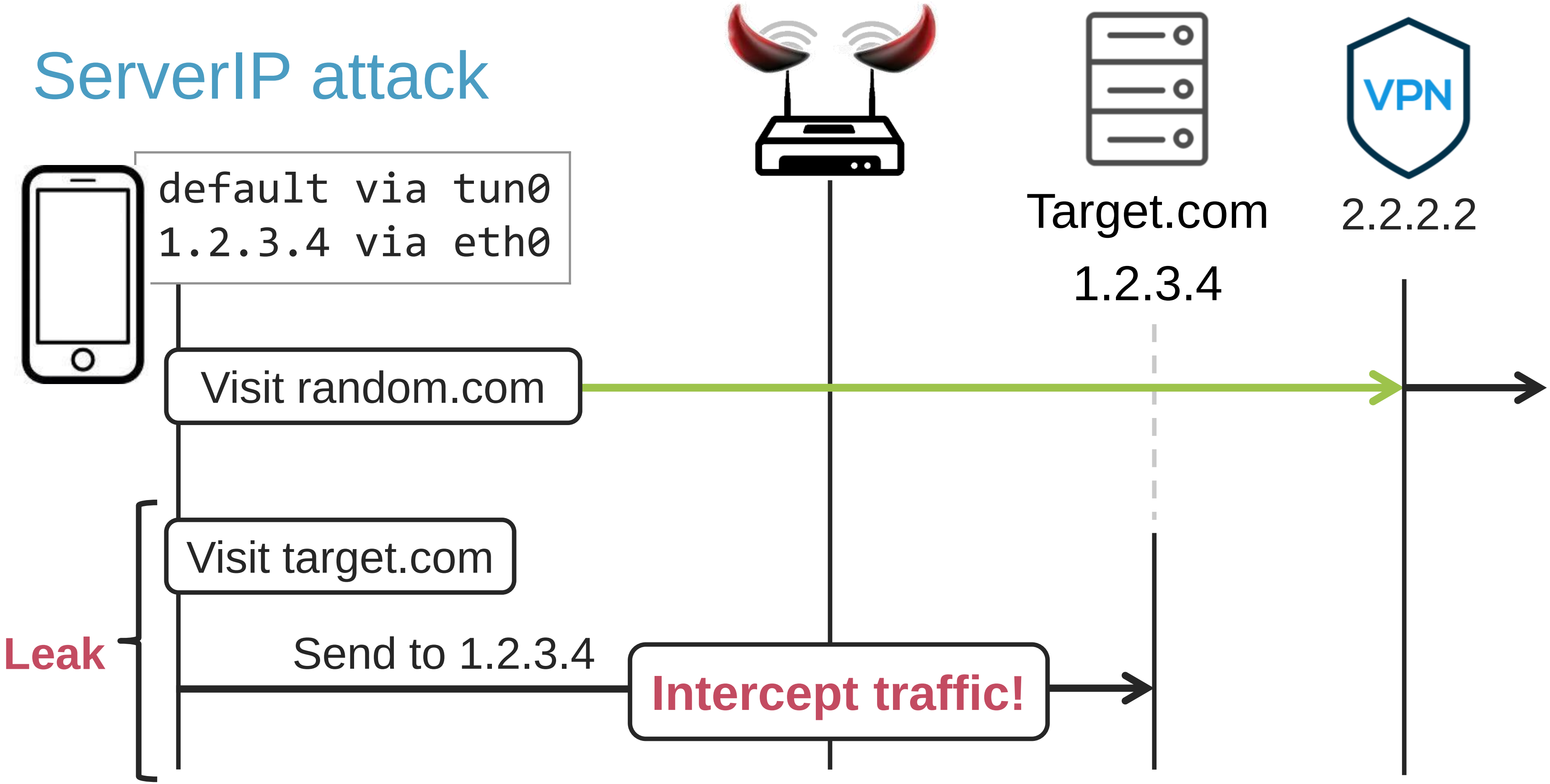
ServerIP attack



ServerIP attack



ServerIP attack



Reflection

- Cryptography of VPNs was widely studied…
- …but not their integration into real-world systems

- Attack the weakest link! In this case the routing tables.
- The DNS poisoning happens *before* VPN is enabled!

Conclusion

- Established protocols, when used in new situations or under new thread models, may become vulnerable to new attacks
- When reading about attacks, learn about the threat model, that may be the most useful thing to know in the long term.
- “Attacks only get better” -- but why?
  - Either by finding new vulnerabilities…
  - …or by considering new threat models!