

רשותות תקשורת – תרגיל מס' 2

## שאלה 1

.N

תהליך handshake עם הלוקה הראשון:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	76	58870 → 12345 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=1980196353 TSecr=0 WS...
2	0.000018	127.0.0.1	127.0.0.1	TCP	76	12345 → 58870 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=1980196353...
3	0.000033	127.0.0.1	127.0.0.1	TCP	68	58870 → 12345 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=1980196353 TSecr=1980196353

ניתן לראות כי הפורט בו הלקוח הראשון משתמש הוא פорт מס' 58870.

**כעת נפרט את כל התהליין:**

Wireshark - Packet 1 · ex2part1Client1.pcap

Frame 1: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)  
Linux cooked capture  
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
Transmission Control Protocol, Src Port: 58870, Dst Port: 12345, Seq: 0, Len: 0

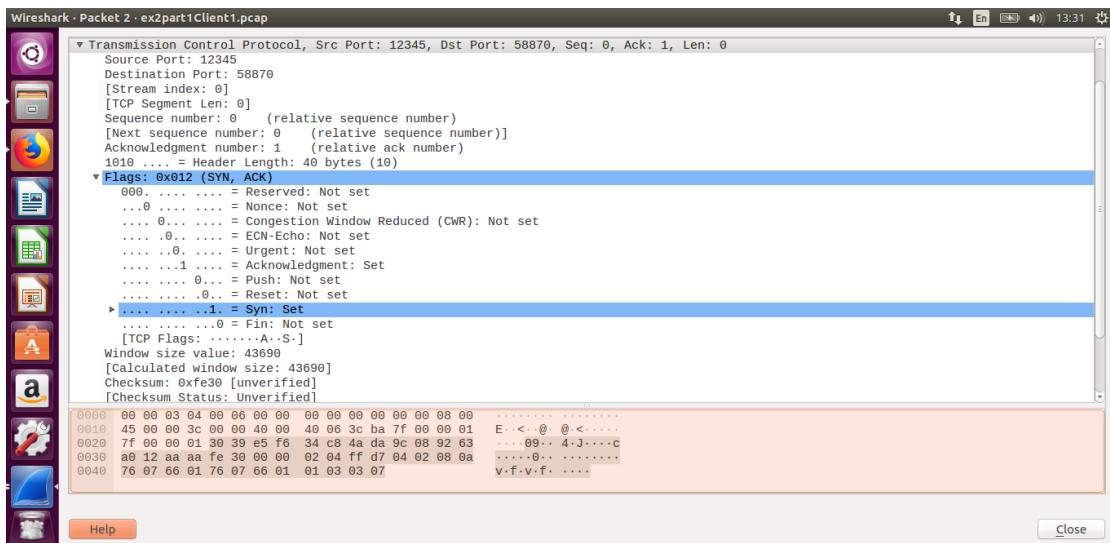
Source Port: 58870  
Destination Port: 12345  
[Stream index: 0]  
[TCP Segment Len: 0]  
Sequence number: 0 (relative sequence number)  
[Next sequence number: 0 (relative sequence number)]  
Acknowledgment number: 0  
1010 .... = Header Length: 40 bytes (10)  
Flags: 0x002 (SYN)  
000. .... = Reserved: Not set  
....0. .... = Nonce: Not set  
....0. .... = Congestion Window Reduced (CWR): Not set  
....0. .... = ECN-Echo: Not set  
....0. .... = Urgent: Not set  
....0. .... = Acknowledgment: Not set  
....0. .... = Push: Not set  
....0. .... = Reset: Not set  
....0. .... = Syn: Set  
....0. .... = Fin: Not set  
[TCP Flags: ....S.]  
Window size value: 43690  
[Calculated window size: 43690]

0000 00 00 03 04 00 06 00 00 00 00 00 00 00 00 08 00  
0010 45 00 00 3c 34 bf 40 00 40 06 07 fb 7f 00 00 01 E-<4 @ @ . . . .  
0020 7f 00 00 01 e5 f6 30 39 9c 08 92 62 00 00 00 00 . . . . 09 . . b . . .  
0030 a0 02 aa aa fe 30 00 00 02 04 ff d7 04 02 08 0a . . . . 0 . . . .  
0040 76 07 66 00 00 00 00 00 01 03 03 07 v f . . . .

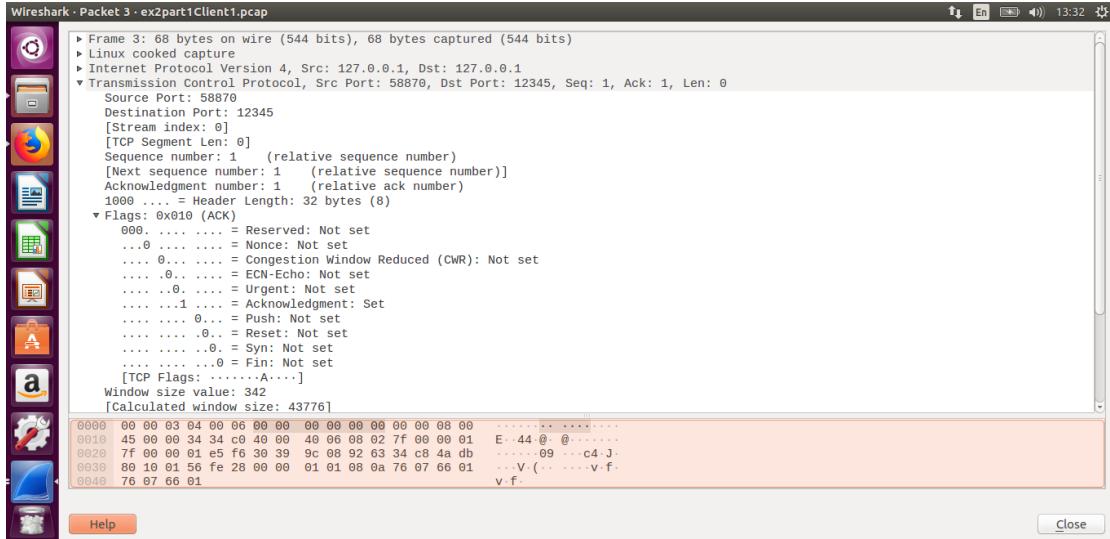
No.: 1 · Time: 0.000000 · Source: 127.0.0.1 · Destination: 127.0.0.1 · Protocol: TCP · Length: 76 · Info: 58870 → 12345 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK\_PERM=1 TStamp=1980196353 TSectr=0 WS=128

ניתן לראות כי ההפניה הראשונה ב握手 handshake ששולחת היא מהלкова לסרבר לצורך הקמת שיחה.

ניתן לאות כי הסרבר משתמש בפורט 58870 והלkoח בפורט 12345, כמו כן ניתן לראות בדגלים כי דגל החסס דולקן.



כעת השרת שולח הודעה חוזרת ללקוח שמצוירה ללקוח למעשה שהוא פניו , והם יכולים להתחיל לנחל שיחה.  
שוב ניתן לראות כי דגל החסус דלוק.

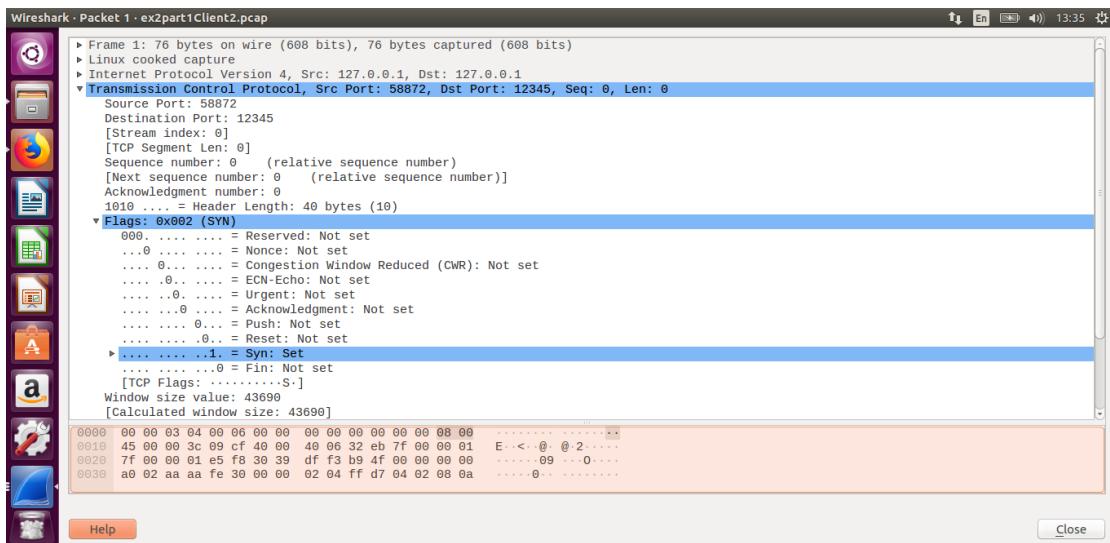


כעת הלקוח מקבל את ההודעה מהשרת , וכעת הוא יודע שהוא יכול להתחיל במטרת השיחה מבחןינו  
שאצלנו זה שליחת הרודעות hello ו world .

#### כעת נראה את תהליך handshake עם הלקוח השני:

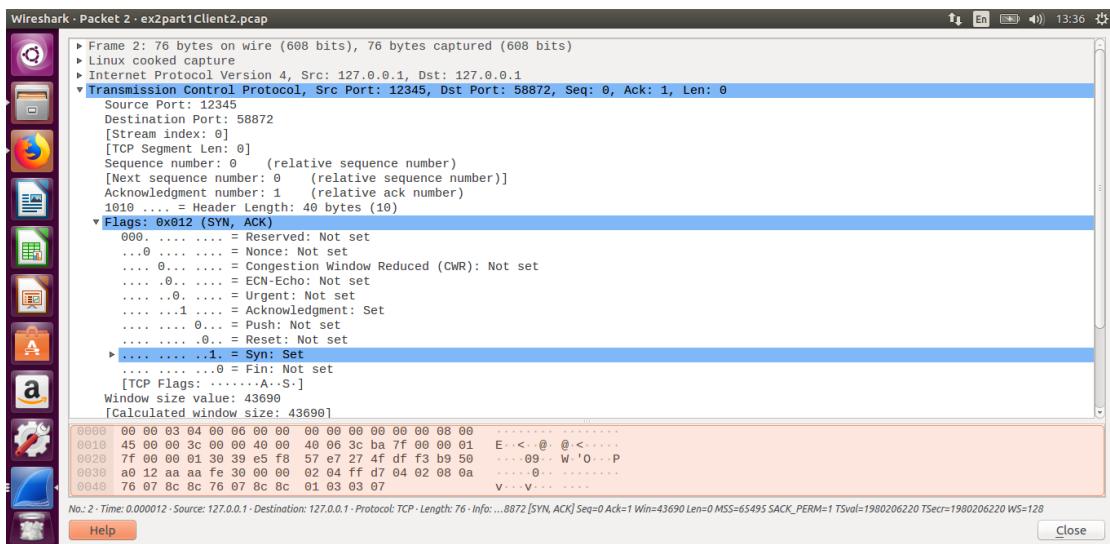
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	76	58872 → 12345 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1..
2	0.000012	127.0.0.1	127.0.0.1	TCP	76	12345 → 58872 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 ..
3	0.000025	127.0.0.1	127.0.0.1	TCP	68	58872 → 12345 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=198020622..

ניתן לראות כי הפורט בו הלקוח הראשון משתמש הוא פורט מס' 58872  
כעת נפרט את כל התהילן:



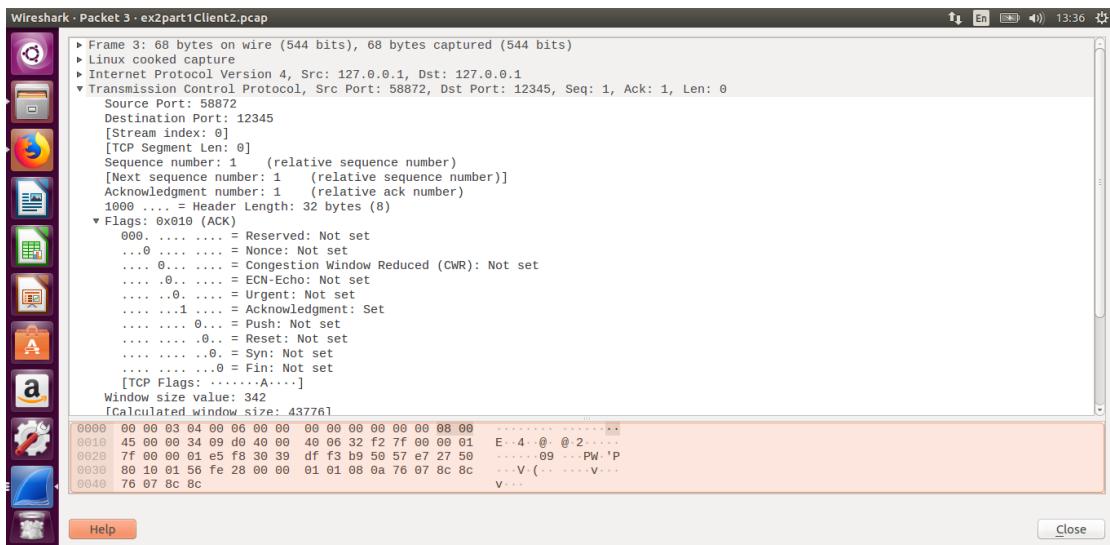
ניתן לראות כי החבילה הראשונה ב שנסלחת היא מהלך לסרבר לצורך הקמת שיחה.

ניתן לראות כי השרת משתמש בפורט 12345 והלך בפורט 58872 , כמו כן ניתן לראות בדרכים כי דגל החסוד לא נקבע.



כעת השרת שולח הודעה חוזרת ללקוח שמצוירה לעמשה שהוא פניו , והם יכולים להתחיל לנהל שיחה.

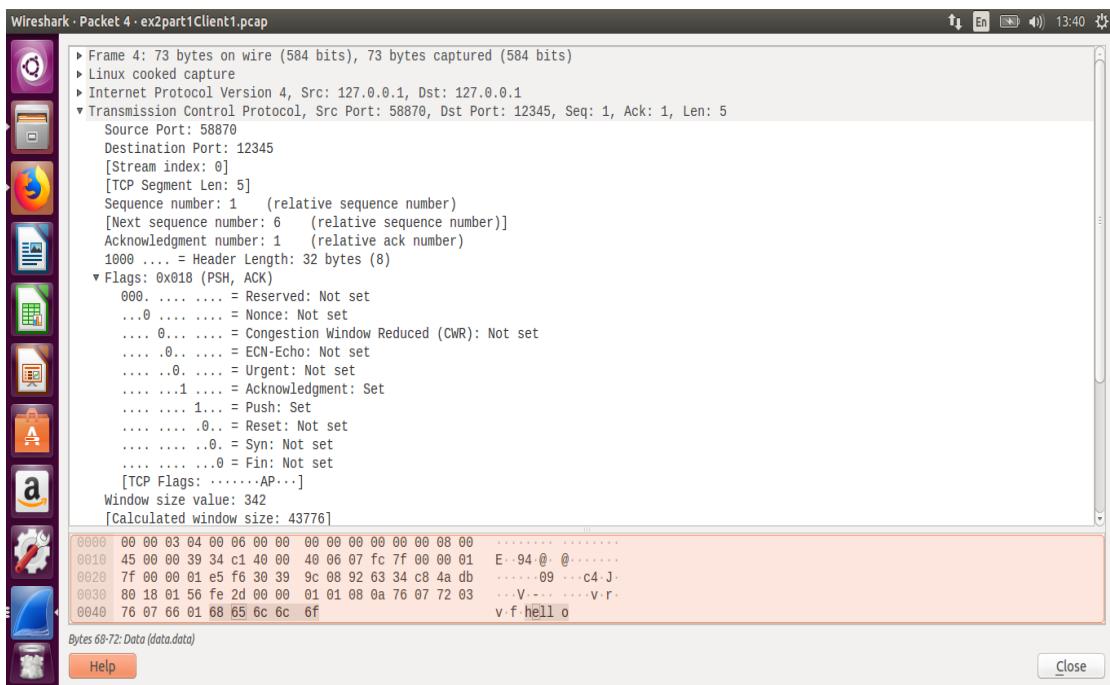
שוב ניתן לראות כי דגל החסוד נקבע.



כעת הלקוח מקבל את ההודעה מהсер버 , וכעת הוא יודע שהוא יכול להתחל במתרת השיחה מבחןינו שאצלנו זה שליחת ההודעות hello ו-world .

#### סעיף ב':

כעת נפרט את החביבות שלוקח הראשון שלוח לשרת והשרת החזיר להם בהקשר של ההודעות hello ו-world . (וכמובן שבסוף הסעיף נציג את ההבדלים שיש בין הלקוח הראשון לשני מכיוון שהורוב הדברים הם זמינים ).



ניתן לראות כי הלקוח שלוח לסרבר כאן את הודעת הסיסמה hello .

פרטים על שכבת התעבורה בpacket הנ"ל:

**58870 :Source port**

**12345 :Dest port**

ניתן לראות שהdagלים ACK ו- PSH דלוקים:

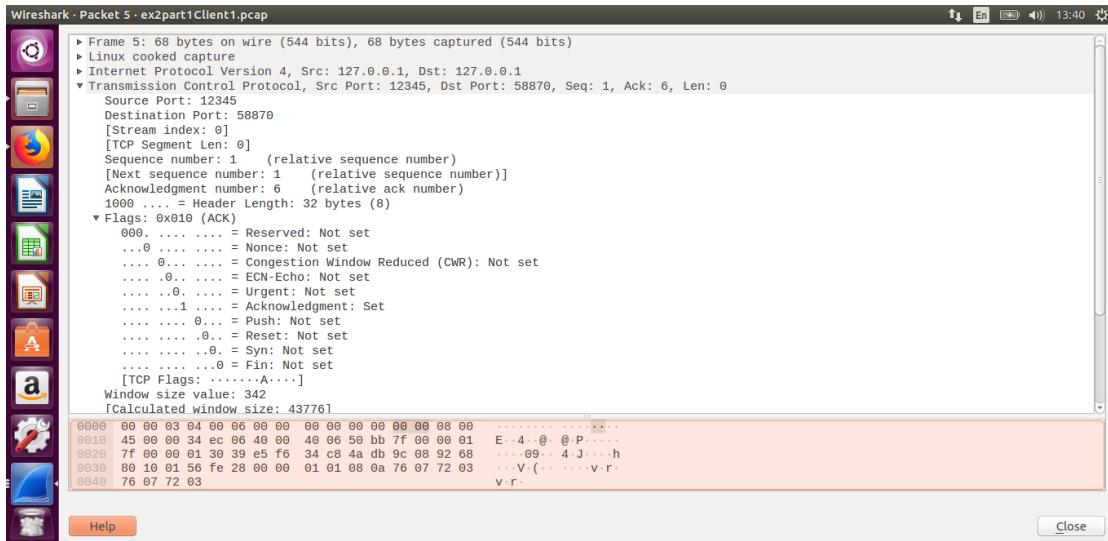
**Sequence number רלוּטִיבִי : 1**

**הערך האמיתי של ה** Sequence number **2617807459**

## Ack number רלטיבי: 1

**הערך האמיתי של Ack number**

## 5 :header גודל



כעת הסרבר שולח ללקוח הודעה ack על ה-hello.

**פרטים על שכבת התעבורה בpacket הנ"ל:**

**הסרבר שלנו** :Source port 12345

**הסבר שלנו** 58870 :Dest port

**ניתן לראות שהדגל ACK דלוק.**

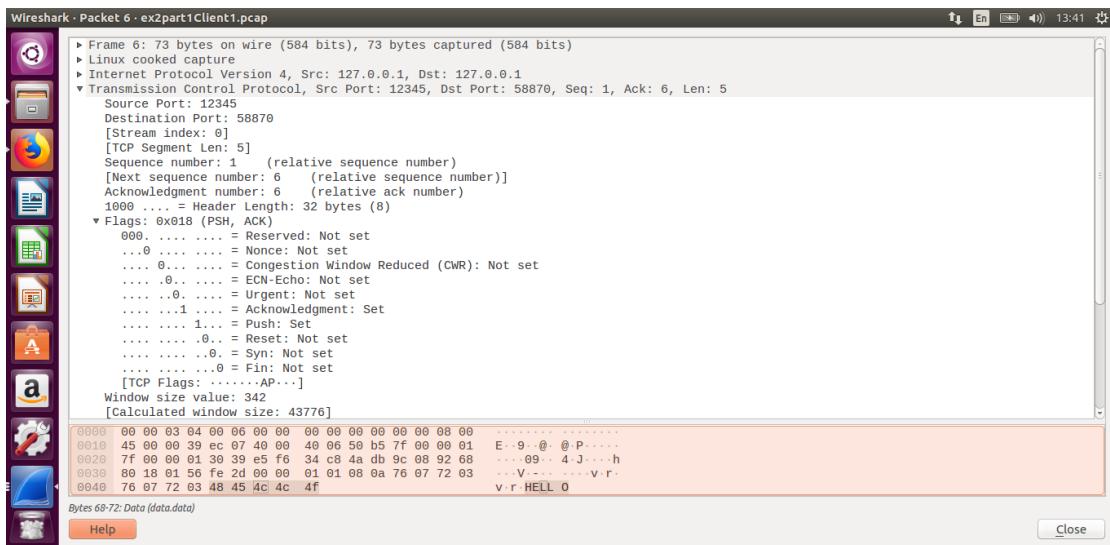
1 רלטיבי Sequence number

**הערך האמיתי של ה** Sequence number **885541595**

## **6 רלטיבי Ack number**

**הערך האמיתי של הnumer : Ack number**

## 5 :header גודל



**cutet\_the\_server\_refuses\_to\_reject\_hello.h**

**פרטים על שכבת התעבורה בpacket הנ"ל:**

**הסרבר שלנו** 12345 :Source port

**58870 (הסבר שלנו) :Dest port**

ניתן לראות שדגלי ה-ACK וה-PSH דלוקים.

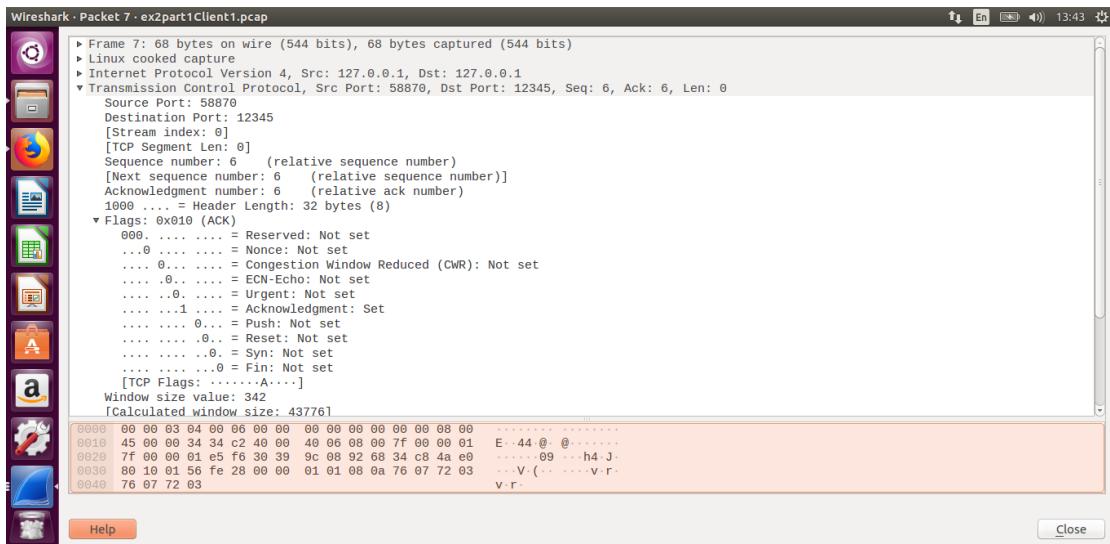
## מספר סדרה : רלטיבי 1

הערך האמיתי של ה-Sequence number

## **אך נומר רלטיבי: 6**

**הערך האמיתי של ה-**

## 5 :header גודל



כעת הלקוח שולח ack על קבלת הודעה hello מהסרבר.

**פרטים על שכבת התעבורה בpacket הנ"ל:**

**58870 :Source port (הפורט של הלקוח הראשוני)**

12345 :Dest port (הסברר שלנו)

ניתן לראות שדגל ה- ACK דלוק:

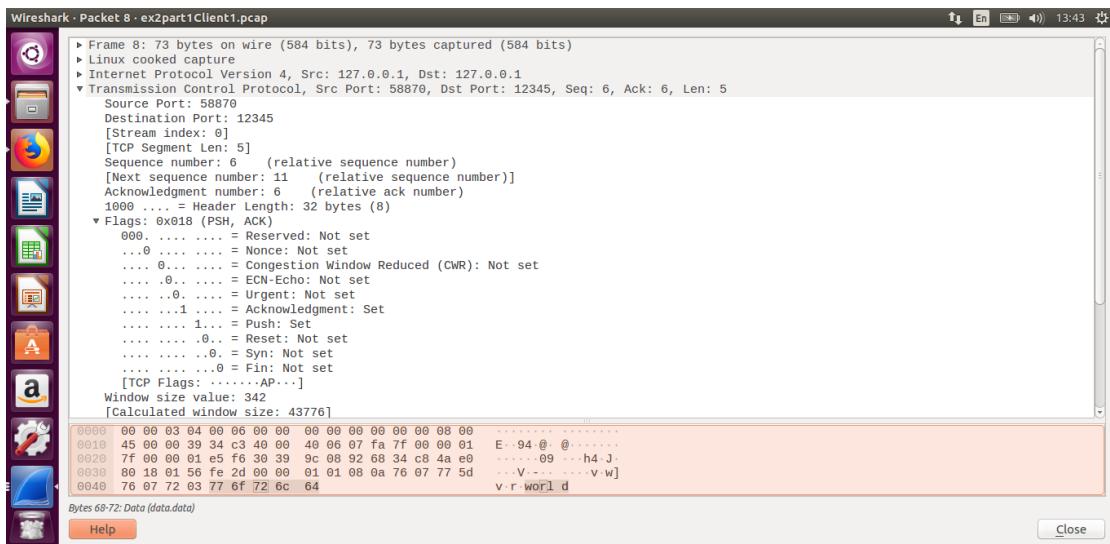
Sequence number רלוֹטִיבָּי : 6

הערך האמיתי של ה Sequence number 2617807464

Ack number רלוֹטִיבָּי: 6

הערך האמיתי של ה Ack number 885541600

גודל header 5



כעת הלקוח שורת לשורת את הודעת world.

פרטים על שכבת התעבורה בpacket הנ"ל:

58870 :Source port (ה포רט של הלקוח הראשון)

12345 :Dest port (הסברר שלנו)

ניתן לראות שהדגלים ACK ו- PSH דלוקים:

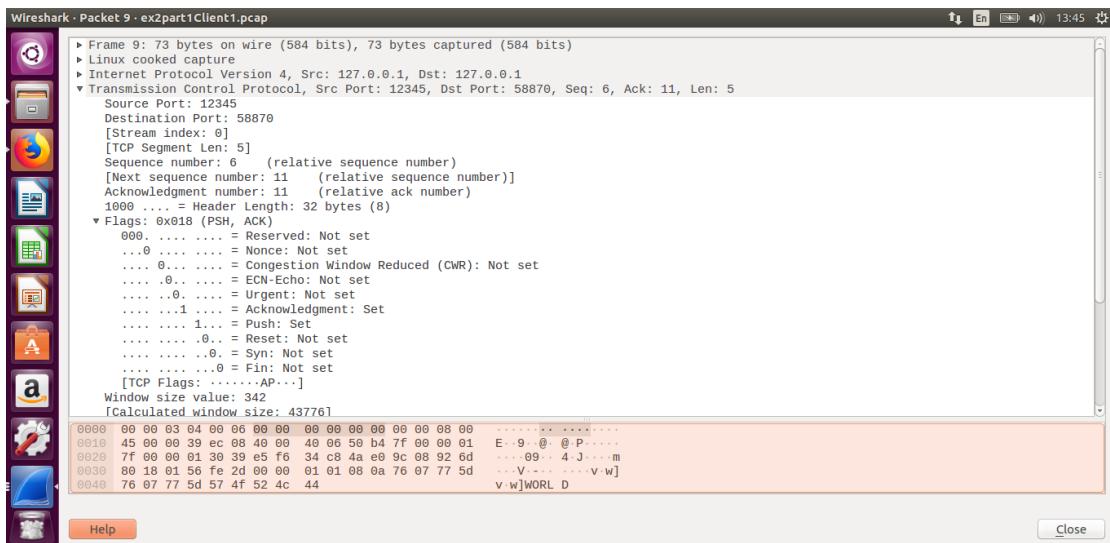
Sequence number רלוֹטִיבָּי : 6

הערך האמיתי של ה Sequence number 2617807464

Ack number רלוֹטִיבָּי: 6

הערך האמיתי של ה Ack number 885541600

גודל header 5



כעת הסרבר בפקטה אחת שולח גם הודעת ACK על ה world וגם הודעת PSH על ה world.(שולח ללקוח את הודעת ה world).

פרטים על שכבת התעבורה בpacket הנ"ל:

**12345**: Source port

**58870**: Dest port

ניתן לראות שדגלי ה-ACK וה-PSH דלוקים.

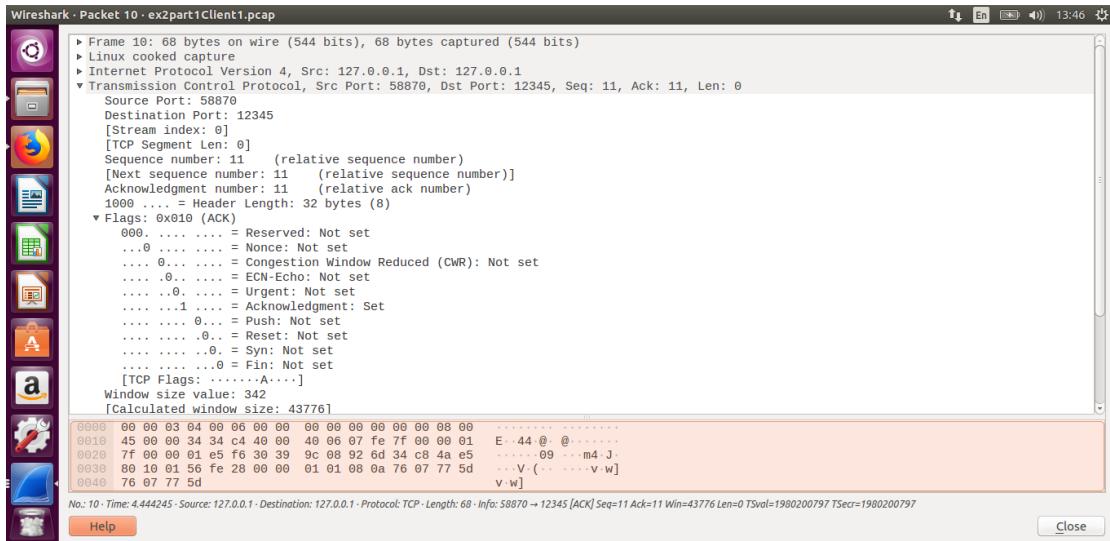
**Sequence number**

**885541600** : Sequence number

**11** Ack number

**2617807469** : Ack number

**5** header length



כעת הלקוח מוחדר לשרת ACK על הודעת hello ובירך מסת"ימת התקשרות מביניהם בהעברת ההודעות בין world hello לשרת.

פרטים על שכבה התעבורה בpacket הנ"ל:

ה포רט של הלוקוֹת הראשון (Source port) : **58870**

הסְרֶבֶר שלוֹנוֹ (Dest port) : **12345**

ניתן לראות שהדגל ACK דלוק.

Sequence number רלוֹטִיבֵי : **11**

הערך האמִתִי של Sequence number : **2617807469**

Ack number רלוֹטִיבֵי : **11**

הערך האמִתִי של Ack number : **885541605**

גודל header : **5**

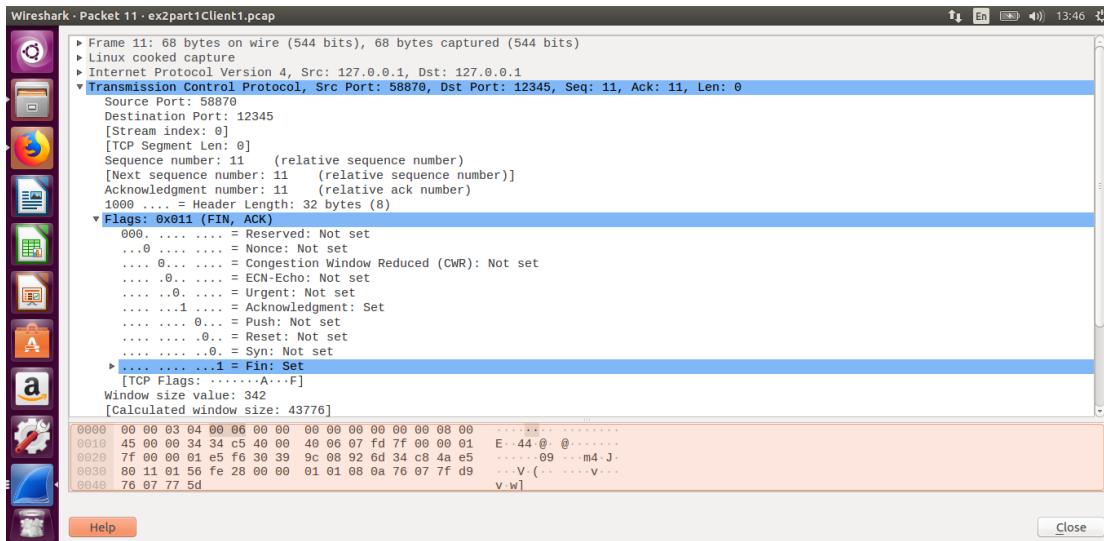
עבור הלוקוֹת השנִי קיבלנו את אותו נתונים בשכבה התעבורה רק ה port של הקלינט השתנה מ- **58870** ל- **58872**

כמו כן sequence number ההתחלתי (הRELATIVE = 0) של הקלינט (הלא רלוֹטִיבֵי) הוא: **3757291855**

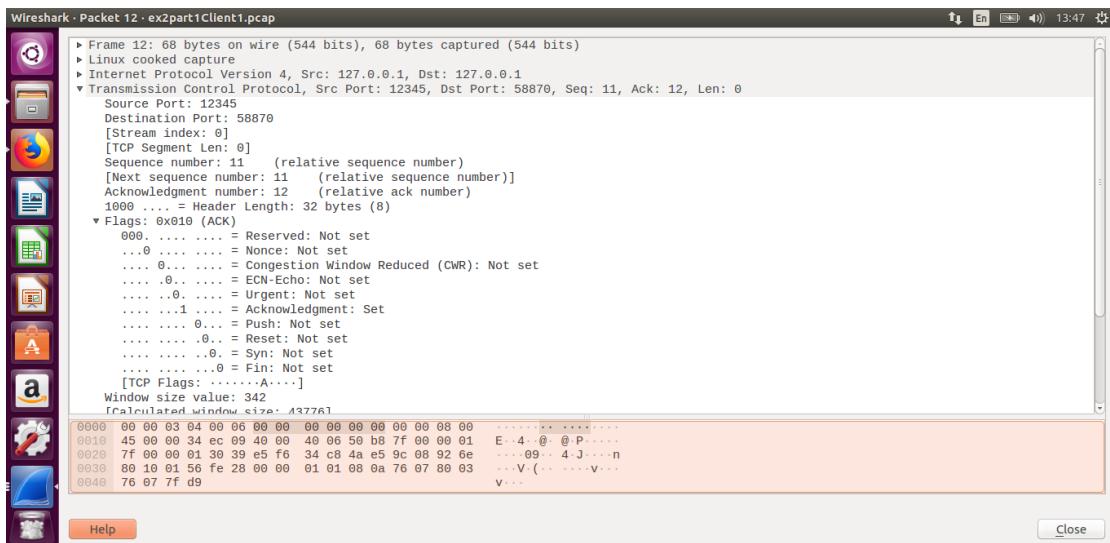
כמו כן sequence number ההתחלתי (הRELATIVE = 0) של הסְרֶבֶר (הלא רלוֹטִיבֵי) הוא: **1474766671**

#### סעיף ג'

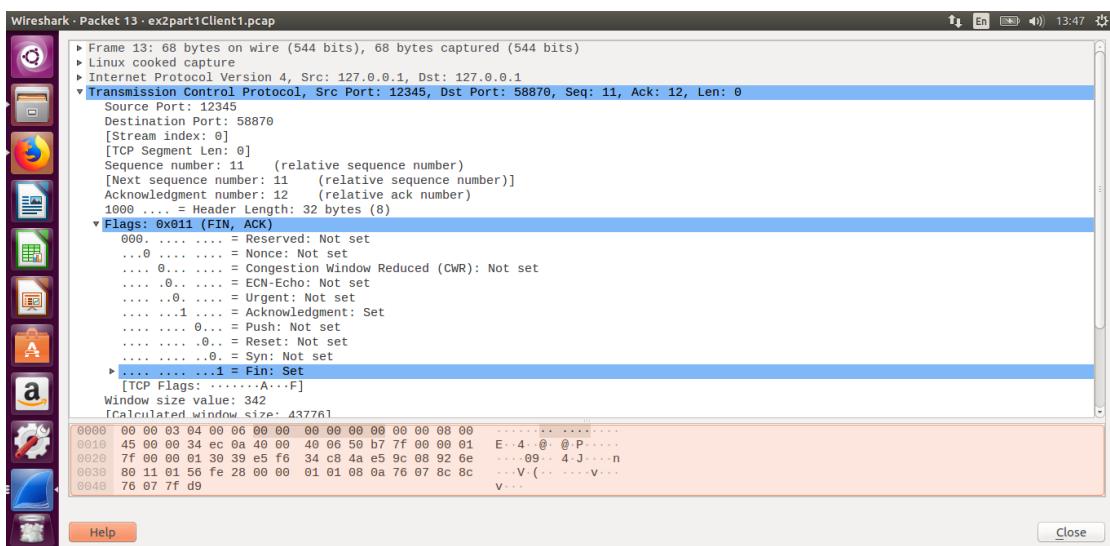
cut נדגים את תהליך teardown בין הלוקוֹת הראשון לשרת.



כאן ניתן לראות כי הלוקוֹת שולח בקשה סגירת שיחה לשרת, כמו כן דגל ack וה-fin דלוקים.



כאן ניתן לראות כי השרת קיבל את בקשת הסגירה ושלח ACK ללקוח.



וכאן השרת שולח את בקשת הסגירה ללקוח, ניתן לראות כי דגלי ACK וfin דלוקים.

אותו דבר קורה עם הלקוח השני, וכך למעשה מtbodyה תהליך teardown בסעיף הזה.

## שאלה 2:

סעיף א':

נתבונן בחבילה שהלכו שלח לשרת:

The screenshot shows the Wireshark interface with two windows. The top window displays a list of 10 captured TCP packets. The bottom window is a detailed view of the fourth packet (No. 4), which is highlighted in orange. The details pane shows the packet's structure: Source: 192.168.42.12, Destination: 192.168.42.203, Protocol: TCP, Length: 62 bytes, Info: 14656 → 12345 [ACK] Seq=3900412207 Ack=2697510411 Win=525568. The bytes pane shows the raw hex and ASCII data of the packet.

סך הכל הלכו צירק לשלוח 15000 בטים, בהודעה הראשונה נשים לב שהלכו שלוח לשרת מידע של 14600 בתים (סה"כ 14656 עם header 4bytes), כאשר כל המידע הזה הוא A, נשים לב שההבדל בין הידיעה 14,600 לבין ההודעה הזאת(הידיעה מס' 4) לבין ההודעה הבאה(הידיעה מס' 6) הוא 14,600 (כי הלקוות הגדיל את הסeq number שלו בעקבות ה-ack של השרת), נשים לב שיש את הדגל don't fragment בשכבות הרשות.

נשים לב שבאופן אחד מההודעות לא הייתה פרוגמנטציה בשכבות הרשות:

Wireshark - Packet 4 · ex2part2Aclient1.pcap

Frame 4: 14656 bytes on wire (117248 bits), 14656 bytes captured (117248 bits)  
 ▶ Linux cooked capture  
 ▶ Internet Protocol Version 4, Src: 192.168.42.12, Dst: 192.168.42.203  
 0100 .... = Version: 4  
 .... 0101 = Header Length: 20 bytes (5)  
 ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
 Total Length: 14640  
 Identification: 0x7efa (32586)  
 Flags: 0x4000 [Don't fragment]  
 0... .... .... = Reserved bit: Not set  
 .1.. .... .... = Don't fragment: Set  
 ..0. .... .... = More fragments: Not set  
 ...0 0000 0000 0000 = Fragment offset: 0  
 Time to live: 128

0000 00 00 01 00 06 02 52 55 06 36 33 00 00 08 00	.....R U 63 ..
0010 45 00 39 7e fa 40 00 80 06 c6 a5 c0 a8 2a 00	E 90- @. .1 ..*
0020 c0 a8 2a cb c3 9c 30 39 e8 7b 91 2f a0 c8 be 0b	*.*. 09 { / ...
0030 50 10 08 05 cf b0 00 00 41 41 41 41 41 41 41 41	P....K . AAAAAAAA
0040 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
0050 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
0060 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
0070 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
0080 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
0090 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
00a0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
00b0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
00c0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
00d0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
00e0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
00f0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
0100 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA

No.: 4 - Time: 0.002704 - Source: 192.168.42.12 - Destination: 192.168.42.203 - Protocol: TCP - Length: 14656 - Info: 50076 → 12345 [ACK] Seq=3900412207 Ack=2697510411 Win=525568 Len=14600

Help Close

Wireshark - Packet 6 · ex2part2Aclient1.pcap

Frame 6: 456 bytes on wire (3648 bits), 456 bytes captured (3648 bits)  
 ▶ Linux cooked capture  
 ▶ Internet Protocol Version 4, Src: 192.168.42.12, Dst: 192.168.42.203  
 0100 .... = Version: 4  
 .... 0101 = Header Length: 20 bytes (5)  
 ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
 Total Length: 440  
 Identification: 0x7f04 (32516)  
 Flags: 0x4000 [Don't fragment]  
 0... .... .... = Reserved bit: Not set  
 .1.. .... .... = Don't fragment: Set  
 ..0. .... .... = More fragments: Not set  
 ...0 0000 0000 0000 = Fragment offset: 0  
 Time to live: 128

0000 00 00 01 00 06 02 52 55 06 36 33 00 00 08 00	.....R U 63 ..
0010 45 00 01 b8 7f 04 40 00 80 06 a4 13 c0 a8 2a 00	E .....@. ....*.
0020 c0 a8 2a cb c3 9c 30 39 e8 7b ca 37 a0 c8 be 0b	*.*. 09 { 7 ..
0030 50 10 08 05 cf b0 00 00 41 41 41 41 41 41 41 41	P..... AAAAAAAA
0040 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
0050 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
0060 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
0070 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
0080 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
0090 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
00a0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
00b0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
00c0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
00d0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
00e0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
00f0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA
0100 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA AAAAAAAA

No.: 6 - Time: 0.002882 - Source: 192.168.42.12 - Destination: 192.168.42.203 - Protocol: TCP - Length: 456 - Info: 50076 → 12345 [PSH, ACK] Seq=3900426807 Ack=2697510411 Win=525568 Len=400

Help Close

בכל הודעה של רצפי A בשכבה הרשת גם הדגל don't fragment הוא 0 כי אין פרוגמנטציה.

התשובה שהתקבלה מהשרת היא:

Wireshark - Packet 5 · ex2part2Aclient1.pcap

Frame 5: 56 bytes on wire (448 bits), 56 bytes captured (448 bits)  
 ▶ Linux cooked capture  
 ▶ Internet Protocol Version 4, Src: 192.168.42.203, Dst: 192.168.42.12  
 ▶ Transmission Control Protocol Src Port: 12345, Dst Port: 50076, Seq: 2697510411, Ack: 3900426807, Len: 0

Source Port: 12345
Destination Port: 50076
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 2697510411
[Next sequence number: 2697510411]
Acknowledgment number: 3900426807
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
000. .... .... = Reserved: Not set
...0 .... .... = Nonce: Not set
...0. .... .... = Congestion Window Reduced (CWR): Not set
...0. .... .... = ECN-Echo: Not set
...0. .... .... = Urgent: Not set
...0.1 .... = Acknowledgment: Set
...0.0....0... = Push: Not set
...0.0....0... = Reset: Not set
...0.0....0... = Syn: Not set
...0.0....0... = Fin: Not set
[TCP Flags: .....A....]

0000 00 04 00 01 00 06 08 00 27 8b 4b c5 00 00 08 00	.....R K ..
0010 45 00 00 28 a1 45 40 00 40 06 c3 62 c0 a8 2a cb	E ..@. 09 ..*
0020 c0 a8 2a 0c 30 39 c3 9c a0 c8 be 0b e8 7b ca 37	*. 09 ..{ .7
0030 50 10 01 c9 d6 42 00 00	P....B ..

Help Close

השרת שלח ACK על ההודעה שקיבל, ניתן לראות שה-ACK NUMBER שהשרות שלח גדול מ-14600 מה-number שלוח לו הלקוח עם ההודעה, ככלומר השרת קיבל את כל ההודעה(שאורךה 14600).

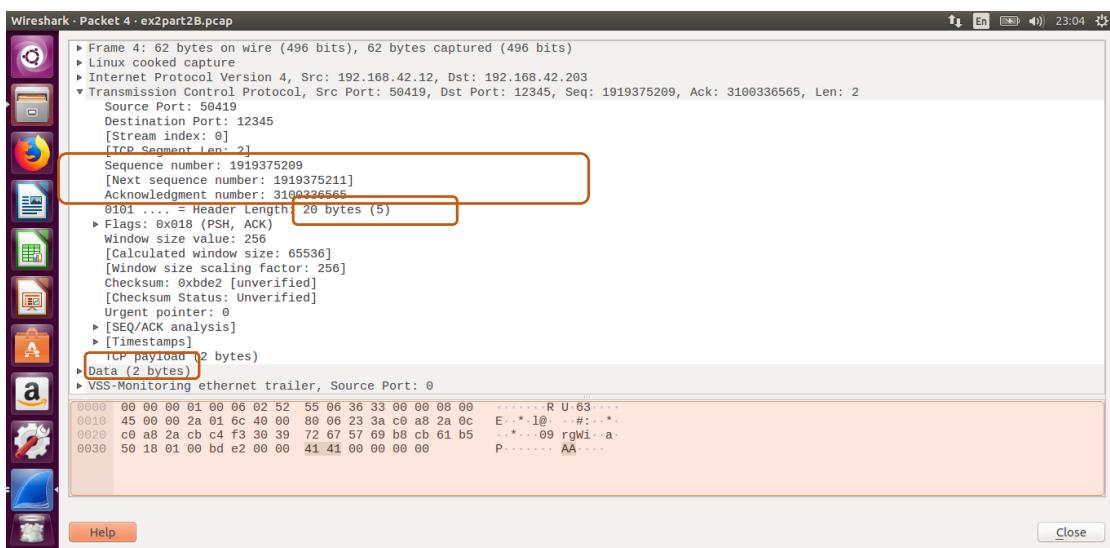
באופן זה ניתן לראות שהליך שלח לשרת מספר הודעות בהן רצפים של 14600 A ולא הודהה אחת של 15000 A ככלומר בוצע תהליך סגמנטציה ע"י שכבת התעבורה.

**סעיף ב':**

הקו של הרשות הוא 192.168.42.12 , הקו של הלקוח הוא 192.168.42.203 . הפורט של הסרבר הוא 50419 ופורט הלקוח הוא 12345.

בהתיל'ר התקשרות בין הלקוח לשרת, השרת קיבל פעמיים הודעה AA (שני A רצופים באותה הודעה) ובשאר ההודעות התקבל כל A בפרט.

כאן רואים את החבילה הראשונה שנשלחה לשרתן: AA (אך המידע הוא 2 בתים – 4141 שבעיota זה AA)



בזהותה זו ניתן לראות את שליחת ההודעה AA מהלך לשרת. אנחנו יודעים שבוקוד הלהקה send מבוצע פעמיים בפרט, אך כמו שניתן לראות ב data של ההודעה השילוחת "התליכו" להודעה אחת והמידע נשלח הפעם נשלח בהודעה אחת. גשים לב ש ה seq number הוא 1919375209, ונסתכל בצללים הבא:

Wireshark - Packet 5 · ex2part2B.pcap

Frame 5: 56 bytes on wire (448 bits), 56 bytes captured (448 bits)

Internet Protocol Version 4, Src: 192.168.42.203, Dst: 192.168.42.12

Transmission Control Protocol, Src Port: 12345, Dst Port: 50419, Seq: 3100336565, Ack: 1919375211, Len: 0

Source Port: 12345  
Destination Port: 50419  
[Stream index: 0]  
[TCP Segment Len: 1]  
Sequence number: 3100336565  
[Next sequence number: 3100336565]  
Acknowledgment number: 1919375211  
0101 ... = Header length: 20 bytes (5)  
[Flags: 0x010 (ACK)]  
Window size value: 229  
[Calculated window size: 29312]  
[Window size scaling factor: 128]  
Checksum: 0xd642 [unverified]  
[Checksum Status: Unverified]  
Urgent pointer: 0  
[SEQ/ACK analysis]  
[timestamps]

0008 00 04 00 01 00 06 08 00 27 8b 4b c5 00 00 08 00 .....K...  
0010 45 00 28 c2 f9 40 00 40 06 a1 ae c0 a8 2a cb E..( ..@. #9...\*.  
0020 c0 a8 2a c0 30 39 c4 f3 b8 cb 61 b5 72 67 57 6b ..\*..09..a.rgwk  
0030 50 10 00 e5 d6 42 00 00 P....B..

Help Close

הואים שהשרת שלוח ללקוח ACK עם seq number ack 199375211 כלומר גדול ב 2 מ ה seq number של הודעתה AA שאורכה 2 - כלומר ההודעה התקבלה במלואה בשרת, והתקבלה כהוודעה אחת כנראה הרि נשלח עלייה ACK נפרד.

נתבונן בחבילות הבאות שנשלחו מהלקוח לשרת שמכילה גם שני AA.

Wireshark - Packet 8 · ex2part2B.pcap

Frame 8: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)

Internet Protocol Version 4, Src: 192.168.42.12, Dst: 192.168.42.203

Transmission Control Protocol, Src Port: 50419, Dst Port: 12345, Seq: 1919375211, Ack: 3100336566, Len: 1

Source Port: 50419  
Destination Port: 12345  
[Stream index: 0]  
[TCP Segment Len: 1]  
Sequence number: 1919375211  
[Next sequence number: 1919375212]  
Acknowledgment number: 3100336566  
0101 ... = Header Length: 20 bytes (5)  
[Flags: 0x018 (PSH, ACK)]  
Window size value: 256  
[Calculated window size: 65536]  
[Window size scaling factor: 256]  
Checksum: 0xbe21 [unverified]  
[Checksum Status: Unverified]  
Urgent pointer: 0  
[SEQ/ACK analysis]  
[timestamps]  
TCP payload (1 byte)  
Data (1 byte)  
VSS-Monitoring ethernet trailer, Source Port: 0

0008 00 00 00 01 00 06 02 52 55 06 36 33 00 00 08 00 .....R U.63...  
0010 45 00 29 01 6e 40 00 80 06 23 39 c0 a8 2a 0c E..) .0@. #9...\*.  
0020 c0 a8 2a cb c4 f3 30 39 72 67 57 6b b8 cb 61 b6 ..\*..09.rgwk.a.  
0030 50 18 01 00 be 21 00 00 41 00 00 00 00 00 00 P....! A...

No. 8 - Time: 2.273888 - Source: 192.168.42.12 - Destination: 192.168.42.203 - Protocol: TCP - Length: 62 - Info: 50419 → 12345 [PSH, ACK] Seq=1919375211 Ack=3100336566 Win=65536 Len=1

Help Close

Wireshark - Packet 9 · ex2part2B.pcap

Frame 9: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)

Internet Protocol Version 4, Src: 192.168.42.12, Dst: 192.168.42.203

Transmission Control Protocol, Src Port: 50419, Dst Port: 12345, Seq: 1919375212, Ack: 3100336566, Len: 1

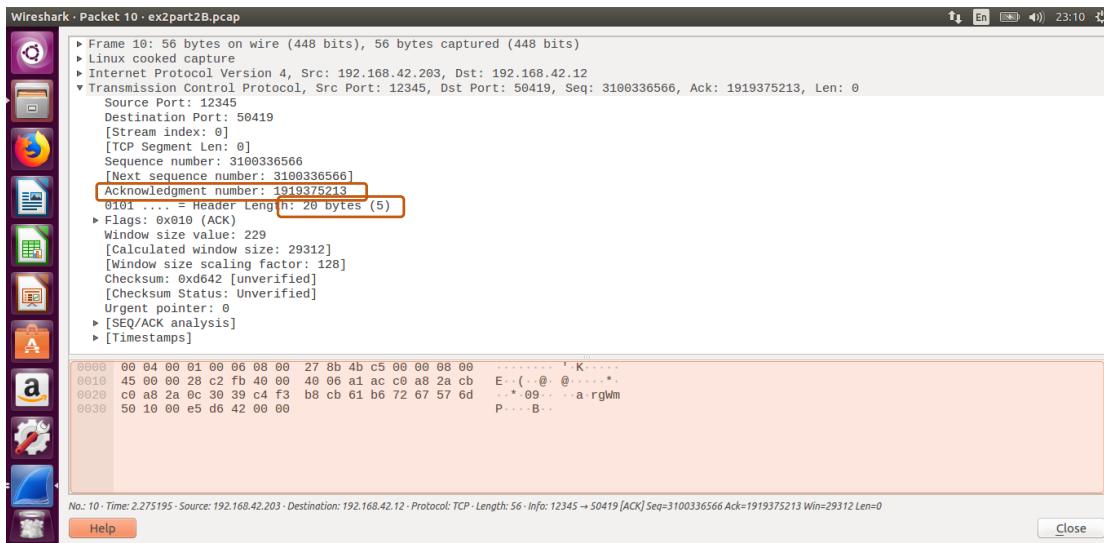
Source Port: 50419  
Destination Port: 12345  
[Stream index: 0]  
[TCP Segment Len: 1]  
Sequence number: 1919375212  
[Next sequence number: 1919375213]  
Acknowledgment number: 3100336566  
0101 ... = Header Length: 20 bytes (5)  
[Flags: 0x018 (PSH, ACK)]  
Window size value: 256  
[Calculated window size: 65536]  
[Window size scaling factor: 256]  
Checksum: 0xbe20 [unverified]  
[Checksum Status: Unverified]  
Urgent pointer: 0  
[SEQ/ACK analysis]  
[timestamps]  
TCP payload (1 byte)  
Data (1 byte)  
VSS-Monitoring ethernet trailer, Source Port: 0

0008 00 00 00 01 00 06 02 52 55 06 36 33 00 00 08 00 .....R U.63...  
0010 45 00 29 01 6f 40 00 80 06 23 38 c0 a8 2a 0c E..) .0@. #8...\*.  
0020 c0 a8 2a cb c4 f3 30 39 72 67 57 6c b8 cb 61 b6 ..\*..09.rgwl.a.  
0030 50 18 01 00 be 20 00 00 41 00 00 00 00 00 00 P....! A...

Help Close

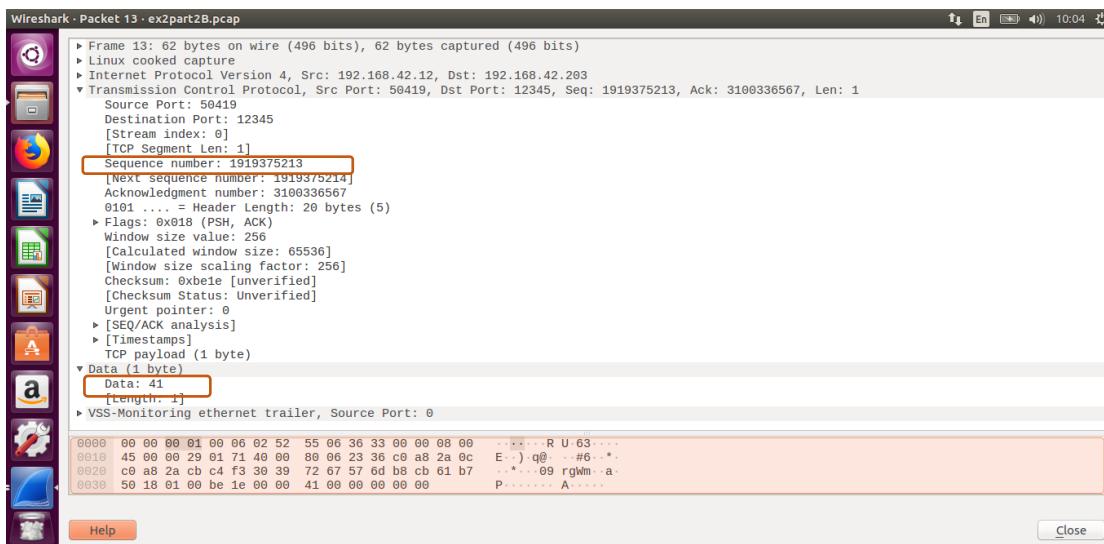
נשים לב שגודל כל הודעה הוא 1, מתקיים ששלוח A מהלך לשרת(גודל ה-data הוא 1), גודל header הוא 20, ה-seq number של הודעה הראשונה של הלקוח הוא 1919375211 והשנייה 1919375212 והוא ה-ack number של השרת שהוא לא השתנה מהשליחה הקודמת של השרת(כיוון ששלוח .(ACK)).

נתבונן ב-ack של השרת ללקוח:



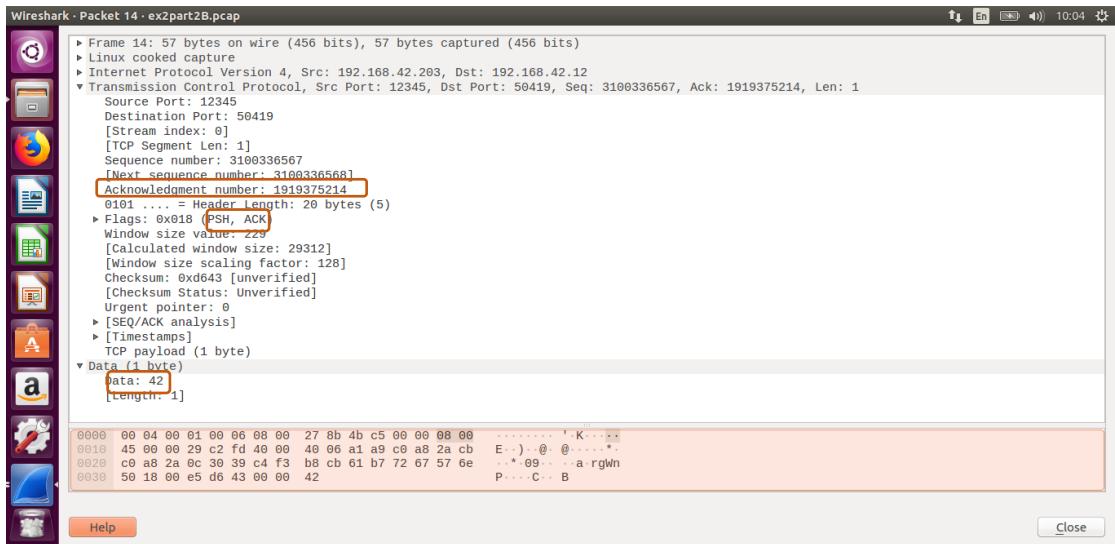
נשים לב שהאיך הוא 1919375213 ,-column המהווה גודל(seq number) ב-2 מקודם , כך השרת מאשר ללקוח שהוא קרא מידע של שני ביטים(AA) , נשים לב שהגודל של tcp header הוא 20 וdagל ack דלק.

כעת נתבונן בחבילה שהלקוח שלח לשרת A בודד והשרת תפס אותו כבודד(שהזהה בפועל כל החבילות שנשארו).



נשים לב שגודל הודעה הוא 1, מתקיים ששלוח A מהלך לשרת(גודל ה-data הוא 1), גודל header הוא 20, ה-seq number של הודעה הראשונה של הלקוח הוא ackn 1919375213 והשנייה 1919375212 והוא ה-ack number של השרת שהוא לא השתנה מהשליחה הקודמת של השרת(כיוון ששלוח .(ACK)).

נתבון בפקט ack של השרת ללקוח:



נשים לב שהערך ack הוא 1919375214, כלומר הערך seq number של הלקוח גדול ב1 ממקודם, כך השרת מאשר ללקוח שהוא קרא מידע של בית אחד(A), נשים לב שהגודל של header tcp הוא 20 וערך ack דלוק.

כל הibilities הבאות זהות בdata של him (כולומר שלוחות רק A אחד, חז' מה 2 הראשונות ששלוחות 2 (A) וכן אין צורך להראות את כלן, נציג כי בהודעה האחרונות הערך seq number יהיה גדול ב22 ממה שהיה בהתחלה, מה שמשמעותו ש坎坷ו הכל ההודעות ביחד 22.

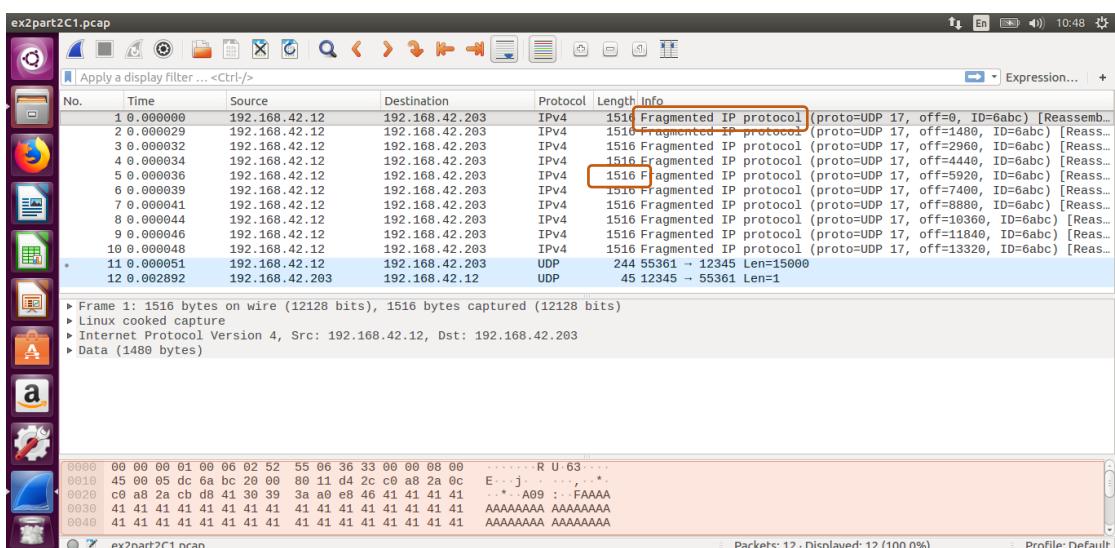
תוצאה זו צפוייה להתקבל לא משנה אם שני הודעות send מתלכדות או נשלחות בנפרד שחרי בכל מקרה "שלחו 2 בתים בסופו של דבר".

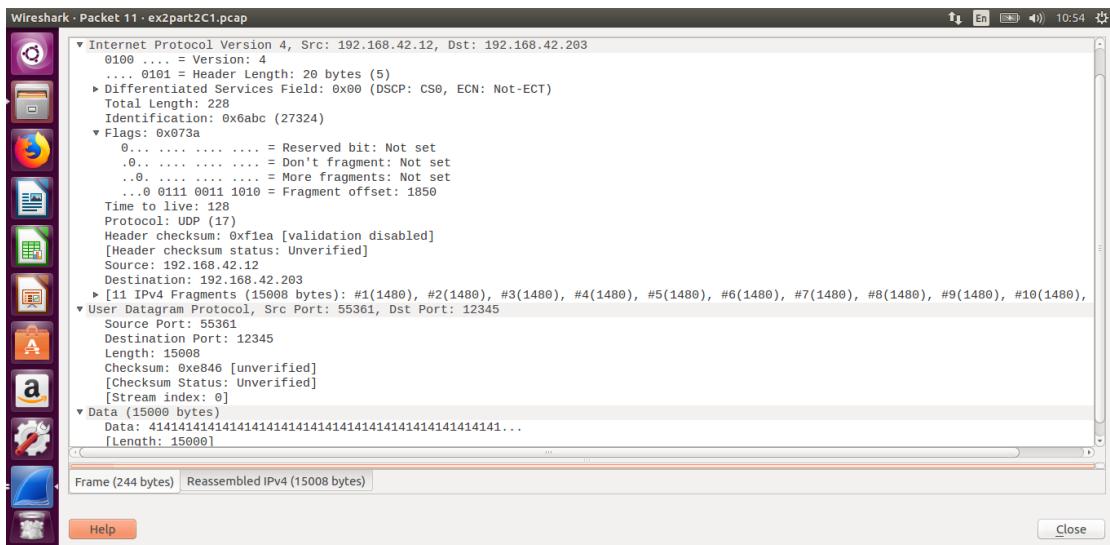
בתשובה השרת ללקוח נשלח פעמיים אחת, B ניתן לראות שב הודעה הבאה יחזיר לו הלקוח ACK שערק ה-ack number שלו גדול ב 1 מערך ה seq של השרת בהודעה הקודמת כמצופה.

## שאלה 2 סעיף ג':

.1

שליחת 15,000 תווים A מהלקוח לשרת בקפdas:



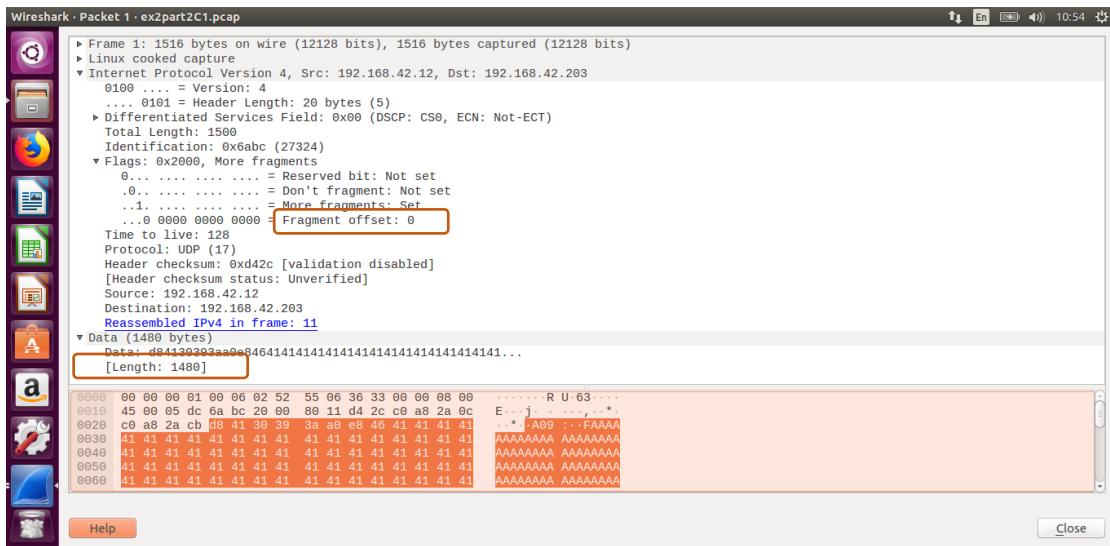


ההודעה המצלמת כאן היא הודעה האחורונה שהלקוח שלח לשרת.

נשים ללב לתהליך שמתרכז כאן, הקפּן מורייד לשכבות הרשות בקשה לשלווה 15000 בתים של התו A, אפשר לראות בהודעות כי שכבות הרשות עושות פרגמנטציה, כלומר מפרקת את החבילה לפי המטא, שניתן לראות כאן 1516 שווא.

כיוון ש乾坤 איננו פרוטוקול אמין, הוא לא מבצע חלוקה של החבילות.

**נסתכל על ההודעה הראשונה שנשלחה:**



נשים לברוח מהעיר המוקסימל (MSS) שיכל להישלח לא כולל הדרוג הוא 1480 (רואים את זה לפני האורך של header), עבורה מתווסף header של שכבות הרשות(20 בתים) ושל שאר שכבות(16 בתים) ורק הכל גודל של 1516.

נתבונן בכל הבעיות חוץ מהabituta האחרונה כיון שגם חビלה שנשלחה מהשרות ללקוח, ומה שמעניין אותו זה הבעיות שנשלחו מהמלך לשרת.

לכל חיבור הinfo כולל protocol ip fragmented, כלומר חלק מחייב לה שפה עברית פרגמנטציה.

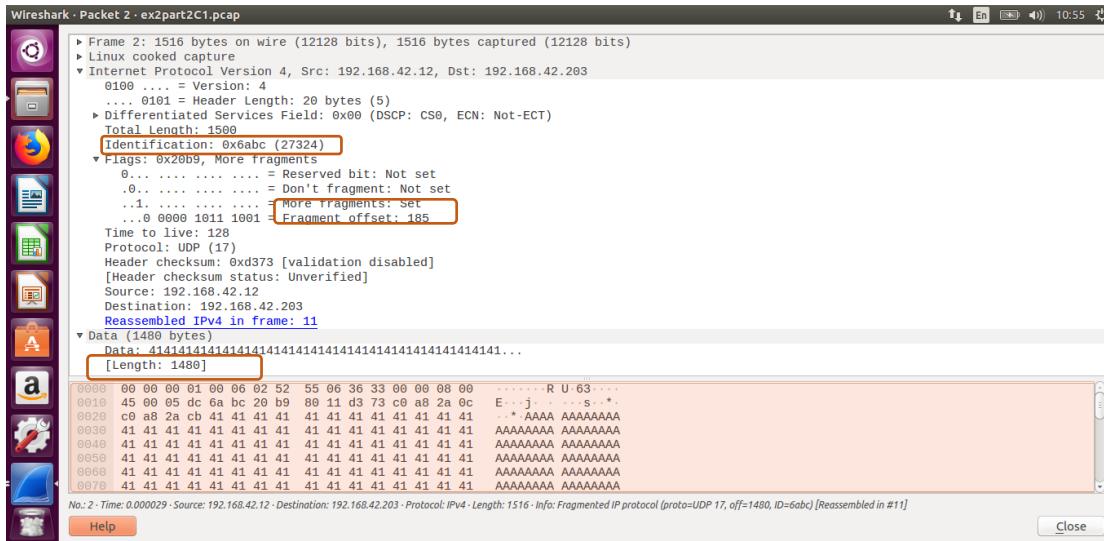
נשים לב שבעור החבילה הראשונה והחbillות שהרמיה דלוק הדגל zw (כלומר, בוצעה פרגמנטציה), ויש עוד חbillות אחרי החבילה pz). נשים לב שבערכות הינה דלוק הדגל zf בשכבת הרשת.

בכל הabiliaות חוץ מhabilaות אחת לא יהיה header של udp, כיון ששכבות הרשת מוצעת פרוגמנטיצה, לא משנה לה איך המידע שהייא קיבלה הגיע, היא סך הכל מחלקת אותוhabilaות בגדים המתאימים ומוסיפה את headers הרכזונטיים שלה, ולכן קיימ header בודד של udp בכל הabiliaות שנשלחו.

נשים לב שהidentification (שדה שוחש בפרוגמנטיצה על מנת שטכל הabiliaות קשורות אחת לשניה) הוא זהה בכל הabiliaות והוא 0x6abc.

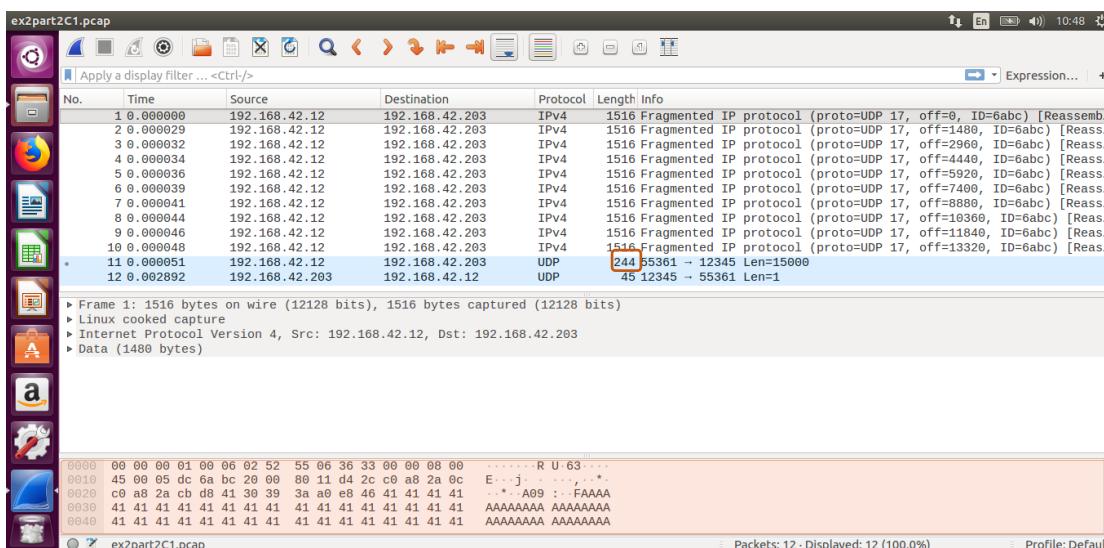
נשים לב כי offset שלhabilaות הראשונה הוא 0.

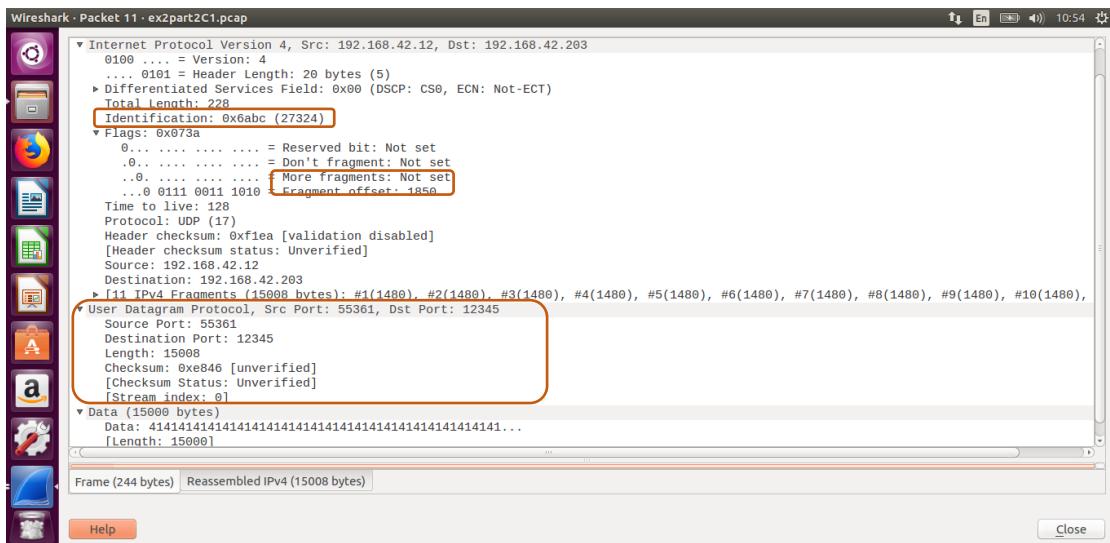
#### cut נtabonhabilaות השניה שנשלחה:



נשים לב שכן offset הוא 185, וזה המידע מוחלק ל8 הבודעות ולכן  $1480 = 185 * 8$ , שכןhabilaות הראשונה שנשלחה כוללת 1480 bytes מהhabilaות המקורית, והhabilaות השניה תתחילה עם המידע שנמצא אחריו אוטם 1480 בתים, נשים לב שהidentification יcano זהה להודעה הקודמת וגם כן דלוק דgal המדו.

#### cut נtabonhabilaות האחורונה שהלכו שלח אל הרשת:

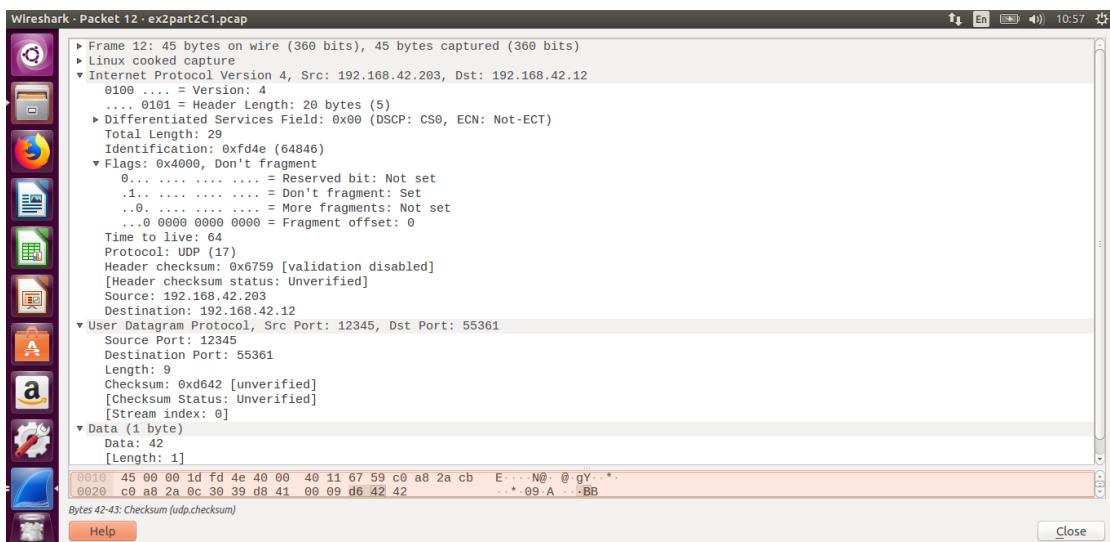




גודל החבילה צריך להיות 15000 פחות 1480 \* מספר החבילות שנשלחו , מחישוב פשוט אנו יודעים כי זה שווה ל 200 , וכיון שיש גם את headers udp , את שאר headers שביחד הם 44 בתים נקבל שגודל החבילה הוא 244 בתים.

נשים לב שההעדר identification הוא 0 כיון שגם החבילה האחורה שנשלחה מהלкова לשרת בתהילר הrogramnetzcia הנ"ל, אפשר לראות שהמידע בשכבת הקdp הוא מאוד מצומצם , כיון שהיא נותנת לשכבת הרשת למעשה הרבה יותר עבודה.

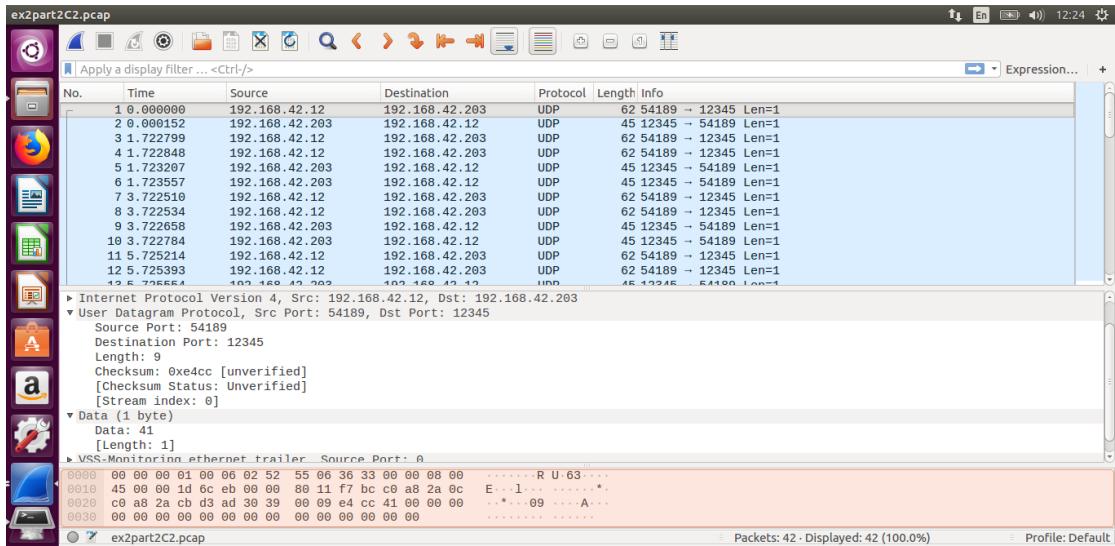
#### נתבען בחבילה(היחידה) שהשרת שלח ללקוח:



ניתן לדוחות כי זו ההודעה שהсер버 שלח ללקוח ע"פ ה source portו שהוא 12345

נשים לב שלחבילה זו אין דגלים בחבילת הרשות שקשורים לrogramnetzcia(כיון שהוא לא צריך לעבור פrogramnetzcia), וכך נעביר רק בית אחד של מידע בתוספת headers משאר השכבות ונגיע ל 45 בתים של מידע בסך הכל.

## حلק ב' של סעיף ג':



שים לב שבמקרה זה כל ההודעות של תווים A יתקבלו בנפרד (גודל כל הודעה כאן היא 1) , במקרה של TCP ראיינו לפעמים שתני הودעות A רצופות שמתחרבות להודעת AA אחת.

ידוע לנו שtcp מעביר את המידע כרצף של בתים , זאת אומרת שהוא לא מודיע לגבולות של הודעה, הוא יודיע רק שיש לו רצף של בתים שעליו להעביר הלהה בסדר זהה.

כאשר עשינו ב-tcp send רצופים, אז הבטים שהבادر היה צריך להעביר נוכנסו אליו ביחד ב един האלול tcp כמו שהוא יודיע העביר אותם byte stream כמזה:

כלומר , tcp לא הספיק לשולח את מה הראשון לפני שהשני כבר נכנס, tcp כאמור שולח את כל הבטים שיש לו בבאפר בלי מודעות למאיפה הגעה הודעה ולכך לפעמים(כאשר tcp לא הספיק לשולח את מה הראשון לפני השני נכנס) הם יתמזגו להודעה אחת(הר' בבאפר ישם מידע AA).

לעומת זאת , בudp אין את אותו רצף בתים ש모וברים מקופה לנצח בudp תמיד כתבנו:

```
Socket.sendto(msg,(ip_dest,port_dest))
```

כלומר את הודעה הוא היה שולח מיד ליעד, ולא בהתבסס על החיבור ביניהם.

לכן כל שליחת הודעה הייתה בלתי תליה בשליחות ההודעות האחרות, וכך כל הודעה A נשלחה בפני עצמה והתקבלה אצל הרשות בנפרד.