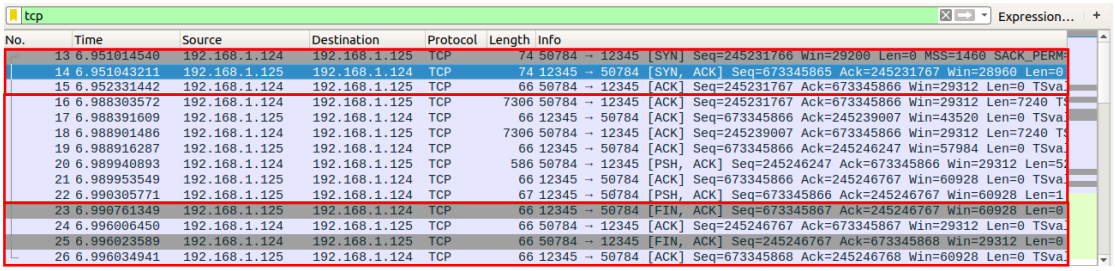
שאלה 2:

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'.



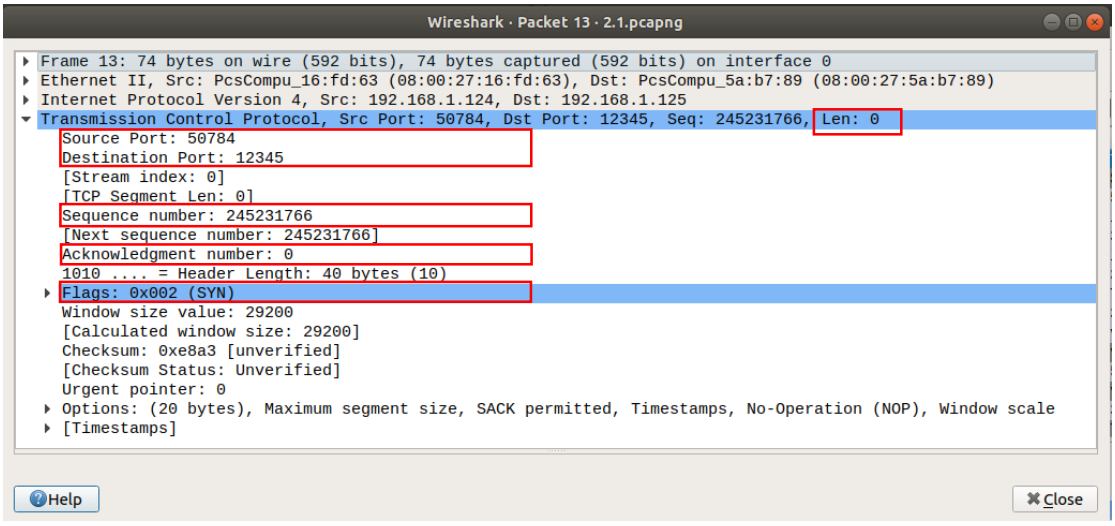
No.	Time	Source	Destination	Protocol	Length	Info
13	6.951014540	192.168.1.124	192.168.1.125	TCP	74	50784 → 12345 [SYN] Seq=245231766 Win=29200 Len=0 MSS=1460 SACK_PERM=
14	6.951043211	192.168.1.125	192.168.1.124	TCP	74	12345 → 50784 [SYN, ACK] Seq=673345865 Ack=245231767 Win=28960 Len=0
15	6.952331442	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 TS
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 TS
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
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
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 בhandshake (חבילות 13-15). לאחר מכן מועבר הdata (חבילות 16-22) בהרצה זאת על כל חבילה ששולח הלקוח השרת עונה לו מיד בack, אך אין זה קורה בכל הרצה. לבסוף מתרחש teardown (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:



Sequence number : 245231766

ack number : 0

source port :50784

destination port: 12345

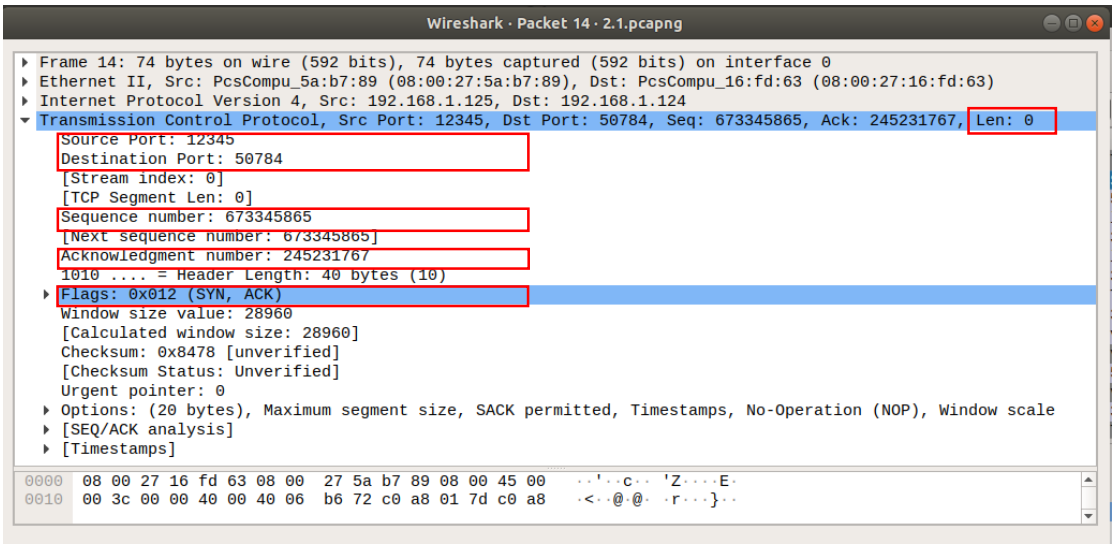
flags: SYN

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 הack כבוי בheader (דבר שמייצג את תחילת ההתקשרות בhandshake כפי שציינתי בשאלה הקודמת) , לחבילה מצורף הSequence number ההתחלתי של הלקוח.

חבילה מספר 14:



Sequence number : 673345865

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

source port :12345

destination port: 50784

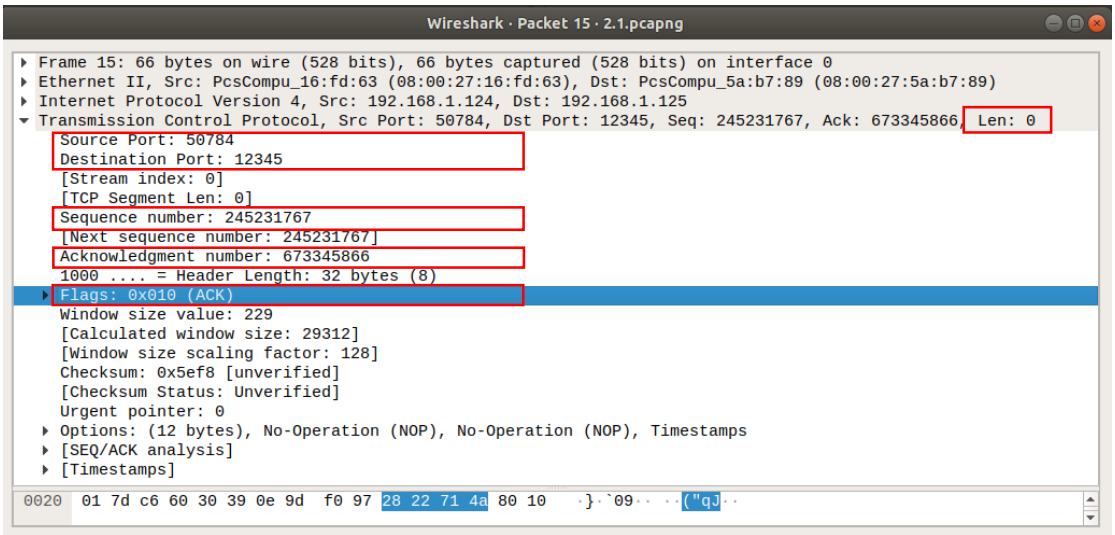
flags: SYN,ACK

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 הack דלוקים בheader, בכך השרת מאשר ללקוח כי קיבל את רצונו להתקשרות. לחבילה מצורף Sequence number התחלתי של השרת וה-ack number הינו הSequence number של הלקוח + 1.

חבילה מספר 15:



Sequence number : 245231767

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

source port :50784

destination port: 12345

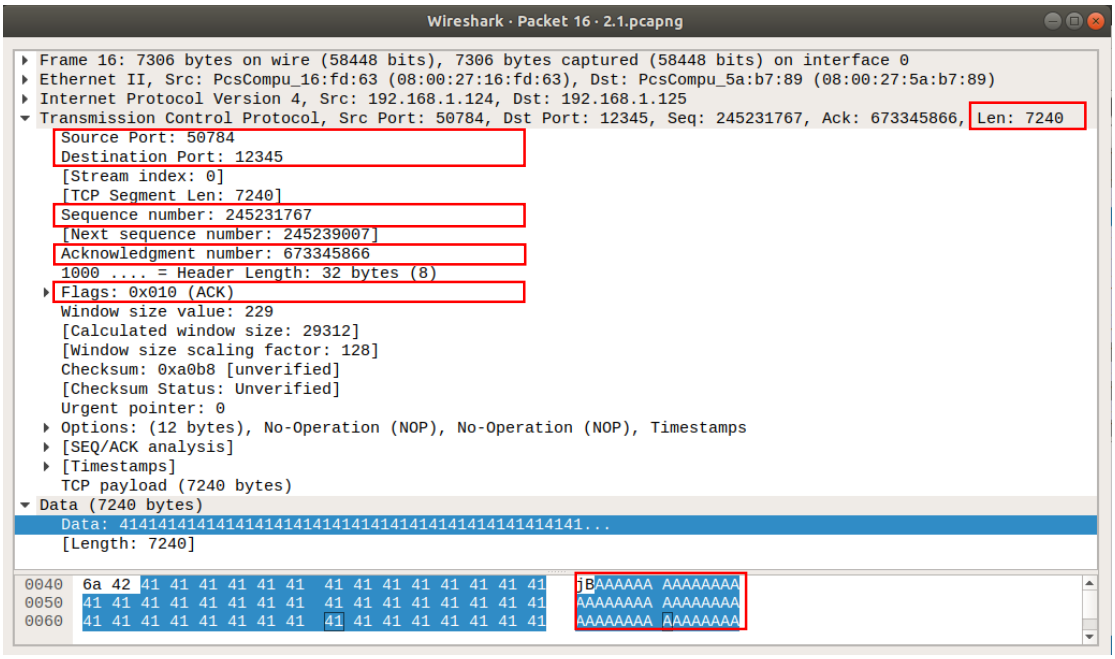
flags: ACK

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 הack דלוק ב header ובכך מסתיים תהליך הhandshake. לחבילה מצורף ack number שהוא ה-Sequence number של השרת +1 . ה-Sequence number הוא ה ack number ששלח הלקוח בחבילה הקודמת.

חבילה מספר 16:



Sequence number : 245231767 (prev seq.num)

ack number : 673345866 (prev ack.num)

source port :50784

destination port: 12345

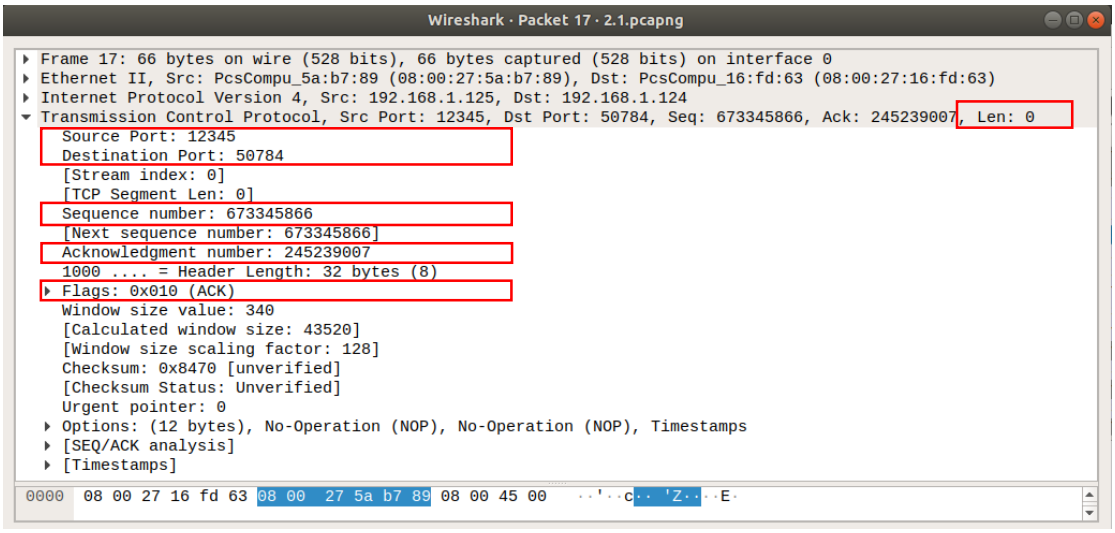
flags: ACK

length: 7240

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 הack 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

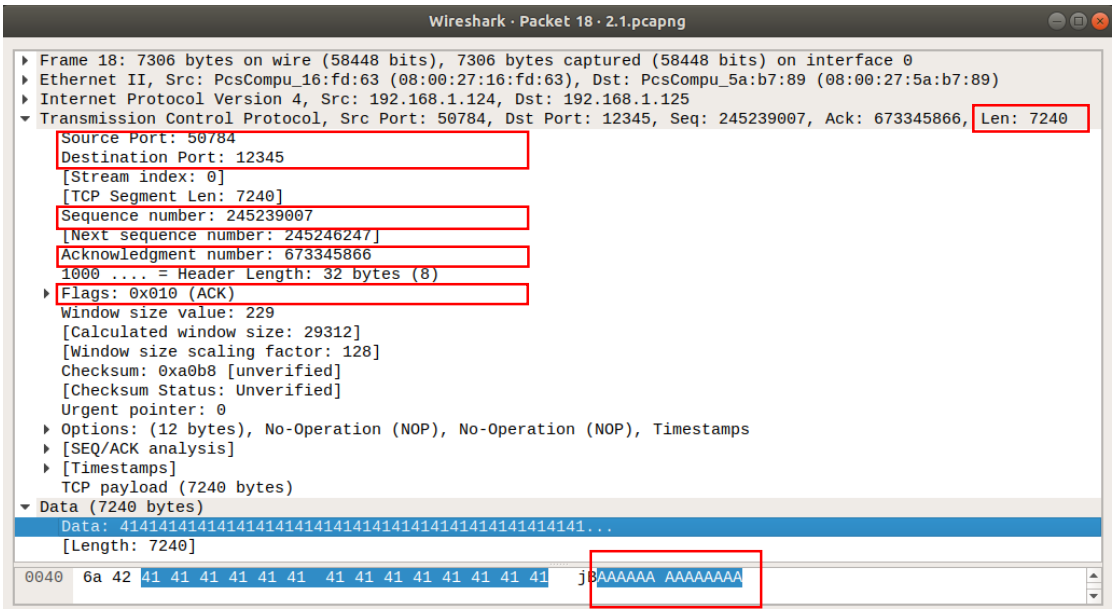
flags: ACK

length: 0

data :NO DATA

בחלק זה 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 הdata (7240 B)שהלקוח שלח בחבילה הקודמת .

חבילה מספר 18:



Sequence number : 245239007 (prev ack.num)

ack number : 673345866 (prev seq.num)

source port :50784

destination port: 12345

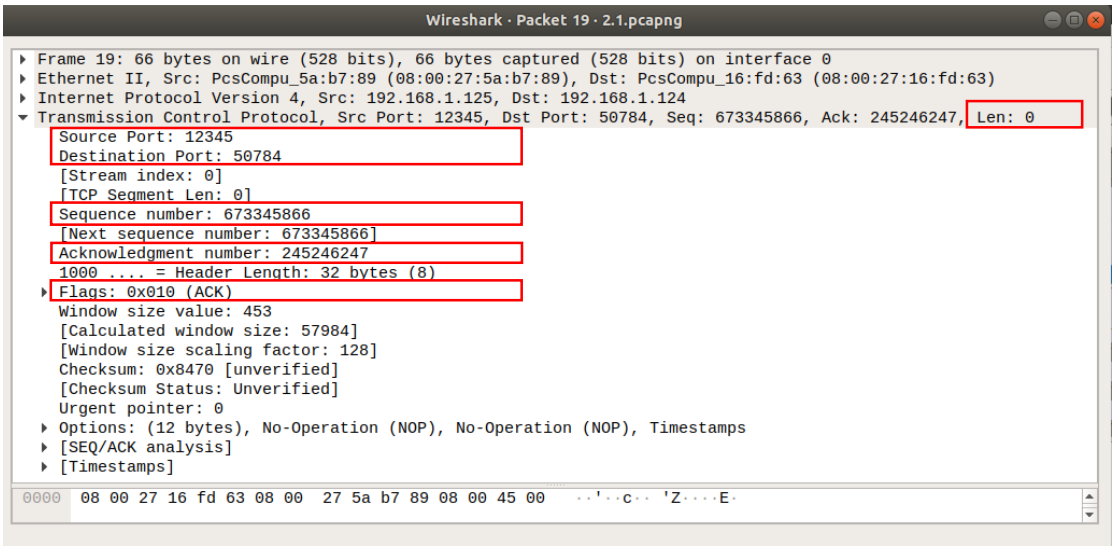
flags: ACK

length: 7240

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 הack 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

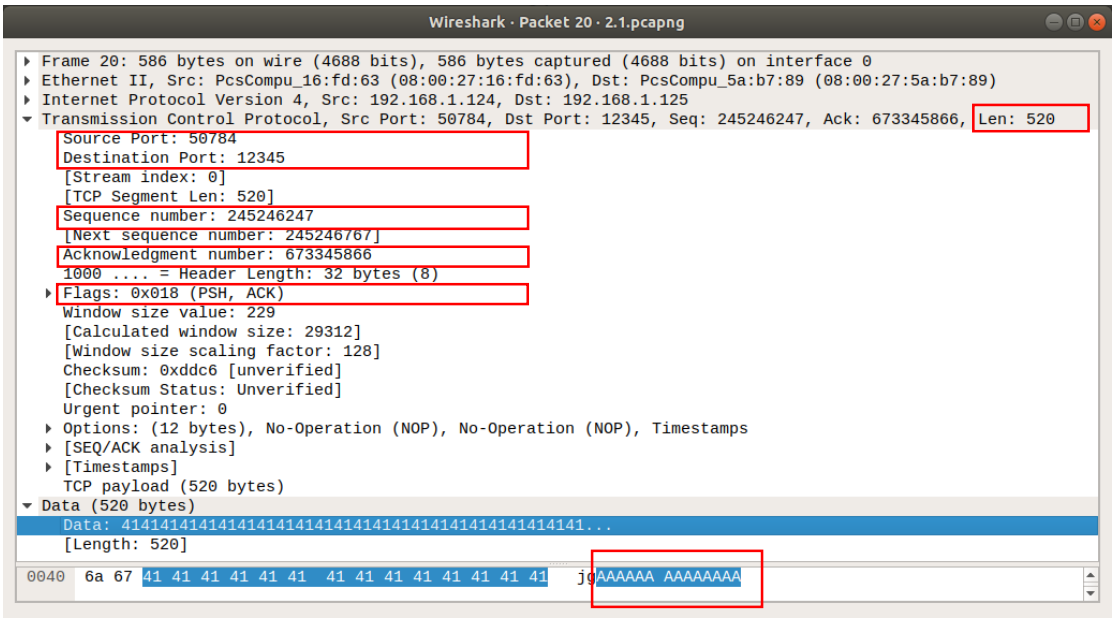
flags: ACK

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 הdata (7240 Bשהלקוח שלח בחבילה הקודמת .

חבילה מספר20:



Sequence number : 245246247 (prev ack.num)

ack number : 673345866 (prev seq.num)

source port :50784

destination port: 12345

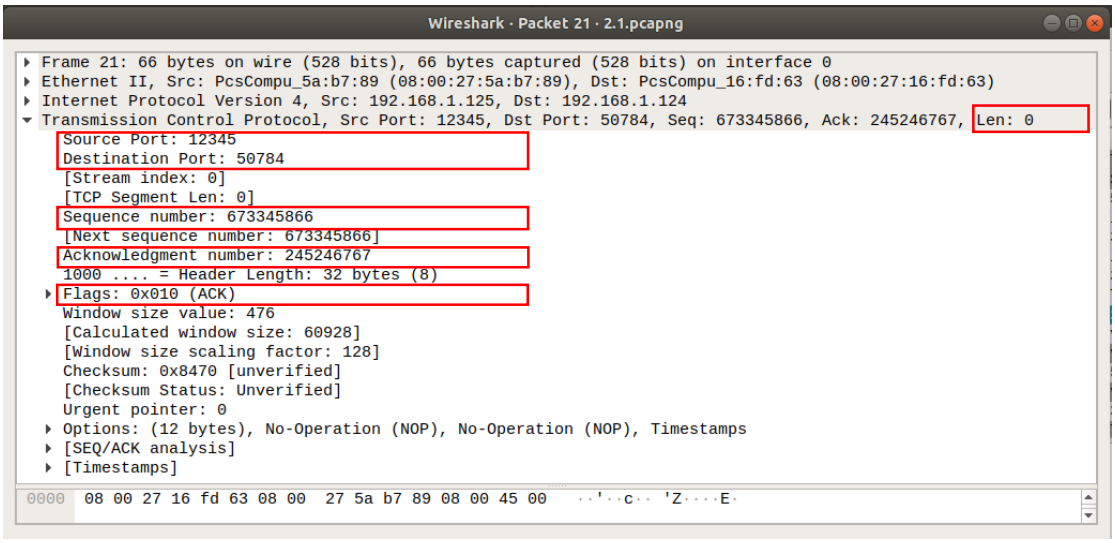
flags: PSH,ACK

length: 520

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 דגלי הpsh והack דלוקים ב header.
ה ack number וה-Sequence number מתחלפים מכפי שהיו בחבילה הקודמת ששלח השרת.

חבילה מספר 21:



Sequence number : 673345866 (prev ack.num)

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

source port :12345

destination port: 50784

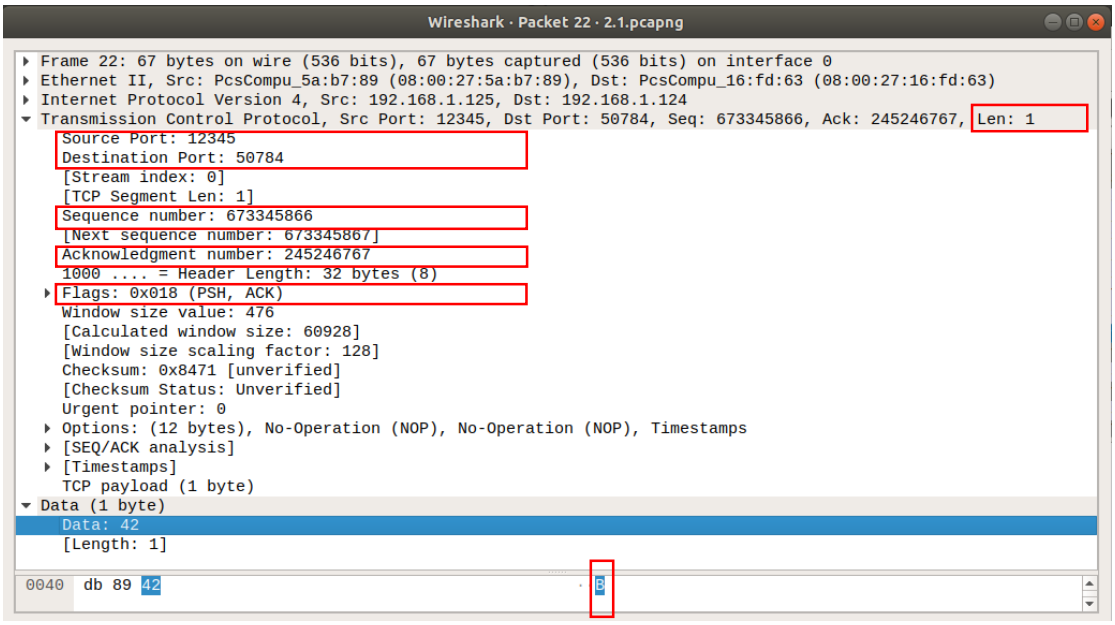
flags: ACK

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 הdata (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

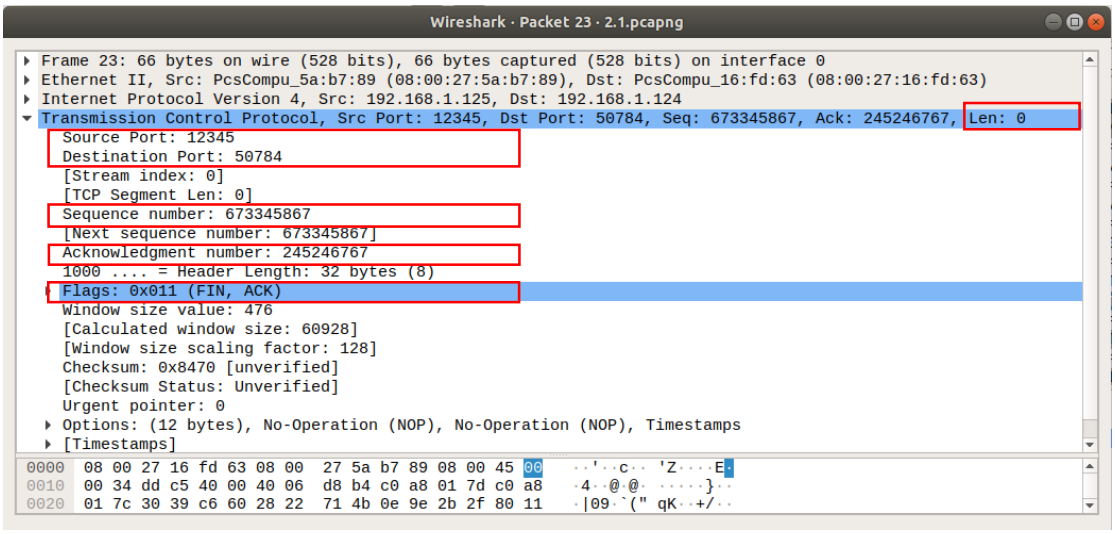
flags: PSH,ACK

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 וה-ack דלוקים ב header, דגל psh מסמן שהשרת סיים לשלוח את כל ה data, ולא צפויות להגיע עוד חבילות. ה ack number וה-Sequence 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

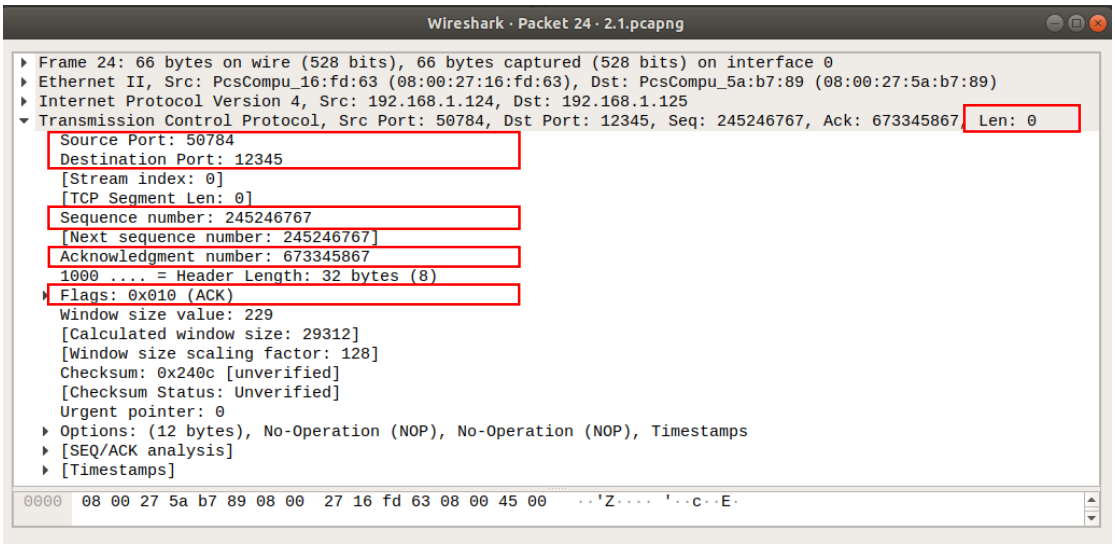
flags: FIN,ACK

length: 0

data : NO DATA

בחלק זה port 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

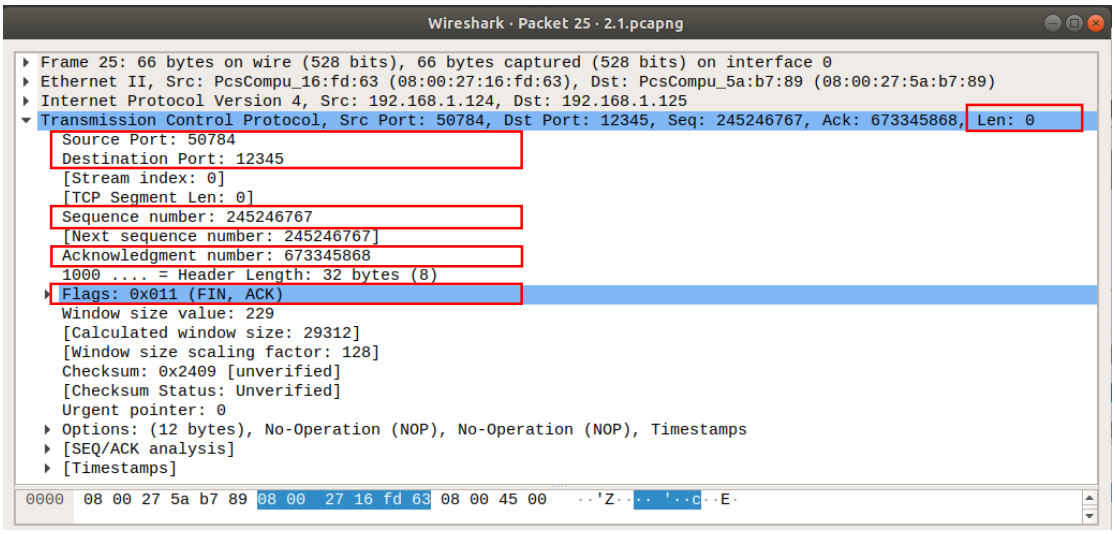
flags: ACK

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 הdata (1 B)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

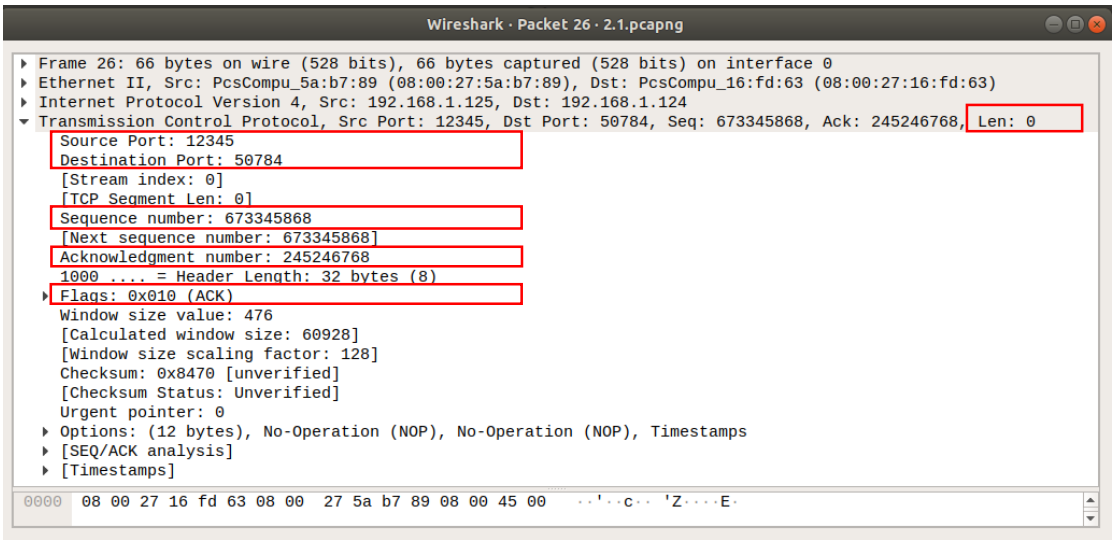
flags: FIN,ACK

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 ה teardown). 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 הteardown). לחבילה מצורף Sequence number שזהה לack number המחבילה הקודמת, והack number זהה ל Sequence number + 1 מהחבילה הקודמת ששלח השרת.

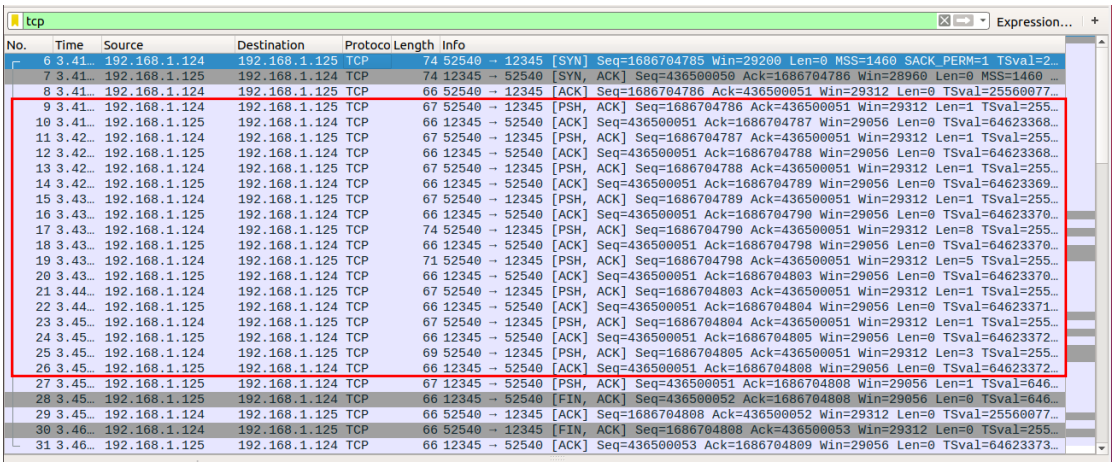
סעיף ב:

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 הdata שונה מלקוח ללקוח. כפי שלמדנו בפרוטוקול TCP החבילות מועברות כstream של בתים, ולכן כאשר מספר ההודעות שנשלחות 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 באדום את החבילות שמהן שלח הלקוח לשרת לראשונה חבילה עם data ועד ה ack האחרון ששלח השרת ללקוח.

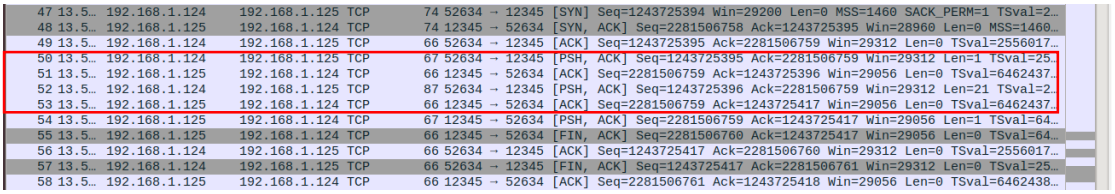
לקוח מספר 1 :



No.	Time	Source	Destination	Protoco	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=2556017…
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…

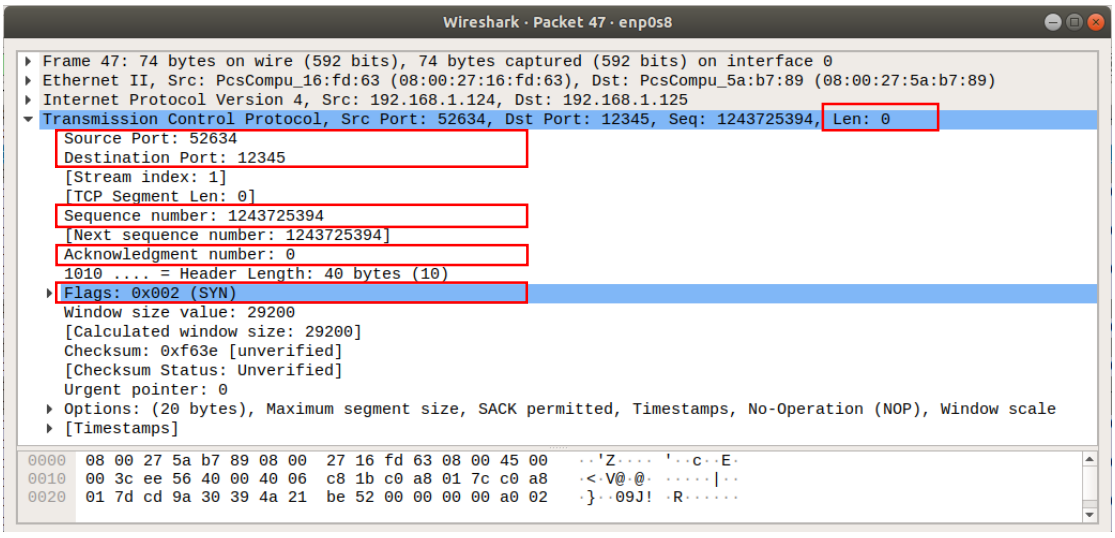
לעומת זאת בלקוח זה, נשלחו רק 2 חבילות עם data, על מנת להעביר את אותו תוכן כמו בלקוח מספר 1.

כעת אנתח את לקוח מספר 2. ניתן לראות כי כל התהליך מתחיל בhandshake (חבילות 47-49). לאחר מכן מועבר הdata (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

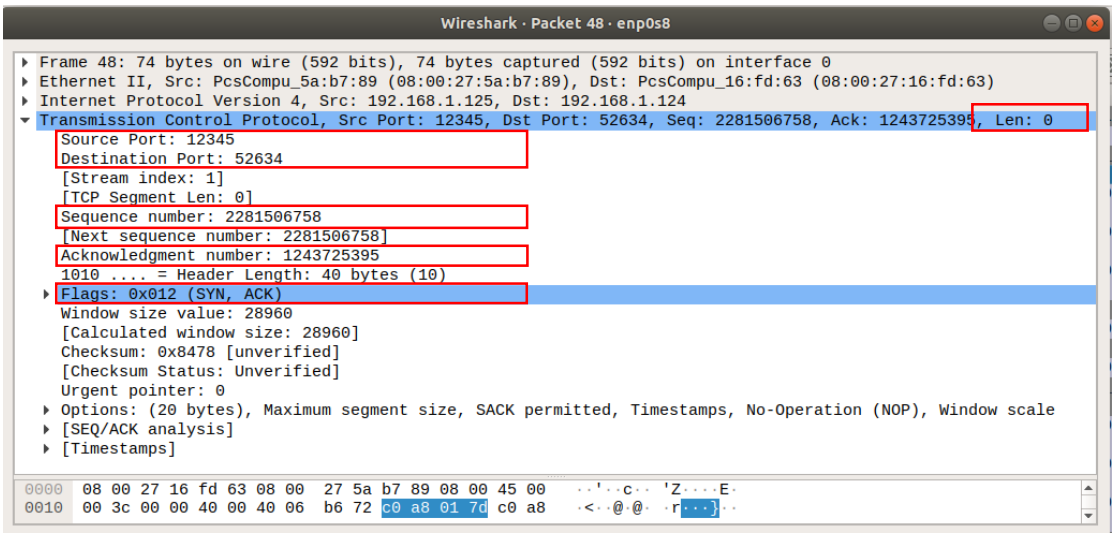
flags: SYN

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 הack 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

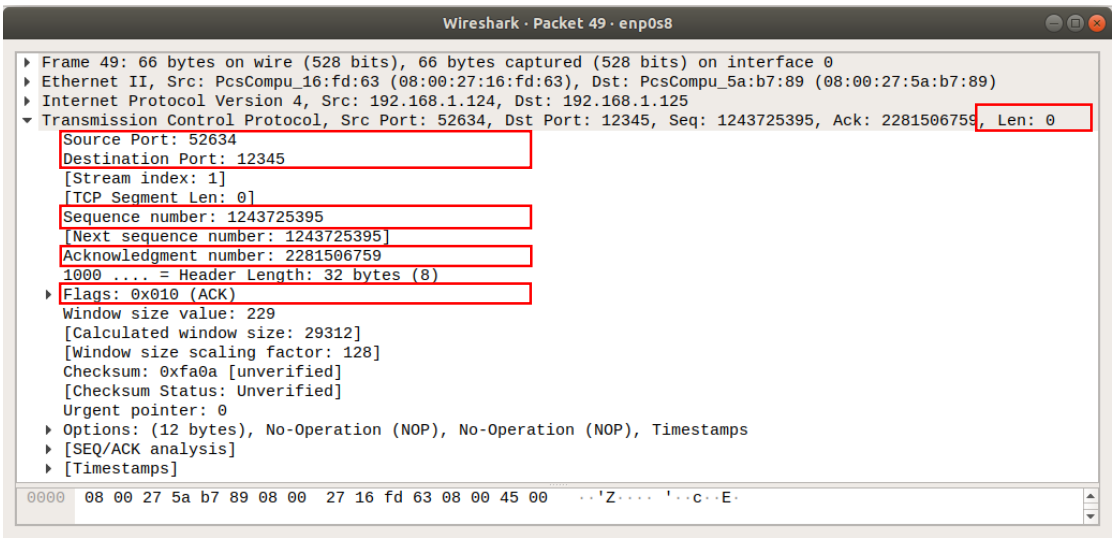
flags: SYN,ACK

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 הsyn ודגל הack דלוקים בheader, בכך השרת מאשר ללקוח כי קיבל את רצונו להתקשרות. לחבילה מצורף Sequence number התחלתי של השרת וה-ack number הינו הSequence number של הלקוח + 1.

חבילה מספר 49:



Sequence number : 1243725395

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

source port :52634

destination port: 12345

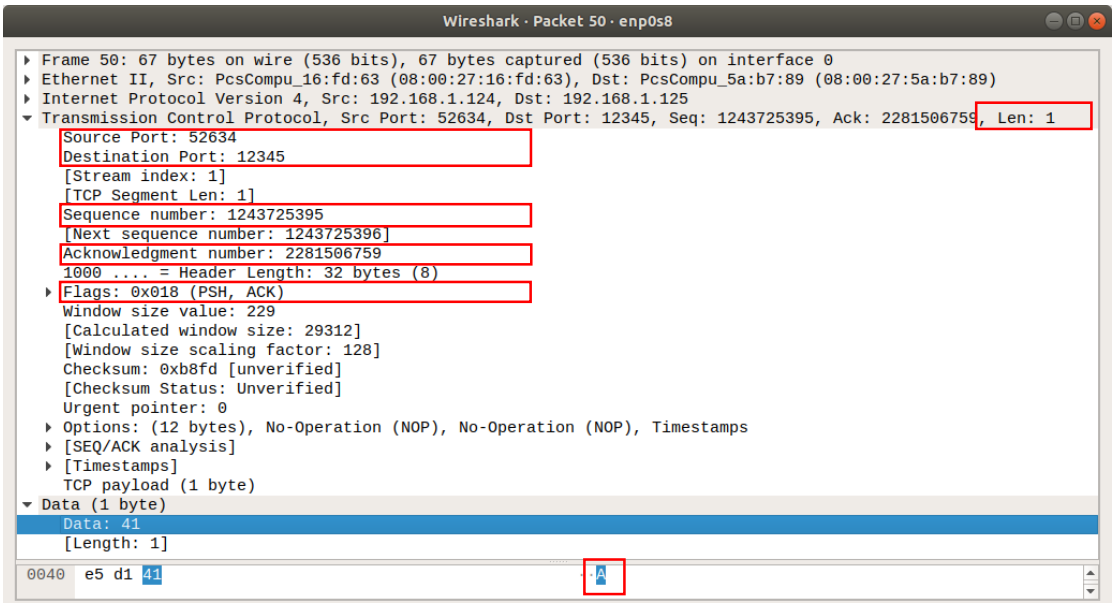
flags: ACK

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 הack דלוק ב header ובכך מסתיים תהליך הhandshake. 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

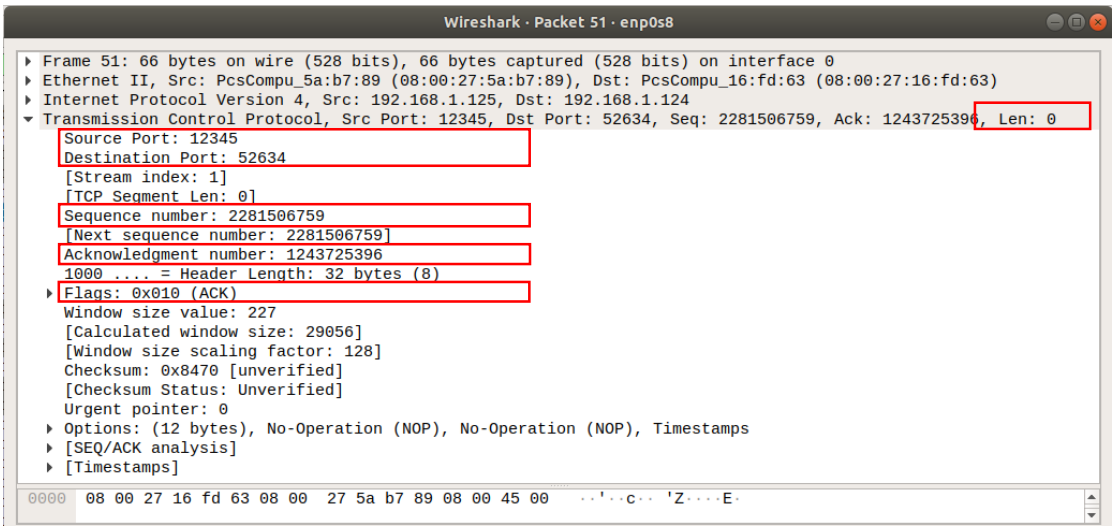
flags: PSH,ACK

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 והpsh 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

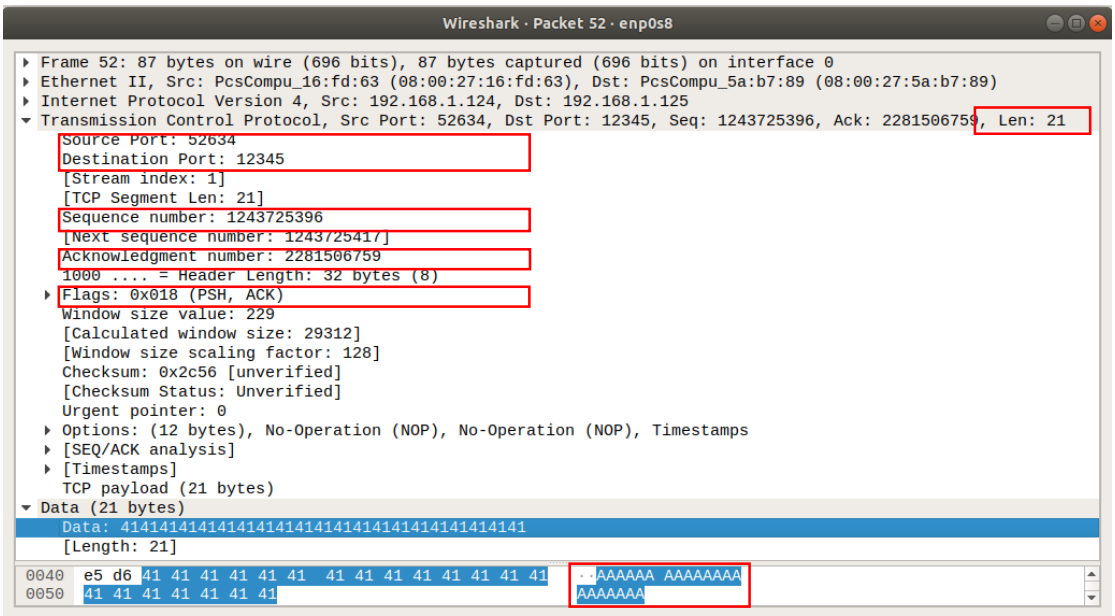
flags: ACK

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 הdata (1 B שהלקוח שלח בחבילה הקודמת).

חבילה מספר 52:



Sequence number : 1243725396 (prev seq.num)

ack number : 2281506759 (prev ack.num)

source port :52634

destination port: 12345

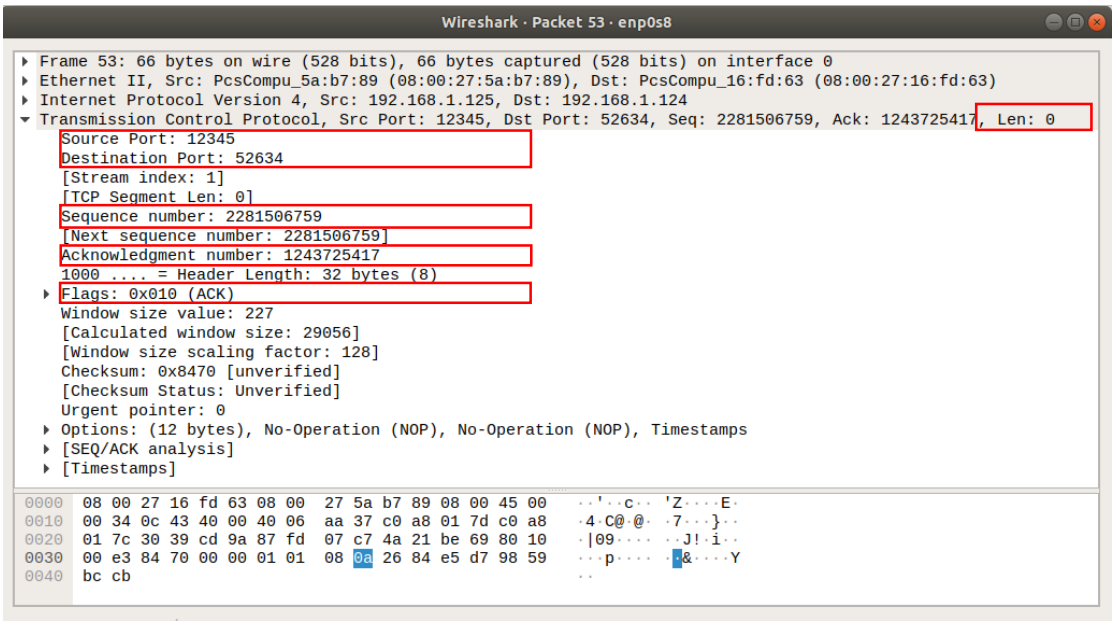
flags: PSH,ACK

length: 21

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 הack והpsh 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

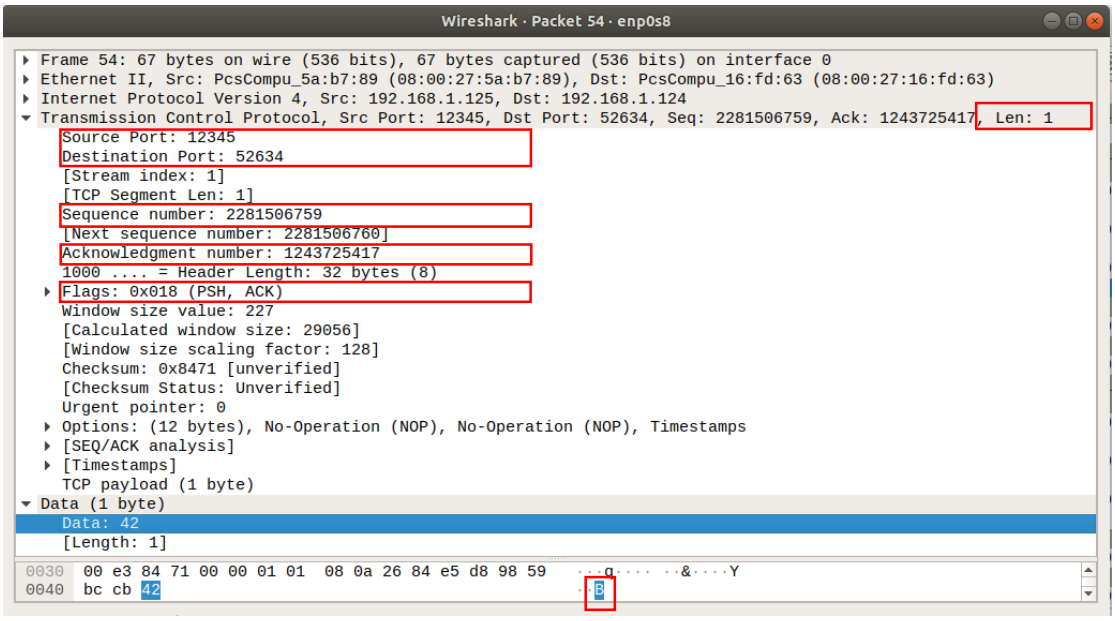
flags: ACK

length: 0

data :NO DATA

בחלק זה 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 הdata (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

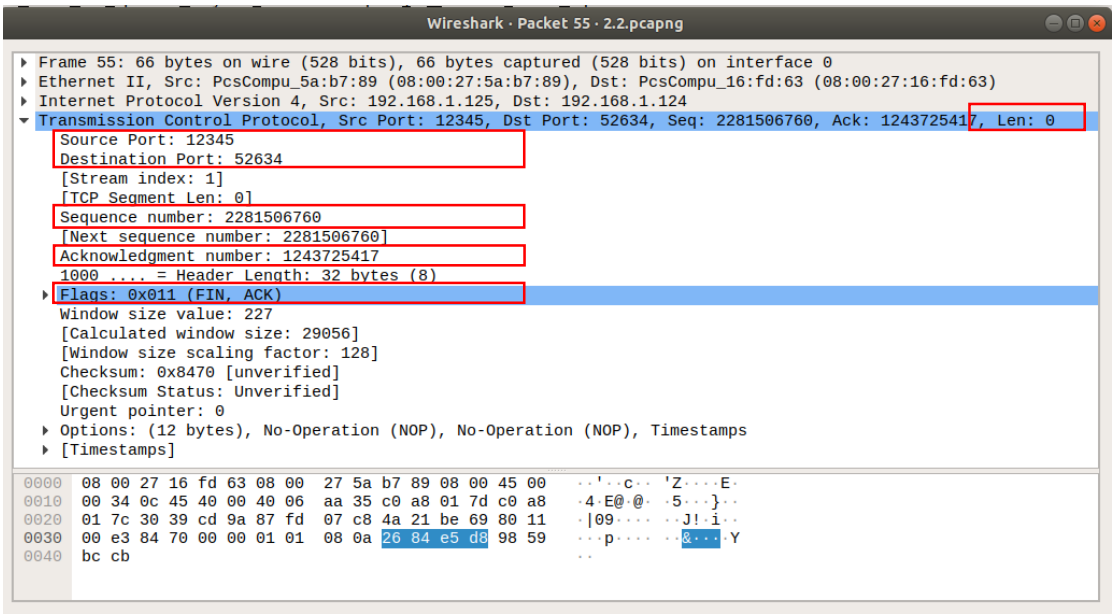
flags: PSH,ACK

length: 1

data :B

בחלק זה port מספר 12345(שרת) שולח חבילה ל port מספר 52634 (לקוח), ניתן לראות שחבילה זאת מכינה data ולכן length הוא 1. ניתן לראות כי דגלי psh וack 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

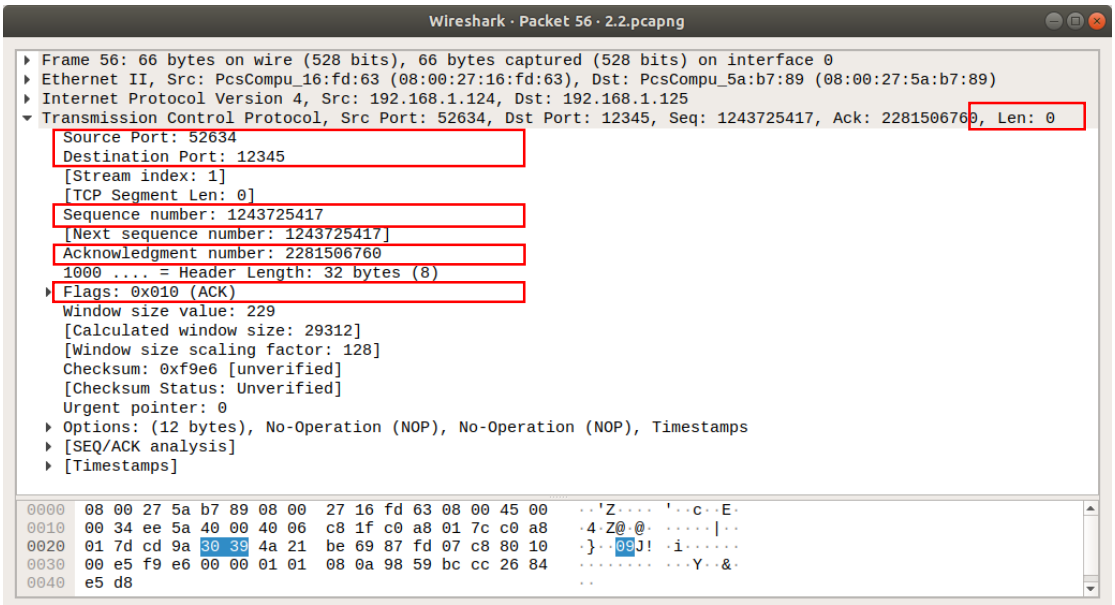
flags: FIN,ACK

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

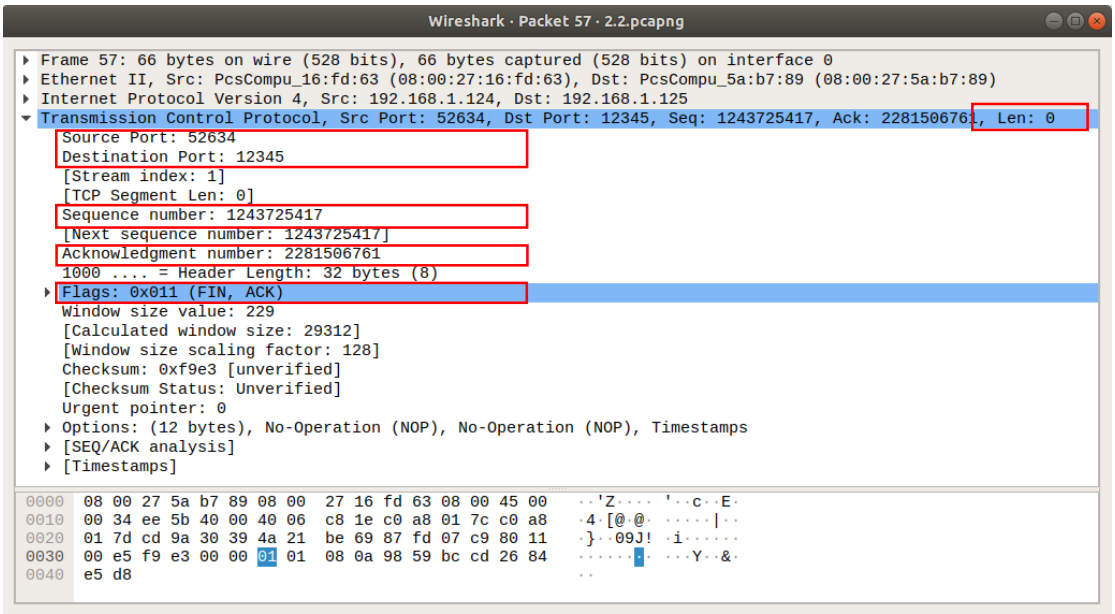
flags: ACK

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 הdata (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

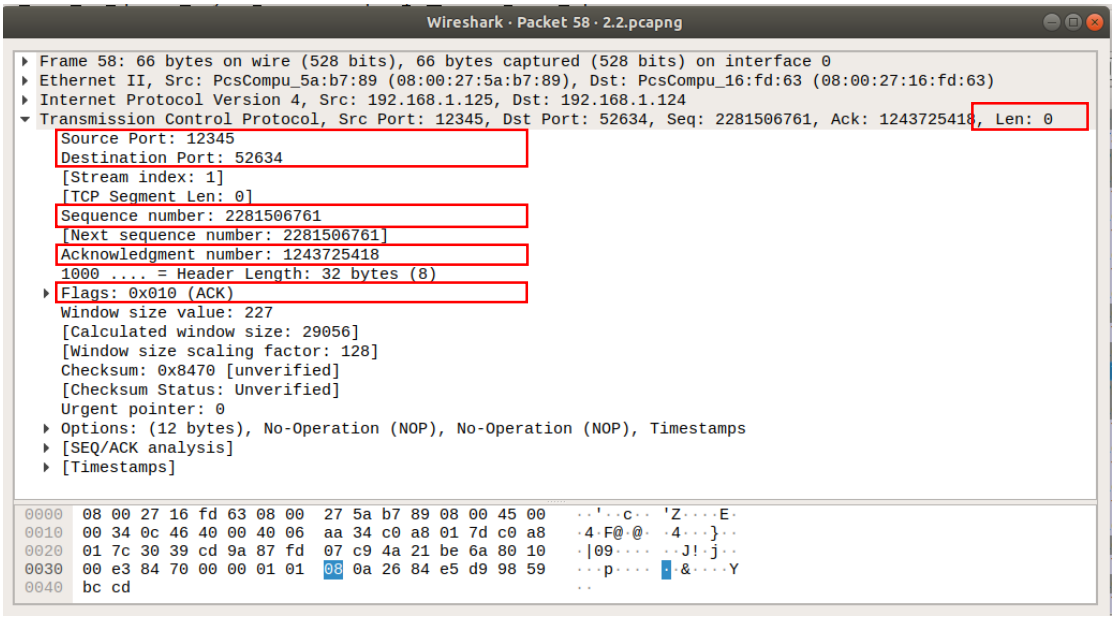
flags: ACK

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 ה teardown). 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 הack 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':

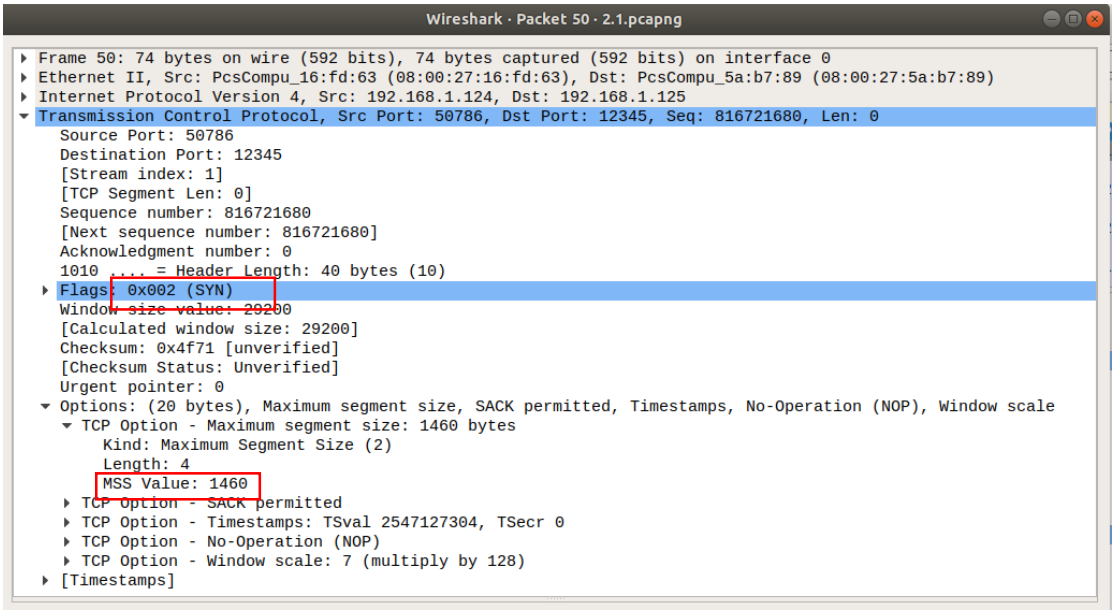
בסעיף זה אראה את ההבדלים בין הפרוטוקולים UDP,TCP שבאו לידי ביטוי בסעיפים א' וב'.

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.

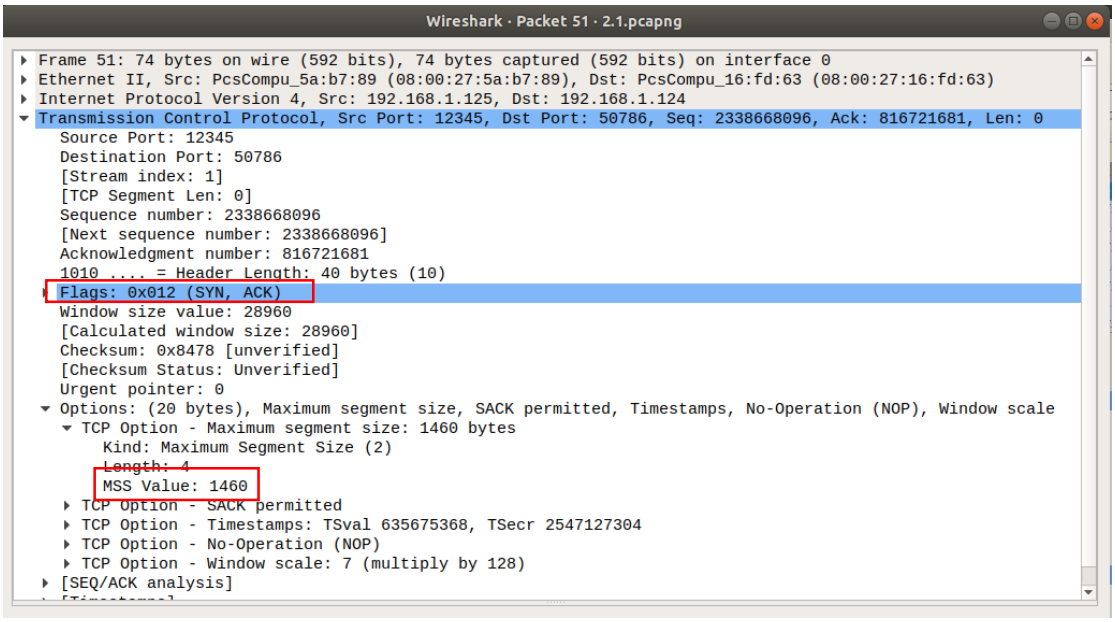
פרוטוקול TCP עושה שימוש בתהליך ה handshake, ובכך "קובעים" הלקוח והשרת את גודל המידע שאפשר להעביר, ובכך הם קובעים את ה- (maximum segment size)**MSS** . בכך הצדדים מסכימים על גודל סגמנט מקסימלי שניתן להעביר ומוודאים שאכן כל חבילה שנשלחת מאחד הצדדים תוכל להתקבל בצד השני. לאחר שנקבע ה MSS במידה ואחד הצדדים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

חבילה ראשונה בhandshake מלקוח לשרת:

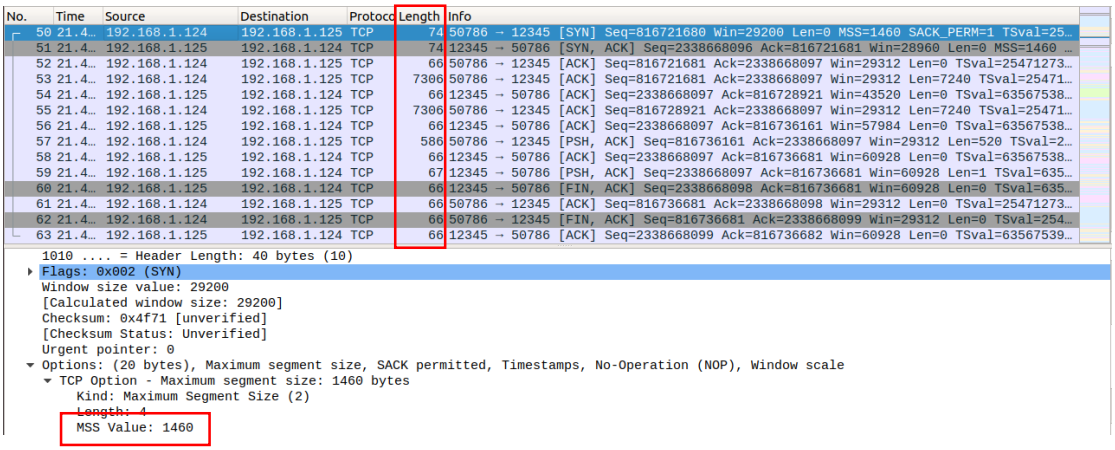


ניתן לראות כי הMSS הוא 1460, כלומר גודל הdata ללא הheaders חסום מלמעלה בB 1460

חבילה שניה בhandshake משרת ללקוח:



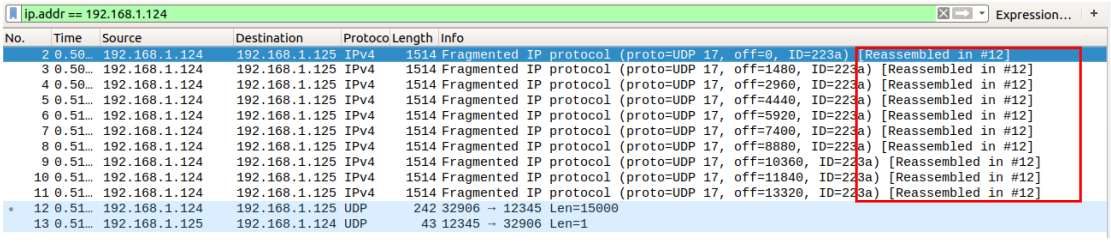
ואכן גודל הdata שנשלח לא עלה על 1460.



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

פרוטוקול UDP עושה שימוש בנתוני ה **MTU**(maximum transmission unit) של הרשת. בין הלקוח והשרת ישנם מספר ראוטרים ולכל חיבור שכזה קיים MTU. הלקוח אינו מודע לגודל הMTU שמחבר בין השרת לראוטר,בהתחשב בכך ובעובדה כי הפרוטוקול "אינו אמין" מעדיף הפרוטוקול לחלק את 15,000B של data לחתיכות קטנות יחסית(לעומת הTCP) תהליך זה נקרא **פרגמנטציה**. הפרגמנטציה מתבצעת בשכבת הרשת (לעומת TCP בשכבת התעבורה).

כעת אציג את חבילות בין הלקוח לשרת המקביל לסעיף א', אשים דגש על ההבדלים בין סוגי הפרוטוקולים.

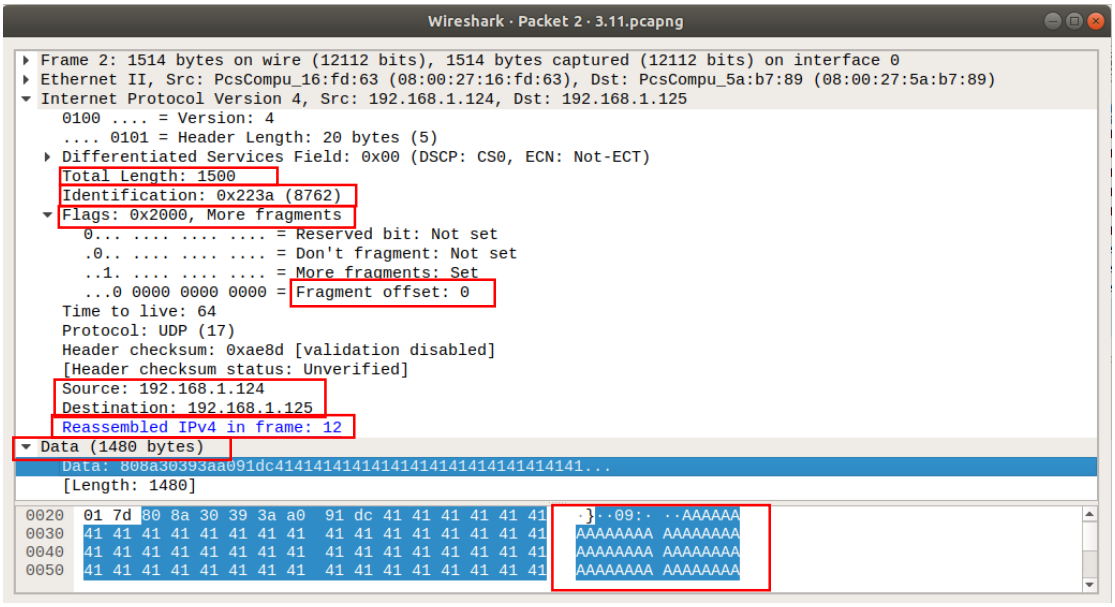


בצילום ניתן לראות כי חבילות 2-11 הן הפרגמנטציה של ה data ששלח הלקוח ועליהן כתוב כי הן יורכבו בחבילה מספר 12, חבילה 12 היא סיום קבלת הdata מהלקוח וגם איחוד החבילות , וחבילה 13 היא ה data ששלח השרת ללקוח.

שרת: כתובת IP : 192.168.1.125

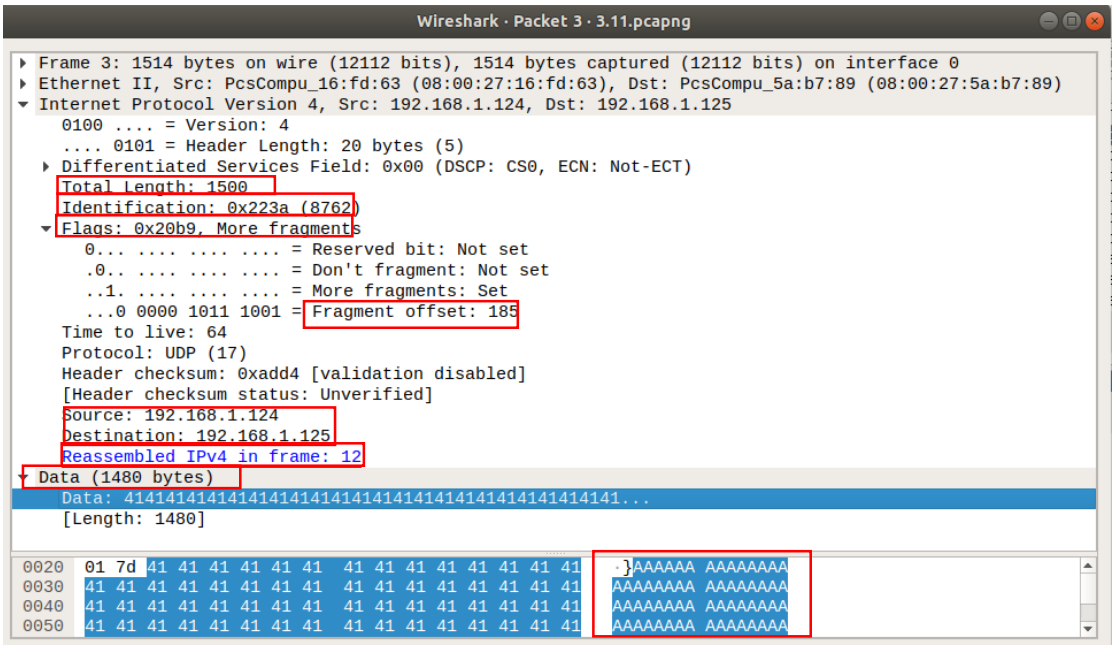
לקוח: כתובת IP: 192.168.1.124

חבילה מספר 2:



בחלק זה הלקוח שולח חבילה לשרת, ניתן לראות שחבילה זאת בעלת data ולכן הlength הוא 1480, פלוס 20B של header IP, 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 הראשונה שנשלחה, הoffset הוא 0/8=0. ב wireshark ניתן לראות כי החבילות יורכבו בחבילה מספר 12.

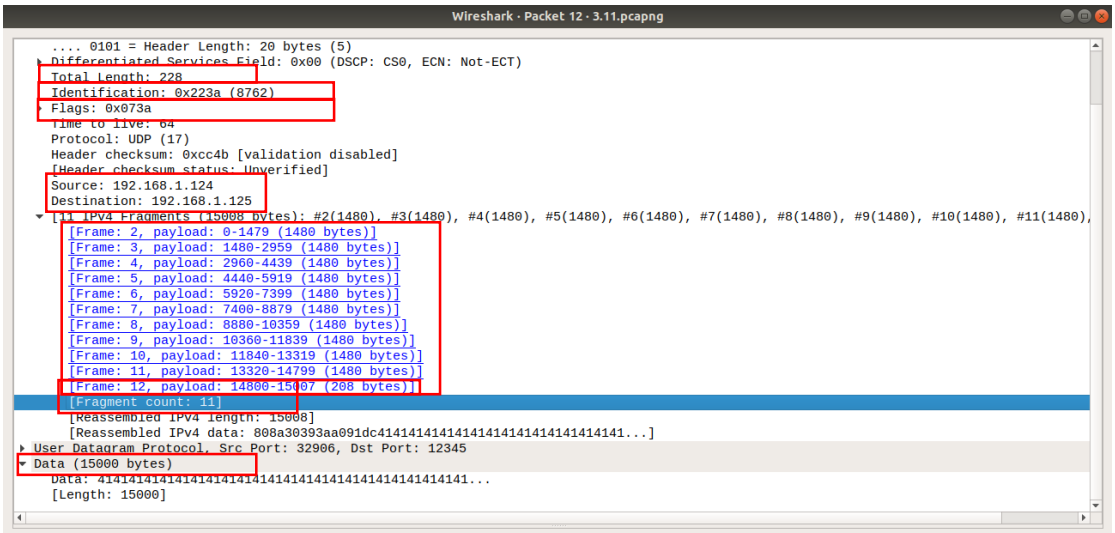
חבילה מספר 3:



בחלק זה הלקוח שולח חבילה לשרת, ניתן לראות שחבילה זאת בעלת data ולכן lengthה הוא 1480, פלוס 20B של header IP, 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 ב1480/8=185

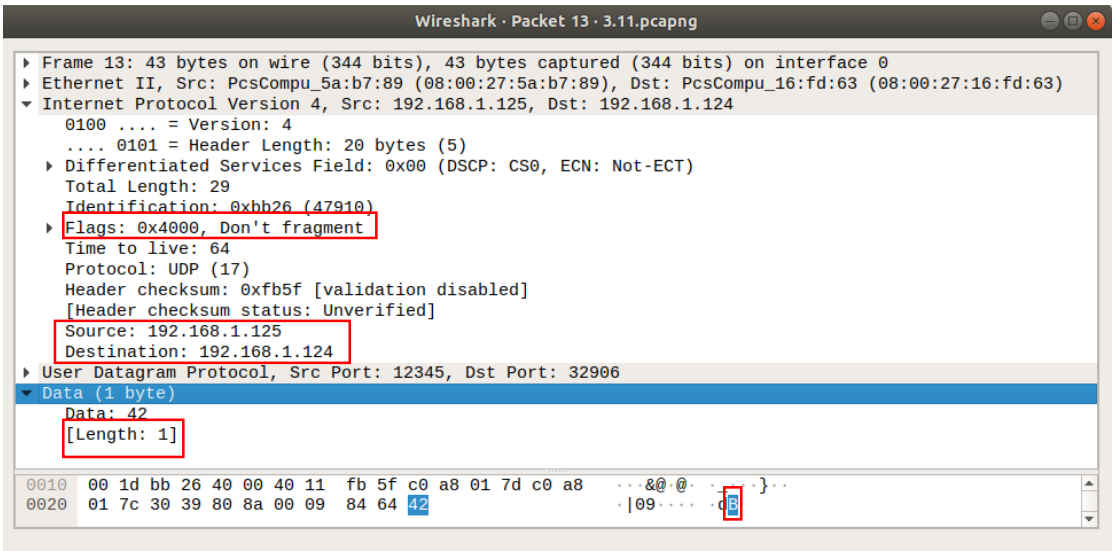
חבילה מספר 12:



בחלק זה הלקוח שולח את חבילת הdata האחרונה לשרת, ניתן לראות שחבילה זאת בעלת data ולכן lengthה 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 בheader, בכך הלקוח מסמן לשרת כי לא

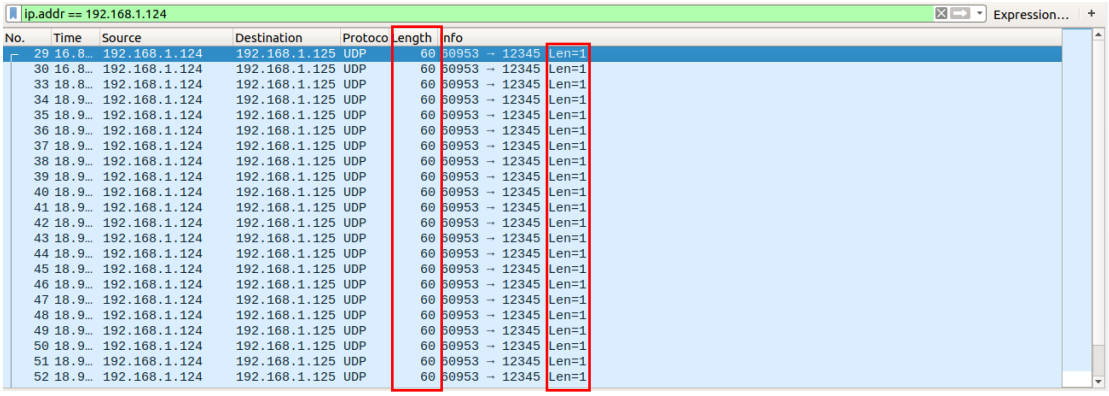
קיימות עוד חבילות עם data בדרך, ושהוא יכול להתחיל להרכיב את החבילות שהתקבלו עד כה.
כיוון שזוהי החבילה האחרונה שנשלחה, הoffset הוא 15,000/8=1850. ב wireshark ניתן לראות את פירוט החבילות בעלות המזהה המשותף שהתקבלו עד עתה.

חבילה מספר 13:



בחלק זה השרת שולח את התו B ללקוח, גודל ה data הינו 1 וכיוון שהוא קטן מהMTU אין צורך בלפרק את החבילה לפרגמנטים ולכן ודגל הDF אכן דלוק בheader 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 הdata. 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 headers IP,UDP וסה"כ גודל החבילה הוא 60B על כל 1B של data. לכן על מנת להעביר את ה data שהלקוח שלח נעשה שימוש ב60B * 22 = 1,320B.
בפרוטוקול TCP המבצע סגמנטציה מסעיף ב' נעשה שימוש ב266B כולל הack מהשרת. ניתן בקלות לראות כי הoverhead של הheaders בפרוטוקול UDP מגדיל משמעותית את סה"כ גודל החבילות שנשלחו.

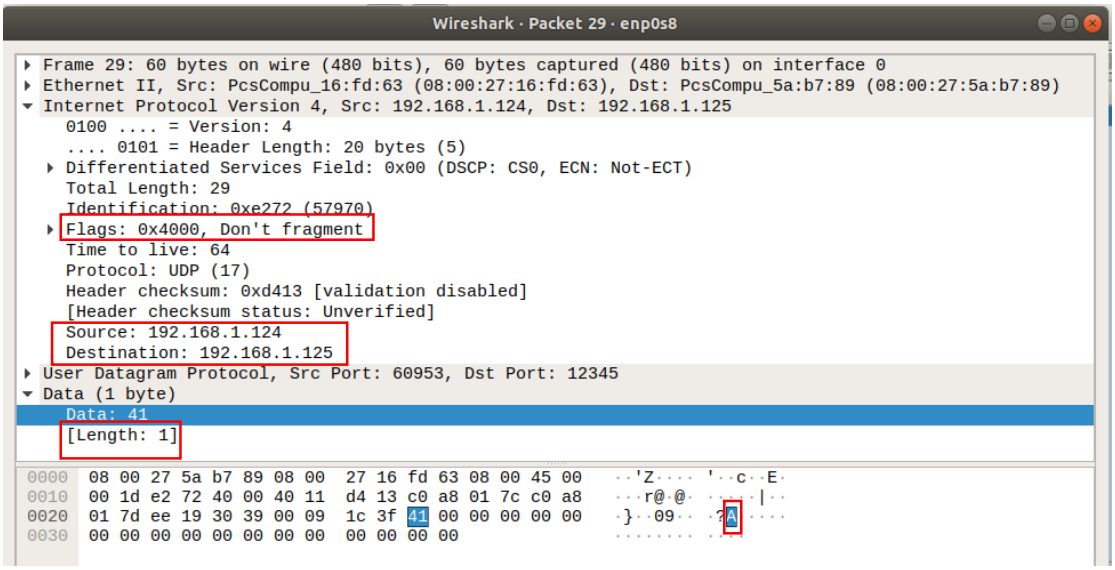


ניתן לראות כי חבילות 29-51 הן מהלקוח לשרת, וחבילה 52 היא מהשרת ללקוח.

שרת: כתובת IP : 192.168.1.125

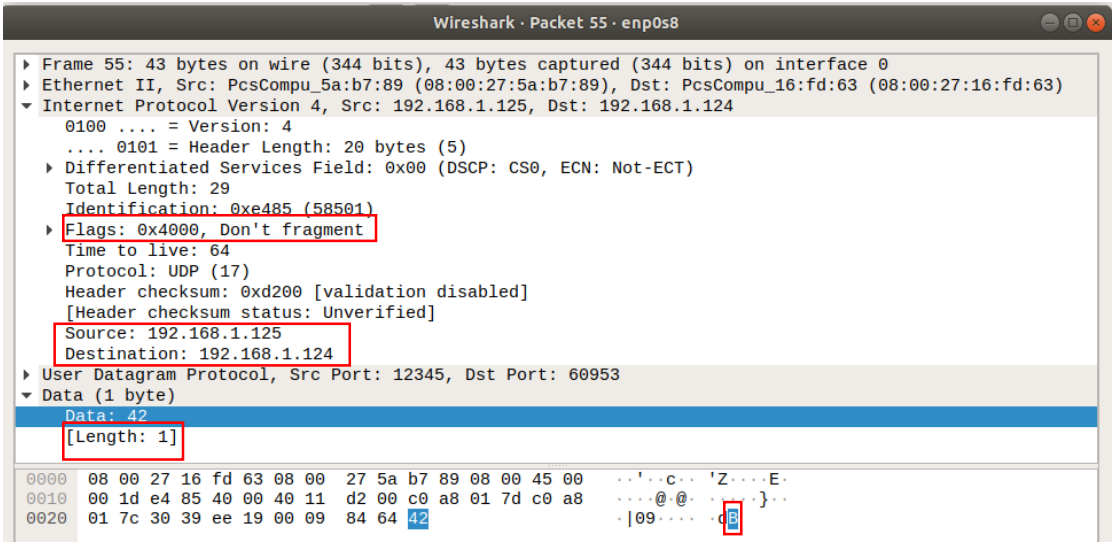
לקוח: כתובת IP: 192.168.1.124

חבילה מספר 29:



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

חבילה מספר 55:



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