

רשותות תרגיל 2 חלק 1

מצורף כאן צילום מסך של תקשורת בין שני מחשבים שונים. הלקוח (10.0.0.16) מרים את הקובץ `tcp_server.py` אשר פונה לשרת (10.0.0.32) המרים את הקובץ `tcp_client.py`

The figure shows a Wireshark capture window with the following details:

- Display Filter:** client-server-wireshark.pcapng
- Selected Filter:** Apply a display filter... <esc>/
- Selected Column:** Info
- Number of Packets:** 13
- Summary:** The table lists 13 network packets. The first 12 packets are from 10.0.0.16 to 10.0.0.32, and the last one is from 10.0.0.32 to 10.0.0.16. The protocols shown are TCP and ACK.
- Details:** Each row provides information about a specific packet, including its number, timestamp, source, destination, protocol, length, and detailed info.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.0.16	10.0.0.32	TCP	78	55748 → 12345 [SYN] Seq=3694215514 Win=65535 Len=0 MSS=1460 WS=64 TStamp=36133664
2	0.040079	10.0.0.32	10.0.0.16	TCP	74	12345 → 55748 [SYN, ACK] Seq=1667296290 Win=3694215515 Len=0 MSS=1460 TStamp=36133664
3	0.040246	10.0.0.16	10.0.0.32	TCP	66	55748 → 12345 [ACK] Seq=3694215515 Ack=1667296291 Win=131776 Len=0 TStamp=36133664
4	0.040296	10.0.0.16	10.0.0.32	TCP	78	55748 → 12345 [PSH, ACK] Seq=3694215515 Ack=1667296291 Win=131776 Len=0 TStamp=36133664
5	0.044234	10.0.0.32	10.0.0.16	TCP	66	12345 → 55748 [ACK] Seq=1667296291 Win=3694215527 Len=0 TStamp=32595633
6	0.045122	10.0.0.32	10.0.0.16	TCP	78	12345 → 55748 [PSH, ACK] Seq=1667296291 Win=3694215527 Len=0 TStamp=32595633
7	0.045124	10.0.0.32	10.0.0.16	TCP	66	12345 → 55748 [FIN, ACK] Seq=1667296303 Win=3694215527 Len=0 TStamp=32595633
8	0.045223	10.0.0.16	10.0.0.32	TCP	66	55748 → 12345 [ACK] Seq=3694215527 Ack=1667296303 Win=131776 Len=0 TStamp=36133664
9	0.045270	10.0.0.16	10.0.0.32	TCP	66	55748 → 12345 [ACK] Seq=3694215527 Ack=1667296304 Win=131776 Len=0 TStamp=36133664
10	0.045349	10.0.0.16	10.0.0.32	TCP	75	55748 → 12345 [PSH, ACK] Seq=3694215527 Ack=1667296304 Win=131776 Len=9 TStamp=36133664
11	0.045388	10.0.0.16	10.0.0.32	TCP	66	55748 → 12345 [FIN, ACK] Seq=3694215536 Ack=1667296304 Win=131776 Len=0 TStamp=36133664
12	0.050230	10.0.0.32	10.0.0.16	TCP	54	12345 → 55748 [RST] Seq=1667296304 Win=0 Len=0
13	0.051229	10.0.0.32	10.0.0.16	TCP	54	12345 → 55748 [RST] Seq=1667296304 Win=0 Len=0

תהליך הקמת החיבור (Three-Way Handshake)

תהליך זה מתרחש בחבילות 1 עד 3. זהו תהליך לחיצת היד המשולשת.

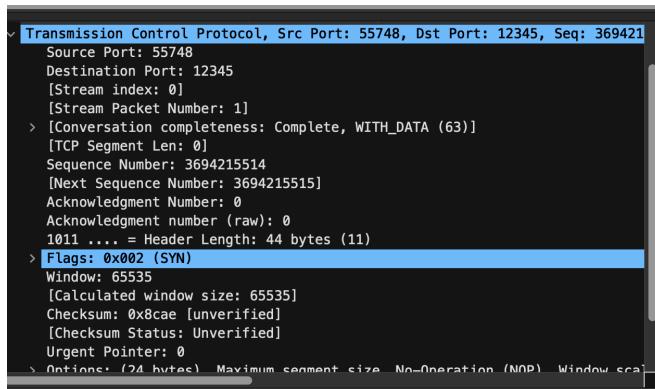
1. שלב ראשון (Packet 1) - שליחת SYN:

  - הלקוח (10.0.0.16) יוזם את החיבור ושולח הודעה SYN לשרת (10.0.0.32).
  - מספרים סידוריים: המוקור מגיריל מספר סידורי תחלה.

Seq = 3694215514 ■

Ack = 0 ■

  - משמעותו: אני רוצה לפתוח איתך שיחה, המספר הסידורי שלי מתחילה ב-3694215514.



2. שלב שני (Packet 2) - שליחת ACK, SYN: ○ תיאור: השרת מקבל את הבקשה, ומחזיר הודעה המכילה גם NY (כדי לפתח ערכן משלו) וגם ACK (אישור הבקשה של הלקובץ). ○ מספרים סידוריים: ■  $\text{Seq} = 1667296290$  ■  $\text{Ack} = 3694215515$

- משמעות: קיבלתי את ההודעה שלך והוספתי לה 1, ואני גם רוצה לפתח איתך שיחה במספר הסידורי שלו.

```

    > Transmission Control Protocol, Src Port: 12345, Dst Port: 55748, Seq: 166729
      Source Port: 12345
      Destination Port: 55748
      [Stream index: 0]
      [Stream Packet Number: 2]
    > [Conversation completeness: Complete, WITH_DATA (63)]
      [TCP Segment Len: 0]
      Sequence Number: 1667296290
      [Next Sequence Number: 1667296291]
      Acknowledgment Number: 3694215515
      1010 .... = Header Length: 40 bytes (10)
    > Flags: 0x012 (SYN, ACK)
      Window: 65160
      [Calculated window size: 65160]
      Checksum: 0x2eaf [unverified]
      [Checksum Status: Unverified]
  
```

- שלב שלישי (Packet 3) - שליחת ACK:  
הלקוח מאשר את קבלת ה-SYN של השרת ע"י שליחת ACK. החיבור כעת הוקם.  
מספרים סידוריים:
- Seq (התקדם ב-1 לעומת חבילת 1, כי דגל ה-SYN צורך מספר סידורי אחד).  
■ Ack (הчисלוב הוא: ה-Seq של השרת + 1).

```

> Frame 3: Packet, 66 bytes on wire (528 bits), 66 bytes captured (528 bits) o
> Ethernet II, Src: de:6e:31:1c:f5:de (de:6e:31:1c:f5:de), Dst: Intel_46:ba:9a
> Internet Protocol Version 4, Src: 10.0.0.16, Dst: 10.0.0.32
> Transmission Control Protocol, Src Port: 55748, Dst Port: 12345, Seq: 369421
  Source Port: 55748
  Destination Port: 12345
  [Stream index: 0]
  [Stream Packet Number: 3]
> [Conversation completeness: Complete, WITH_DATA (63)]
  [TCP Segment Len: 0]
  Sequence Number: 3694215515
  [Next Sequence Number: 3694215515]
  Acknowledgment Number: 1667296291
  1000 .... = Header Length: 32 bytes (8)
> Flags: 0x010 (ACK)
  Window: 2059
  [Calculated window size: 131776]
  [Window size scaling factor: 64]
  Checksum: 0x53d0 [unverified]
  
```

## העברת נתונים וניתוח המספרים

### 1. שליחת מידע (Packet 4):

- תיאור: הלקוח (10.0.0.16) שולח מידע לשרת. ניתן לראות את הדגל שմבוקש להעביר את המידע לאפליקציה מיד.
- גודל המידע:  $Len = 12$  (רואים זאת בעמודת Info או Length).
- מספר סידורי:  $.Seq = 3694215515$

The screenshot shows a single TCP packet from a client to a server. The details pane at the top displays the following information:

- Sequence Number: 3694215515
- [Next Sequence Number: 3694215527]
- Acknowledgment Number: 1667296291
- 1000 .... = Header Length: 32 bytes (8)
- > Flags: 0x018 (PSH, ACK)
- Window: 2059
- [Calculated window size: 131776]
- [Window size scaling factor: 64]
- Checksum: 0x4960 [unverified]
- [Checksum Status: Unverified]
- Urgent Pointer: 0
- > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
- > [Timestamps]
- > [SEQ/ACK analysis]
- [Client Contiguous Streams: 1]
- [Server Contiguous Streams: 1]
- TCP payload (12 bytes)

The bytes pane at the bottom shows the raw hex and ASCII data of the payload:

Offset	Hex	ASCII
0000	f4 06 69 46 ba 9a de 6e 31 1c f5 de 08 00 45 00	iF...n 1.....E..
0010	00 40 00 00 40 00 40 06 26 89 0a 00 00 10 0a 00	@...@. @ &.....
0020	00 20 d9 c4 30 39 dc 31 41 5b 63 60 ec 23 80 18	...09.1 A[c`.#..
0030	08 0b 49 60 00 00 01 01 08 0a d7 5f 97 5d c2 4a	I.....[.J
0040	5e 9b 59 6f 61 76 20 45 6c 68 61 64 61 64	^Yoav E lhadad

### 2. אישור קבלת מידע (Packet 5):

- תיאור: השרת (10.0.0.32) מאשר את קבלת המידע.
- חישוב ה-Ack: השרת מבצע חישוב פשוט: המספר הסידורי הקודם שקיבל + גודל המידע שקיבל.
- החישוב במספרים:
  - $3694215515$  (ה-.Seq של חבילה 4)
  - $+ 12$  (ה-Len של חבילה 4)
  - $= 3694215527$
- לכן חבילה 5 אנו רואים Ack = 3694215527. זהו האישור שהמידע התקבל בשלםותיו והשרת מצפה לבית הבא במספר זה.

```

!Stream Packet Number: 5]
> [Conversation completeness: Complete, WITH_DATA (63)]
[TCP Segment Len: 0]
Sequence Number: 1667296291
[Next Sequence Number: 1667296291]
Acknowledgment Number: 3694215527
1000 .... = Header Length: 32 bytes (8)
> Flags: 0x010 (ACK)
Window: 509
[Calculated window size: 65152]
[Window size scaling factor: 128]
Checksum: 0x59cd [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
> [Timestamps]
> [SEQ/ACK analysis]
[Client Contiguous Streams: 1]
[Server Contiguous Streams: 1]

```

client-server-wireshark.pcapng

### 3. תשובה השרת (Packet 6)

- תיאור: השרת (10.0.0.32) שולח חזרה ללקוח את המחרוזת "YOAV ELHADAD".
- ה Seq הוא 1667296291 (זהה ל-ACK הקודם, כי השרת עדין לא שלח מידע, רק אישר).
- ה Ack הוא 3694215527 (השרת עדין מצפה לבית ה-27, כולם מאשר שקיבל את השם).
- האורך הוא 12 בתים

```

* Transmission Control Protocol, Src Port: 12345, Dst Port: 55748, Seq: 1, Ack: 13, Len: 12
Source Port: 12345
Destination Port: 55748
[Stream index: 0]
[Stream Packet Number: 6]
> [Conversation completeness: Complete, WITH_DATA (63)]
[TCP Segment Len: 12]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 1667296291
[Next Sequence Number: 13 (relative sequence number)]
Acknowledgment Number: 13 (relative ack number)
Acknowledgment number (raw): 3694215527
1000 .... = Header Length: 32 bytes (8)
> Flags: 0x018 (PSH, ACK)
Window: 509
[Calculated window size: 65152]
[Window size scaling factor: 128]
Checksum: 0xcffd [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
> [Timestamps]
> [SEQ/ACK analysis]
[Client Contiguous Streams: 1]
[Server Contiguous Streams: 1]
TCP payload (12 bytes)

```

### 4. שליחת תעודת זהות (Packet 8)

- תיאור: הלקוח (10.0.0.16) שולח את ההודעה השנייה המכילה את תעודת הזהות: "315853793".

הчисוב במספרים:

- Seq=3694215527, המספר הסידורי הקודם של הלקוח בחבילה 4 היה 3694215515, שלחנו 12 בתים ולכן  $3694215515 + 12 = 3694215527$
- Ack=1667296303, הלקוח קיבל מהשרת את תשובה Echo החבילה 6 שהייתה באורך 12 בתים והתחילה ב 1667296291 لكن הלקוח מאשר קבלה ומצפה לו:  $1667296303 + 12 = 1667296291$ .

## האורך הוא 9 בתים ■

```
▼ Transmission Control Protocol, Src Port: 55748, Dst Port: 12345, Seq: 13, Ack: 13, Len: 0
  Source Port: 55748
  Destination Port: 12345
  [Stream index: 0]
  [Stream Packet Number: 8]
  > [Conversation completeness: Complete, WITH_DATA (63)]
    [TCP Segment Len: 0]
    Sequence Number: 13      (relative sequence number)
    Sequence Number (raw): 3694215527
    [Next Sequence Number: 13      (relative sequence number)]
    Acknowledgment Number: 13      (relative ack number)
    Acknowledgment number (raw): 1667296303
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x010 (ACK)
    Window: 2059
    [Calculated window size: 131776]
    [Window size scaling factor: 64]
    Checksum: 0x53af [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  > [Timestamps]
  > [SEQ/ACK analysis]
    [Client Contiguous Streams: 1]
    [Server Contiguous Streams: 1]
```

## חלק 3: סיום החיבור

1. ניסיון סיום (Packet 7):
  - הרשות (10.0.0.32) שולח FIN, ACK (10.0.0.32).
  - משמעות: הרשות מודיע "סימתי לשלוח מידע, אני רוצה לסגור את החיבור מציד".
2. סגירה מיידית / תקלה (Packet 12-13):
  - רואים הודעות של RST Reset (10.0.0.32).
  - הסבר: במקום תחיליך סגירה מסודר (שבו הצד השני שולח FIN משול ו-ACK), נשלחה פקודה "Reset". זה קורה בכך כלל כאשר צד אחד סגור את התוכנה בכוח, או שהפורט כבר לא מאזין, והוא "שובר" את החיבור מיד במקום לסגור אותו בעדינות.

```
11 0.045388 10.0.0.16 10.0.0.32 TCP 66 55748 -> 12345 [FIN, ACK] Seq=3694215536 Ack=1667296304 Win=131776 Len=0 Tsva1=3
12 0.050230 10.0.0.32 10.0.0.16 TCP 54 12345 -> 55748 [RST] Seq=1667296304 Win=0 Len=0
13 0.051229 10.0.0.32 10.0.0.16 TCP 54 12345 -> 55748 [RST] Seq=1667296304 Win=0 Len=0
```

> Frame 12: Packet, 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
> Ethernet II, Src: Intel\_A6:ba:9a (f4:06:69:46:ba:9a), Dst: de:6e:31:1c:f5:de
> Internet Protocol Version 4, Src: 10.0.0.32, Dst: 10.0.0.16
> Transmission Control Protocol, Src Port: 12345, Dst Port: 55748, Seq: 1667296304
 Source Port: 12345
 Destination Port: 55748
 [Stream index: 0]
 [Stream Packet Number: 12]
 > [Conversation completeness: Complete, WITH\_DATA (63)]
 [TCP Segment Len: 0]
 Sequence Number: 1667296304
 [Next Sequence Number: 1667296304]
 Acknowledgment Number: 0
 Acknowledgment number (raw): 0
 0101 .... = Header Length: 20 bytes (5)
 > Flags: 0x004 (RST)
 Window: 0
 [Calculated window size: 0]
 [Window size scaling factor: 128]

◦

תומך:  
הקוד של הלקוח:

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a project structure under "ASS2" named "part1" containing "client-server-wireshark.pcapng", "Dockerfile", "tcp\_client.py", and "tcp\_server.py".
- Code Editor:** Displays the contents of "tcp\_client.py".

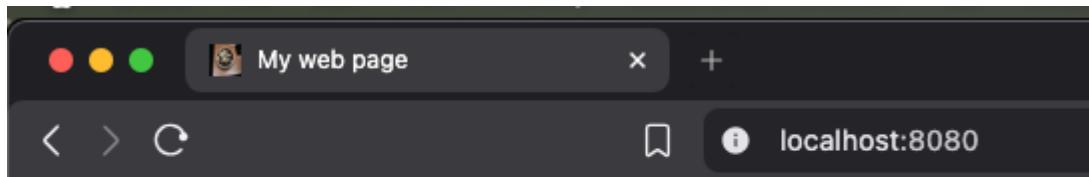
```
part1 > 🐍 tcp_client.py > ...
1 import socket
2 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
3 s.connect(('10.0.0.32', 12345))
4 s.send(b'Yoav Elhadad')
5 data = s.recv(100)
6 print("Server sent: ", data)
7 s.send(b'315053793')
8 data = s.recv(100)
9 print("Server sent: ", data)
10 s.close()
```
- Terminal:** Shows the output of running the client twice.

```
● part1|> python3 ./tcp_client.py
Server sent: b'YOAV ELHADAD'
Server sent: b''
● part1|> python3 ./tcp_client.py
Server sent: b'YOAV ELHADAD'
Server sent: b''
○ part1|>
```

## רשתות תרגיל 2 חלק 2

### דוגמא של עמוד הבית:

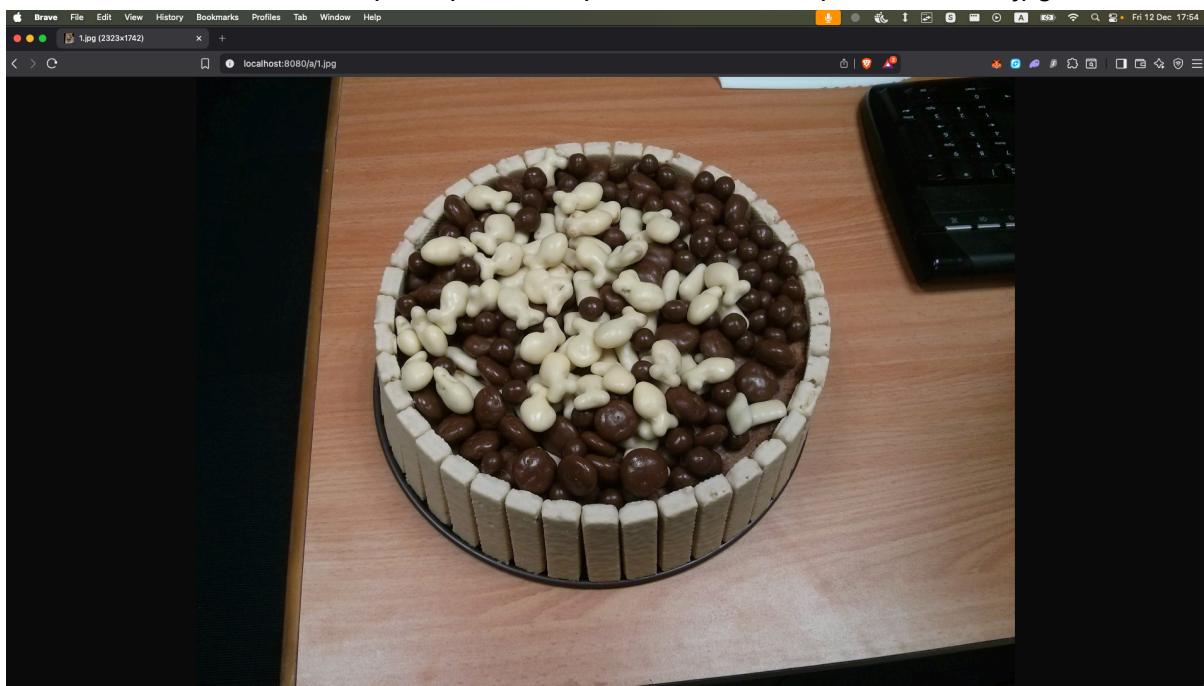
גישה ל / שמובילה לקבלת הקובץ index.html.



hello

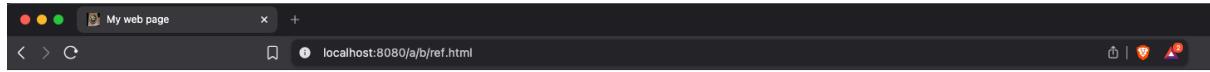
### דוגמא של גישה לתמונה באופן ישירות:

גישה לכתובת 1/a שמובילה לקבלת התמונה 1 בתיקיה a שבתוך התיקייה files.



## דוגמה של גישה לדף מורכב:

גישה לכתובות a/b/ref.html הוא דף אשר טוען תמונות נוספות וכאן מתבצעות קריאות רבות עם גישת connection keep alive עד אשר הדפסן מסיים לקבל את כל הקבצים אשר צריכים לו להציג העמוד.

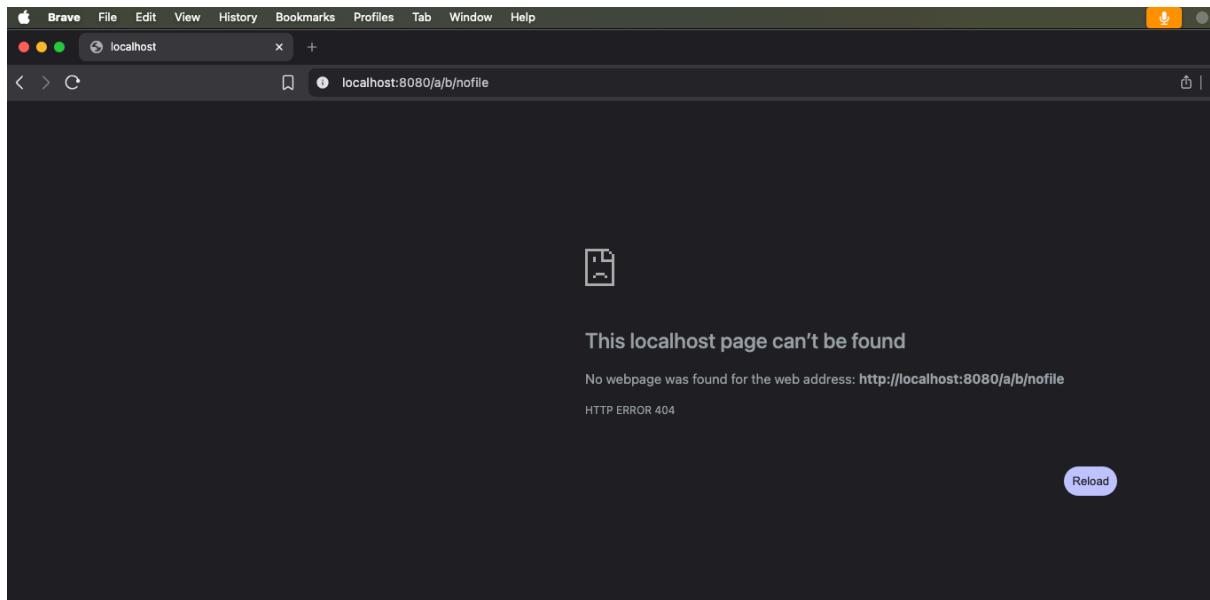


Amazing:



## דוגמה של גישה לדף שלא קיים:

גישה לכתובות a/b/nofile גורמת לנו לקבל את התשובה 404 not found כי הקובץ זהה לא קיים ב filesystem.



Wireshark Screenshot showing captured traffic on the Loopback interface (lo0). The packet list displays 26 entries, primarily TCP connections between 127.0.0.1 and 127.0.0.1. The details and bytes panes are visible below the main packet list.

No.	Time	Source	Destination	Protocol	Length	Info
11	47.192137	127.0.0.1	127.0.0.1	TCP	68	56875 → 8080 [SYN] Seq=3015514758 Win=65535 Len=0 MSS=16344 WS=64 TSval=26659
12	47.192184	127.0.0.1	127.0.0.1	TCP	68	8080 → 56875 [SYN, ACK] Seq=1528830962 Ack=3015514759 Win=65535 Len=0 MSS=16344 TSval=26659
13	47.192195	127.0.0.1	127.0.0.1	TCP	56	56875 → 8080 [ACK] Seq=3015514759 Ack=1528830963 Win=408320 Len=0 TSval=26659
14	47.192203	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 8080 → 56875 [ACK] Seq=1528830963 Ack=3015514759 Win=408320 TSval=26659
15	47.192335	127.0.0.1	127.0.0.1	HTTP	689	GET / HTTP/1.1
16	47.192351	127.0.0.1	127.0.0.1	TCP	56	8080 → 56875 [ACK] Seq=1528830963 Ack=3015515392 Win=407744 Len=0 TSval=23671
17	47.193252	127.0.0.1	127.0.0.1	HTTP	279	HTTP/1.1 200 OK
18	47.193264	127.0.0.1	127.0.0.1	TCP	56	56875 → 8080 [ACK] Seq=3015515392 Ack=1528831186 Win=408128 Len=0 TSval=26659
19	47.558855	127.0.0.1	127.0.0.1	HTTP	654	GET /favicon.ico HTTP/1.1
20	47.558878	127.0.0.1	127.0.0.1	TCP	56	8080 → 56875 [ACK] Seq=1528831186 Ack=3015515990 Win=407168 Len=0 TSval=23671
21	47.559292	127.0.0.1	127.0.0.1	HTTP	3247	HTTP/1.1 200 OK
22	47.559303	127.0.0.1	127.0.0.1	TCP	56	56875 → 8080 [ACK] Seq=3015515990 Ack=1528834377 Win=404992 Len=0 TSval=26659
23	48.560171	127.0.0.1	127.0.0.1	TCP	56	8080 → 56875 [FIN, ACK] Seq=1528834377 Ack=3015515990 Win=407168 Len=0 TSval=26659
24	48.560253	127.0.0.1	127.0.0.1	TCP	56	56875 → 8080 [ACK] Seq=3015515990 Ack=1528834378 Win=404992 Len=0 TSval=26659
25	49.114447	127.0.0.1	127.0.0.1	TCP	56	56875 → 8080 [FIN, ACK] Seq=3015515990 Ack=1528834378 Win=404992 Len=0 TSval=26659
26	49.114498	127.0.0.1	127.0.0.1	TCP	56	8080 → 56875 [ACK] Seq=1528834378 Ack=3015515991 Win=407168 Len=0 TSval=23671

הסבר קצר על חלק מהפקות

**פקטה 15 (HTTP GET):** זהוי בבקשת **HTTP** מסוג **GET** שנשלחת מהלוקה אל השרת. בבקשת זו, הלוקה מבקש לקבל את המשאב הראשי של האתר כלומר את דף הבית. התקשרות מתבצעת בתווך המחשב המקומי. זהוי תחילת השיחה ברמת האפליקציה, שבה הדפסון או התוכנה מבקשים מידע מהשרת המאזין.

**פקטה 16 (TCP ACK):** זהה הودעת אישור (ACK) בرمת פרוטוקול התעבורה TCP, הנשלחת מהשרת (פורט 8080) חזקה ללקוח. השרת מאותת ללקוח שהוא קיבל בהצלחה את בקשת-GET שנשלחה בפקטה הקודמת (מספר 15). נשים לב שגודל המידע (Len) הוא 0, מכיוון שהזיה הודעת בקרה טכנית בלבד ללא תוכן של דף אינטרנט. הودעה זו הכרחית כדי לשמר על אמינות החיבור ול證וד ששם מידע לא אבד בדרך לפני שהשרט מתחליל לעבד את המשובча.

**פקטה 17 HTTP 200 OK:** זהוי התשובה של השרת לבקשת המקורית, הכוללת את קוד הסטטוס **HTTP 200 OK**. המשמעות היא שהשרת מצא את הדף המבוקש, עיבד את הבקשת הצלחה, וכעת הוא שולח את הנתונים צורה ליקוט. בתור הפקטה זו נמצא ה-**Payload** (הטען), קלונר תוכן המקורי של דף **HTML** או המידע שהложен בבקשת. כאן מוצאים מחזור הבקשה-תגובה (Request-Response) המוצלח של פרוטוקול ה-**HTTP** עבור מסגרת זה.

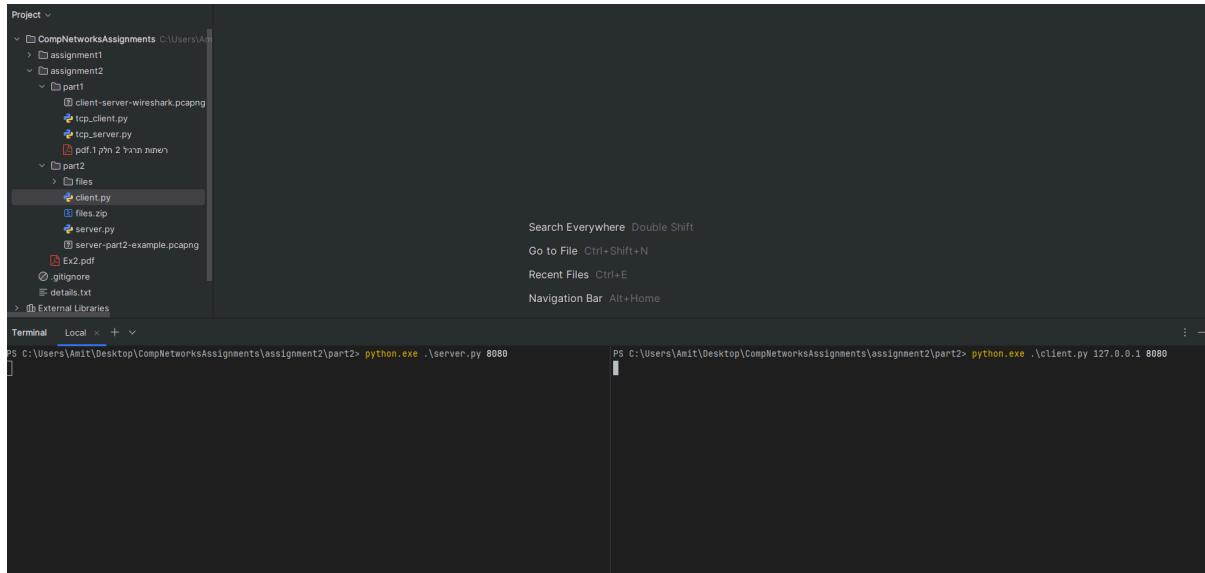
The Wireshark interface displays a network capture with the following details:

- Packets:** 697 · **Displayed:** 644 (92.4%) · **Dropped:** 0 (0.0%)
- Profile:** Default
- Selected Frame:** 17 (47.193252, 127.0.0.1 → 127.0.0.1, HTTP, 279 bytes)
- Conversation:**
  - Frame 17: 127.0.0.1 → 127.0.0.1 [HTTP] 279 HTTP/1.1 200 OK
  - Frame 18: 127.0.0.1 → 127.0.0.1 [TCP] 56 56875 → 8080 [ACK] Seq=3015515392 Ack=1528831186 Win=408128 Len=0 TSval=26659
  - Frame 19: 127.0.0.1 → 127.0.0.1 [HTTP] 654 GET /favicon.ico HTTP/1.1
  - Frame 20: 127.0.0.1 → 127.0.0.1 [TCP] 56 8080 → 56875 [ACK] Seq=1528831186 Ack=3015515990 Win=407168 Len=0 TSval=23671
  - Frame 21: 127.0.0.1 → 127.0.0.1 [HTTP] 3247 HTTP/1.1 200 OK
  - Frame 22: 127.0.0.1 → 127.0.0.1 [TCP] 56 56875 → 8080 [ACK] Seq=3015515990 Ack=1528834377 Win=404992 Len=0 TSval=26659
  - Frame 23: 127.0.0.1 → 127.0.0.1 [TCP] 56 8080 → 56875 [FIN, ACK] Seq=1528834377 Ack=3015515990 Win=407168 Len=0 TSval=23671
  - Frame 24: 127.0.0.1 → 127.0.0.1 [TCP] 56 56875 → 8080 [ACK] Seq=3015515990 Ack=1528834378 Win=404992 Len=0 TSval=26659
  - Frame 25: 127.0.0.1 → 127.0.0.1 [TCP] 56 56875 → 8080 [FIN, ACK] Seq=3015515990 Ack=1528834378 Win=404992 Len=0 TSval=23671
  - Frame 26: 127.0.0.1 → 127.0.0.1 [TCP] 56 8080 → 56875 [ACK] Seq=1528834378 Ack=3015515991 Win=407168 Len=0 TSval=23671
- Selected Conversation:**

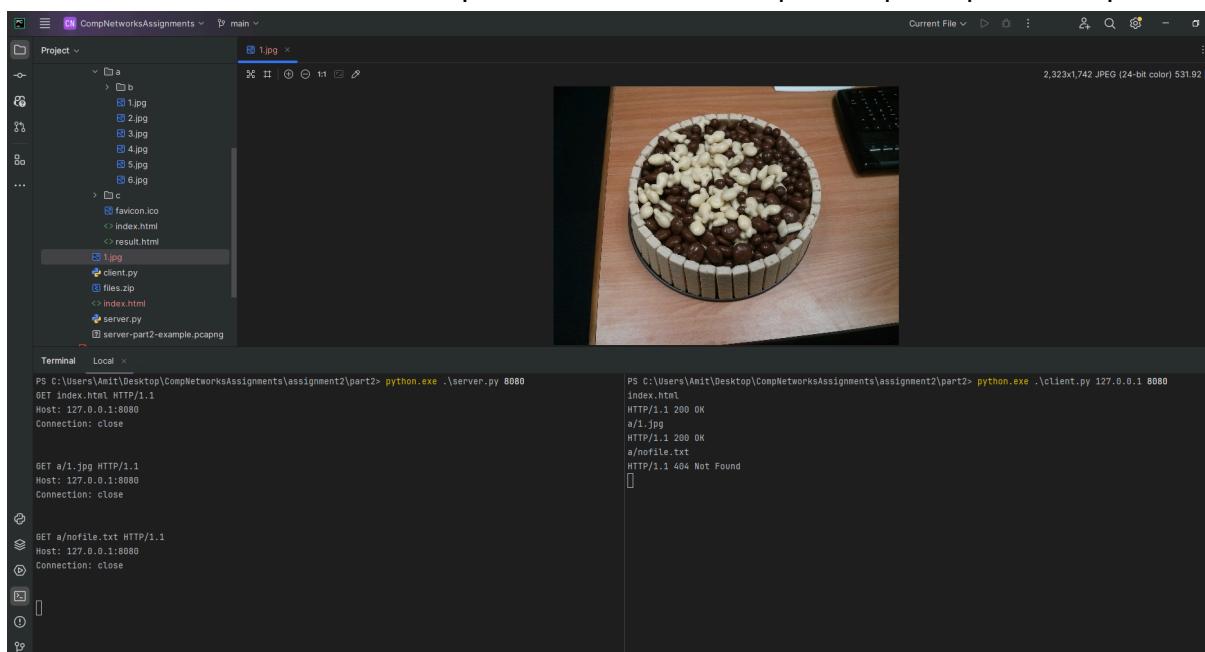
```
> Frame 17: Packet, 279 bytes on wire (2232 bits), 279 bytes captured (2232 bits)
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
< Transmission Control Protocol, Src Port: 8080, Dst Port: 56875, Seq: 1528830
  Source Port: 8080
  Destination Port: 56875
  [Stream index: 3]
  [Stream Packet Number: 7]
< [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 223]
  Sequence Number: 1528830963
  [Next Sequence Number: 1528831186]
  Acknowledgment Number: 3015515392
  1000 .... = Header Length: 32 bytes (8)
> Flags: 0x018 (PSH, ACK)
  Window: 6371
  [Calculated window size: 407744]
  [Window size scaling factor: 64]
  Checksum: 0xf0f7 [unverified]
```

## דוגמא של הרצה של הלקוטן:

ניתן לראות מה הרכזה של הלקוטן ושל השרת מהתרגיל, הלקוטן מבקש קבצים מהשרת, מקבל אותם בתור תשובה ושומר אותם מחדש על המחשב.  
בתמונה הראשונה רואים שאין עדין את הקבצים index.html ואת התמונה 1.



לאחר מכן אחורי שהלקוטן מבקש את הקבצים האלו הם נשמרים וניתן לראות אותם בתור תמונה גם.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	45	60978 → 61001 [ACK] Seq=1 Ack=1 Win=10231 Len=1
2	0.000017	127.0.0.1	127.0.0.1	TCP	56	61001 → 60978 [ACK] Seq=1 Ack=2 Win=10232 Len=0 SLE=1 SRE=2
3	1.393950	127.0.0.1	127.0.0.1	TCP	45	60976 → 61002 [ACK] Seq=1 Ack=1 Win=10228 Len=1
4	1.393971	127.0.0.1	127.0.0.1	TCP	56	61002 → 60976 [ACK] Seq=1 Ack=2 Win=10085 Len=0 SLE=1 SRE=2
5	7.593636	127.0.0.1	127.0.0.1	TCP	56	49470 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
6	7.593686	127.0.0.1	127.0.0.1	TCP	56	8080 → 49470 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
7	7.593746	127.0.0.1	127.0.0.1	TCP	44	49470 → 8080 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
8	7.593756	127.0.0.1	127.0.0.1	HTTP	112	GET index.html HTTP/1.1
9	7.593777	127.0.0.1	127.0.0.1	TCP	44	8080 → 49470 [ACK] Seq=1 Ack=69 Win=2619648 Len=0
10	7.594682	127.0.0.1	127.0.0.1	HTTP	271	HTTP/1.1 200 OK
11	7.594699	127.0.0.1	127.0.0.1	TCP	44	49470 → 8080 [ACK] Seq=69 Ack=228 Win=2619392 Len=0
12	7.594717	127.0.0.1	127.0.0.1	TCP	44	8080 → 49470 [FIN, ACK] Seq=228 Ack=69 Win=2619648 Len=0
13	7.594725	127.0.0.1	127.0.0.1	TCP	44	49470 → 8080 [ACK] Seq=69 Ack=229 Win=2619392 Len=0
14	7.594756	127.0.0.1	127.0.0.1	TCP	44	49470 → 8080 [FIN, ACK] Seq=69 Ack=229 Win=2619392 Len=0
15	7.594777	127.0.0.1	127.0.0.1	TCP	44	8080 → 49470 [ACK] Seq=229 Ack=70 Win=2619648 Len=0
16	14.849438	127.0.0.1	127.0.0.1	TCP	45	55883 → 61067 [ACK] Seq=1 Ack=1 Win=10206 Len=1
17	14.849480	127.0.0.1	127.0.0.1	TCP	56	61067 → 55883 [ACK] Seq=1 Ack=2 Win=10002 Len=0 SLE=1 SRE=2

## הסבר קצר על חלק מהפקודות

**פקודות 5-7 (TCP 3-Way Handshake):** זה הפקות שמבצעות את תהליך להקמת החיבור.

- **פקטה 5 (SYN):** הלוקו (פורט אקראי 49470) שלוח בקשה סנכרון לשרת (פורט 8080) כדי ליזום חיבור.
- **פקטה 6 (SYN, ACK):** השרת מאשר את הבקשה (ACK) ושלוח בקשה סנכרון משלו (SYN).
- **פקטה 7 (ACK):** הלוקו מאשר את סנכרון השרת.

**פקטה 8 (HTTP GET Request):** זאת בקשה של האפליקציה. הלוקו שלוח הודעה index.html לשרת. בפקטה זו הלוקו מבקש את הקובץ index.html שנמצא בתיקיה בשרת.

**פקטה 9 (TCP ACK):** הודעה אישור ברמת ה-TCP הנשלחת לлокו. השרת מאותת לлокו שהוא קיבל את בקשה GET שלו בהצלחה.

**פקטה 10 (HTTP 200 OK Response):** זאת התשובה של השרת. השרת איתר את הקובץ index.html ושולח את תוכן הקובץ.

**פקטה 11 (TCP ACK):** הלוקו מאשר לשרת שהוא קיבל את התשובה (אף פקטה 10).

**פקטה 12 (TCP FIN, ACK):** השרת יוזם את סגירת החיבור. הדגל FIN מסמן שהשרת סיים לשלוח מייד והוא מעוניין לנתק את התחברות.

**פקודות 13-15 (Connection Teardown):** תהליך סיום החיבור משני הצדדים:

- **פקטה 13:** הלוקו מאשר (ACK) את בקשה הסגירה של השרת.
- **פקטה 14:** הלוקו שלוח FIN משלו כדי לסגור את הצד שלו בחיבור.
- **פקטה 15:** השרת מאשר (ACK) את הסגירה של הלוקו.