Sicherheitsrisiko in der libpcap

Wann sind tcpdump, snort und wireshark blind?

Agenda.

1. tcpdump und libpcap
2. Berkeley Packet Filter
3. VLAN Bug
4. Zusammenfassung

Netzwerkanalyse

Werkzeuge

- tcpdump
- wireshark
- etherape
- dsniff
- snort

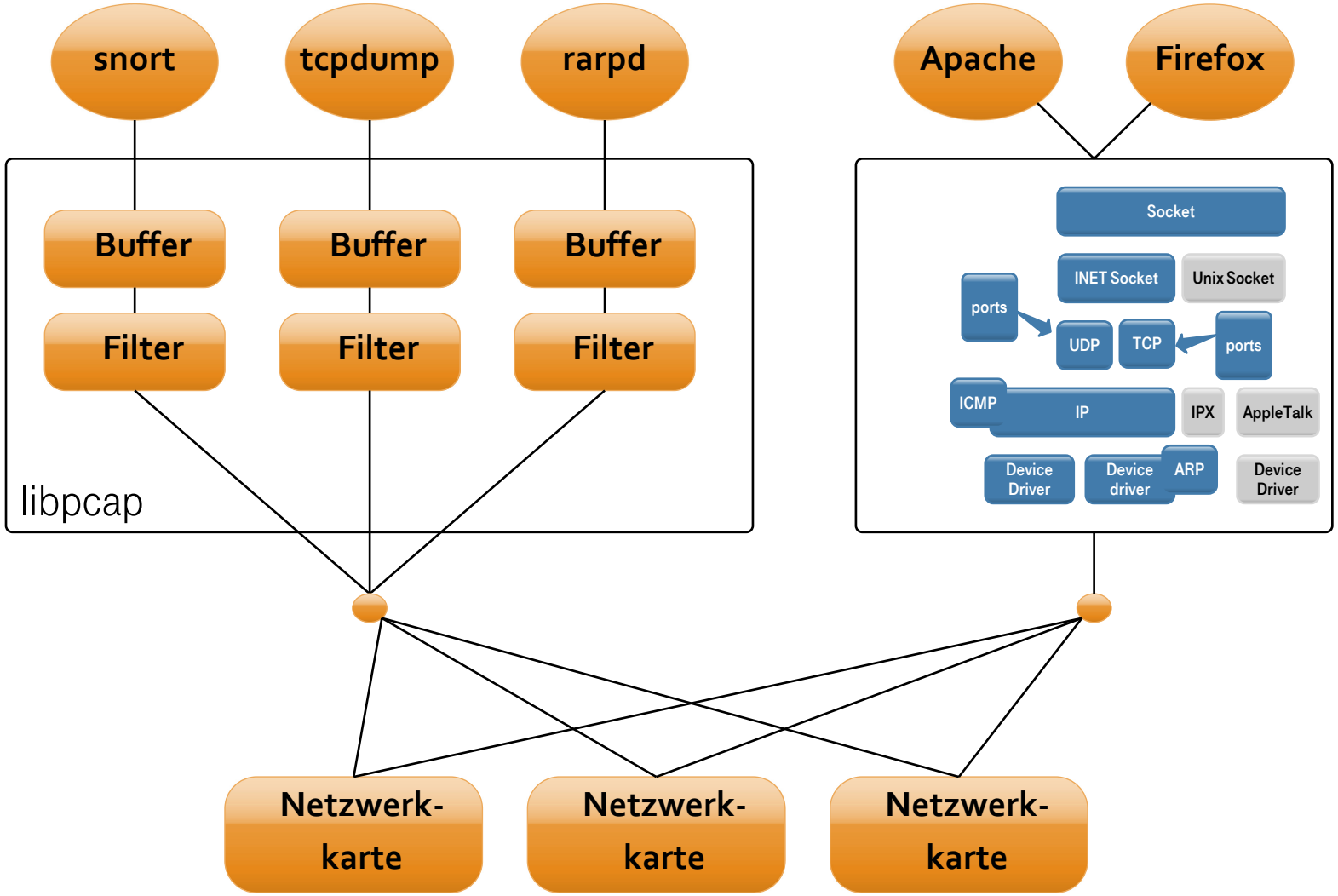
Anforderungen

- Zugriff auf die Rohdaten des Netzwerks
- Pufferung der Daten
- Filterung der Daten
- Werkzeug darf keine zusätzliche Fehlerquelle sein

Der Protokollstack des Betriebssystems ist ungeeignet

libpcap

libpcap


snort
tcpdump
rarpd
Apache
Firefox
Buffer
Buffer
Buffer
Filter
Filter
Filter
libpcap
Socket
INET Socket
Unix Socket
ports
UDP
TCP
ports
ICMP
IP
IPX
AppleTalk
Device Driver
Device driver
ARP
Device Driver
Netzwerk-karte
Netzwerk-karte
Netzwerk-karte

Agenda.

1. tcpdump und libpcap
2. Berkeley Packet Filter
3. VLAN Bug
4. Zusammenfassung

Berkeley Packet Filter
Beispiel

- `root@host:~# tcpdump -n -c 5 -e -i eth0 host foo`

- Sammeln nur von interessanten Paketen
  - Der Filter beschreibt „interessant“
  - Zugriff auf Bits und Bytes muss möglich sein **ip[20:2] = 0x1234**
  - Die Beschreibung erlaubt allgemeine Begriffe wie **host** oder **src port 20**
- **Problem:**
  - Verschachtelte Header
  - Variable Headerlänge
- **Lösung:** BPF Pseudo Maschine
  - Eingeschränkter Befehlssatz
  - Nur Vorwärtssprünge

Berkeley Packet Filter
Behind the scene

```
tcpdump -d host 1.2.3.4
(000) ldh      [12]
(001) jeq      #0x800           jt 2    jf 6
(002) ld       [26]
(003) jeq      #0x1020304       jt 12   jf 4
(004) ld       [30]
(005) jeq      #0x1020304       jt 12   jf 13
(006) jeq      #0x806           jt 8    jf 7
(007) jeq      #0x8035          jt 8    jf 13
(008) ld       [28]
(009) jeq      #0x1020304       jt 12   jf 10
(010) ld       [38]
(011) jeq      #0x1020304       jt 12   jf 13
(012) ret      #65535
(013) ret      #0
```

Berkeley Packet Filter
Behind the scene

```
tcpdump -d host 1.2.3.4
(000) ldh      [12]
(001) jeq      #0x800           jt 2    jf 6
```

Übersetzung:

- **(000)** Lade 2 Byte beginnend beim Offset 12 des Ethernetpaketes
- **(001)** Ist der Wert ox800, dann springe zu Zeile 2 ansonsten zu Zeile 6

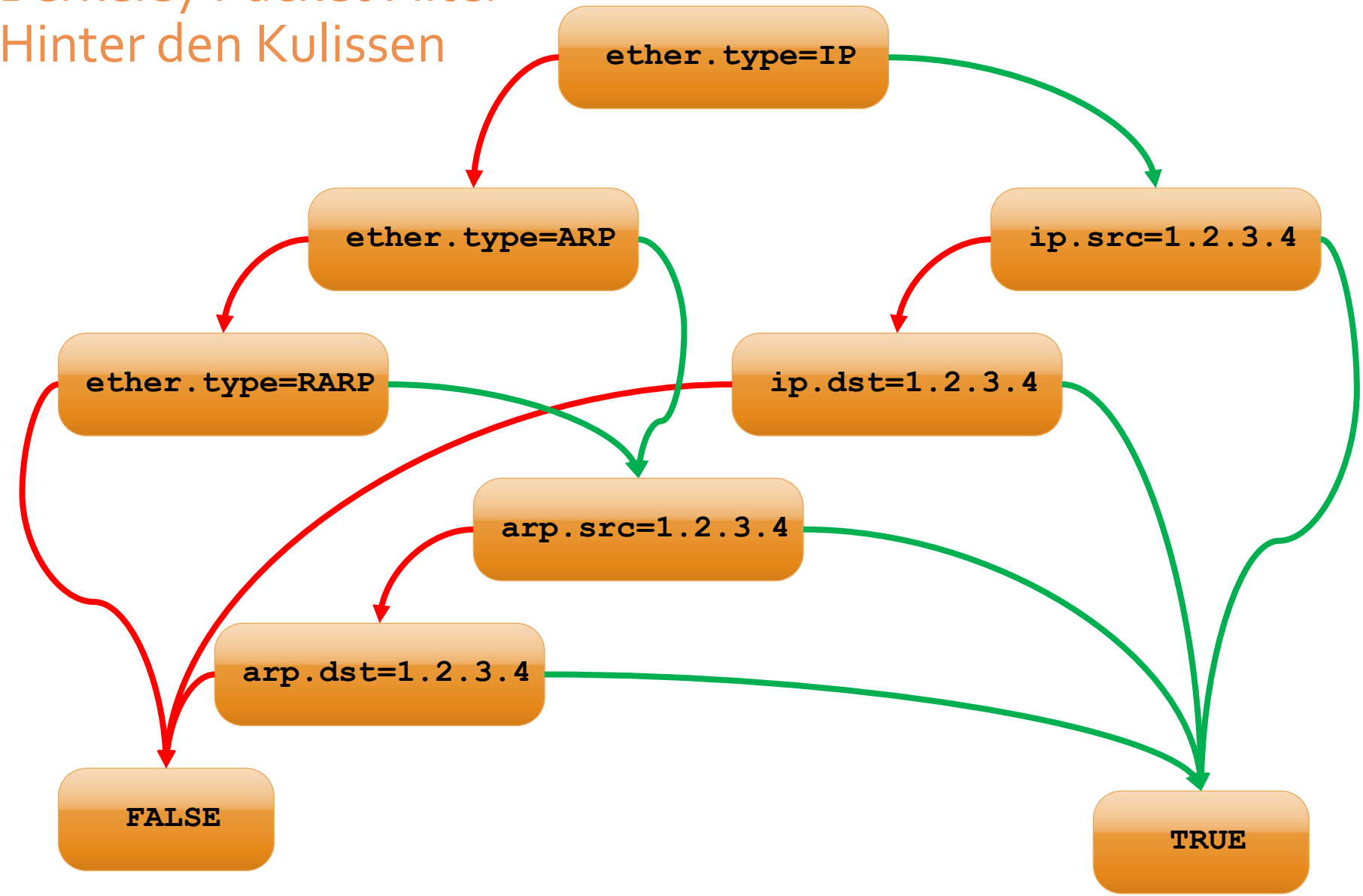
Bedeutung:

- **(000)** Lade das Typ-Feld aus dem Ethernet Header
- **(001)** Ist es ein IP-Paket? Ja: Zeile 2 Nein: Zeile 6

Kurzfassung:

- **ether.type = IP**

Berkeley Packet Filter
Hinter den Kulissen



Agenda.

1. tcpdump und libpcap
2. Berkeley Packet Filter
3. VLAN Bug
4. Zusammenfassung

VLAN

- Virtual LAN
- 802.1q
- Headerlänge: 4 Byte
- „Eingeschoben zwischen Ethernet-Header und IP-Header“
- Beliebt in der Industrie
  - Netzwerker (weniger Kabel)
  - Virtuelle Systeme (beinahe unumgänglich)
  - Gefühlte Sicherheit (leider)

- Berkeley Packet Filter: **`vlan [vlanid]`**

Live Demo I

- Übersicht
- `windump -n -l -r demo_vlan.pcap`
- Suche nach Port 6000 – fehlende Pakete
- `windump -n -l -r demo_vlan.pcap „port 6000“`
- Suche nach Port 6000 – fehlende Pakete -VLAN
- `windump -e -n -l -r demo_vlan.pcap „port 6000“`
- Suche nach Port 6000 – fehlende Pakete -VLAN
- `windump -e -n -l -r demo_vlan.pcap „vlan 32 or port 6000“`

Live Demo II

- Suche nach Port 6000 – Quellcode
- `windump –d „port 6000“`
- Suche nach Port 6000 oder VLAN – Quellcode
- `windump –d „vlan 32 or port 6000“`
- Suche nach Port 6000 oder VLAN– Quellcode
- `windump –d „ port 6000 or vlan 32“`
- Verschiebung der Adressen
- `windump -n -l -r demo_vlan.pcap "not vlan or src host 10.3.47.39"`

Agenda.

1. tcpdump und libpcap
2. Berkeley Packet Filter
3. VLAN Bug
4. Zusammenfassung

Zusammenfassung

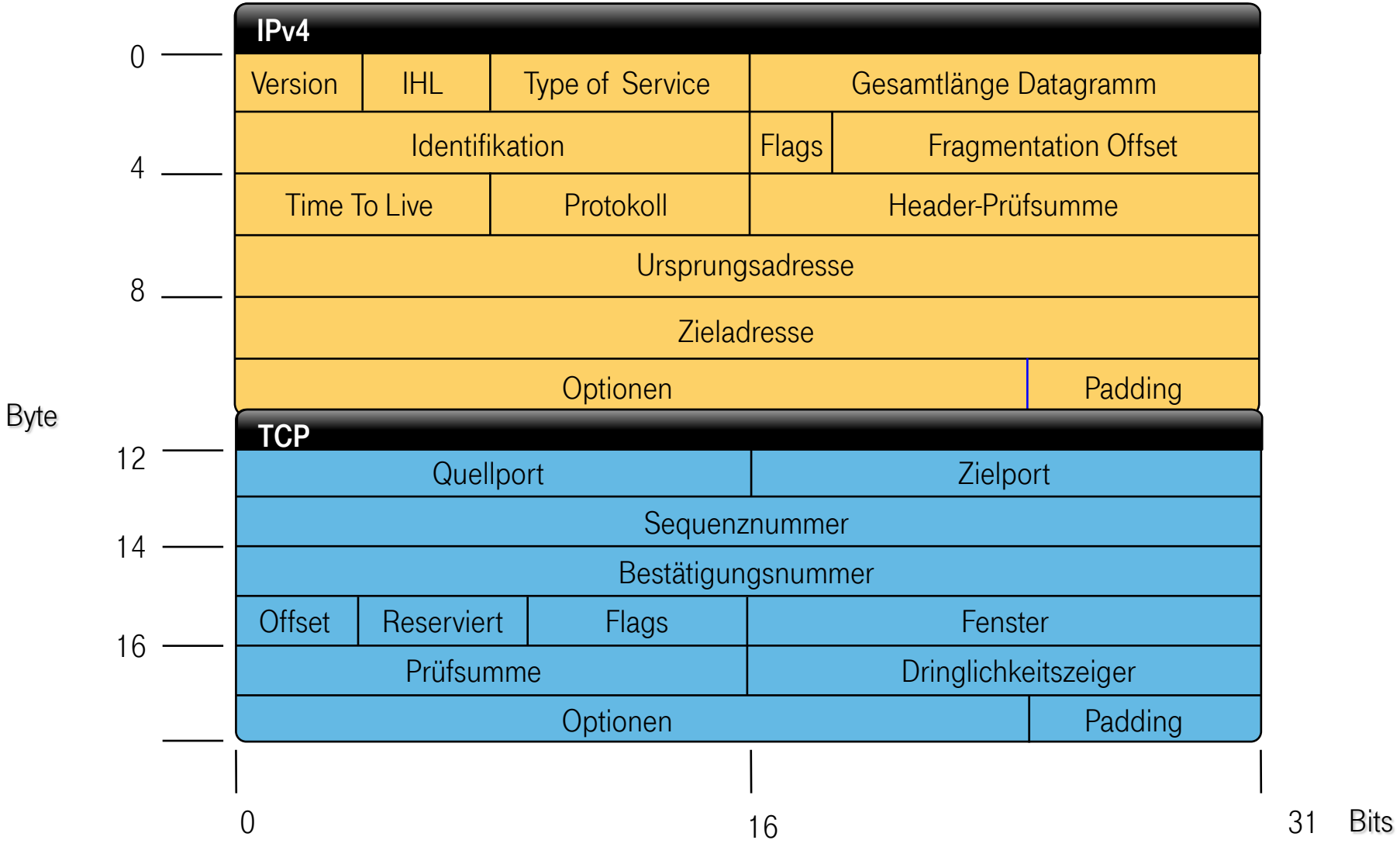
- tcpdump, snort, wireshark, etc.
  - bauen auf der libpcap auf
  - nutzen somit Berkeley Packet Filter
- aktuelle Implementierungen der libpcap
  - haben einen Designfehler bei der Analyse von Vlans
  - blenden zu viele Pakete aus
- Fehlerbehebung
  - komplex
  - die Autoren wissen Bescheid

Literatur

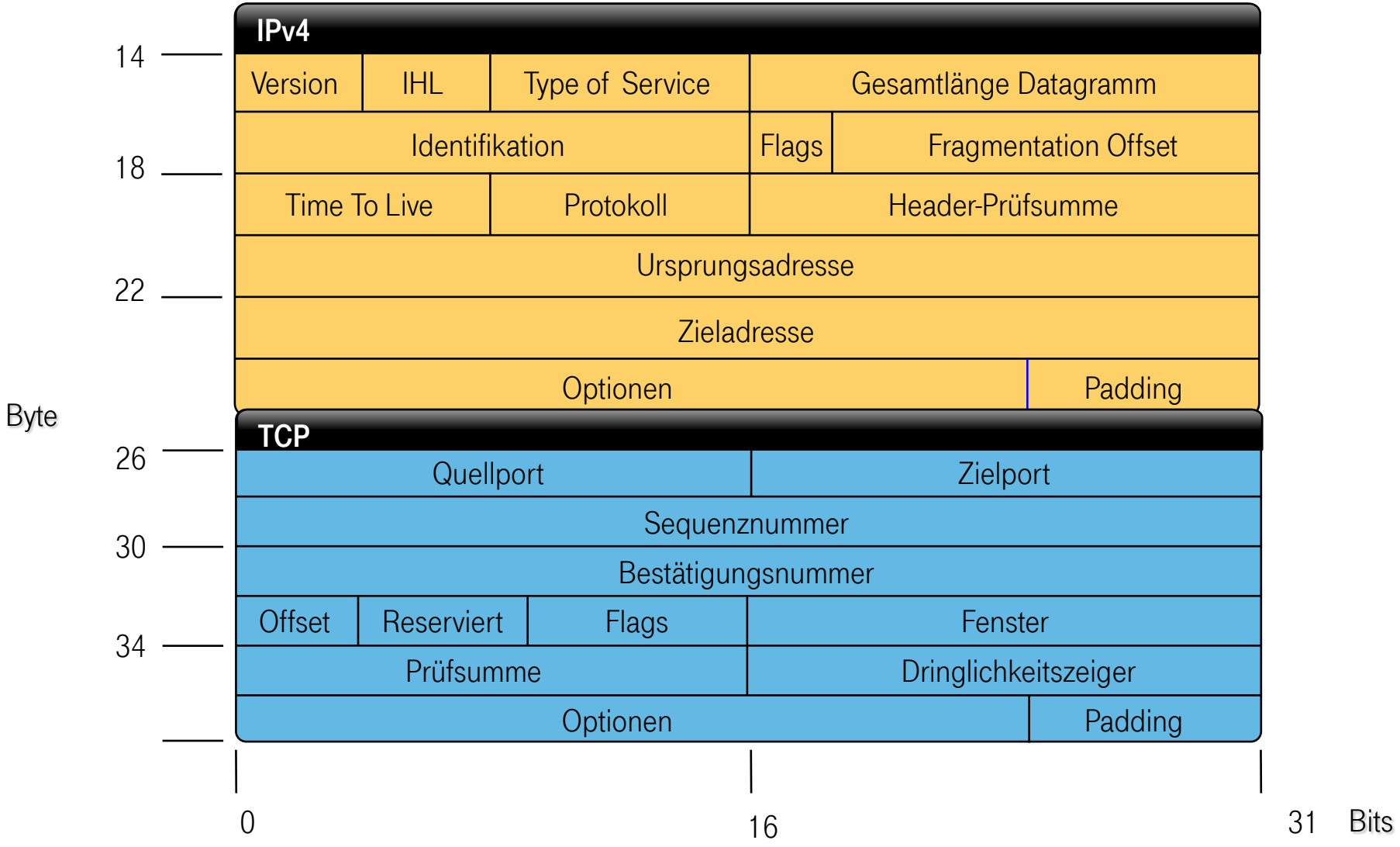
- TCP/IP
  - W. Richard Stevens
- The BSD Packet Filter: A New Architecture for User-level Packet Capture
  - Steven McCanne & Van Jacobson

Vielen Dank!

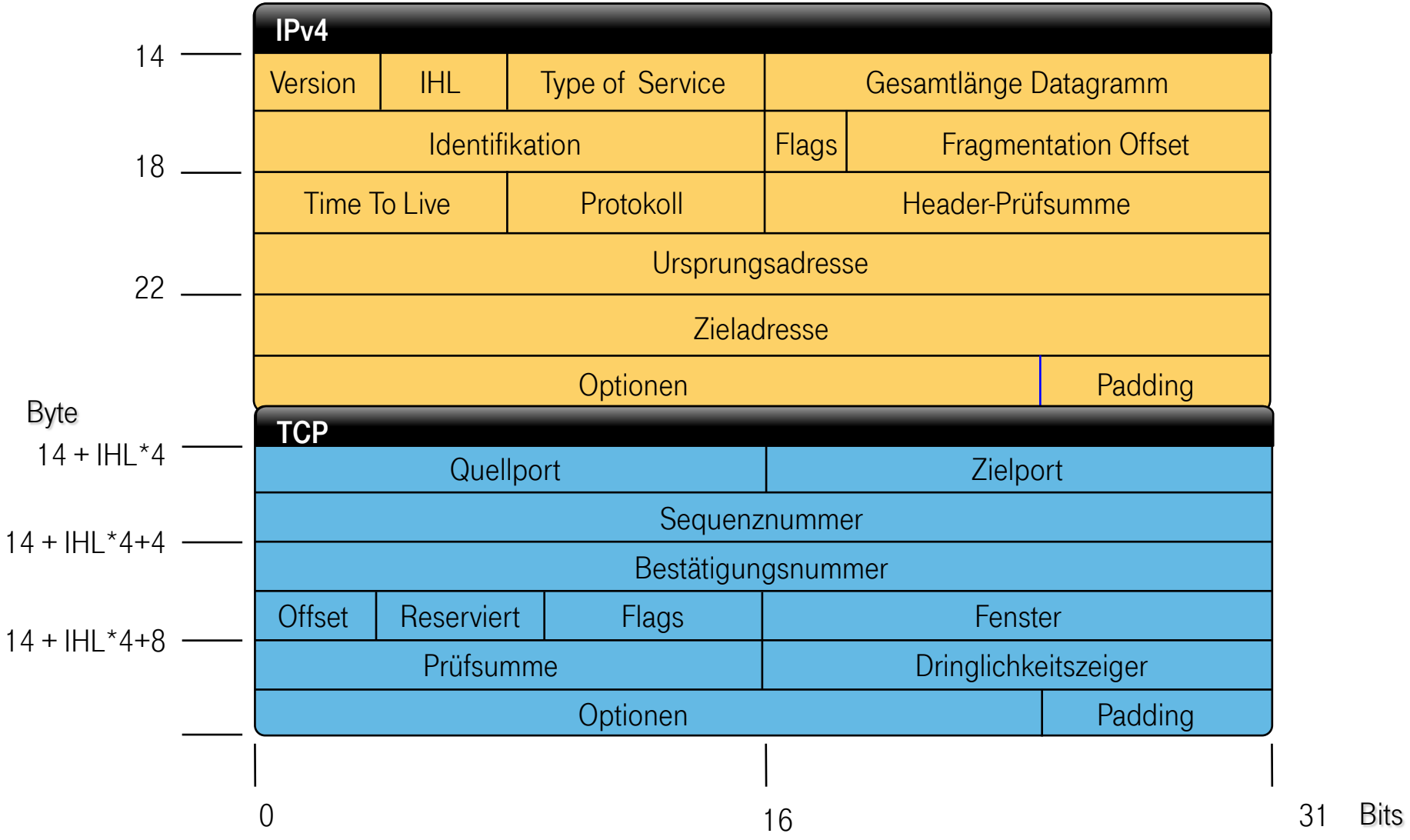
TCP/IP


IPv4
Version
IHL
Type of Service
Gesamtlänge Datagramm
Identifikation
Flags
Fragmentation Offset
Time To Live
Protokoll
Header-Prüfsumme
Ursprungsadresse
Zieladresse
Optionen
Padding
TCP
Quellport
Zielport
Sequenznummer
Bestätigungsnummer
Offset
Reserviert
Flags
Fenster
Prüfsumme
Dringlichkeitszeiger
Optionen
Padding
Byte
0
4
8
12
14
16
0
16
31
Bits

TCP/IP


IPv4
Version
IHL
Type of Service
Gesamtlänge Datagramm
Identifikation
Flags
Fragmentation Offset
Time To Live
Protokoll
Header-Prüfsumme
Ursprungsadresse
Zieladresse
Optionen
Padding
TCP
Quellport
Zielport
Sequenznummer
Bestätigungsnummer
Offset
Reserviert
Flags
Fenster
Prüfsumme
Dringlichkeitszeiger
Optionen
Padding
Byte
14
18
22
26
30
34
0
16
31
Bits

TCP/IP


IPv4
14
Version
IHL
Type of Service
Gesamtlänge Datagramm
18
Identifikation
Flags
Fragmentation Offset
Time To Live
Protokoll
Header-Prüfsumme
22
Ursprungsadresse
Zieladresse
Optionen
Padding
Byte
TCP
14 + IHL*4
Quellport
Zielport
14 + IHL*4+4
Sequenznummer
Bestätigungsnummer
14 + IHL*4+8
Offset
Reserviert
Flags
Fenster
Prüfsumme
Dringlichkeitszeiger
Optionen
Padding
0
16
31
Bits