

Thema

Laborprotokoll TCP-UPD Header Verleich



Figure 1: memes klauen ist nicht ethisch

Unterrichtsgegenstand: NWT1|ZIVK
Jahrgang: 2BHIT
Name: Stefan Fürst, Marcel Raichle
Betreuer: ZIVK
Übungsdaten: 24.5.2024, 31.5.2024, 7.6.2024
Abgabedatum: 7.6.2024

Contents

1	Aufgabenstellung	3
2	Zusammenfassung	3
3	Übungsdurchführung	4
3.1	Aufsetzen der Server	4
3.1.1	Verbinden mit dem Sever (TCP)	4
3.1.2	TCP-Verbindungsaufbau	4
3.1.3	Nachrichten verschicken und empfangen	4
3.1.4	TCP-Flags	5
3.1.5	TCP-Header	6
3.1.6	Verbindungsabbruch	7
3.1.7	Verbinden mit dem Sever (UDP)	7
3.1.8	UDP-Verbindungsaufbau	7
3.1.9	UDP-Nachrichten	7
3.1.10	UDP-Header	8
3.1.11	UDP-Verbindungsabbruch	8
3.2	TCP und UDP Headervergleich	9
3.3	Nach Ports filtern	9
4	Quellen	10
5	Abbildungsverzeichnis	11

1 Aufgabenstellung

Zusammenfassung der Aufgabenstellung von Cat-gpt:

1. Work in groups of 2, each with their own PC and Virtual Box with Kali Linux. Make sure network settings are correct and virtual machines have IP addresses. Verify with a ping. Start Wireshark and record the ping. You'll need to capture other transmissions too. Document every step in a log.
2. Use netcat to start a TCP server on port 5000. Connect with your partner using netcat. Document with screenshots and Wireshark. Answer questions about TCP connections and flags. Find and document TCP header fields. Document what happens when the server is closed.
3. Use netcat to start a UDP server on port 5000. Connect with your partner, document with screenshots and Wireshark. Answer questions about UDP connections and headers. Document server closure process.
4. Research TCP and UDP headers. Show and describe the most important fields in a one-page document. Compare header sizes and differences between TCP and UDP headers.
5. Bonus: Document how TCP and UDP are recorded in Wireshark. Research how port addresses are displayed. Show which Wireshark filters to use.

2 Zusammenfassung

Netcad um einen TCP/UPD Server starten, mit Netcad verbinden und mit Wireshark die Verbindungen analysieren.

3 Übungsdurchführung

3.1 Aufsetzen der Server

Listing 1: Commands

```
# TCP
nc -l -p 5000
# UDP
nc -l -u -p 5000
```

3.1.1 Verbinden mit dem Sever (TCP)

nc 10.23.38.117 4201

3.1.2 TCP-Verbindungsaufbau

No.	Time	Source	Destination	Protocol	Length	Info
70	17.48970	10.23.38.100	10.23.38.117	TCP	74	43440 → 4201 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=3589641073 T...
71	17.49217	10.23.38.117	10.23.38.100	TCP	74	4201 → 43440 [ACK, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3589641073 T...
72	17.49226	10.23.38.100	10.23.38.117	TCP	66	43440 → 4201 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=3589641076 TScsr=352486392

Figure 2: TCP 3 Way Handshake

3.1.3 Nachrichten verschicken und empfangen

The screenshot shows a Wireshark packet capture. The top pane displays a list of packets. Packet 32 is selected, showing its details in the middle pane. The packet is a TCP Reset (RST) from 10.23.38.117 to 10.23.38.117. The bottom pane shows the raw packet data in hexadecimal and ASCII. The ASCII column shows the text "banana trinkt" followed by a green arrow pointing to the end of the data.

No.	Time	Source	Destination	Protocol	Length	Info
70	17.48970..	10.23.38.100	10.23.38.117	TCP	74	43440 → 4201 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=3589641073 T...
71	17.49217..	10.23.38.117	10.23.38.100	TCP	74	4201 → 43440 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3...
72	17.49226..	10.23.38.100	10.23.38.117	TCP	66	43440 → 4201 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=3589641076 TSecr=352486492
32	33.71605..	10.23.38.100	10.23.38.117	TCP	66	43440 → 4201 [RST, ACK] Seq=2 Ack=1 Win=32128 Len=0 TSval=3589955344 TSecr=35...
32	33.74172..	10.23.38.100	10.23.38.117	TCP	74	40586 → 4201 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=3589955731 T...
32	33.74184..	10.23.38.117	10.23.38.100	TCP	74	4201 → 40586 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3...
32	33.7485..	10.23.38.100	10.23.38.117	TCP	66	40586 → 4201 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=3589955732 TSecr=35280103...
32	33.72752..	10.23.38.117	10.23.38.100	TCP	71	4201 → 40586 [FSH, ACK] Seq=1 Ack=1 Win=65280 Len=5 TSval=352801431 TSecr=3589...
32	33.72752..	10.23.38.100	10.23.38.117	TCP	42	40586 → 4201 [ACK] Seq=1 Ack=6 Win=32128 Len=0 TSval=3589955752 TSecr=352801431
33	33.7792..	10.23.38.117	10.23.38.100	TCP	71	4201 → 40586 [FSH, ACK] Seq=6 Ack=1 Win=65280 Len=5 TSval=352805937 TSecr=3589...
33	33.7792..	10.23.38.100	10.23.38.117	TCP	66	40586 → 4201 [ACK] Seq=1 Ack=11 Win=32128 Len=0 TSval=3589960363 TSecr=3528059...
34	36.53155..	10.23.38.117	10.23.38.100	TCP	79	4201 → 40586 [FSH, ACK] Seq=11 Ack=1 Win=65280 Len=13 TSval=352834488 TSecr=35...
34	36.53156..	10.23.38.100	10.23.38.117	TCP	66	40586 → 4201 [ACK] Seq=1 Ack=24 Win=32128 Len=0 TSval=3589988899 TSecr=3528344...
34	37.31783..	10.23.38.117	10.23.38.100	TCP	78	4201 → 40586 [ACK] Seq=24 Ack=1 Win=65280 Len=12 TSval=352842354 TSecr=35...
34	37.31784..	10.23.38.100	10.23.38.117	TCP	66	40586 → 4201 [ACK] Seq=1 Ack=36 Win=32128 Len=0 TSval=3589959752 TSecr=3528423...
35	38.8073..	10.23.38.117	10.23.38.100	TCP	78	4201 → 40586 [FSH, ACK] Seq=36 Ack=1 Win=65280 Len=12 TSval=352855990 TSecr=35...
35	38.8074..	10.23.38.100	10.23.38.117	TCP	66	40586 → 4201 [ACK] Seq=1 Ack=48 Win=32128 Len=0 TSval=3590010391 TSecr=3528559...
37	40.67568..	10.23.38.117	10.23.38.100	TCP	93	4201 → 40586 [FSH, ACK] Seq=48 Ack=1 Win=65280 Len=27 TSval=352875949 TSecr=35...
37	40.67568..	10.23.38.100	10.23.38.117	TCP	66	40586 → 4201 [ACK] Seq=1 Ack=75 Win=32128 Len=0 TSval=3590030340 TSecr=3528759...
37	41.6954..	10.23.38.117	10.23.38.100	TCP	89	4201 → 40586 [FSH, ACK] Seq=75 Ack=1 Win=65280 Len=19 TSval=352886152 TSecr=35...
37	41.6954..	10.23.38.100	10.23.38.117	TCP	66	40586 → 4201 [ACK] Seq=1 Ack=94 Win=32128 Len=0 TSval=3590040538 TSecr=3528861...

No.	Time	Source	Destination	Protocol	Length	Info
Frame 3775:	85 bytes on wire (capture length 85)	Ethernet II, Src: PCSSystem	Internet Protocol Version 4	Transmission Control Protocol	Data (19 bytes)	<pre> 0000 fc 45 96 3a 68 9f 08 00 27 93 9d 01 08 00 45 00 0010 00 47 b7 71 40 00 04 06 22 39 9d 01 26 75 0a 17 0020 26 64 10 69 9e 8a 0c b1 b5 32 53 fa fe 9a 80 18 0030 01 fe ab 38 00 00 01 01 08 0a 15 08 9d 88 45 fb 0040 04 84 02 61 6e 61 6e 65 20 74 72 69 6e 6b 74 20 0050 73 61 6f 66 74 0a [Length: 19] </pre>

Figure 3: Nachricht

3.1.4 TCP-Flags

TCP-Flags dienen dazu um den Zustand, oder andere zusätzliche Informationen der Verbindung anzuzeigen. Diese dienen zum Troubleshooten. In der Übung ist die Push flag gesetzt, was bedeutet, dass die Nachricht sofort übertragen wird, ohne darauf zu warten, dass zusätzliche Informationen auf der Senderseite gebuffert werden.[3]

Wird oft in Echtzeitanwendung benutzt.

```
▼ Transmission Control Protocol, Src Port: 4201, Dst Port: 40586, Seq: 75, Ack: 1, Len: 19
  Source Port: 4201
  Destination Port: 40586
  [Stream index: 32]
  ▶ [Conversation completeness: Incomplete, DATA (15)]
  [TCP Segment Len: 19]
  Sequence Number: 75      (relative sequence number)
  Sequence Number (raw): 212972850
  [Next Sequence Number: 94      (relative sequence number)]
  Acknowledgment Number: 1      (relative ack number)
  Acknowledgment number (raw): 1408958106
  1000 .... = Header Length: 32 bytes (8)
  ▼ Flags: 0x018 (PSH, ACK)
    000. .... = Reserved: Not set
    ...0 .... = Accurate ECN: Not set
    .... 0... = Congestion Window Reduced: Not set
    .... .0.. = ECN-Echo: Not set
    .... ..0. = Urgent: Not set
    .... ...1 = Acknowledgment: Set
    .... .... 1... = Push: Set
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
    [TCP Flags: .....AP....]
  Window: 510
  [Calculated window size: 65280]
  [Window size scaling factor: 128]
  Checksum: 0xab38 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▶ [Timestamps]
  ▶ [SEQ/ACK analysis]
  TCP payload (19 bytes)
```

Figure 4: TCP-Flags

3.1.5 TCP-Header

Vorhandenen Felder

- Source Port
- Destination Port
- Sequence Number
 - Zeigt an wie viele Daten in der TCP Session übertragen werden.
- Acknowledgment Number
 - Vom Empfänger benutzt um das Nächste TCP Segment anzufordern

Nicht Vorhandene Felder

- Window
 - Gibt an wie viele bytes die der Empfänger empfangen will. Wird genutzt damit der Empfänger sagen kann, dass er mehr Daten empfangen will.
- RSV
 - 3 Reservierte Bits, die immer 0 sind.
- Urgent Pointer
 - Wenn die URG-Flag gesetzt ist, zeigt dieser Pointer an, wo die Urgent Daten enden.
- DO
 - Länge des Headers.
- Flags
 - Vorher bereits erklärt.
- Checksum
 - Benutzt für eine Prüfsumme um sicherzugehen, dass der TCP header korrekt is.

[4]

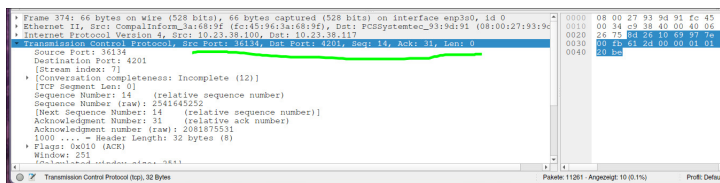


Figure 5: TCP-Header

3.1.6 Verbindungsabbruch

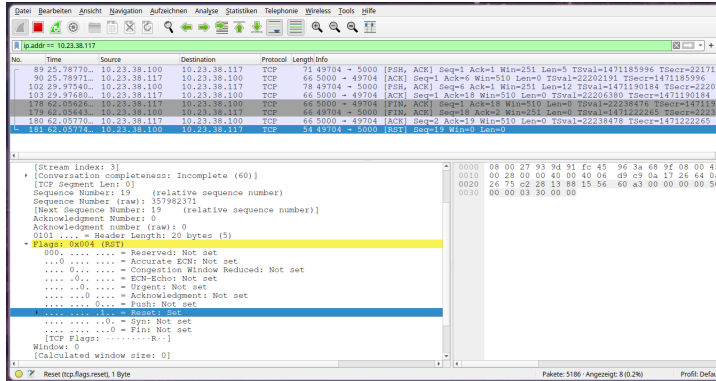


Figure 6: Verbindungsabbruch

Die TCP-Flag Reset wird im Packet gesetzt und dann ist die Verbindung mit diesem Packet beendet.

3.1.7 Verbinden mit dem Sever (UDP)

```
nc -u 10.23.38.117 4201
```

3.1.8 UDP-Verbindungs Aufbau

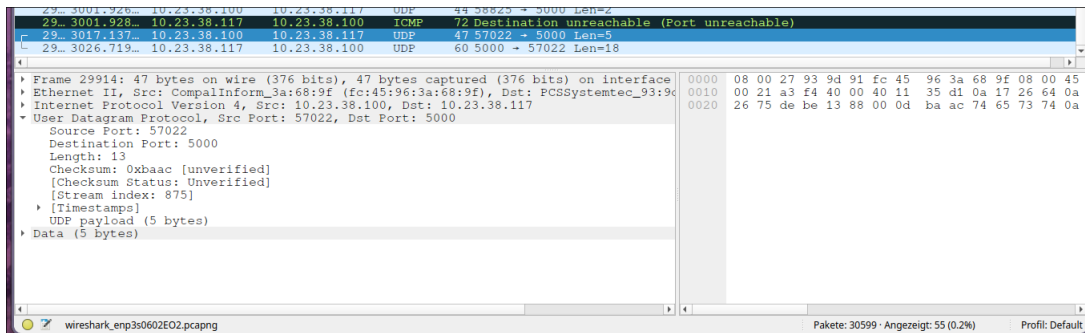


Figure 7: UDP-Verbindungs Aufbau

Im gegensatz zu TCP gibt es hier keinen 3-Way-Handsake und die Verbindung beginnt direkt.

3.1.9 UDP-Nachrichten

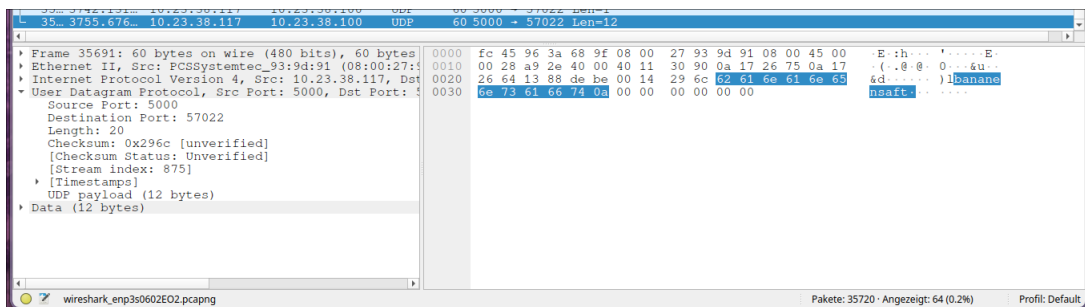


Figure 8: UDP-Nachricht

3.1.10 UDP-Header

- Source Port
- Destination Port
- Länge
- Checksum

Diese Felder haben die selben bedeutungen wie bei TCP. [2]

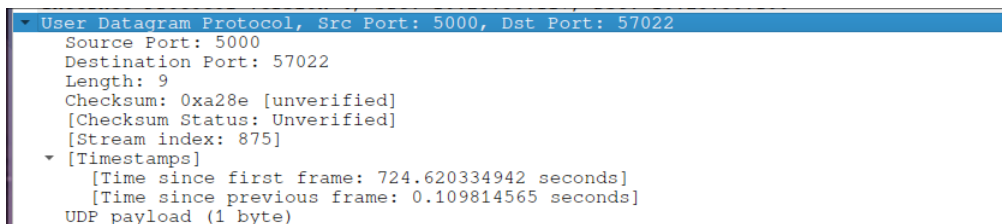


Figure 9: UDP-Header

3.1.11 UDP-Verbindungsabbruch

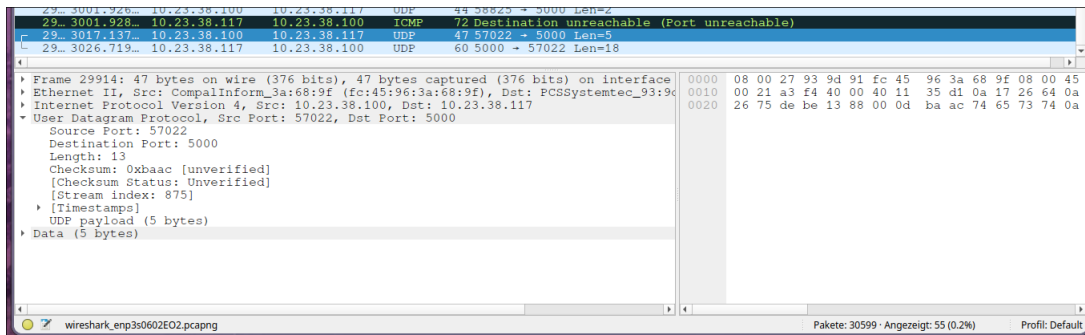


Figure 10: UDP-Verbindungsabbruch

Die Verbindung endet einfach und es wird kein Packet zum Beenden der Verbindung geschickt.

3.2 TCP und UDP Headervergleich

Im Gegensatz zu TCP, UDP hat weniger Felder, da extra Felder wie Window, Sequence und Acknowledgment Number einfach nicht benötigt, da der Sinn von UDP es ist, weniger Overhead zu haben, damit eine schnellere Übertragung möglich ist.

Der TCP-Header hat eine Minimale Größe von 20 Bytes, während der UDP-Header immer 8 Bytes groß ist. [4][1] UDP hat nur die essenziellsten Felder, die benötigt werden, um eine Kommunikation durchzuführen.

Wichtigste Felder:

- Src/Destination Port
 - Ohne dieses Feld kann Kommunikation nicht funktionieren.
- Checksum
 - Wichtig, um sicherzugehen, dass das Paket nicht korrupt ist.

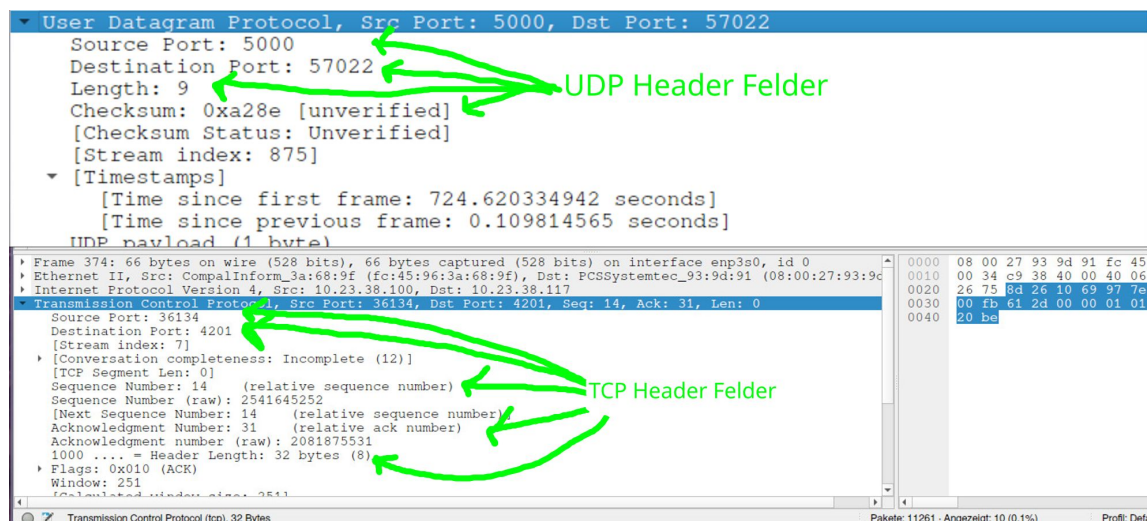


Figure 11: Header Vergleich

3.3 Nach Ports filtern

Nur Nach Port filtern:

Protokoll.port == port

Logische Operatoren können für komplexere Filter angewandt werden[5]

Listing 2: Commands

```
# tcp port 80 oder udp port 80
tcp.port == 80 || udp.port == 80

# auf port 80 nach verlorenen tcp segmenten filtern
tcp.port == 80 && tcp.analysis.lost_segment
```

4 Quellen

References

- [1] User Datagram Protocol (UDP) in Computer Network, December 2021. [Online; accessed 7. Jun. 2024].
- [2] Autoren der Wikimedia-Projekte. User Datagram Protocol – Wikipedia, January 2003. [Online; accessed 2. Jun. 2024].
- [3] GeeksforGeeks. TCP flags. *GeeksforGeeks*, April 2023.
- [4] Rene Molenaar. TCP Header. *NetworkLessons*, October 2019.
- [5] Lee Stanton. How to Filter by Port with Wireshark. *Alphr*, June 2021.

5 **Abbildungsverzeichnis**

List of Figures

1	memes klauen ist nicht ethisch	1
2	TCP 3 Way Handshake	4
3	Nachricht	4
4	TCP-Flags	5
5	TCP-Header	6
6	Verbindungsabbruch	7
7	UDP-Verbindungsaufbau	7
8	UDP-Nachricht	7
9	UDP-Header	8
10	UDP-Verbindungsabbruch	8
11	Header Vergleich	9