

שאלה 2:

סעיף א:

בסעיף זה הלקוח מייצר מחרוזת שמכלה 15,000 פעם את התו 'A' ושולח את המחרוזת הנ"ל לשרת, השרת קורא את ההודעה ומשיב בהודעה שמכילה את התו 'B'.

No.	Time	Source	Destination	Protocol	Length	Info
13	6.951014540	192.168.1.124	192.168.1.125	TCP	74	50784 → 12345 [SYN, Seq=245231766 Win=29200 Len=0 MSS=1460 SACK_PERM=1]
14	6.951043211	192.168.1.125	192.168.1.124	TCP	74	12345 → 50784 [SYN, ACK] Seq=673345865 Ack=245231767 Win=28960 Len=0
15	6.952331442	192.168.1.124	192.168.1.125	TCP	66	50784 → 12345 [ACK] Seq=245231767 Ack=673345866 Win=29312 Len=0 TSval=...
16	6.988303572	192.168.1.124	192.168.1.125	TCP	7306	50784 → 12345 [ACK] Seq=245231767 Ack=673345866 Win=29312 Len=7240 TSval=...
17	6.988391609	192.168.1.125	192.168.1.124	TCP	66	12345 → 50784 [ACK] Seq=673345866 Ack=245239007 Win=43520 Len=0 TSval=...
18	6.988901486	192.168.1.124	192.168.1.125	TCP	7306	50784 → 12345 [ACK] Seq=245239007 Ack=673345866 Win=29312 Len=7240 TSval=...
19	6.988916287	192.168.1.125	192.168.1.124	TCP	66	12345 → 50784 [ACK] Seq=673345866 Ack=245246247 Win=57984 Len=0 TSval=...
20	6.989940893	192.168.1.124	192.168.1.125	TCP	586	50784 → 12345 [PSH, ACK] Seq=245246247 Ack=673345866 Win=29312 Len=5
21	6.989953549	192.168.1.125	192.168.1.124	TCP	66	12345 → 50784 [ACK] Seq=673345866 Ack=245246767 Win=60928 Len=0 TSval=...
22	6.990305771	192.168.1.125	192.168.1.124	TCP	67	12345 → 50784 [PSH, ACK] Seq=673345866 Ack=245246767 Win=60928 Len=1
23	6.990761349	192.168.1.125	192.168.1.124	TCP	66	12345 → 50784 [FIN, ACK] Seq=673345867 Ack=245246767 Win=60928 Len=0 TSval=...
24	6.996006450	192.168.1.124	192.168.1.125	TCP	66	50784 → 12345 [ACK] Seq=245246767 Ack=673345867 Win=29312 Len=0 TSval=...
25	6.996023589	192.168.1.124	192.168.1.125	TCP	66	50784 → 12345 [FIN, ACK] Seq=245246767 Ack=673345868 Win=29312 Len=0 TSval=...
26	6.996034941	192.168.1.125	192.168.1.124	TCP	66	12345 → 50784 [ACK] Seq=673345868 Ack=245246768 Win=60928 Len=0 TSval=...

ניתן לראות כי כל התהליך מתחיל בהandshake (חבילות 13-15). לאחר מכן מועבר הנתונים (חבילות 16-22) בהרצת זאת על כל חבילה ששולח הלקוח השרת עונה לו מיד באck, אך אין זה קורה בכל הרצת. לבסוף מתרחש הteardown (חבילות 23-26). אפרט על כל חבילה וחבילה בהמשך.

שרת: כתובת IP : 192.168.1.125 , port : 12345

לקוח: כתובת IP : 192.168.1.124 , port : 50784

חבילה מספר 13:

Wireshark - Packet 13 - 2.1.pcapng

Frame 13: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0

Ethernet II, Src: PcsCompu_16:fd:63 (08:00:27:16:fd:63), Dst: PcsCompu_5a:b7:89 (08:00:27:5a:b7:89)

Internet Protocol Version 4, Src: 192.168.1.124, Dst: 192.168.1.125

Transmission Control Protocol, Src Port: 50784, Dst Port: 12345, Seq: 245231766, Len: 0

Source Port: 50784

Destination Port: 12345

[Stream index: 0]

[TCP Segment Len: 0]

Sequence number: 245231766

[Next sequence number: 245231766]

Acknowledgment number: 0

1010 = Header Length: 40 bytes (10)

Flags: 0x002 (SYN)

Window size value: 29200

[Calculated window size: 29200]

Checksum: 0xe8a3 [unverified]

[Checksum Status: Unverified]

Urgent pointer: 0

Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale

[Timestamps]

Sequence number : 245231766

ack number : 0

source port : 50784

destination port: 12345

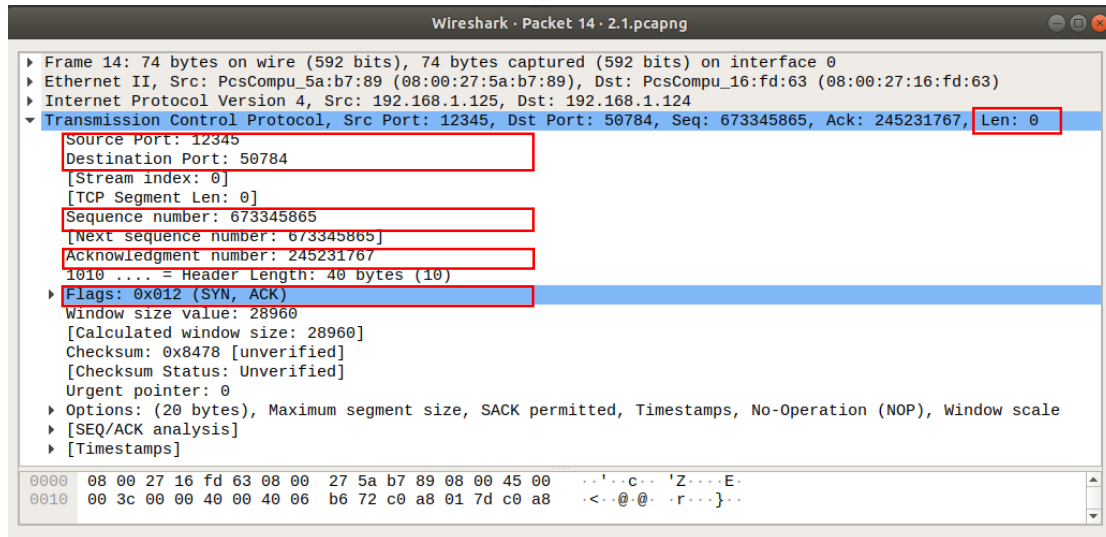
flags: SYN

length: 0

data :NO DATA

בחלק זה port מספר 50784(לקוח) שולח חבילה ל port מספר 12345 (שרת), ניתן לראות שחבילה זאת ללא data ולכן אורכה היא 0. ניתן לראות כי דגל הsyn דלוק ואילו דגל האck כבוי, header (דבר שמייצג את תחילת ההתקשרות בhandshake כפי שצינתי בשאלה הקודמת), לחבילה מצורף Sequence number ההתחלתי של הלקוח.

חבילה מספר 14:



Sequence number : 673345865

ack number : 245231767 (prev Seq.num + 1)

source port :12345

destination port: 50784

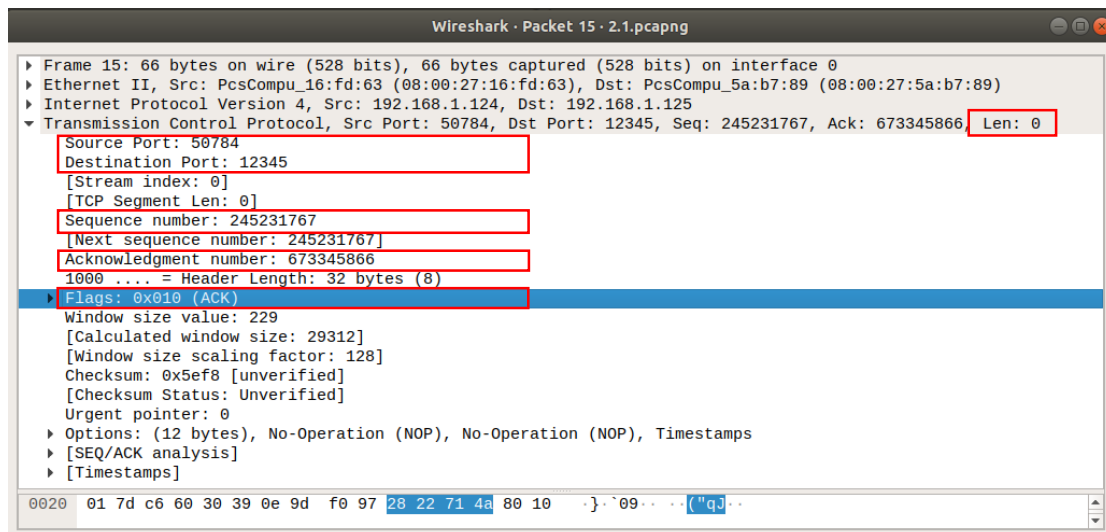
flags: SYN,ACK

length: 0

data :NO DATA

בחלק זה port מספר 12345(שרת) שולח חבילה ל port מספר 50784 (לקוח), ניתן לראות שחבילה זאת ללא data ולכן אורכה היא 0. ניתן לראות כי דגל הsyn ודגל האck דלוקים בheader, בכך השרת מאשר ללקוח כי קיבל את רצונו להתקשרות. לחבילה מצורף Sequence number התחלתי של השרת וה-ack number הינו Sequence number של הלקוח + 1.

חבילה מספר 15:



Sequence number : 245231767

ack number : 673345866 (prev Seq.num + 1)

source port :50784

destination port: 12345

flags: ACK

length: 0

data :NO DATA

בחלק זה port מספר 50784 (לקוח) שולח חבילה ל port מספר 12345 (שרת), ניתן לראות שחבילה זאת ללא data ולכן אורכה היא 0. ניתן לראות כי דגל האק דלוק ב header ובכך מסתיים תהליך הhandshaken. לחבילה מצורף ack number שהוא ה-Sequence number של השרת +1. ה-Sequence number הוא ה ack number ששלח הלקוח בחבילה הקודמת.

The screenshot shows a Wireshark interface with a single packet selected. The packet details pane displays the following information:

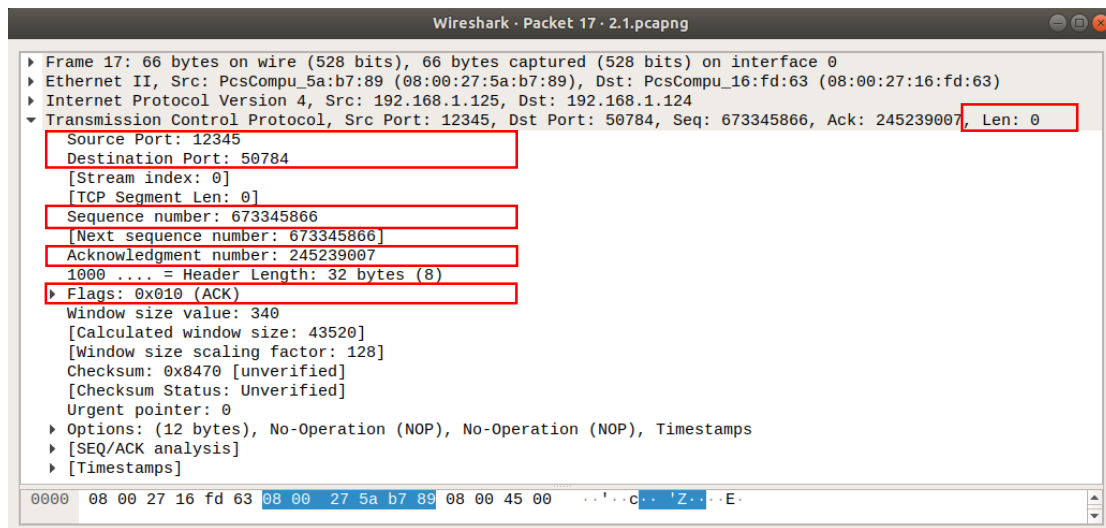
- Ethernet II**: Src: PcsCompu_16:fd:63 (08:00:27:16:fd:63), Dst: PcsCompu_5a:b7:89 (08:00:27:5a:b7:89)
- Internet Protocol Version 4**: Src: 192.168.1.124, Dst: 192.168.1.125
- Transmission Control Protocol**: Src Port: 50784, Dst Port: 12345, Seq: 245231767, Ack: 673345866, Len: 7240
 - Source Port: 50784
 - Destination Port: 12345
 - [Stream index: 0]
 - [TCP Segment Len: 7240]
 - Sequence number: 245231767
 - [Next sequence number: 245239007]
 - Acknowledgment number: 673345866
 - 1000 = Header Length: 32 bytes (8)
 - Flags: 0x010 (ACK)
 - Window size value: 229
 - [Calculated window size: 29312]
 - [Window size scaling factor: 128]
 - Checksum: 0xa0b8 [unverified]
 - [Checksum Status: Unverified]
 - Urgent pointer: 0
 - Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
 - [SEQ/ACK analysis]
 - [Timestamps]
 - TCP payload (7240 bytes)
- Data (7240 bytes)**: Data: 41... [Length: 7240]

The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII. The first few bytes are 0040 6a 42, followed by several 41 41 pairs, and then a series of AAAAAA pairs.

```
data : 'AAA...'
```

בחלק זה port מספר 50784(לקוח) שולח חבילה ל port מספר 12345 (שרת), ניתן לראות שחבילה זאת מכילה data בגודל 7240. ניתן לראות כי דגל ack דלוק ב header. ה ack number וה-Sequence number נשארים כפי שהיו בחבילה הקודמת ששלח הלקוח.

חבילה מספר 17:



Sequence number : 673345866 (prev ack.num)

ack number : 245239007 (prev seq.num + 7240 (data len))

source port :12345

destination port: 50784

flags: ACK

length: 0

data :NO DATA

בחלק זה מספר 12345 (שרת) שולח חבילה ל port מספר 50784 (לקוח), ניתן לראות שחבילה זאת ללא data ולכן הlength הוא 0. ניתן לראות כי דגל הack דלוק בheader, בכך השרת מאשר ללקוח כי קיבל את ה data ששלח. לחבילה מצורף Sequence number שזהה ל ack number המחבילה הקודמת, והack number זהה ל Sequence number + גודל data (7240) (B) שהלקוח שלח בחבילה הקודמת.

The screenshot shows the Wireshark interface with a single packet selected. The packet list pane on the left shows three packets, with the first one highlighted. The packet details pane on the right shows the structure of the selected packet, which is a Transmission Control Protocol (TCP) segment. The packet length is 7240 bytes.

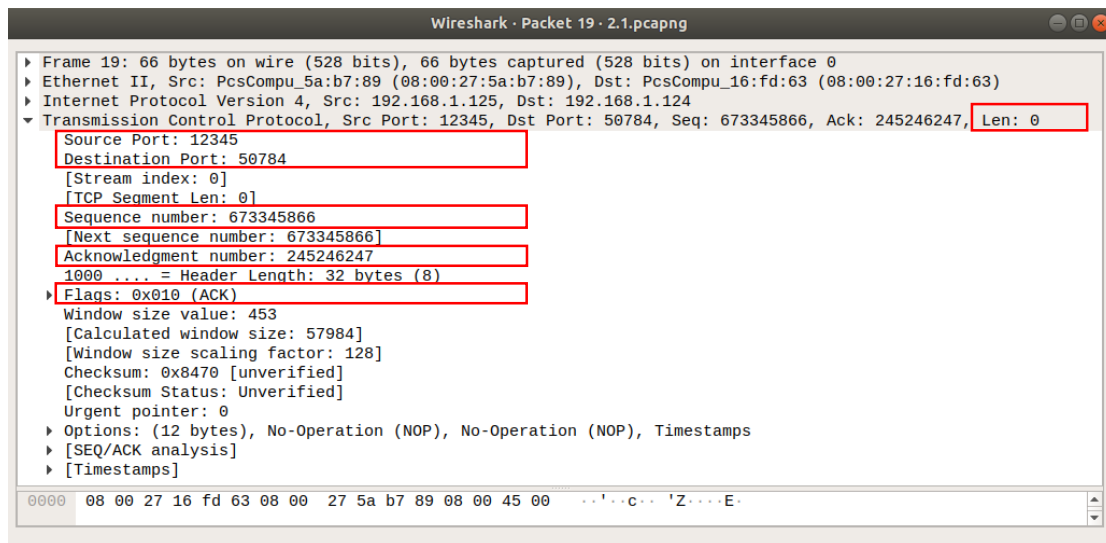
- Ethernet II:** Src: PcsCompu_16:fd:63 (08:00:27:16:fd:63), Dst: PcsCompu_5a:b7:89 (08:00:27:5a:b7:89)
- Internet Protocol Version 4:** Src: 192.168.1.124, Dst: 192.168.1.125
- Transmission Control Protocol:** Src Port: 50784, Dst Port: 12345, Seq: 245239007, Ack: 673345866, Len: 7240
 - Source Port: 50784
 - Destination Port: 12345
 - [Stream index: 0]
 - [TCP Segment Len: 7240]
 - Sequence number: 245239007
 - [Next sequence number: 245246247]
 - Acknowledgment number: 673345866
 - 1000 = Header Length: 32 bytes (8)
 - Flags: 0x010 (ACK)
 - Window size value: 229
 - [Calculated window size: 29312]
 - [Window size scaling factor: 128]
 - Checksum: 0xa0b8 [unverified]
 - [Checksum Status: Unverified]
 - Urgent pointer: 0
 - Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
 - [SEQ/ACK analysis]
 - [Timestamps]
 - TCP payload (7240 bytes)
- Data (7240 bytes):** 41... [Length: 7240]

The packet bytes pane at the bottom shows the raw data of the packet. It starts with the Ethernet II header (0040), followed by the Internet Protocol Version 4 header (6a 42), and then the Transmission Control Protocol header (41 41 41 41 41 41 41 41 41 41 41 41 41 41). The rest of the packet is the TCP payload, which consists of a series of 'A' characters (AAAAAA AAAAAA).

```
data : 'AAA...'
```

בחלק זה port מספר 50784(לקוח) שולח חבילה ל port מספר 12345 (שרת), ניתן לראות שחבילה זאת מכילה data בגודל 7240. ניתן לראות כי דגל ack דלוק ב header. ה ack number-וה Sequence number מתחלפים מכפי שהיו בחבילה הקודמת ששלח השרת.

חבילה מספר 19:



Sequence number : 673345866 (prev ack.num)

ack number : 245246247 (prev seq.num + 7240 (data len))

source port :12345

destination port: 50784

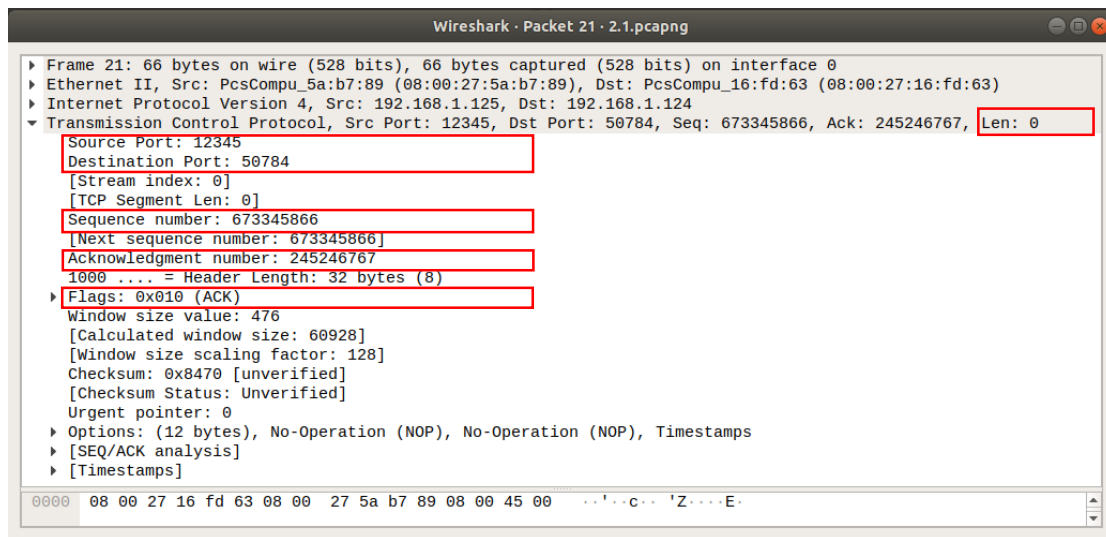
flags: ACK

length: 0

data :NO DATA

בחלק זה port מספר 12345 (שרת) שולח חבילה ל port מספר 50784 (לקוח), ניתן לראות שחבילה זאת ללא data ולכן הlength הוא 0. ניתן לראות כי דגל הack דלוק בheader, בכך השרת מאשר ללקוח כי קיבל את ה data ששלח. לחבילה מצורף Sequence number שזהה ל ack number המחבילה הקודמת, והack number זהה ל Sequence number + גודל ה data (7240) (שהלקוח שלח בחבילה הקודמת).

חבילה מספר 21:



Sequence number : 673345866 (prev ack.num)

ack number : 245246767 (prev seq.num + 520 (data len))

source port :12345

destination port: 50784

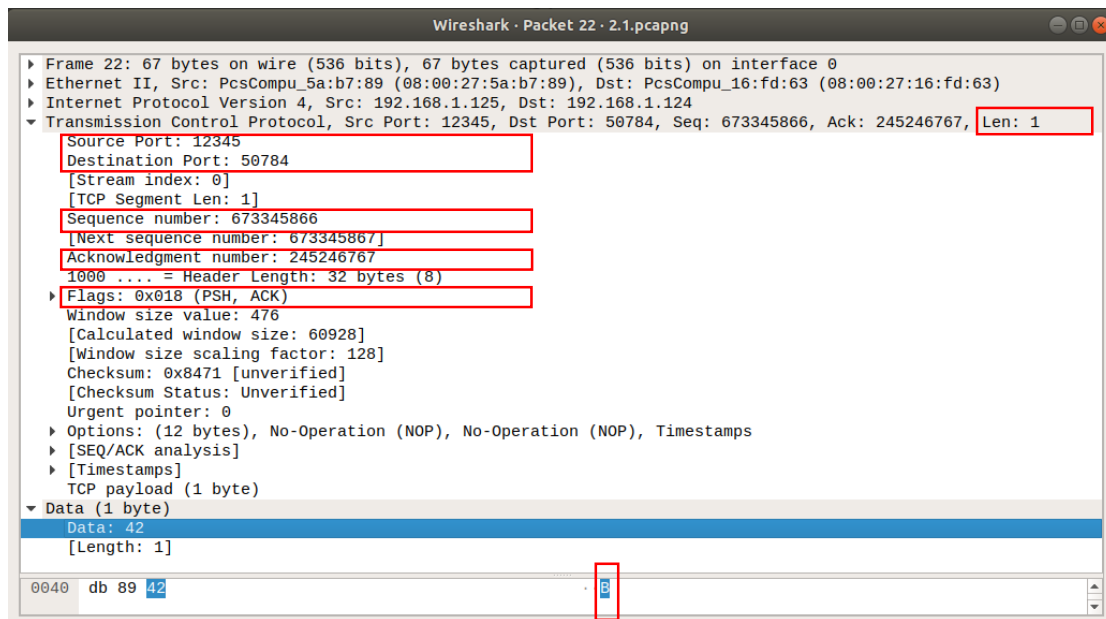
flags: ACK

length: 0

data :NO DATA

בחלק זה port מספר 12345 (שרת) שולח חבילה ל port מספר 50784 (לקוח), ניתן לראות שחבילה זאת ללא data ולכן הlength הוא 0. ניתן לראות כי דגל הack דלוק בheader, בכך השרת מאשר ללקוח כי קיבל את ה data ששלח. לחבילה מצורף Sequence number שזהה ל ack number המחבילה הקודמת, והack number זהה ל Sequence number + גודל ה data (520 B) (שהלקוח שלח בחבילה הקודמת).

חבילה מספר 22:



Sequence number : 673345866 (prev seq.num)

ack number : 245246767 (prev ack.num)

source port :12345

destination port: 50784

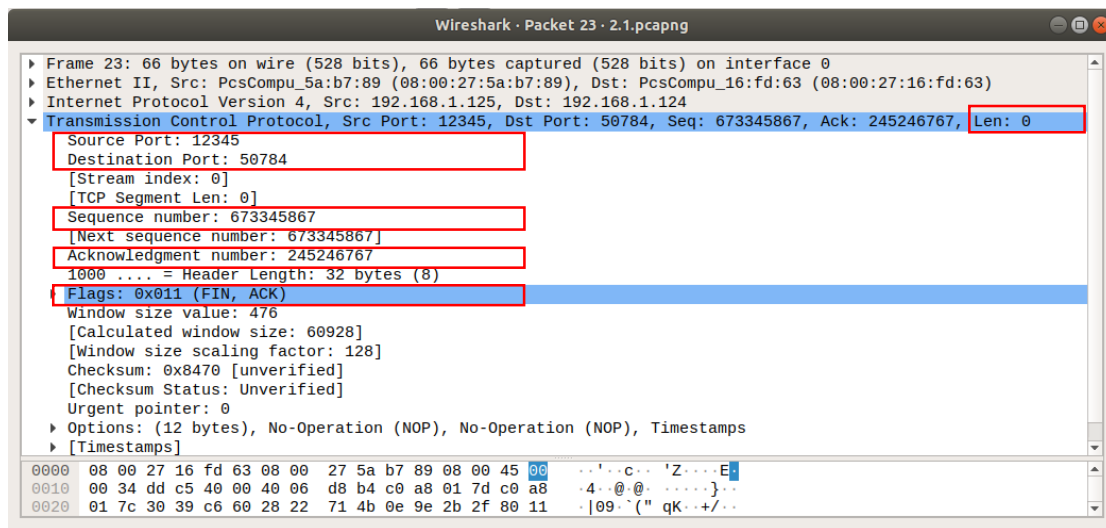
flags: PSH,ACK

length: 1

data :B

בחלק זה port מספר 12345(שרת) שולח חבילה ל port מספר 50784 (לקוח), ניתן לראות שחבילה זאת מכינה data ולכן length הוא 1. ניתן לראות כי דגלי psh והack דלוקים ב header, דגל psh מסמן שהשרת סיים לשלוח את כל ה data, ולא צפויות להגיע עוד חבילות. ה Sequence number וה-ack number נשארים כפי שהיו בחבילה הקודמת ששלח השרת.

חבילה מספר 23:



Sequence number : 245246767 (prev ack.num)

ack number : 673345867 (prev seq.num + 1)

source port :12345

destination port: 50784

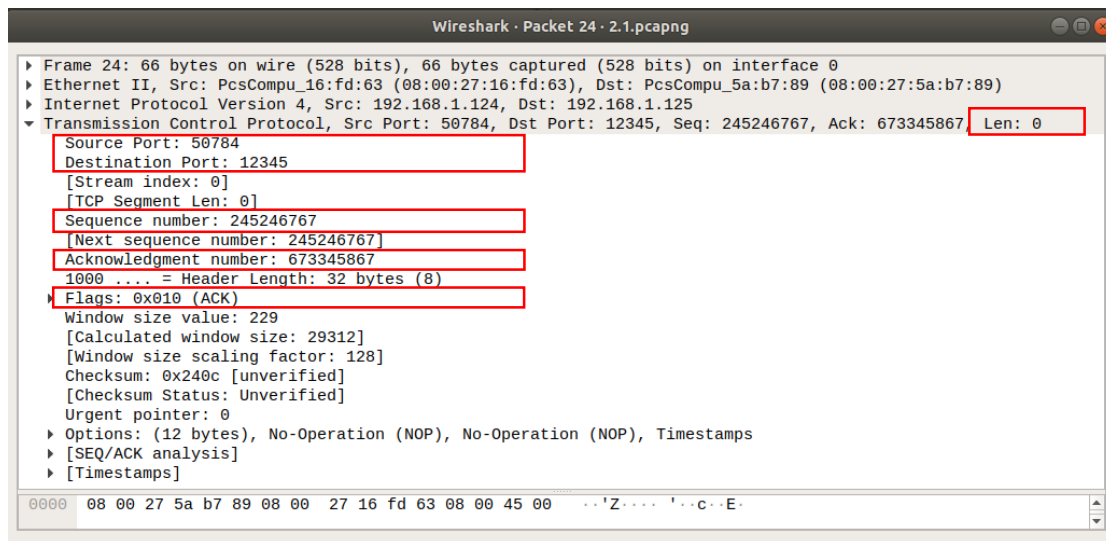
flags: FIN,ACK

length: 0

data : NO DATA

בחלק זה מספר 12345 (שרת) שולח חבילה ל port מספר 50784 (לקוח), ניתן לראות שחבילה זאת ללא data ולכן הlength הוא 0. ניתן לראות כי דגלי הfin והack דלוקים בheader, בכך השרת מסמן ללקוח כי ברצונו לסיים את ההתקשרות (תחילת תהליך ה teardown). לחבילה מצורף Sequence number שזהה לack number המחבילה הקודמת, והack number זהה ל Sequence number + 1 מהחבילה הקודמת ששלח השרת.

חבילה מספר 24:



Sequence number : 245246767 (prev ack.num)

ack number : 673345867 (prev seq.num + 1(data len))

source port :50784

destination port: 12345

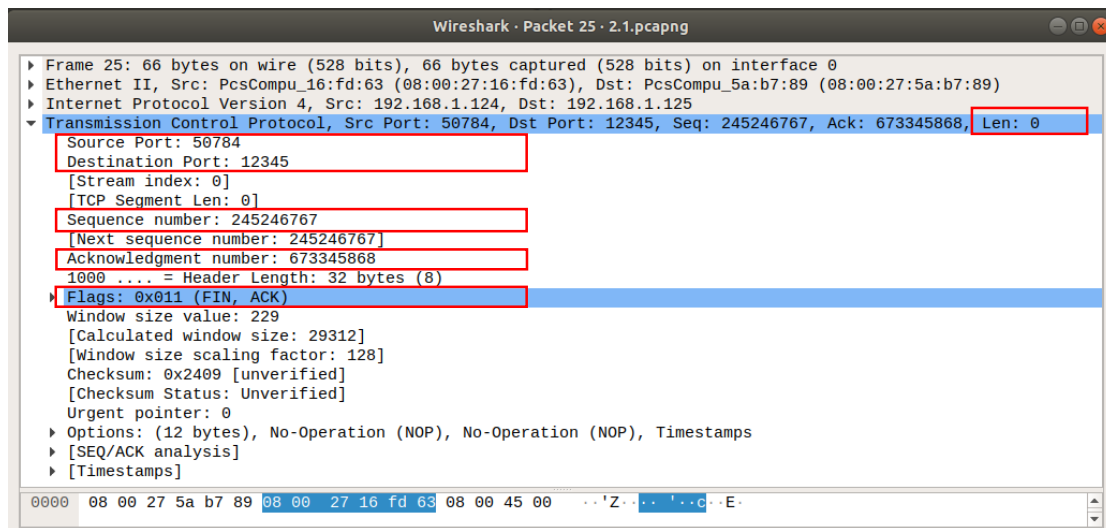
flags: ACK

length: 0

data : NO DATA

בחלק זה port מספר 50784 (לקוח) שולח חבילה ל port מספר 12345 (שרת), ניתן לראות שחבילה זאת ללא data ולכן הlength הוא 0. ניתן לראות כי דגל הack דלוק בheader, בכך הלקוח מסמן לשרת כי קיבלת את ה data ששלח קודם. לחבילה מצורף Sequence number שזהה ל ack number המחבילה הקודמת, והack number זהה ל Sequence number + גודל ה data (1) (שהשרת שלח בחבילה הקודמת).

חבילה מספר 25:



Sequence number : 245246767 (prev ack.num)

ack number : 673345868 (prev seq.num + 1)

source port :50784

destination port: 12345

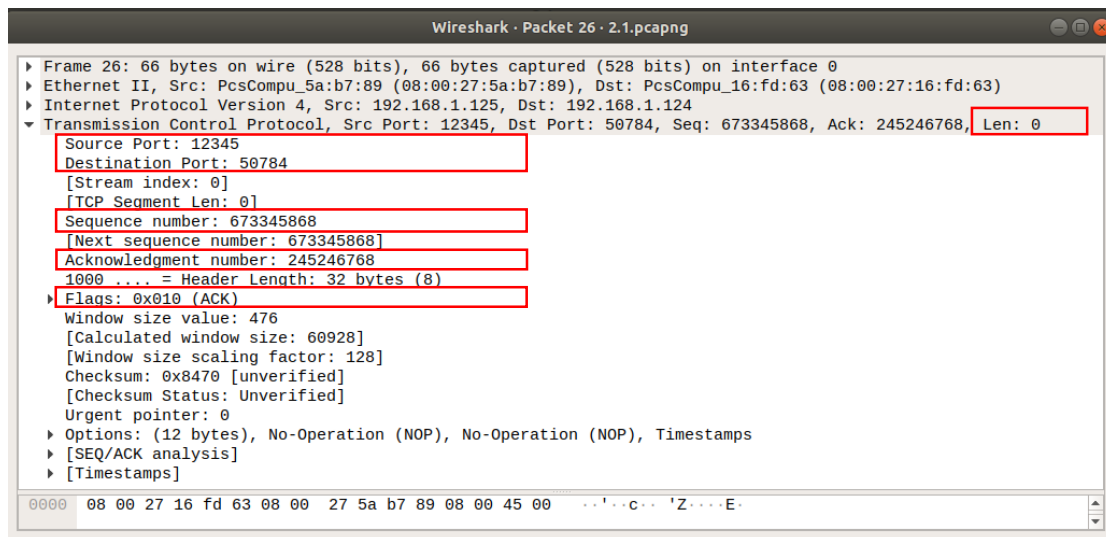
flags: FIN,ACK

length: 0

data : NO DATA

בחלק זה port מספר 50784 (לקוח) שולח חבילה ל port מספר 12345 (שרת), ניתן לראות שחבילה זאת ללא data ולכן הlength הוא 0. ניתן לראות כי דגלי הfin והack דלוקים בheader, בכך הלקוח מסמן לשרת כי קיבלת את הודעתו שברצונו לסיים את ההתקשרות(תהליך הteardown). לחבילה מצורף Sequence number שזהה לack number המחבילה הקודמת, והack number זהה ל Sequence number + 1 מהחבילה הקודמת ששלח השרת.

חבילה מספר 26:



Sequence number : 673345868 (prev ack.num)

ack number : 245246768 (prev seq.num + 1)

source port :12345

destination port: 50784

flags: FIN,ACK

length: 0

data : NO DATA

בחלק זה port מספר 12345 (שרת) שולח חבילה ל port מספר 50784 (לקוח), ניתן לראות שחבילה זאת ללא data ולכן הlength הוא 0. ניתן לראות כי דגל הack דלוק בheader, בכך השרת מסיים את ההתקשרות עם הלקוח (סוף תהליך ההתנתקות). Sequence number מצורף שזהה לack number המחבילה הקודמת, והack number זהה ל Sequence number + 1 מהחבילה הקודמת ששלח השרת.

סעיף ב:

בסעיף זה הלקוח שולח 2 הודעות 'A' רצופות, ממתין 2 שניות ולאחר מכן חוזר 10 פעמים נוספות על משימת השליחה הכפולה.

השאלה שמועלית בסעיף: " האם כל send היה הודעה נפרדת? האם התוצאה הייתה זהה בכל השליחות? :

תשובה: לא, לאחר הרצת מספר לקוחות, ניתן בבירור לראות כי מספר החבילות שבהן מועבר datan שונה מלקוח ללקוח. כפי שלמדנו בפרוטוקול TCP החבילות מועברות כstreams של בתים, ולכן כאשר מספר ההודעות שנסלחות בצמידות, יגיעו לצד המקבל במספר שונה של חבילות ואף בחבילה אחת. אתן 2 דוגמאות למקרים בהן מספר החבילות שבהן עבר ה data מהלקוח לשרת, אסמן ב**אדום** את החבילות שממן שלח הלקוח לשרת לראשונה חבילה עם data ועד ה ack האחרון ששלח השרת ללקוח.

לקוח מספר 1 :

No.	Time	Source	Destination	Protocol	Length	Info
6	3.41	192.168.1.124	192.168.1.125	TCP	74	52540 → 12345 [SYN, Seq=1686704785 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2...
7	3.41	192.168.1.125	192.168.1.124	TCP	74	12345 → 52540 [SYN, ACK] Seq=436500050 Ack=1686704786 Win=28960 Len=0 MSS=1460
8	3.41	192.168.1.124	192.168.1.125	TCP	66	52540 → 12345 [ACK] Seq=1686704786 Ack=436500051 Win=29312 Len=0 TSval=25560077...
9	3.41	192.168.1.124	192.168.1.125	TCP	67	52540 → 12345 [PSH, ACK] Seq=1686704786 Ack=436500051 Win=29312 Len=1 TSval=255...
10	3.41	192.168.1.125	192.168.1.124	TCP	66	12345 → 52540 [ACK] Seq=436500051 Ack=1686704787 Win=29056 Len=0 TSval=64623368...
11	3.42	192.168.1.124	192.168.1.125	TCP	67	52540 → 12345 [PSH, ACK] Seq=1686704787 Ack=436500051 Win=29312 Len=1 TSval=255...
12	3.42	192.168.1.125	192.168.1.124	TCP	66	12345 → 52540 [ACK] Seq=436500051 Ack=1686704788 Win=29056 Len=0 TSval=64623368...
13	3.42	192.168.1.124	192.168.1.125	TCP	67	52540 → 12345 [PSH, ACK] Seq=1686704788 Ack=436500051 Win=29312 Len=1 TSval=255...
14	3.42	192.168.1.125	192.168.1.124	TCP	66	12345 → 52540 [ACK] Seq=436500051 Ack=1686704789 Win=29056 Len=0 TSval=64623369...
15	3.43	192.168.1.124	192.168.1.125	TCP	67	52540 → 12345 [PSH, ACK] Seq=1686704789 Ack=436500051 Win=29312 Len=1 TSval=255...
16	3.43	192.168.1.125	192.168.1.124	TCP	66	12345 → 52540 [ACK] Seq=436500051 Ack=1686704790 Win=29056 Len=0 TSval=64623370...
17	3.43	192.168.1.124	192.168.1.125	TCP	74	52540 → 12345 [PSH, ACK] Seq=1686704790 Ack=436500051 Win=29312 Len=8 TSval=255...
18	3.43	192.168.1.125	192.168.1.124	TCP	66	12345 → 52540 [ACK] Seq=436500051 Ack=1686704798 Win=29056 Len=0 TSval=64623370...
19	3.43	192.168.1.124	192.168.1.125	TCP	71	52540 → 12345 [PSH, ACK] Seq=1686704798 Ack=436500051 Win=29312 Len=5 TSval=255...
20	3.43	192.168.1.125	192.168.1.124	TCP	66	12345 → 52540 [ACK] Seq=436500051 Ack=1686704803 Win=29056 Len=0 TSval=64623370...
21	3.44	192.168.1.124	192.168.1.125	TCP	67	52540 → 12345 [PSH, ACK] Seq=1686704803 Ack=436500051 Win=29312 Len=1 TSval=255...
22	3.44	192.168.1.125	192.168.1.124	TCP	66	12345 → 52540 [ACK] Seq=436500051 Ack=1686704804 Win=29056 Len=0 TSval=64623371...
23	3.45	192.168.1.124	192.168.1.125	TCP	67	52540 → 12345 [PSH, ACK] Seq=1686704804 Ack=436500051 Win=29312 Len=1 TSval=255...
24	3.45	192.168.1.125	192.168.1.124	TCP	66	12345 → 52540 [ACK] Seq=436500051 Ack=1686704805 Win=29056 Len=0 TSval=64623372...
25	3.45	192.168.1.124	192.168.1.125	TCP	69	52540 → 12345 [PSH, ACK] Seq=1686704805 Ack=436500051 Win=29312 Len=3 TSval=255...
26	3.45	192.168.1.125	192.168.1.124	TCP	66	12345 → 52540 [ACK] Seq=436500051 Ack=1686704808 Win=29056 Len=0 TSval=64623372...
27	3.45	192.168.1.125	192.168.1.124	TCP	67	12345 → 52540 [PSH, ACK] Seq=436500051 Ack=1686704808 Win=29056 Len=1 TSval=646...
28	3.45	192.168.1.125	192.168.1.124	TCP	66	12345 → 52540 [FIN, ACK] Seq=436500052 Ack=1686704808 Win=29056 Len=0 TSval=646...
29	3.45	192.168.1.124	192.168.1.125	TCP	66	52540 → 12345 [ACK] Seq=1686704808 Ack=436500052 Win=29312 Len=0 TSval=25560077...
30	3.46	192.168.1.124	192.168.1.125	TCP	66	52540 → 12345 [FIN, ACK] Seq=1686704808 Ack=436500053 Win=29312 Len=0 TSval=255...
31	3.46	192.168.1.125	192.168.1.124	TCP	66	12345 → 52540 [ACK] Seq=436500053 Ack=1686704809 Win=29056 Len=0 TSval=64623373...

בדוגמא זאת הלקוח נאלץ לשלוח 9 חבילות עם data על מנת להעביר את 21 הודעות ה'A'.

לקוח מספר 2:

47	13.5	192.168.1.124	192.168.1.125	TCP	74	52634 → 12345 [SYN, Seq=1243725394 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2...
48	13.5	192.168.1.125	192.168.1.124	TCP	74	12345 → 52634 [SYN, ACK] Seq=2281506758 Ack=1243725395 Win=28960 Len=0 MSS=1460
49	13.5	192.168.1.124	192.168.1.125	TCP	66	52634 → 12345 [ACK] Seq=1243725395 Ack=2281506759 Win=29312 Len=0 TSval=25560077...
50	13.5	192.168.1.124	192.168.1.125	TCP	67	52634 → 12345 [PSH, ACK] Seq=1243725395 Ack=2281506759 Win=29312 Len=1 TSval=25...
51	13.5	192.168.1.125	192.168.1.124	TCP	66	12345 → 52634 [ACK] Seq=2281506759 Ack=1243725396 Win=29056 Len=0 TSval=6462437...
52	13.5	192.168.1.124	192.168.1.125	TCP	87	52634 → 12345 [PSH, ACK] Seq=1243725396 Ack=2281506759 Win=29312 Len=21 TSval=2...
53	13.5	192.168.1.125	192.168.1.124	TCP	66	12345 → 52634 [ACK] Seq=2281506759 Ack=1243725417 Win=29056 Len=0 TSval=6462437...
54	13.5	192.168.1.125	192.168.1.124	TCP	67	12345 → 52634 [PSH, ACK] Seq=2281506759 Ack=1243725417 Win=29056 Len=1 TSval=64...
55	13.5	192.168.1.125	192.168.1.124	TCP	66	12345 → 52634 [FIN, ACK] Seq=2281506760 Ack=1243725417 Win=29056 Len=0 TSval=64...
56	13.5	192.168.1.124	192.168.1.125	TCP	66	52634 → 12345 [ACK] Seq=1243725417 Ack=2281506760 Win=29312 Len=0 TSval=2556017...
57	13.5	192.168.1.124	192.168.1.125	TCP	66	52634 → 12345 [FIN, ACK] Seq=1243725417 Ack=2281506761 Win=29312 Len=0 TSval=25...
58	13.5	192.168.1.125	192.168.1.124	TCP	66	12345 → 52634 [ACK] Seq=2281506761 Ack=1243725418 Win=29056 Len=0 TSval=6462438...

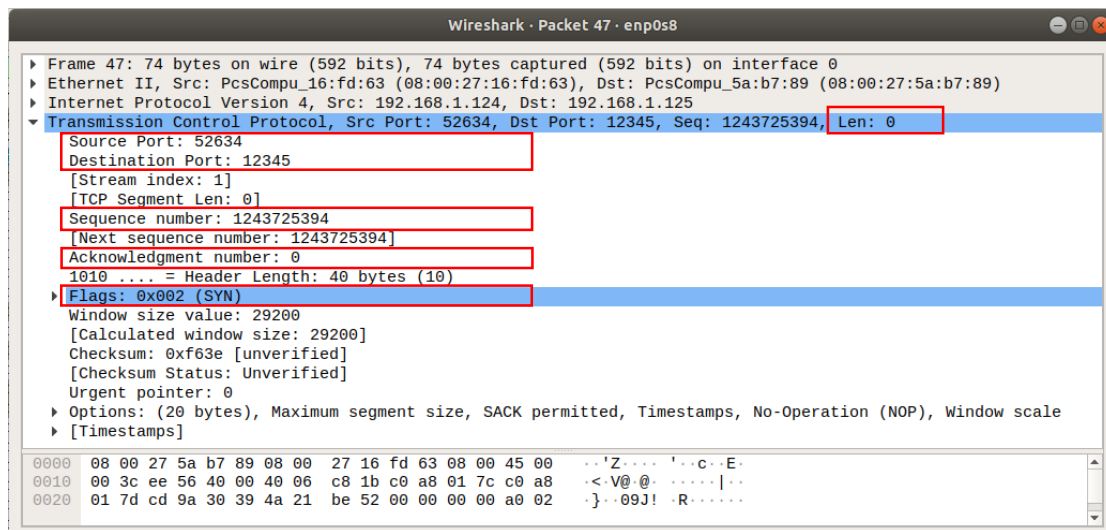
לעומת זאת בלקוח זה, נשלחו רק 2 חבילות עם data, על מנת להעביר את אותו תוכן כמו בלקוח מספר 1.

כעת אנתח את לקוח מספר 2. ניתן לראות כי כל התהליך מתחיל בהandshake (חבילות 47-49). לאחר מכן מועבר datan (חבילות 50-54, ללא הack של הלקוח). לבסוף מתרחש תהליך ה teardown (חבילות 55-58). אפרט על כל חבילה וחבילה בהמשך.

שרת: כתובת IP : 192.168.1.125 , port : 12345

לקוח: כתובת IP : 192.168.1.124 , port : 52634

חבילה מספר 47:



Sequence number : 1243725394

ack number : 0

source port :52634

destination port: 12345

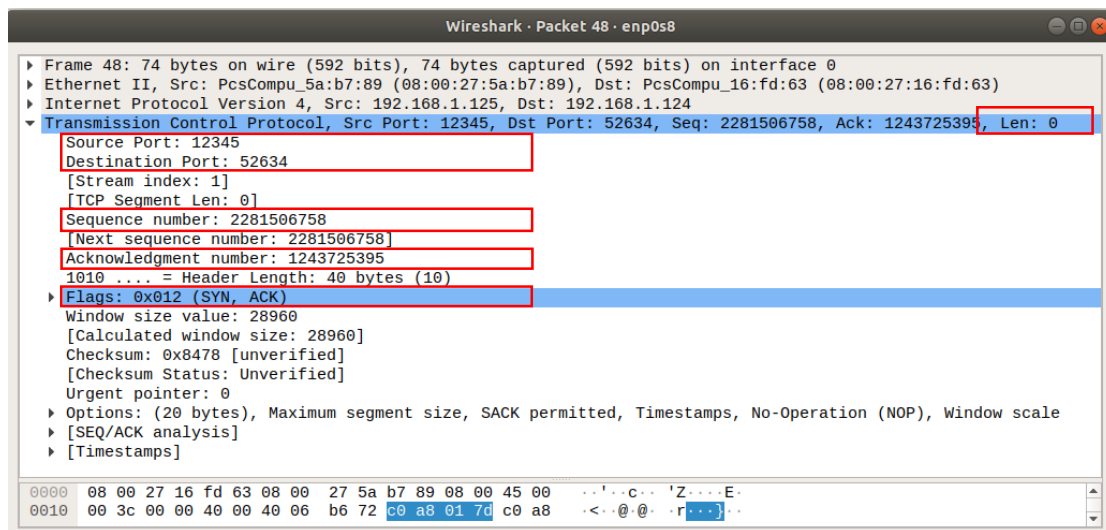
flags: SYN

length: 0

data :NO DATA

בחלק זה port מספר 52634 (לקוח) שולח חבילה ל port מספר 12345 (שרת), ניתן לראות שחבילה זאת ללא data ולכן אורכה היא 0. ניתן לראות כי דגל הsyn דלוק ואילו דגל האck כבוי בheader (דבר שמייצג את תחילת ההתקשרות בhandshake), לחבילה מצורף ה Sequence number ההתחלתי של הלקוח.

חבילה מספר 48:



Sequence number : 2281506758

ack number : 1243725395 (prev Seq.num + 1)

source port :12345

destination port: 52634

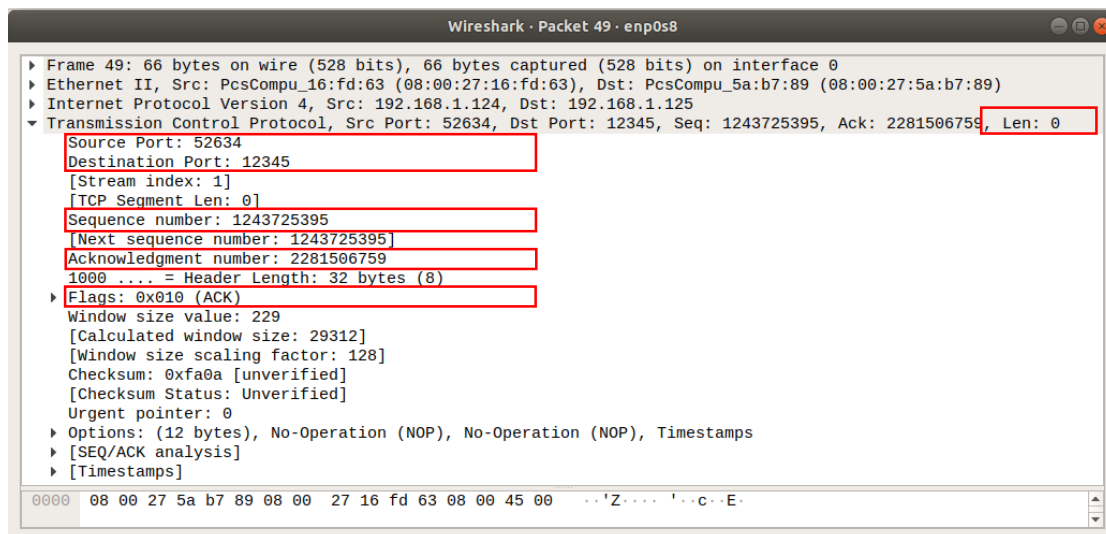
flags: SYN,ACK

length: 0

data :NO DATA

בחלק זה port מספר 12345(שרת) שולח חבילה ל port מספר 52634 (לקוח), ניתן לראות שחבילה זאת ללא data ולכן אורכה היא 0. ניתן לראות כי דגל הsync ודגל האck דלוקים בheader, בכך השרת מאשר ללקוח כי קיבל את רצונו להתקשרות. לחבילה מצורף Sequence number התחלתי של השרת וה-ack number הינו Sequence number של הלקוח + 1.

חבילה מספר 49:



Sequence number : 1243725395

ack number : 2281506759 (prev Seq.num + 1)

source port :52634

destination port: 12345

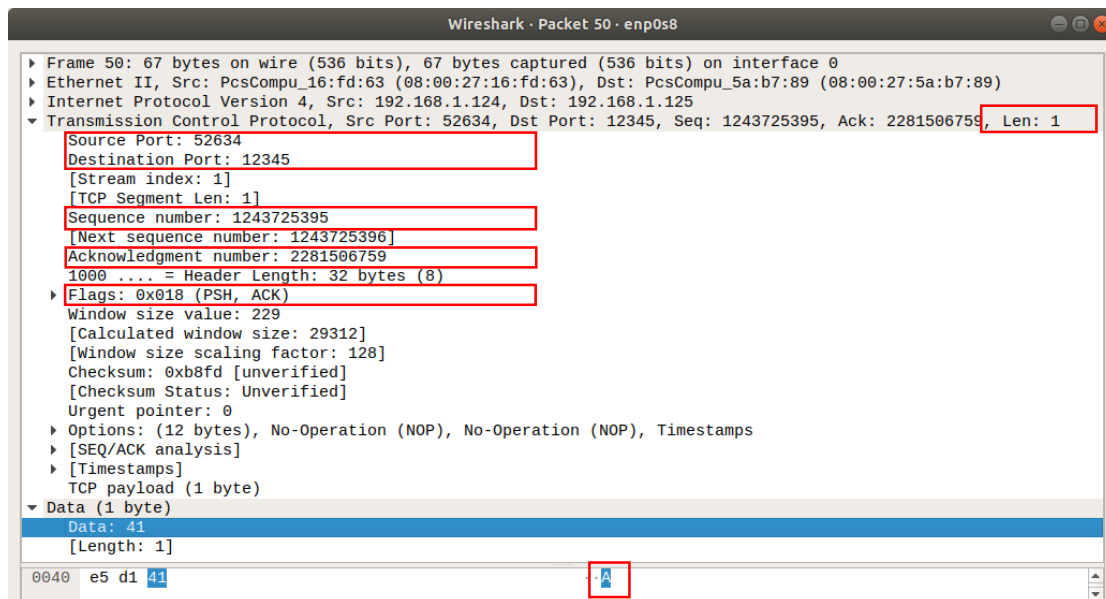
flags: ACK

length: 0

data :NO DATA

בחלק זה port מספר 52634 (לקוח) שולח חבילה ל port מספר 12345 (שרת), ניתן לראות שחבילה זאת ללא data ולכן אורכה היא 0. ניתן לראות כי דגל האק דלוק ב header ובכך מסתיים תהליך הhandshaken. לחבילה מצורף ack number שהוא ה-Sequence number של השרת +1. ה-Sequence number הוא ה ack number ששלח הלקוח בחבילה הקודמת.

חבילה מספר 50:



Sequence number : 1243725395 (prev seq.num)

ack number : 2281506759 (prev ack.num)

source port :52634

destination port: 12345

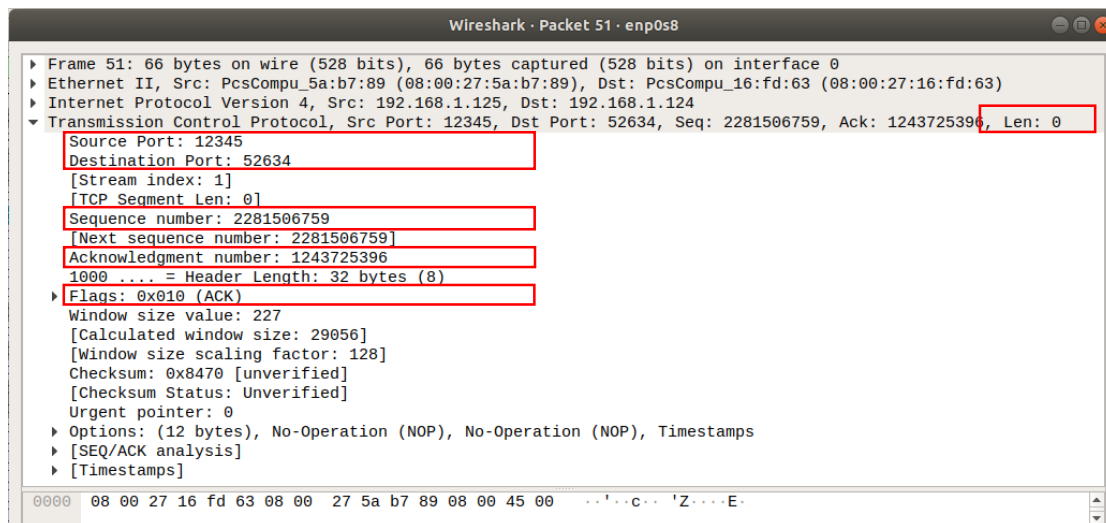
flags: PSH,ACK

length: 1

data : 'A'

בחלק זה port מספר 52634 (לקוח) שולח חבילה ל port מספר 12345 (שרת), ניתן לראות שחבילה זאת מכילה data ולכן length הוא 1. ניתן לראות כי דגלי הack והpsd דלוקים ב header, שכן הלקוח החל לשלוח data. ה ack number וה-Sequence number נשארים כפי שהיו בחבילה הקודמת ששלח הלקוח.

חבילה מספר 51:



Sequence number : 2281506759 (prev ack.num)

ack number : 1243725396 (prev seq.num + 1 (data len))

source port :12345

destination port: 52634

flags: ACK

length: 0

data :NO DATA

בחלק זה port מספר 12345 (שרת) שולח חבילה ל port מספר 52634 (לקוח), ניתן לראות שחבילה זאת ללא data ולכן הlength הוא 0. ניתן לראות כי דגל הack דלוק בheader, בכך השרת מאשר ללקוח כי קיבל את ה data ששלח. לחבילה מצורף Sequence number שזהה ל ack number המחבילה הקודמת, והack number זהה ל Sequence number + גודל data (1 B) (שהלקוח שלח בחבילה הקודמת).

The screenshot shows a Wireshark interface with a single packet selected. The packet details pane on the left lists the following fields:

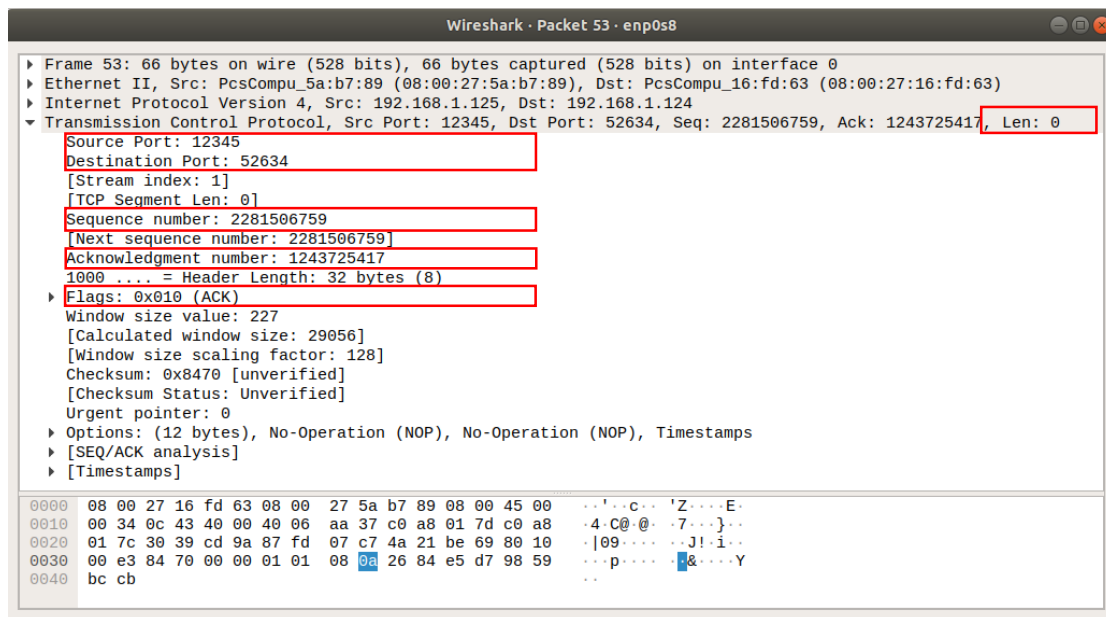
- Ethernet II, Src: PcsCompu_16:fd:63 (08:00:27:16:fd:63), Dst: PcsCompu_5a:b7:89 (08:00:27:5a:b7:89)
- Internet Protocol Version 4, Src: 192.168.1.124, Dst: 192.168.1.125
- Transmission Control Protocol, Src Port: 52634, Dst Port: 12345, Seq: 1243725396, Ack: 2281506759, Len: 21
- TCP Segment Info:
 - Source Port: 52634
 - Destination Port: 12345
 - [Stream index: 1]
 - [TCP Segment Len: 21]
 - Sequence number: 1243725396
 - [Next sequence number: 1243725417]
 - Acknowledgment number: 2281506759
 - 1000 = Header Length: 32 bytes (8)
 - Flags: 0x018 (PSH, ACK)
 - Window size value: 229
 - [Calculated window size: 29312]
 - [Window size scaling factor: 128]
 - Checksum: 0x2c56 [unverified]
 - [Checksum Status: Unverified]
 - Urgent pointer: 0
 - Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
 - [SEQ/ACK analysis]
 - [Timestamps]
- TCP payload (21 bytes)
- Data (21 bytes)
 - Data: 41
 - [Length: 21]

The packet bytes pane at the bottom displays the raw data in hexadecimal and ASCII. The first 21 bytes are all '41' (hex) or 'A' (ASCII). A red box highlights the last six bytes of the packet, which are 'AAAAAA' (hex) or 'AAAAAA' (ASCII).

```
data : 'AAA....'
```

בחלק זה port מספר 52634 (לקוח) שולח חבילה ל port מספר 12345 (שרת), ניתן לראות שחבילה זאת מכילה data ולכן length הוא 21. ניתן לראות כי דגלי ack והפשוט psh דלוקים ב header, שכן הלקוח סיים לשלוח data. ה ack number וה Sequence number נשארים כפי שהיו בחבילה הקודמת ששלח השרת.

חבילה מספר 53:



Sequence number : 2281506759 (prev ack.num)

ack number : 1243725417 (prev seq.num + 21 (data len))

source port :12345

destination port: 52634

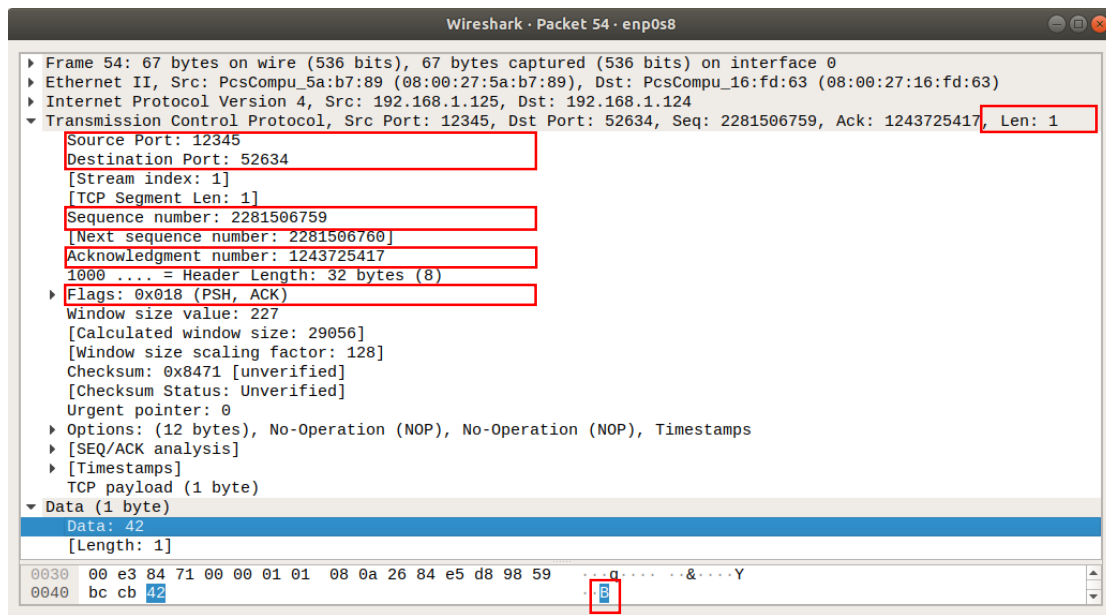
flags: ACK

length: 0

data :NO DATA

בחלק זה מספר 12345 (שרת) שולח חבילה ל port מספר 52634 (לקוח), ניתן לראות שחבילה זאת ללא data ולכן הlength הוא 0. ניתן לראות כי דגל הack דלוק בheader, בכך השרת מאשר ללקוח כי קיבל את ה data ששלח. לחבילה מצורף Sequence number שזהה ל ack number המחבילה הקודמת, והack number זהה ל Sequence number + גודל data (21 B) (שהלקוח שלח בחבילה הקודמת).

חבילה מספר 54:



Sequence number : 2281506759 (prev seq.num)

ack number : 1243725417 (prev ack.num)

source port :12345

destination port: 52634

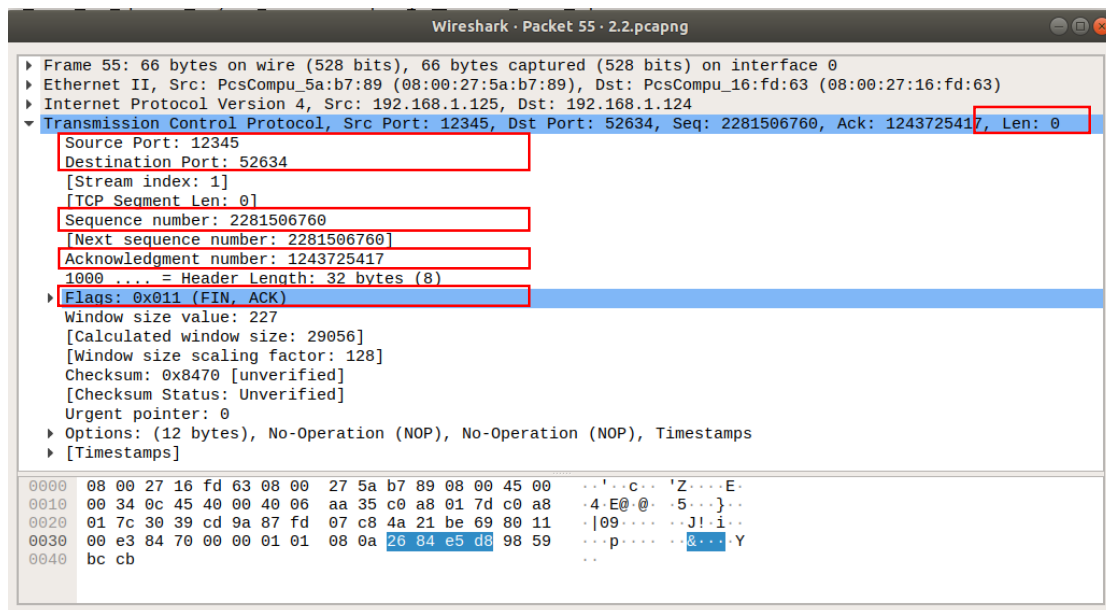
flags: PSH,ACK

length: 1

data :B

בחלק זה port מספר 12345(שרת) שולח חבילה ל port מספר 52634 (לקוח), ניתן לראות שחבילה זאת מכילה data ולכן הlength הוא 1. ניתן לראות כי דגלי הpsb והack דלוקים ב header. ה ack number וה-Sequence number נשארים כפי שהיו בחבילה הקודמת ששלח השרת.

חבילה מספר 55:



Sequence number : 1243725417 (prev ack.num)

ack number : 2281506760 (prev seq.num + 1)

source port :12345

destination port: 52634

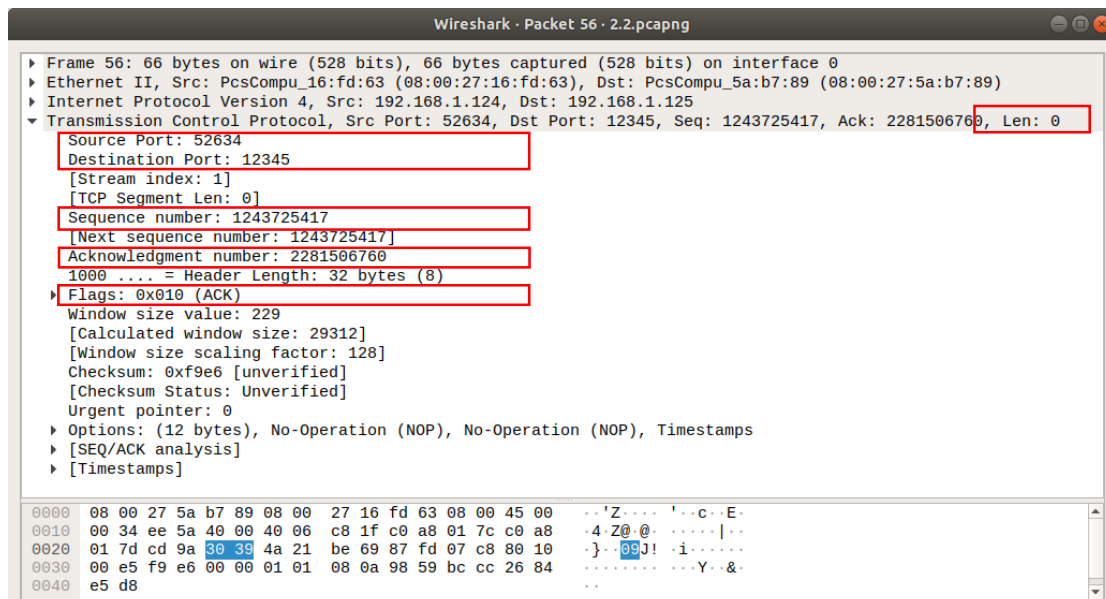
flags: FIN,ACK

length: 0

data : NO DATA

בחלק זה port מספר 12345 (שרת) שולח חבילה ל port מספר 52634 (לקוח), ניתן לראות שחבילה זאת ללא data ולכן הlength הוא 0. ניתן לראות כי דגלי הfin והack דלוקים בheader, בכך השרת מסמן ללקוח כי ברצונו לסיים את ההתקשרות (תחילת תהליך ה teardown). לחבילה מצורף Sequence number שזהה לack number המחבילה הקודמת, והack number זהה ל Sequence number + 1 מהחבילה הקודמת ששלח השרת.

חבילה מספר 56:



Sequence number : 2281506760 (prev ack.num)

ack number : 1243725417 (prev seq.num)

source port :52634

destination port: 12345

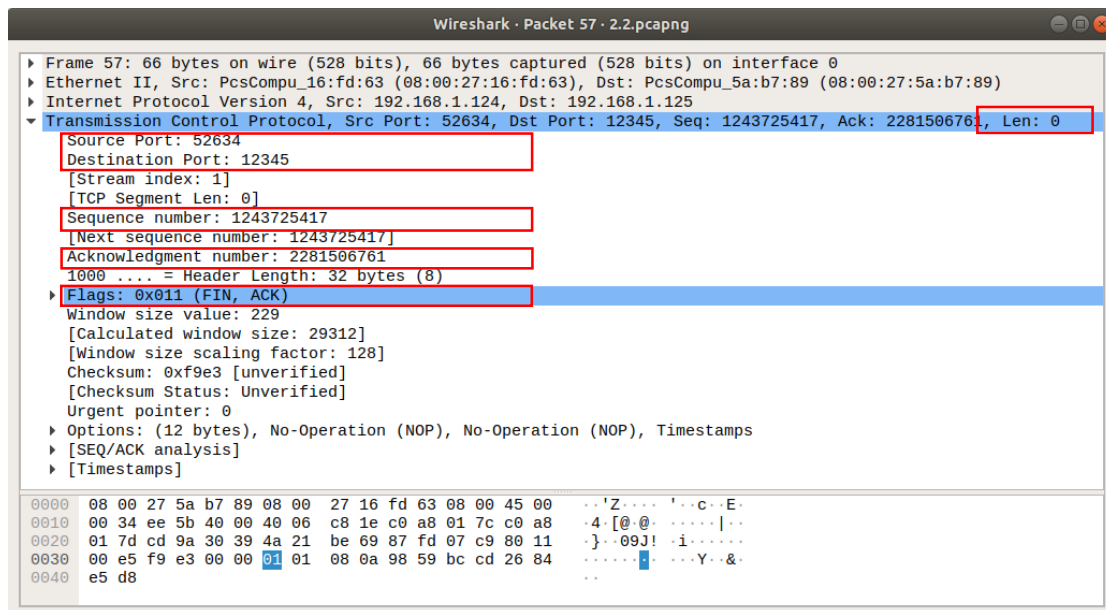
flags: ACK

length: 0

data : NO DATA

בחלק זה port מספר 52634 (לקוח) שולח חבילה ל port מספר 12345 (שרת), ניתן לראות שחבילה זאת ללא data ולכן הlength הוא 0. ניתן לראות כי דגל הack דלוק בheader, בכך הלקוח מסמן לשרת כי קיבלת את ה data ששלח קודם. לחבילה מצורף Sequence number שזהה ל ack number המחבילה הקודמת עם ה data, והack number זהה ל Sequence number + גודל ה data (1 B) שהשרת שלח בחבילה הקודמת.

חבילה מספר 57:



Sequence number : 1243725417 (prev ack.num)

ack number : 2281506761 (prev seq.num + 1(data len))

source port :52634

destination port: 12345

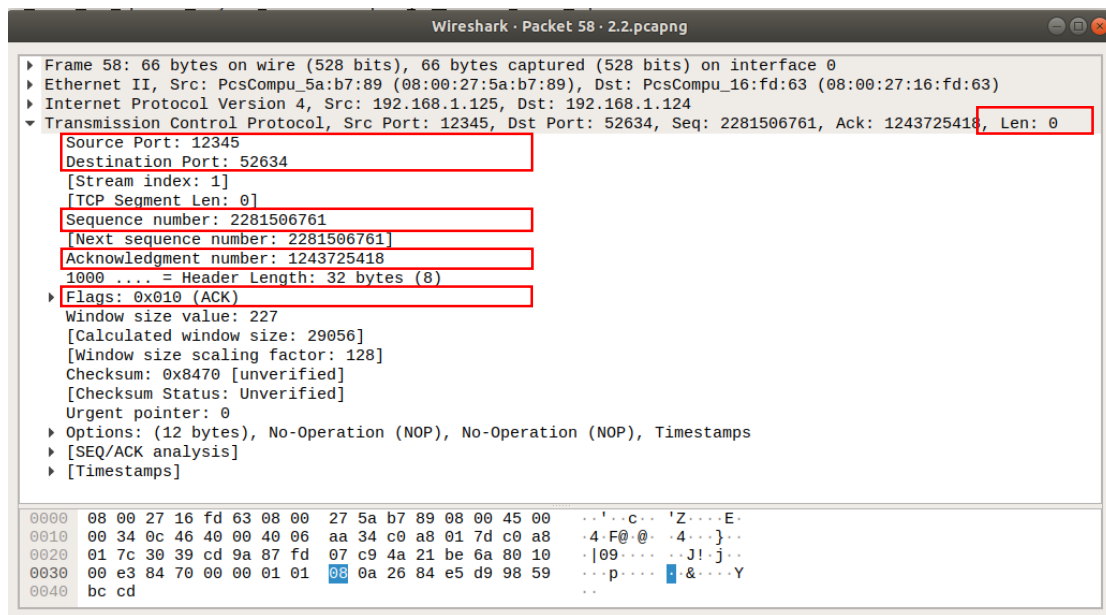
flags: ACK

length: 0

data : NO DATA

בחלק זה port מספר 52634 (לקוח) שולח חבילה ל port מספר 12345 (שרת), ניתן לראות שחבילה זאת ללא data ולכן הlength הוא 0. ניתן לראות כי דגלי הfin והack דלוקים בheader, בכך הלקוח מסמן לשרת כי קיבלת את הודעתו שברצונו לסיים את ההתקשרות(תהליך הteardown). לחבילה מצורף Sequence number שזהה לack number המחבילה הקודמת, והack number זהה ל Sequence number + 1 מהחבילה הקודמת ששלח השרת.

חבילה מספר 58:



Sequence number : 2281506761 (prev ack.num)

ack number : 1243725418 (prev seq.num + 1)

source port :12345

destination port: 52634

flags: FIN,ACK

length: 0

data : NO DATA

בחלק זה port מספר 12345 (שרת) שולח חבילה ל port מספר 52634 (לקוח), ניתן לראות שחבילה זאת ללא data ולכן הlength הוא 0. ניתן לראות כי דגל האck דלוק בheader, בכך השרת מסיים את ההתקשרות עם הלקוח (סוף תהליך הteardown). לחבילה מצורף Sequence number שזהה לack number המחבילה הקודמת, והack number זהה ל Sequence number + 1 מהחבילה הקודמת ששלח השרת.

סעיף ג':

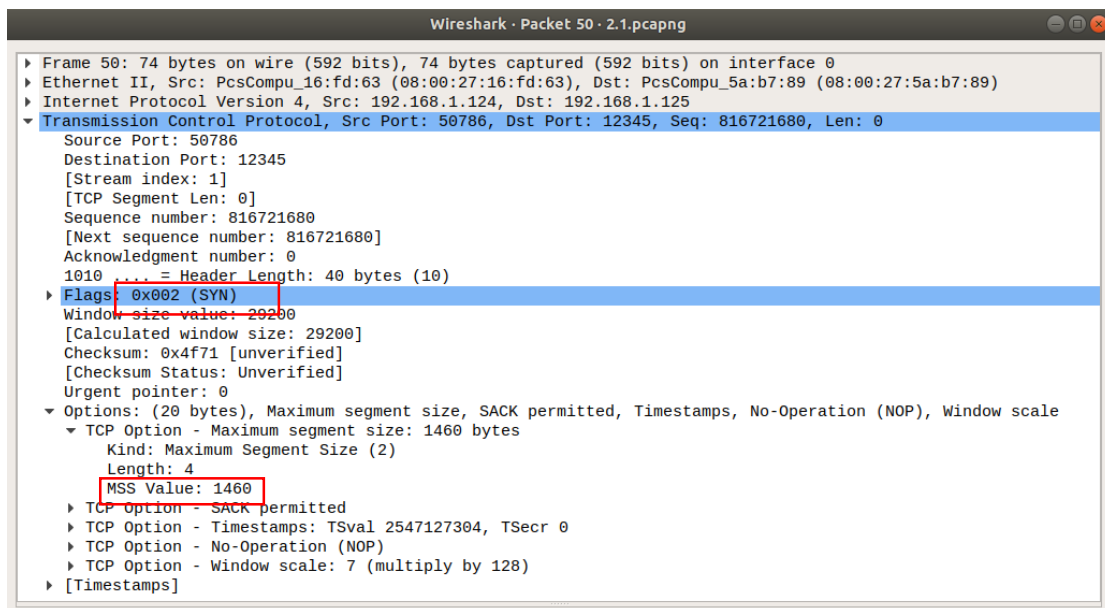
בסעיף זה אראה את ההבדלים בין הפרוטוקולים UDP, TCP שבאו לידי ביטוי בסעיפים א' וב'.

הבדלים מסעיף א':

בסעיף זה נשלחים 15,000 B של data מצד לקוח בבאת אחת, שרתי ה TCP וה UDP מתמודדים עם כמות זאת של data בדרכים שונות.

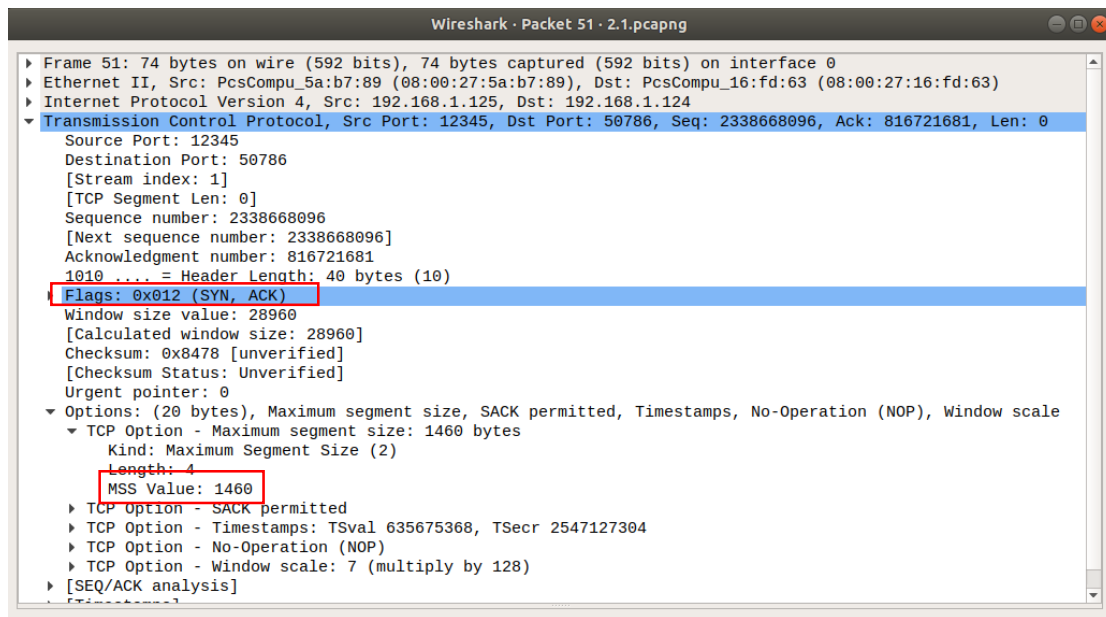
פרוטוקול TCP עושה שימוש בתהליך ה handshake, ובכך "קובעים" הלקוח והשרת את גודל המידע שאפשר להעביר, ובכך הם קובעים את ה- **MSS** (maximum segment size). בכך הצדדים מסכימים על גודל סגמנט מקסימלי שניתן להעביר ומוודאים שאכן כל חבילה שנשלחת מאחד הצדדים תוכל להתקבל בצד השני. לאחר שנקבע ה MSS במידה ואחד הצדדים רוצה לשלוח חבילה שגודל ה data שלה גדול מה-MSS (20 בתים של ה tcp header ו20 בתים של ה IP header אינם נכללים ב-MSS), נאלץ הלקוח במקרה שלנו לפרק את החבילה לחבילות קטנות יותר, תהליך זה נקרא **סגמנטציה**. פירוק זה מתרחש בשכבת התעבורה. ב-wire shark ניתן לראות שכאשר דגל ה syn ב TCP header דלוק, נקבע בין הלקוח לשרת גודל ה-MSS.

חבילה ראשונה בhandshake מלקוח לשרת:



ניתן לראות כי ה-MSS הוא 1460, כלומר גודל הdata ללא הheaders חסום מלמעלה ב1460 B

חבילה שניה בhandshake משרת ללקוח:



ואכן גודל הdata שנשלח לא עלה על 1460.

No.	Time	Source	Destination	Protocol	Length	Info
50	21.4...	192.168.1.124	192.168.1.125	TCP	74	50786 → 12345 [SYN] Seq=816721680 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=25...
51	21.4...	192.168.1.125	192.168.1.124	TCP	74	12345 → 50786 [SYN, ACK] Seq=2338668096 Ack=816721681 Win=28960 Len=0 MSS=1460
52	21.4...	192.168.1.124	192.168.1.125	TCP	66	50786 → 12345 [ACK] Seq=816721681 Ack=2338668097 Win=29312 Len=0 TSval=25471273...
53	21.4...	192.168.1.124	192.168.1.125	TCP	7306	50786 → 12345 [ACK] Seq=816721681 Ack=2338668097 Win=29312 Len=7240 TSval=25471...
54	21.4...	192.168.1.125	192.168.1.124	TCP	66	12345 → 50786 [ACK] Seq=2338668097 Ack=816728921 Win=43520 Len=0 TSval=63567538...
55	21.4...	192.168.1.124	192.168.1.125	TCP	7306	50786 → 12345 [ACK] Seq=816728921 Ack=2338668097 Win=29312 Len=7240 TSval=25471...
56	21.4...	192.168.1.125	192.168.1.124	TCP	66	12345 → 50786 [ACK] Seq=2338668097 Ack=816736161 Win=57984 Len=0 TSval=63567538...
57	21.4...	192.168.1.124	192.168.1.125	TCP	586	50786 → 12345 [PSH, ACK] Seq=816736161 Ack=2338668097 Win=29312 Len=520 TSval=2...
58	21.4...	192.168.1.125	192.168.1.124	TCP	66	12345 → 50786 [ACK] Seq=2338668097 Ack=816736681 Win=60928 Len=0 TSval=63567538...
59	21.4...	192.168.1.125	192.168.1.124	TCP	67	12345 → 50786 [PSH, ACK] Seq=2338668097 Ack=816736681 Win=60928 Len=1 TSval=635...
60	21.4...	192.168.1.125	192.168.1.124	TCP	66	12345 → 50786 [FIN, ACK] Seq=2338668098 Ack=816736681 Win=60928 Len=0 TSval=635...
61	21.4...	192.168.1.124	192.168.1.125	TCP	66	50786 → 12345 [ACK] Seq=816736681 Ack=2338668098 Win=29312 Len=0 TSval=25471273...
62	21.4...	192.168.1.124	192.168.1.125	TCP	66	50786 → 12345 [FIN, ACK] Seq=816736681 Ack=2338668099 Win=29312 Len=0 TSval=254...
63	21.4...	192.168.1.125	192.168.1.124	TCP	66	12345 → 50786 [ACK] Seq=2338668099 Ack=816736682 Win=60928 Len=0 TSval=63567539...

1010 = Header Length: 40 bytes (10)

Flags: 0x002 (SYN)

Window size value: 29200

[Calculated window size: 29200]

Checksum: 0x4f71 [unverified]

[Checksum Status: Unverified]

Urgent pointer: 0

Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale

TCP Option - Maximum segment size: 1460 bytes

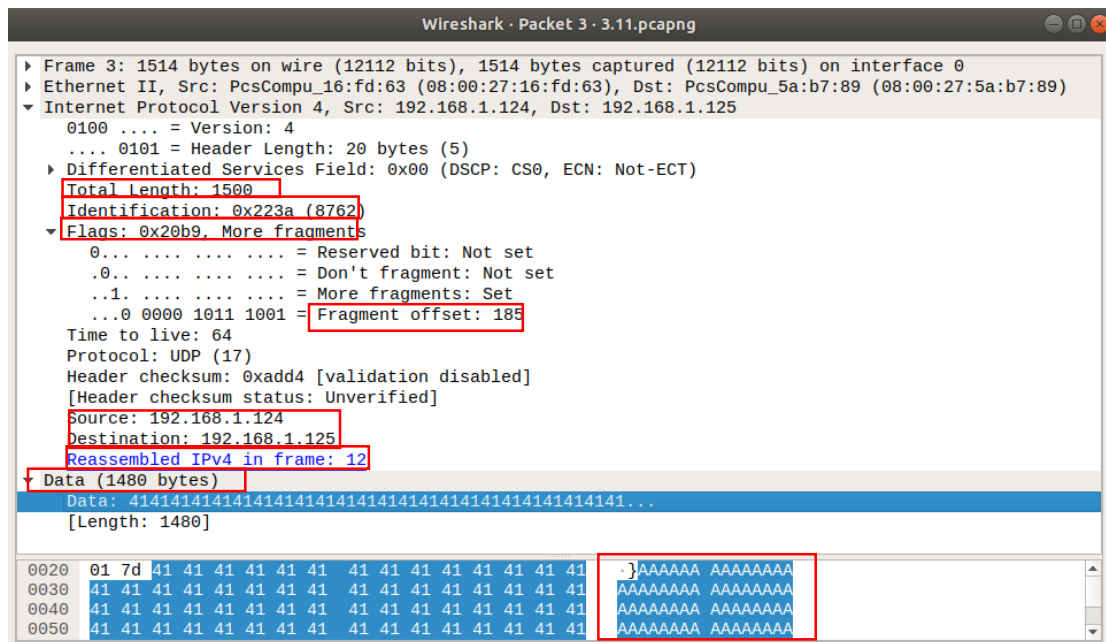
Kind: Maximum Segment Size (2)

Length: 4

MSS Value: 1460

בצילומים ניתן לראות כי הפירוק אכן מתקיים בשכבת התעבורה, שכן בכל החבילות שבהן הועבר data, דגל DF (don't fragment) ב header IP היו דלוקים, ומעולם דגל MF (more fragment) אינו היה דלוק.

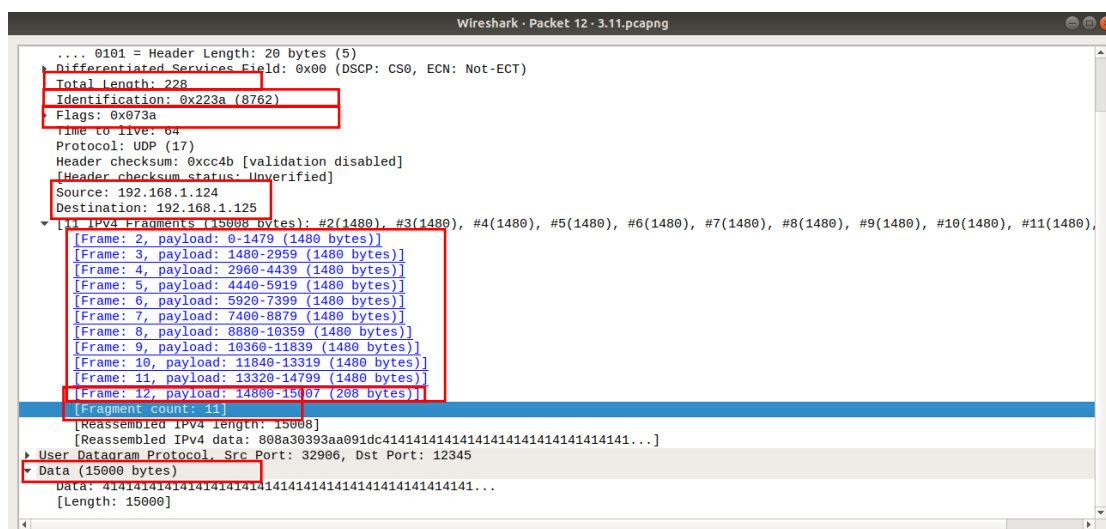
חבילה מספר 3:



בחלק זה הלקוח שולח חבילה לשרת, ניתן לראות שחבילה זאת בעלת data ולכן הlength הוא 1480, פלוס 20B של IP header, ולכן נסיק כי הMTU הינו 1500B. לכל חבילה יש identification מזהה ייחודי ובכך יכול הפרוטוקול לדעת אילו חבילות יש לחבר בסיום התהליך. ניתן לראות כי דגלי MF דלוק בheader, בכך הלקוח מסמן לשרת כי קיבלת קיימות עוד חבילות עם data בדרך, ושלא ירכיב עדיין את החבילות שהתקבלו. כיוון שזוהי החבילה השניה שנשלחה, הoffset הוא $1480/8=185$. ב wireshark ניתן לראות כי החבילות יורכבו בחבילה מספר 12.

חבילות 4-11 זהות פרט לכך שה offset גדל ב185 $1480/8=185$

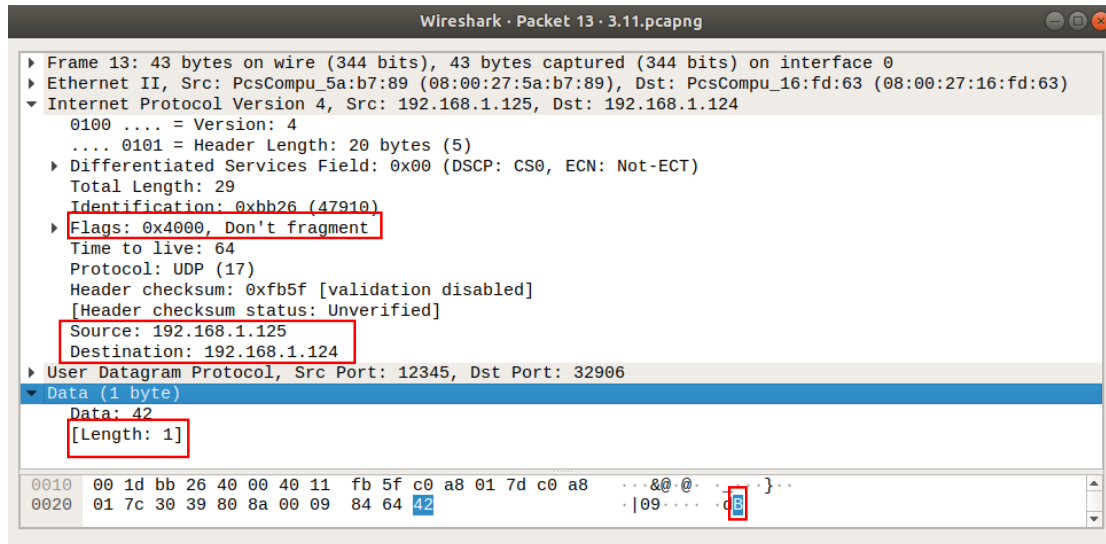
חבילה מספר 12:



בחלק זה הלקוח שולח את חבילת הdata האחרונה לשרת, ניתן לראות שחבילה זאת בעלת data length הוא 15000. לכל חבילה יש identification מזהה ייחודי ובכך יכול הפרוטוקול לדעת אילו חבילות יש לחבר כעת. אפשר לראות כי גודל ה data שהועבר בחבילה זאת הוא 208B וכי מספר הפרגמנטים הוא 11. ניתן לראות כי דגלי MF כבוי header, בכך הלקוח מסמן לשרת כי לא

קיימות עוד חבילות עם data בדרך, ושהוא יכול להתחיל להרכיב את החבילות שהתקבלו עד כה. כיוון שזוהי החבילה האחרונה שנשלחה, offset הוא $15,000/8=1850$. ב Wireshark ניתן לראות את פירוט החבילות בעלות המזהה המשותף שהתקבלו עד עתה.

חבילה מספר 13:



בחלק זה השרת שולח את התו B ללקוח, גודל ה data הינו 1 וכיוון שהוא קטן מה MTU אין צורך בלפרק את החבילה לפרגמנטים ולכן ודגל ה DF אכן דלוק ב header IP.

כעת אציג את חבילות בין הלקוח לשרת המקביל לסעיף ב', אשים דגש על ההבדלים בין סוגי הפרוטוקולים. בסעיף זה נשלחות מהלקוח לשרת 22 חבילות בעלות data בגודל 1B. כיוון ש1B קטן מגודל הMTU הפרוטוקול אינו מבצע פרגמנטציה של data. בניגוד לפרוטוקול TCP בו מתקיימת חלוקה דינמית לסגמנטים, בפרוטוקול UDP החבילות נשלחות כמו שהן. מבחינת נצילות נשים לב כי על כל תו בודד שנשלח מתווסף לו IP, UDP headers וסה"כ גודל החבילה הוא 60B על כל 1B של data. לכן על מנת להעביר את ה data שהלקוח שלח נעשה שימוש ב $22 * 60B = 1,320B$. בפרוטוקול TCP המבצע סגמנטציה מסעיף ב' נעשה שימוש ב266B כולל האck מהשרת. ניתן בקלות לראות כי overhead של headers בפרוטוקול UDP מגדיל משמעותית את סה"כ גודל החבילות שנשלחו.

No.	Time	Source	Destination	Protocol	length	info
29	18.8...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
30	18.8...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
33	18.8...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
34	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
35	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
36	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
37	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
38	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
39	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
40	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
41	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
42	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
43	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
44	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
45	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
46	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
47	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
48	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
49	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
50	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
51	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1
52	18.9...	192.168.1.124	192.168.1.125	UDP	60	60953 → 12345 Len=1

ניתן לראות כי חבילות 29-51 הן מהלקוח לשרת, וחבילה 52 היא מהשרת ללקוח.

שרת: כתובת IP : 192.168.1.125

לקוח: כתובת IP : 192.168.1.124

חבילה מספר 29:

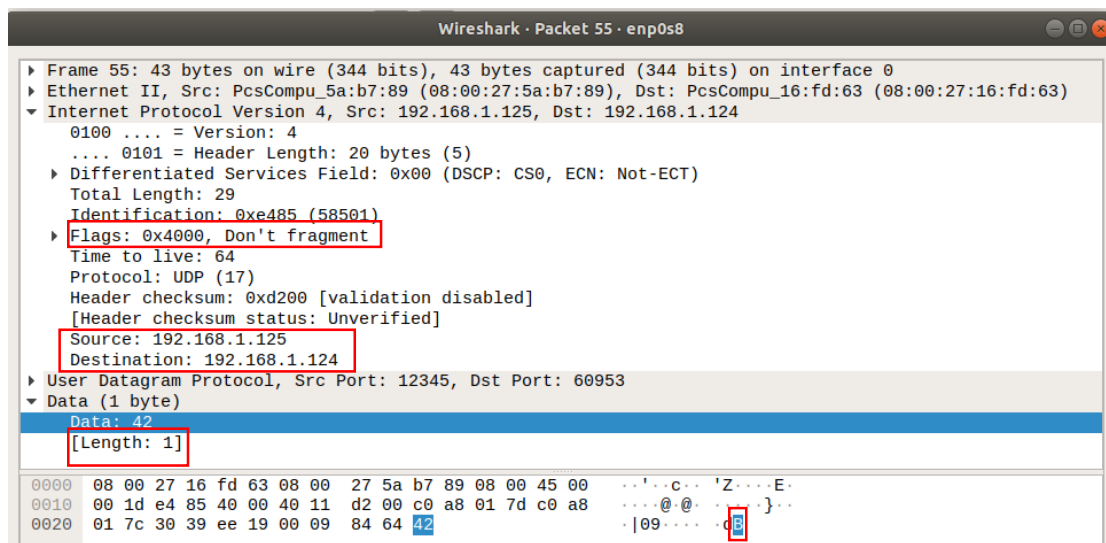
Wireshark · Packet 29 · enp0s8

- Frame 29: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
- Ethernet II, Src: PcsCompu_16:fd:63 (08:00:27:16:fd:63), Dst: PcsCompu_5a:b7:89 (08:00:27:5a:b7:89)
- Internet Protocol Version 4, Src: 192.168.1.124, Dst: 192.168.1.125
 - 0100 = Version: 4
 - 0101 = Header Length: 20 bytes (5)
 - Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 - Total Length: 29
 - Identification: 0xe272 (57970)
 - Flags: 0x4000, Don't fragment**
 - Time to live: 64
 - Protocol: UDP (17)
 - Header checksum: 0xd413 [validation disabled]
 - [Header checksum status: Unverified]
 - Source: 192.168.1.124**
 - Destination: 192.168.1.125**
- User Datagram Protocol, Src Port: 60953, Dst Port: 12345
- Data (1 byte)
 - [Length: 1]**

0000 08 00 27 5a b7 89 08 00 27 16 fd 63 08 00 45 00 ..'Z....'.c..E.
 0010 00 1d e2 72 40 00 00 11 d4 13 c0 a8 01 7c c0 a8 ...r@.@@.|..
 0020 01 7d ee 19 30 39 00 09 1c 3f 41 00 00 00 00 00 ...}..09...A..
 0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00A..

חבילה שנשלחת מהלקוח לשרת, כיוון שאין צורך בסגמנטציה כאשר ה data קטן מהMTU, דגל ה DF דלוק ב header IP, יתר החבילות בין הלקוח לשרת נראות זהה.

חבילה מספר 55:



חבילה שנשלחת מהשרת ללקוח, כיוון שאין צורך בסגמנטציה כאשר ה data קטן מה MTU, דגל ה DF דלוק ב header IP.