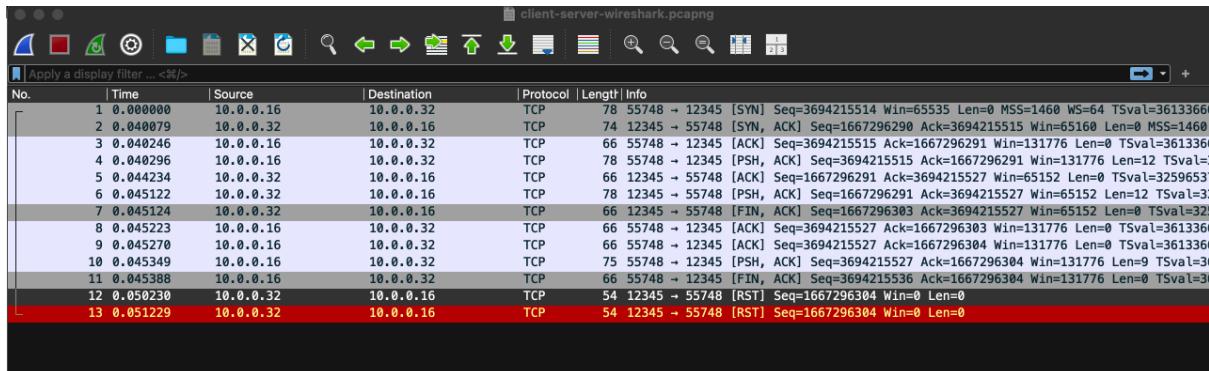


רשתות תרגיל 2 חלק 1

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) הרים את הקובץ `.tcp_client.py`



תהליך הקמת החיבור (Three-Way Handshake)

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

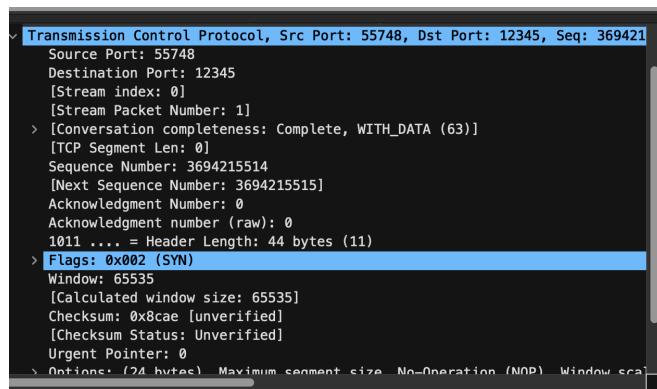
1. שלב ראשון (Packet 1) - שליחת SYN:

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

■ **Seq = 3694215514**

■ **Ack = 0**

משמעות: אני רוצה לפתוח איתך שיחה, המissor הסידורי שלי מתחילה ב- "3694215514". ○



2. שלב שני (Packet 2) - שליחת SYN, ACK:

- תיאור: השרת מקבל את הבקשה, ומחייב הודעה המכילה גם SYN (כדי לפתח ערזע משלו) וגם ACK (אישור הבקשה של הלקוח).

○ מספרים סידוריים:

■ **Seq = 1667296290**

■ **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 חוקם.
מספרים סידוריים:
- Seq (התקדם ב-1 לעומת חבילת 1, כי דגל ה-SYN צורך מספר סידורי אחד).
Ack = 1667296291 (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 two windows from Wireshark. The top window displays the packet details for a TCP segment. The sequence number is 3694215515, and the acknowledgment number is 1667296291. The header length is 32 bytes (0x1000). The flags include PSH and ACK (0x018). The window size is 2059, and the checksum is 0x4960 (unverified). The payload contains 12 bytes of data, which is the string "Yoav E lhadad". The bottom window shows the raw hex and ASCII bytes of the packet, starting with f4 06 69 46 ba 9a de 6e ... and ending with 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 תעודה זהה (Packet 8)

- תיאור: הלקוח (10.0.0.16) שולח את ההודעה השנייה המכילה את תעודה זהה: "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 FIN, ACK (10.0.0.32) שולח (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 Microsoft Visual Studio Code (VS Code) interface. The title bar says "ass2". The left sidebar has icons for file operations like Open, Save, Find, and Delete. The "EXPLORER" tab is open, showing a folder structure for "ASS2" containing a "part1" folder which includes "client-server-wireshark.pcapng", "Dockerfile", "tcp_client.py", and "tcp_server.py". The "tcp_client.py" file is currently selected and its code is displayed in the main editor area:

```
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()
```

Below the editor, there are tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", and "PORTS". The "TERMINAL" tab is active, showing command-line history:

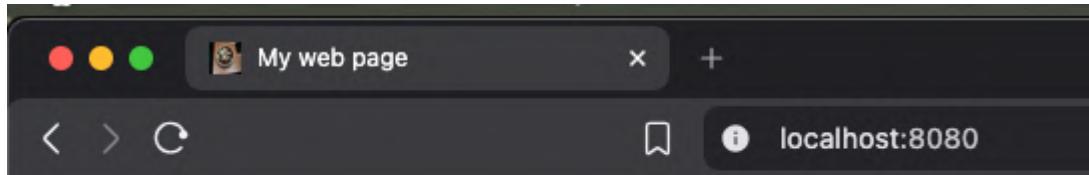
- part1|> python3 ./tcp_client.py
- Server sent: b'YOAV ELHADAD'
- part1|> python3 ./tcp_client.py
- Server sent: b'YOAV ELHADAD'
- part1|>

On the right side, there's a sidebar with icons for "zsh" and "zsh...".

רשתות תרגיל 2 חלק 2

דוגמא של עמוד הבית:

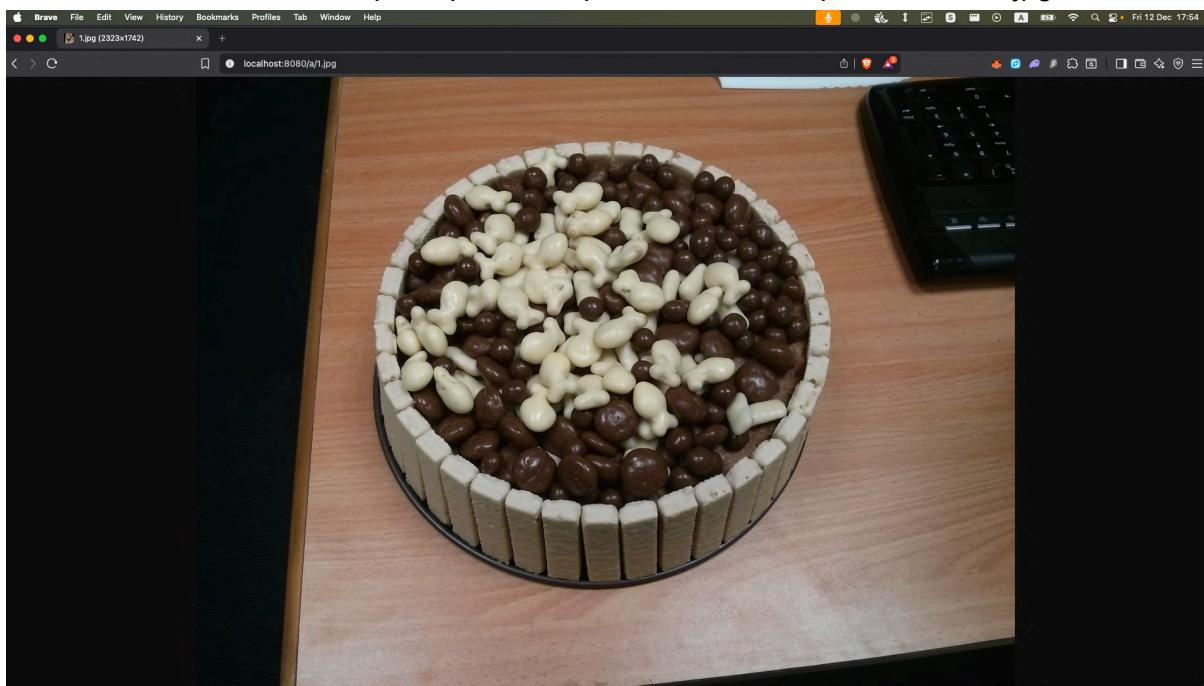
גישה ל / שמובילה לקבלת הקובץ index.html.



hello

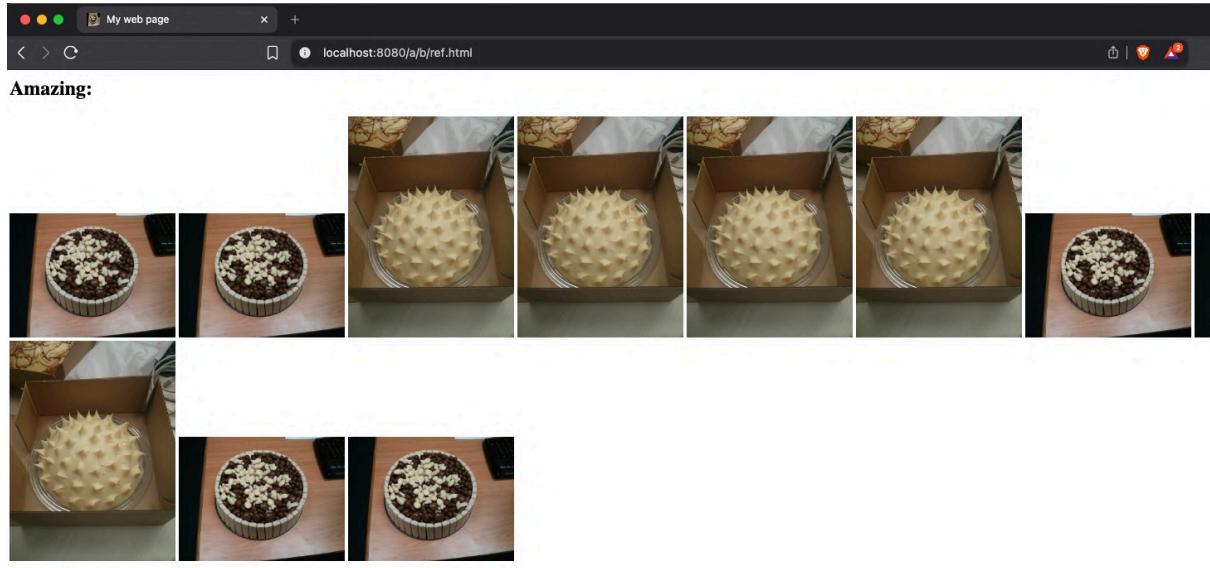
דוגמא של גישה לתמונה באופן ישירות:

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



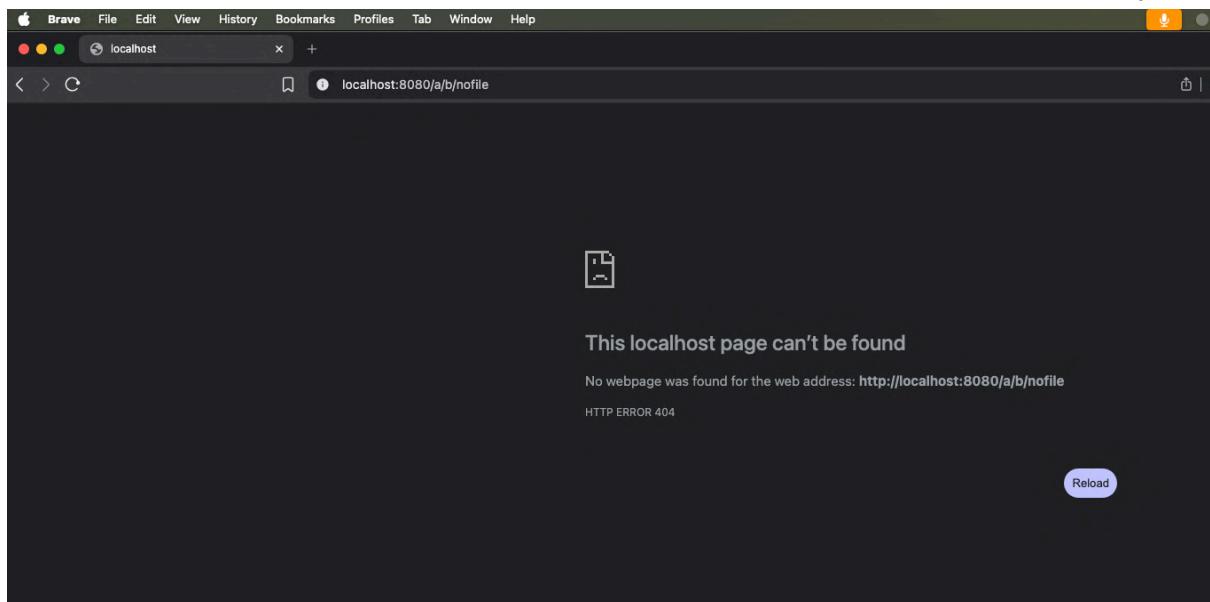
דוגמה של גישה לדף מורכב:

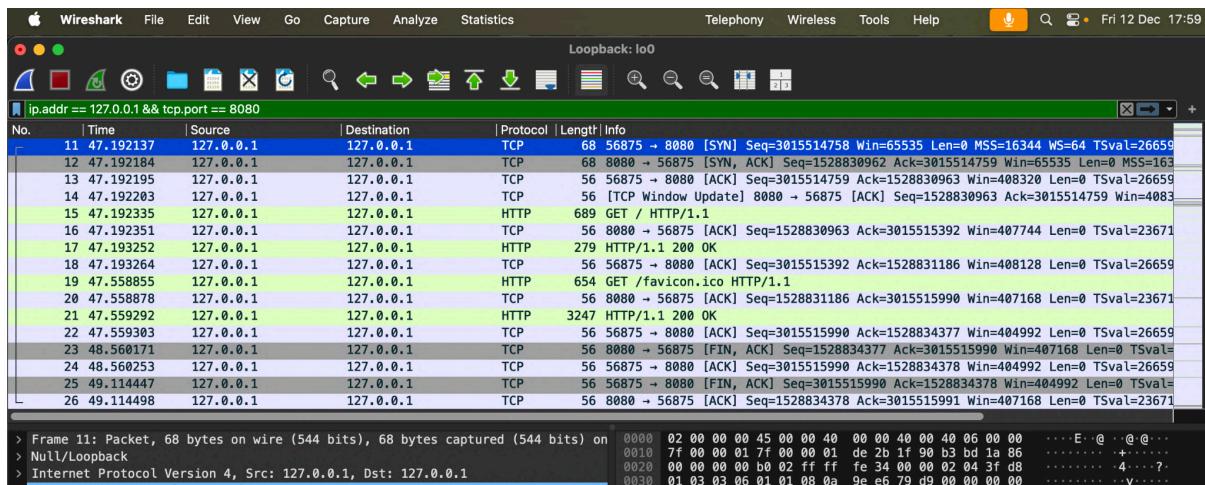
גישה לכתובות a/b/ref.html הוא דף אשר טוען תמונות נוספות וכאן מתבצעות קריאות רבות עם גישת connection keep alive עד אשר הדפסן מסיים לקבל את כל הקבצים אשר צריכים לו להציג העמוד.



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.



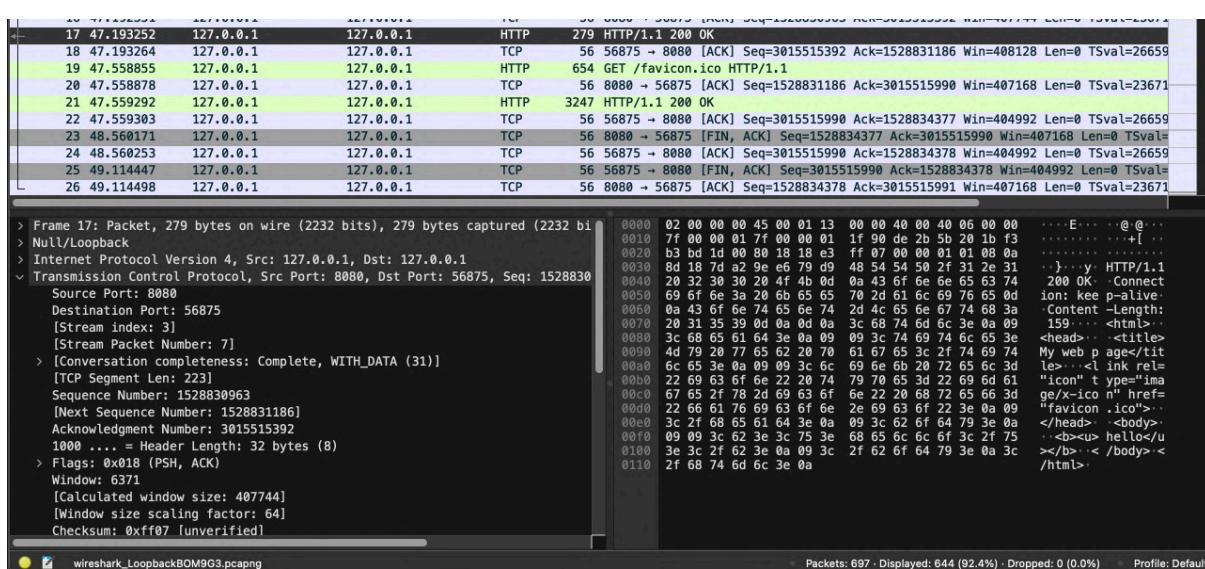


הסבר קצר על חלק מהפקות

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

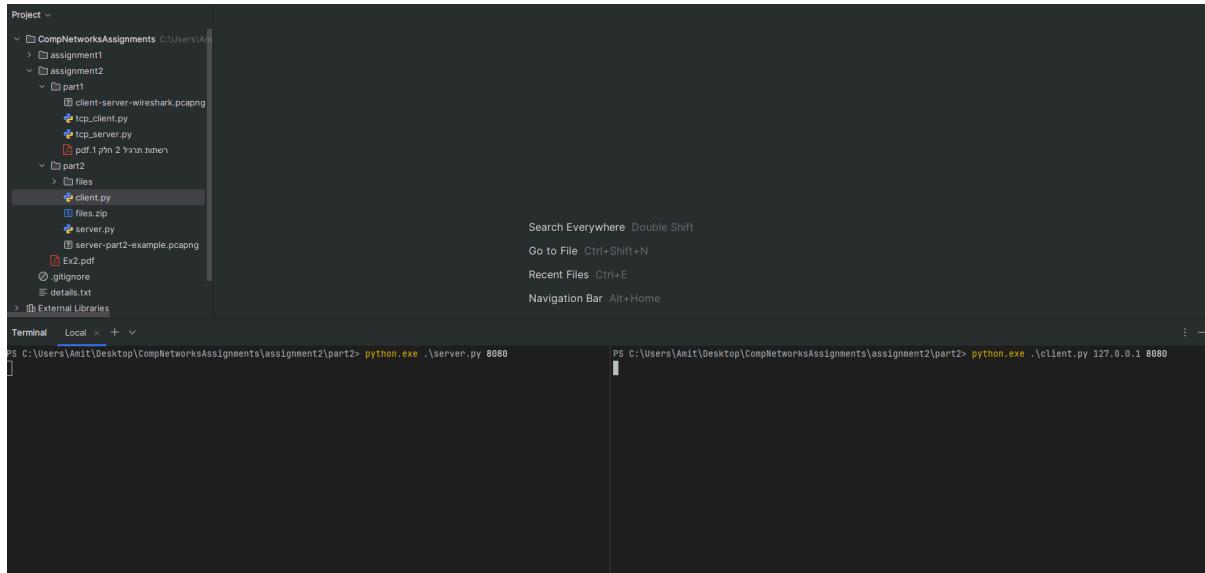
פקטה 16 (TCP ACK): זהה הודעת אישור (**ACK**) בرمת פרוטוקול התעבורה **TCP**, הנשלחת מהשרת (פורט 8080) לחזרה ללוקה. השרת מאותת ללקוח שהוא קיבל בהצלחה את בקשת-GET שנשלחה בפקטה הקודמת (מספר 15). נשים לב שגודלו המידיע (**Len**) הוא 0, מכיוון שהזיהוי הודעת בקרה טכנית בלבד ללא תוכן של דף אינטרנט. הودעה זו הכרחית כדי לשמר על אמינות החיבור ולוודא ששם מידע לא אבד בדרך לפניה. השרת מתחילה לעבד את התשובה.

פקטה 17 (HTTP 200 OK): זהה התשובה של השרת לבקשת המקורית, הכוללת את קוד הסטטוס **HTTP 200 OK**. המשמעות היא שהשרת מצא את הדף המבוקש, עיבד את הבקשה בהצלחה, וicutת הוא שולח את הנתונים לחזרה ללוקה. בתוך הפקטה הזו נמצא ה-**Payload** (הטען), כולם התוכן המקורי של דף ה-**HTML** או המידע שהлокה ביקש. כאן מסתומים מחרוזת הבקשה-תגובה (**Request-Response**) המוצלח של פרוטוקול HTTP עבור משאב זה.

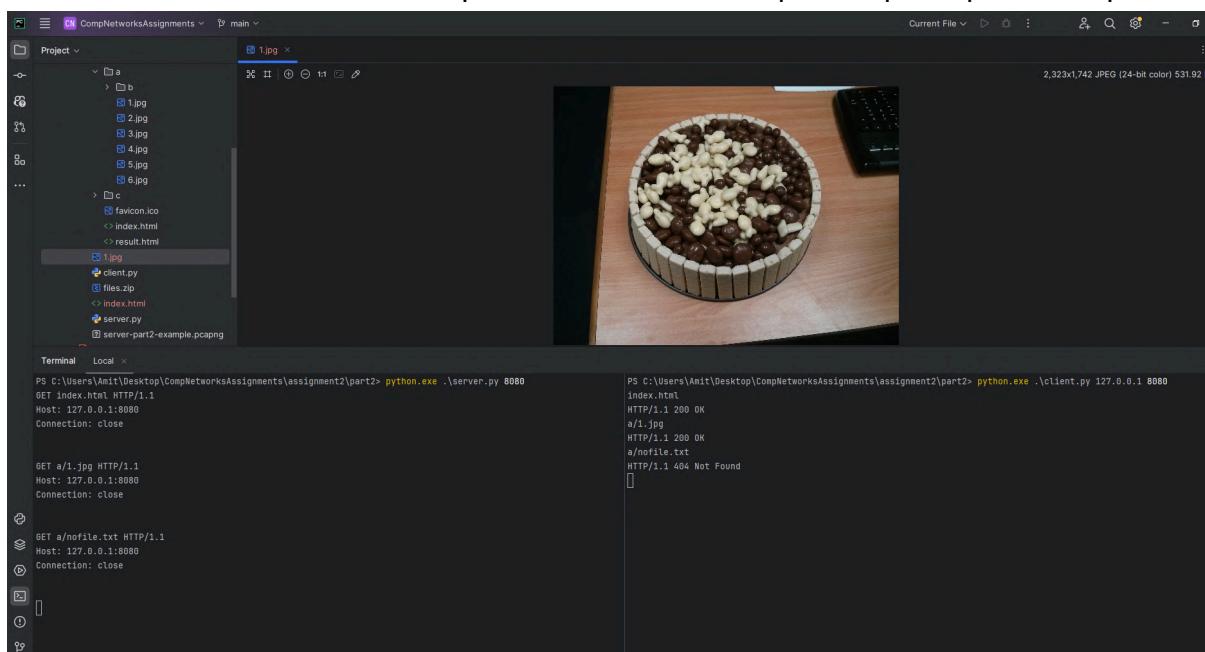


דוגמא של הרצה של הלקוטן:

ניתן לראות פה הרצה 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 אחרי 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

הסביר 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.