

ניתוח תעבורה בפרוטוקול TCP/IP

פרויקט גמר בקורס רשתות תקשורת מחשבים

סטודנט: יוסי אריאל
ת.ז: 209227842
מרצה: אפי פרופוס

תוכן עניינים

חלק ראשון: אריזת נתונים ולכידת מנות בעזרת Wireshark

עמוד 3-4	דרך יצירת קובץ CSV עם הודעות בשכבת היישום (שלבי עבודה)
עמוד 5-7	תהליך והסבר של דרך יצירת מנות
עמוד 8-9	תהליך והסבר של תהליך לכידה

חלק שני: כתיבת יישום רשת וניתוח תעבורה של אותו יישום

עמוד 10	הסבר כללי על מבנה המערכת
עמוד 10	הסבר כללי על מבנה הקוד
עמוד 11	הוראות הרצה והפעלה
עמוד 12-16	דוגמאות קלט ופלט
עמוד 17-22	ניתוח תעבורה של היישום עד שכבת הרשת (כולל)

חלק שלישי: תיאור שימוש בבינה מלאכותית

עמוד 23	מטרות שימוש
עמוד 23	דוגמאות לפרומפטים

אריזת נתונים ולכידת מנות בעזרת wireshark

דרך יצירת קובץ CSV עם הודעות בשכבת היישום-שלבי עבודה

שלב 1: הכנת קובץ CSV

קובץ קלט CSV המכיל הודעות בשכבת היישום בפרוטוקול HTTP נוצר. הקובץ נוצר בעזרת שימוש בבינה מלאכותית (AI) כפי שנרשם שניתן להשתמש בהוראות הפרויקט. הקובץ כולל:

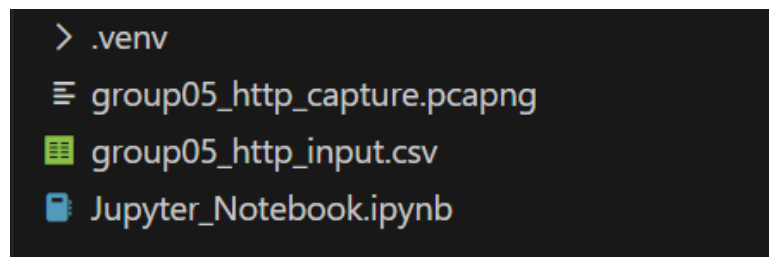
- ❖ מזהה הודעה
- ❖ פרוטוקול יישום
- ❖ מקור
- ❖ יעד
- ❖ תוכן ההודעה
- ❖ חותמת זמן יחסית

שם הקובץ: group05_http_input.csv

msg_id	app_protocol	src_app	dst_app	message	timestamp
1	HTTP	client_browser	web_server	GET /index.html	0.015
2	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.028
3	HTTP	client_browser	web_server	GET /style.css	0.042
4	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.057
5	HTTP	client_browser	web_server	GET /script.js	0.073
6	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.089
7	HTTP	client_browser	web_server	POST /login	0.115
8	HTTP	web_server	client_browser	HTTP/1.1 302 Redirect	0.134
9	HTTP	client_browser	web_server	GET /dashboard	0.162
10	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.187

שלב 2: שימוש במחברת Jupyter

- ❖ המחברת Jupyter נפתחה בצורה תקינה על ידי שימוש ב VS code.
- ❖ קובץ ה CSV נטען בצורה תקינה.
- ❖ שלבי הקוד במחברת הורצו בצורה תקינה.
- ❖ נוצרה הדמיית יצירת אריזת נתונים ב TCP.
- ❖ נוצרה תעבורה שתלכד ב Wireshark.

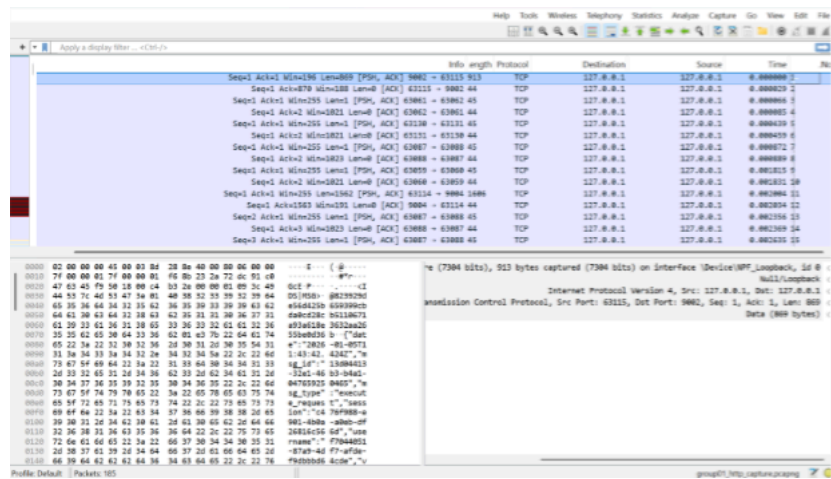


דרך יצירת קובץ CSV עם הודעות בשכבת היישום-שלבי עבודה (המשך)

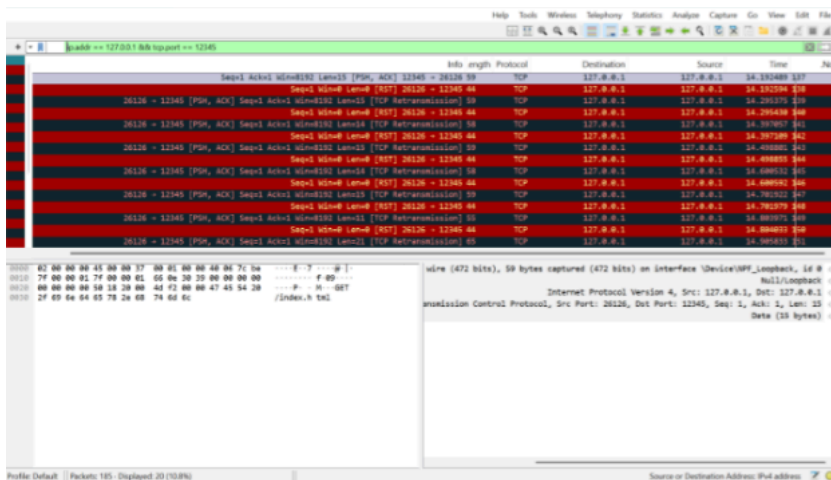
שלב 3: לכידה בWireshark

- ❖ הפעל Wireshark כראוי.
- ❖ המחברת Jupyter הורצה ונוצר תעבורה ב Wireshark.
- ❖ עצרתי את הלכידה והקובץ נשמר בפורמט .pcapng (הקובץ נשמר ללא סינון, הוא נשמר במלואו ולאחר מכן ניתן לסנן כמבוקש).
- ❖ הסימוציה רצה על localhost ובפורט 12345, ולכן התעבורה הרלוונטית לא עברה דרך פורט 80 (פורט 80 רלוונטי לתעבורת HTTP רגילה באינטרנט).
- לכן, לצורך ניתוח השתמשתי במסנן: `tcp.port == 12345 && ip.addr == 127.0.0.1` כפי שונתב במחברת jupyter.

להלן צילום מסך של שמירת הקובץ (ללא מסנן):



להלן צילום מסך של שמירת הקובץ (עם מסנן):



תיאור והסבר של תהליך אריזת מנות

נתאר את תהליך אריזת המנות (הפקטות) לפי שלבים ונראה תיאור של הפרטים בעזרת Wireshark. בקובץ CSV כל שורה שבו מייצגת הודעה בשכבת היישום. כפי שלמדנו, כאשר היא נשלחת היא עוברת תהליך של אריזת מנות. לצורך הדגמת השלבים ופירוט בעזרת Wireshark נבחרה באופן רנדומלי פקטה מס' 141 לצורך המחשה והדגמה.

שלב ראשון:

קיימת רק ההודעה בעמודת message בCSV:

msg_id	app_protocol	src_app	dst_app	message	timestamp
1	HTTP	client_browser	web_server	GET /index.html	0.015
2	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.028
3	HTTP	client_browser	web_server	GET /style.css	0.042
4	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.057
5	HTTP	client_browser	web_server	GET /script.js	0.073
6	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.089
7	HTTP	client_browser	web_server	POST /login	0.115
8	HTTP	web_server	client_browser	HTTP/1.1 302 Redirect	0.134
9	HTTP	client_browser	web_server	GET /dashboard	0.162
10	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.187

שלב שני:

אותן ההודעות נארזות ב-TCP ומשתלבים להן פורט מקור ופורט יעד. (בפרויקט הפורט יעד הוא 12345).

No.	Time	Source	Destination	Protocol	Length	Info
137	14.192489	127.0.0.1	127.0.0.1	TCP	59	Seq=1 Ack=1 Win=8192 Len=15 [PSH, ACK] 12345 → 26126
138	14.192594	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
139	14.295375	127.0.0.1	127.0.0.1	TCP	59	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission]
140	14.295430	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
141	14.397057	127.0.0.1	127.0.0.1	TCP	58	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14 [TCP Retransmission]
142	14.397109	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
143	14.498801	127.0.0.1	127.0.0.1	TCP	59	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission]
144	14.498855	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
145	14.600532	127.0.0.1	127.0.0.1	TCP	58	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14 [TCP Retransmission]
146	14.600592	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
147	14.701922	127.0.0.1	127.0.0.1	TCP	59	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission]
148	14.701979	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
149	14.803971	127.0.0.1	127.0.0.1	TCP	55	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP Retransmission]
150	14.804033	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
151	14.905833	127.0.0.1	127.0.0.1	TCP	65	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=21 [TCP Retransmission]

Frame 141: Packet, 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface \Device\NPF_{...} id 0 <
 Null/Loopback <
 Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1 <
 Transmission Control Protocol, Src Port: 26126, Dst Port: 12345, Seq: 1, Ack: 1, Len: 14 <

ניתן לראות את הפרטים ב-Transmission Control Protocol של אותה פקטה 141:

- ❖ פורט מקור: 26126
- ❖ פורט יעד: 12345
- ❖ הדגלים ACK ו-SNK המסמנים העברת נתונים דרך TCP.
- ❖ ACK=1 הנתונים התקבלו בצורה תקינה בצד המקבל.

תיאור והסבר של תהליך אריזת מנות (המשך)

שלב שלישי:

קטע ה TCP נארז בתוך חבילת IP (כלומר, נארז בשכבת האינטרנט).

No.	Time	Source	Destination	Protocol	Length	Info
137	14.192489	127.0.0.1	127.0.0.1	TCP	59	Seq=1 Ack=1 Win=8192 Len=15 [PSH, ACK] 12345 → 26126
138	14.192594	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
139	14.295375	127.0.0.1	127.0.0.1	TCP	59	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission]
140	14.295436	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
141	14.397057	127.0.0.1	127.0.0.1	TCP	58	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14 [TCP Retransmission]
142	14.397109	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
143	14.498801	127.0.0.1	127.0.0.1	TCP	59	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission]
144	14.498855	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
145	14.600532	127.0.0.1	127.0.0.1	TCP	58	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14 [TCP Retransmission]
146	14.600592	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
147	14.701922	127.0.0.1	127.0.0.1	TCP	59	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission]
148	14.701979	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
149	14.803971	127.0.0.1	127.0.0.1	TCP	55	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP Retransmission]
150	14.804033	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
151	14.905833	127.0.0.1	127.0.0.1	TCP	65	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=21 [TCP Retransmission]

Frame 141: Packet, 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface \Device\NPF_{Loopback}, id 0	<
Null/Loopback	<
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	<
Transmission Control Protocol, Src Port: 26126, Dst Port: 12345, Seq: 1, Ack: 1, Len: 14	<

כאשר נסתכל ב Internet Protocol, ניתן לראות:

❖ כתובת מקור: 127.0.0.1

❖ כתובת יעד: 127.0.0.1

★ הפרויקט מבוצע על מחשב יחיד ולכן הכתובת מקור זהה לכתובת יעד.

תיאור והסבר של תהליך אריזת מנות (המשך)

שלב רביעי:

אריזת נתונים בשכבת הקישור.

No.	Time	Source	Destination	Protocol	Length	Info
137	14.192489	127.0.0.1	127.0.0.1	TCP	59	Seq=1 Ack=1 Win=8192 Len=15 [PSH, ACK] 12345 → 26126
138	14.192594	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
139	14.295375	127.0.0.1	127.0.0.1	TCP	59	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission]
140	14.295430	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
141	14.397057	127.0.0.1	127.0.0.1	TCP	58	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14 [TCP Retransmission]
142	14.397109	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
143	14.498801	127.0.0.1	127.0.0.1	TCP	59	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission]
144	14.498855	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
145	14.600533	127.0.0.1	127.0.0.1	TCP	58	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14 [TCP Retransmission]
146	14.600592	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
147	14.701922	127.0.0.1	127.0.0.1	TCP	59	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission]
148	14.701979	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
149	14.803971	127.0.0.1	127.0.0.1	TCP	55	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP Retransmission]
150	14.804033	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
151	14.905833	127.0.0.1	127.0.0.1	TCP	65	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=21 [TCP Retransmission]

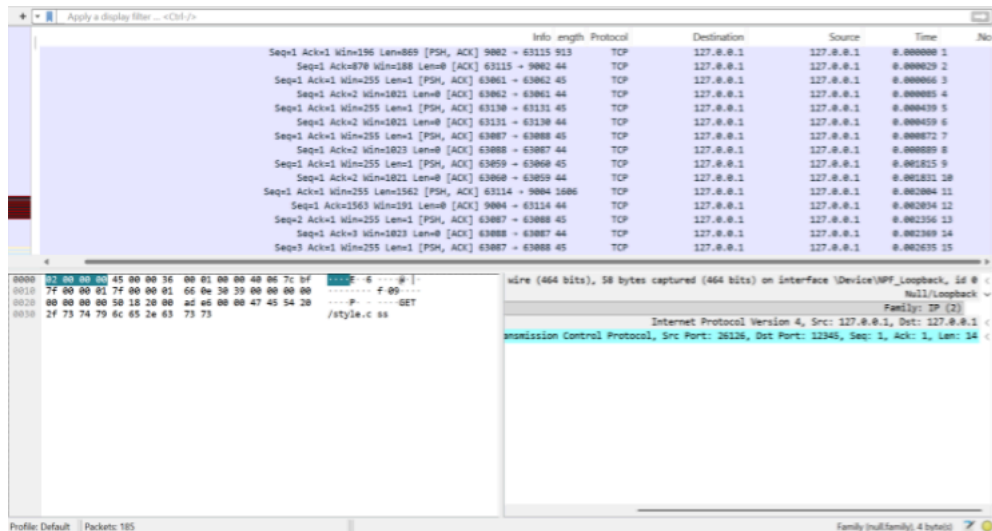
Frame 141: Packet, 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface \Device\NPF_{...} id 0 <
Null/Loopback <
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1 <
Transmission Control Protocol, Src Port: 26126, Dst Port: 12345, Seq: 1, Ack: 1, Len: 14 <

בחלונית המידע עבור פקטה 141, ניתן לראות שמופיע בשורה הראשונה Frame:

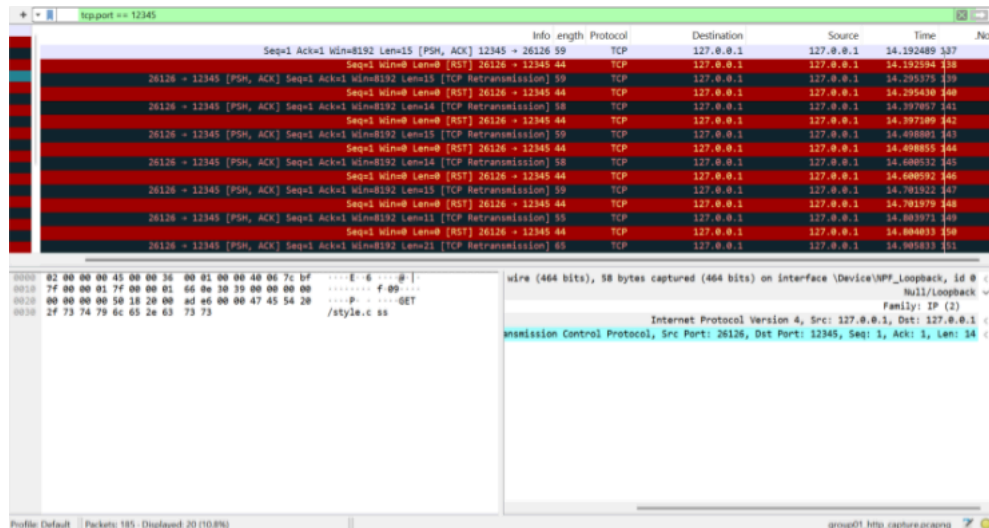
- ❖ מספר פקטה 141.
- ❖ הגודל שהיא נלכדה 58 בתים.

תיאור והסבר של תהליך הלכידה

כאשר הקוד במחברת jupyter הורץ, הופעלה יחדיו מערכת wireshark לפי ההוראות. התקשורת בפרויקט מתבצעת על כתובת 127.0.0.1 ניתן לראות באופן כללי שאכן יש פקטות (185 פקטות) ואכן מתבצעת תעבורה:



לאחר שלוכדה תעבורה, הורץ שלב 5 במחברת jupyter שיצר תעבורה (TCP). כאשר נבצע פילטר של tcp.port==12345, נוכל לראות כמות הפקטות TCP ואת מספר הפורט שאכן תואם לקוד שהורץ:



תיאור והסבר של תהליך הלכידה (המשך)

לבסוף, לאחר שהלכידה נעצרה (הקובץ גם נשמר) ניתן לבחור פקטה ולבצע ניתוח מעמיק של שכבות התקשורת שהיא עברה (כפי שהראיתי קודם לכן).
באופן כללי, אבחר באופן רנדומלי פקטה מספר 153 ונוכל לקבל את הפרטים המתאימים:

The image displays a Wireshark packet capture analysis. The top pane shows a list of packets, with packet 153 highlighted. The middle pane shows the packet details for packet 153, including Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The bottom pane shows the packet bytes in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
141	14.397057	127.0.0.1	127.0.0.1	TCP	58	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14 [TCP Retransmission]
142	14.397109	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
143	14.498801	127.0.0.1	127.0.0.1	TCP	59	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission]
144	14.498855	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
145	14.600532	127.0.0.1	127.0.0.1	TCP	58	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14 [TCP Retransmission]
146	14.600592	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
147	14.701922	127.0.0.1	127.0.0.1	TCP	59	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission]
148	14.701979	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
149	14.803971	127.0.0.1	127.0.0.1	TCP	55	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP Retransmission]
150	14.804033	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
151	14.905833	127.0.0.1	127.0.0.1	TCP	65	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=21 [TCP Retransmission]
152	14.905886	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
153	15.007706	127.0.0.1	127.0.0.1	TCP	58	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14 [TCP Retransmission]
154	15.007762	127.0.0.1	127.0.0.1	TCP	44	Seq=1 Win=0 Len=0 [RST] 26126 → 12345
155	15.109430	127.0.0.1	127.0.0.1	TCP	59	26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15 [TCP Retransmission]

Frame 153: Packet, 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface \Device\NPF_{...} id 0
Null/Loopback
Family: IP (2)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 26126, Dst Port: 12345, Seq: 1, Ack: 1, Len: 14

No.: 153 - Time: 15.007706 - Source: 127.0.0.1 - Destination: 127.0.0.1 - Protocol: TCP - Length: 58 - Info: [TCP Retransmission] 26126 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14

כתיבת יישום רשת וניתוח תעבורה של אותו יישום

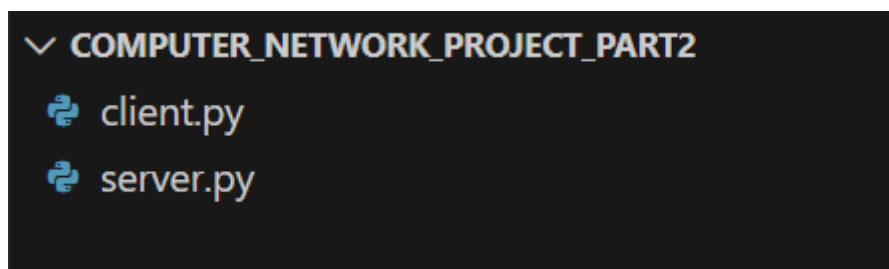
הסבר כללי על המערכת

בחלק השני של הפרויקט, נבנתה מערכת המדמה שרת-לקוח על פי פרוטוקול TCP. המערכת מאפשרת למספר משתמשים (לקוחות) להתחבר לשרת מרכזי שמתווך ביניהם. המערכת מאפשרת יצירת צ'אטים נפרדים בין המשתמשים ובכך ניתן לשלוח ולקבל הודעות פרטיות ממשתמשים ספציפיים. כל זה מתבצע בזמן אמת כאשר ההודעות אינן משודרות לכלל המשתמשים. בצד של השרת, השתמשתי במבנה נתונים מסוג מילון כדי לשמור את המשתמשים (לקוחות) שמתחברים לשרת. לכל משתמש יש מזהה ספציפי. מבנה נתונים מסוג מילון מאפשר גישה מהירה למשתמשים וגם ניהול וניתוב נוחים בין המשתמשים.

- ❖ השרת מאזין לחיבורים נכנסים באמצעות פורט קבוע (5000), ומנהל את כל החיבורים באמצעות sockets.
- ❖ כל לקוח מתחבר לשרת עם שם משתמש שהוא בוחר, והשרת משמש כמתווך בין המשתמשים ואינו משתתף בשיחות עצמן.
- ❖ המערכת תומכת במספר המשתמשים המחוברים בו זמנית, ומטפלת במקרי ניתוק לא צפויים. (במקרה של ניתוק משתמש, השרת מעדכן את רשימת המשתמשים המחוברים, מעדכן על כך, והלקוח מציג הודעת מערכת מתאימה בצ'אט הפרטי).
- ❖ הודעות נשלחות באופן פרטי בלבד בין שני משתמשים נבחרים, ואינן משודרות לשאר המשתמשים המחוברים למערכת.
- ❖ התקשורת מתבצעת על גבי 127.0.0.1 (localhost)
- ❖ בצד הלקוח קיים ממשק גרפי (GUI) שמאפשר עבודה נוחה עם המערכת.

הסבר כללי על מבנה הקוד

- המערכת מחולקת לשני קבצי קוד עיקריים.
1. צד שרת (server.py) שאחראי על החיבורים, ניהול המשתמשים והכמות, וניתוב ההודעות.
 2. צד לקוח (client.py) שאחראי על חיבור לשרת, שליחת וקבלת הודעות (בנוסף גם קיים בצד לקוח מימוש לממשק GUI שנוח יותר למשתמש).



הוראות הרצה והפעלה

ראשית, הפרויקט נבנה בסביבת עבודה: visual studio code בשפת python. לכן, עדיפות להפעיל את המערכת גם בסביבת עבודה visual studio code עם תמיכה ב python או בסביבת עבודה אחרת התומכת ב python.

1. יש לפתוח את תיקיית הפרויקט בסביבת הפיתוח.
2. יש להריץ את קובץ השרת server.py:
פתיחת new terminal.
לאחר מכן, יש להקליד "python server.py" (כמובן ללא גרשיים).
3. יש להריץ את קובץ הלקוח client.py:
פתיחת new terminal.
לאחר מכן, יש להקליד "python client.py" (כמובן ללא גרשיים).
4. לאחר הרצת קובץ הלקוח ייפתח חלון GUI המאפשר למשתמש להזין שם משתמש ולהתחבר לשרת.
כאשר תקליד שם משתמש ותלחץ "connect" - תראה את שם המשתמש בסרגל בצד.
5. ניתן להריץ מספר פעמים את קובץ הלקוח client.py (לפי פתיחת new terminal כפי שמתואר בסעיף 3) ולפתוח מספר משתמשים בו זמנית עם שמות משתמשים שונים.
כל המשתמשים המחוברים יופיעו בסרגל בצד.
6. השרת יציג מידע על חיבורים וניתוקים של משתמשים, כולל פרטי החיבור (כתובת IP ומספר פורט).
7. לאחר התחברות המשתמשים, ניתן לבחור משתמש מהרשימה ולהתחיל שיחה פרטית.
ההודעות נשלחות באופן פרטי בין שני המשתמשים בלבד, ושאר המשתמשים אינם יכולים לראות או לגשת אליהן.

הערה חשובה:

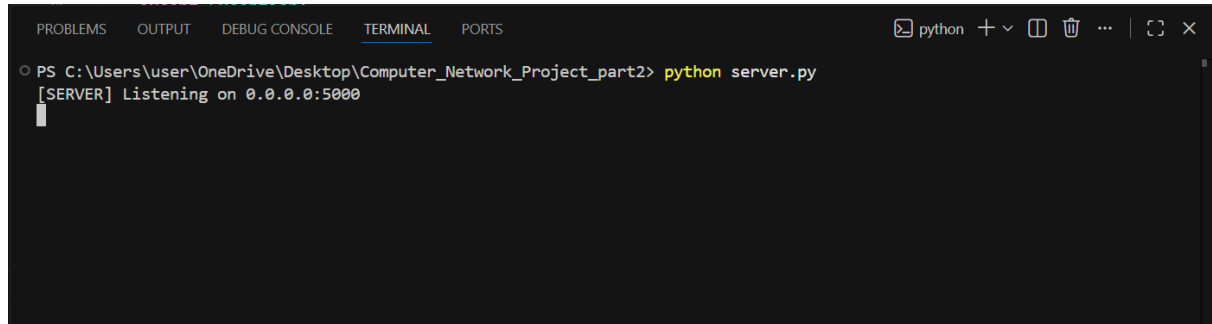
המערכת פועלת על כתובת localhost ובפורט קבוע, ולכן אינה דורשת הגדרות רשת מיוחדות.
במידת הצורך, יש להגדיר:
❖ כתובת שרת 127.0.0.1
❖ פורט 5000

דוגמאות קלט ופלט

❖ חיבור שרת (server)

נריץ ב terminal

python server.py



```
PS C:\Users\user\OneDrive\Desktop\Computer_Network_Project_part2> python server.py
[SERVER] Listening on 0.0.0.0:5000
```

נקבל הודעה שהשרת אכן התחבר + פרטי חיבור (כתובת IP ומספר פורט).

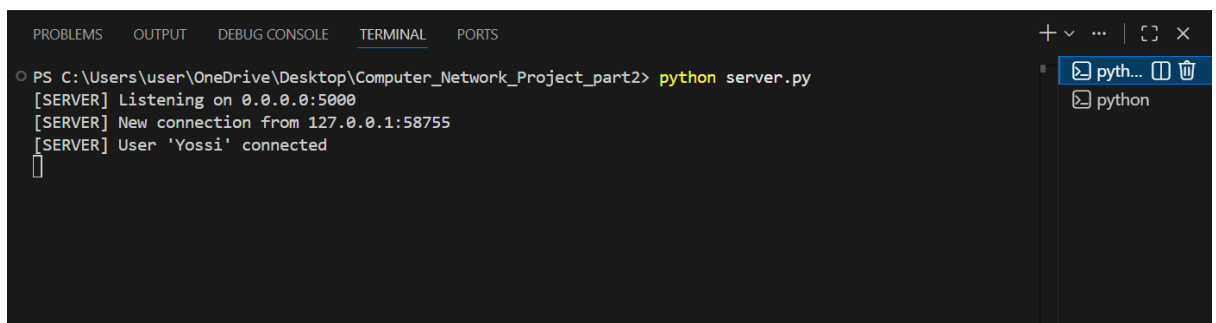
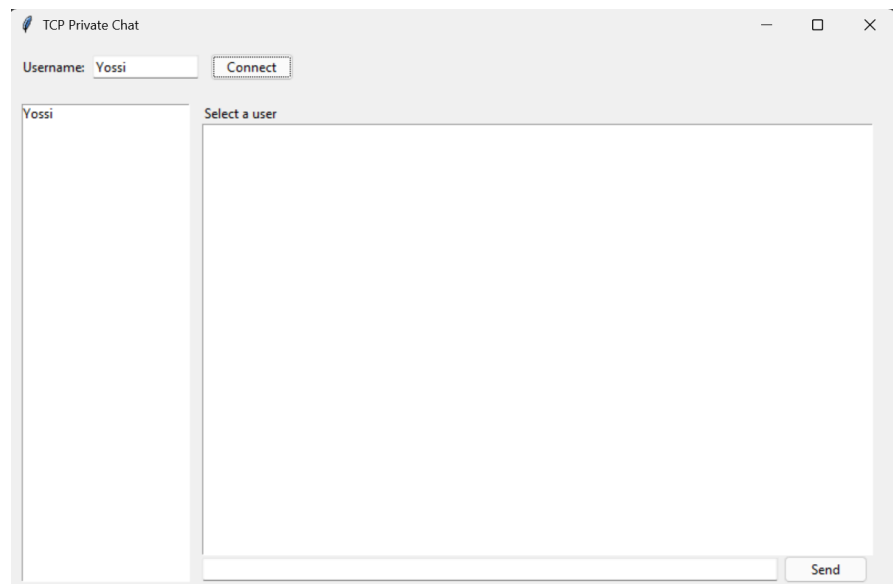
❖ חיבור משתמש (client)

נריץ ב terminal חדש python client.py

נקליד שם משתמש ונתחבר.

נראה בחלונית GUI את שם המשתמש הראשון שהתחבר מופיע בסרגל בצד.

ובחלונית השרת ניתן לראות את פרטי ההתחברות שלו (כתובת IP ומספר פורט) ואת שם המשתמש.

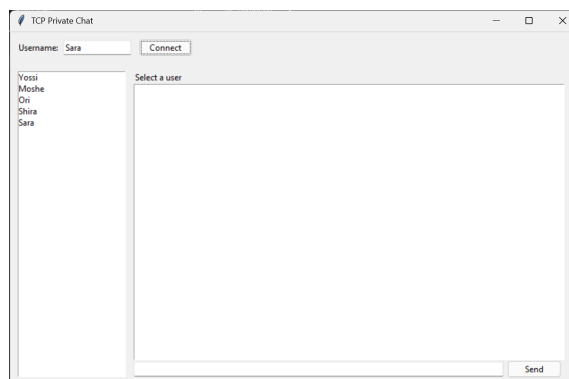
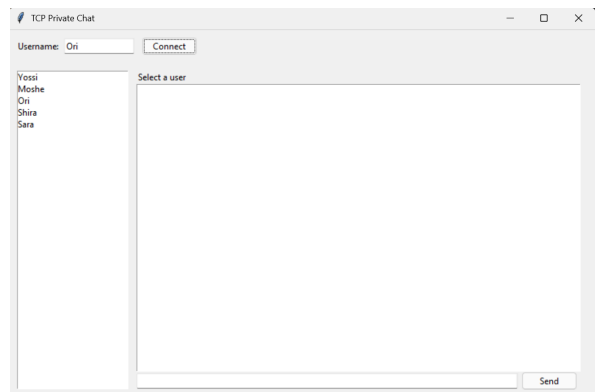
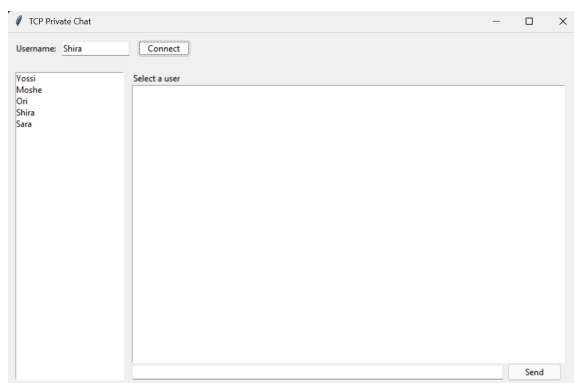
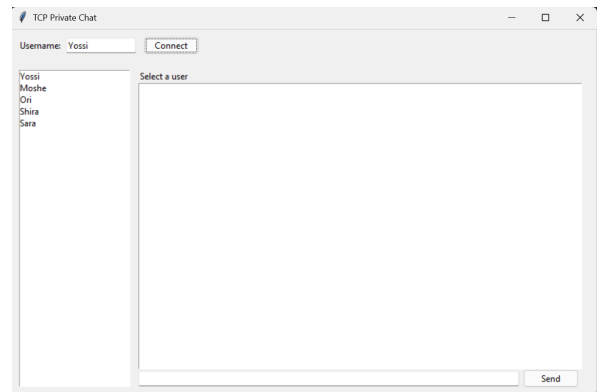
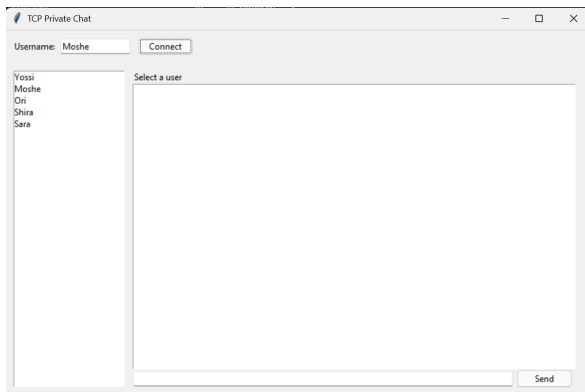


```
PS C:\Users\user\OneDrive\Desktop\Computer_Network_Project_part2> python server.py
[SERVER] Listening on 0.0.0.0:5000
[SERVER] New connection from 127.0.0.1:58755
[SERVER] User 'Yossi' connected
```

דוגמאות קלט ופלט (המשך)

❖ חיבור 5 משתמשים בו זמנית

5 מחוברים בו זמנית התחברו לשרת: "Yossi", "Moshe", "Ori", "Shira", "Sara"
כל שמות המשתמשים מופיעים בסרגל צד בחלונות GUI ובנוסף השרת עדכן על כל החיבורים ופרטי החיבור שלהם (כתובת IP ומספר פורט).

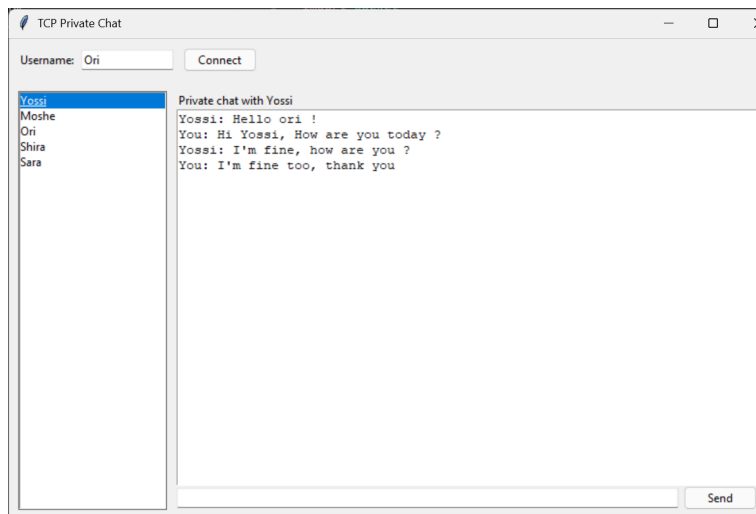
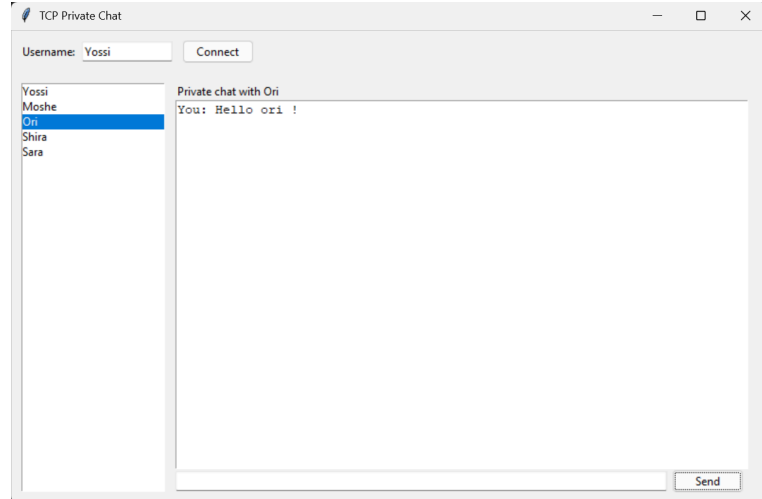
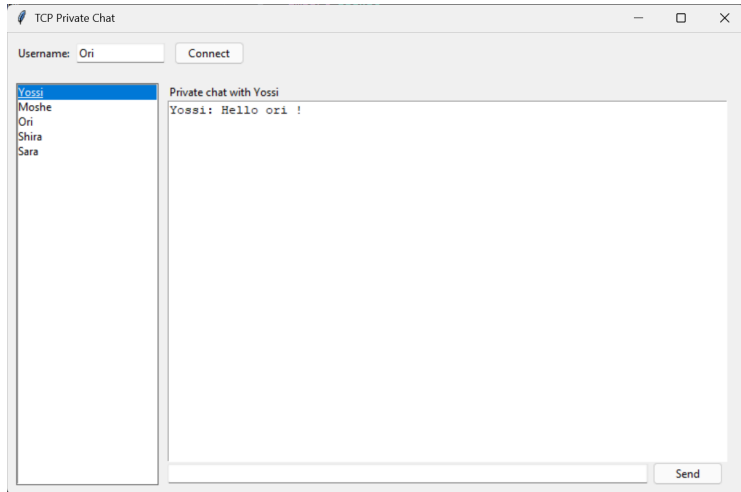


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\User\OneDrive\Desktop\Computer_Network_Project_part2> python server.py
[SERVER] Listening on 0.0.0.0:5000
[SERVER] New connection from 127.0.0.1:58755
[SERVER] User 'Yossi' connected
[SERVER] New connection from 127.0.0.1:58818
[SERVER] User 'Moshe' connected
[SERVER] New connection from 127.0.0.1:58821
[SERVER] User 'Ori' connected
[SERVER] New connection from 127.0.0.1:58824
[SERVER] User 'Shira' connected
[SERVER] New connection from 127.0.0.1:58827
[SERVER] User 'Sara' connected
```

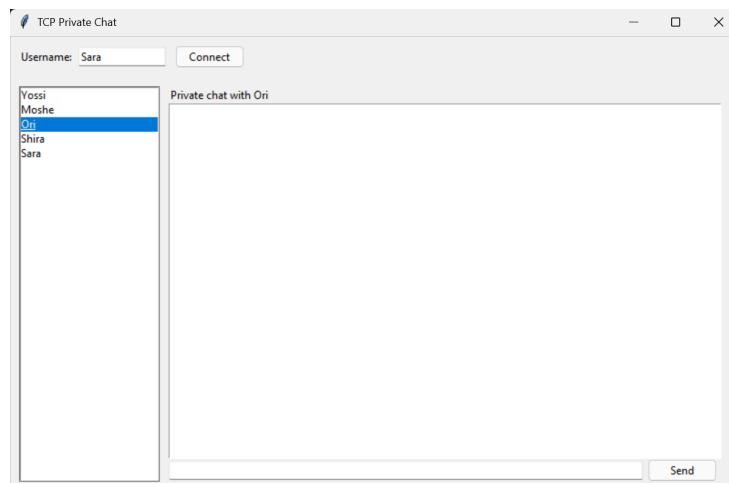
דוגמאות קלט ופלט (המשך)

❖ פתיחת צ'אט פרטי ושליחת הודעה

לצורך הדגמה, המשתמש "Yossi" בוחר את המשתמש "Ori" ושולח לו הודעה.
לאחר מכן, מתנהל ביניהם שיחה פרטית שלא גלויה לשאר המשתמשים.

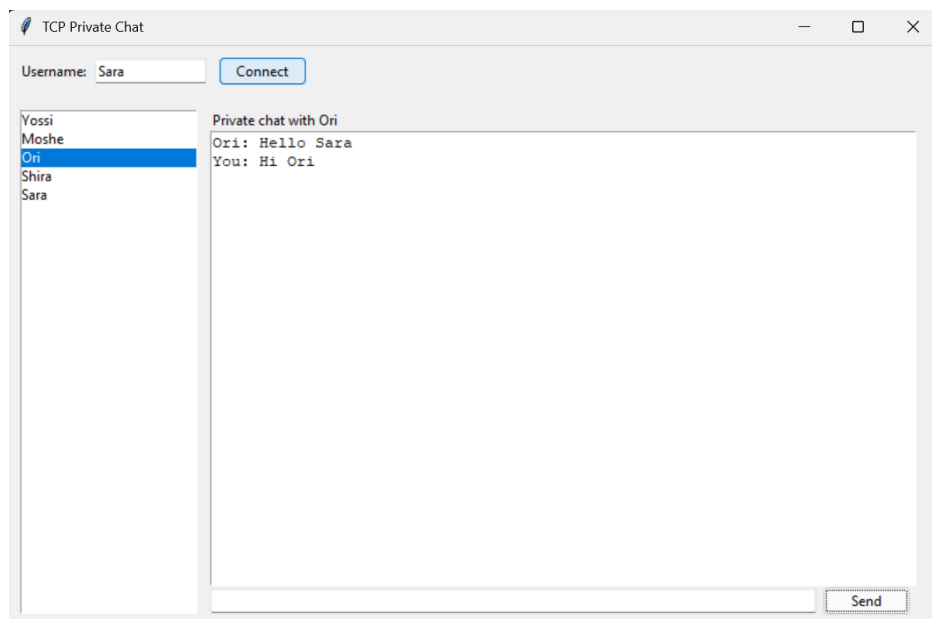
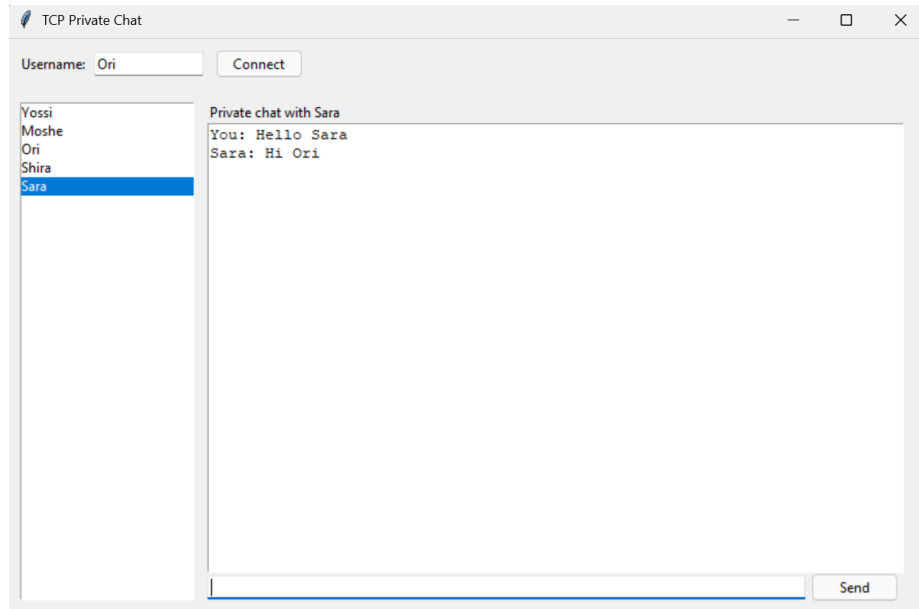


נראה גם שהצ'אט אכן פרטי ומשתמש אחר (לצורך הדוגמה "Shira" ו"Ori") לא יכול לראות:



דוגמאות קלט ופלט (המשך)

❖ פתיחת צ'אט נוסף בין משתמשים תוך כדי צ'אט אחר עם משתמש שונה.
לצורך הדגמה, ניקח את "Ori" וננהל שיחה עם "Sara" בזמן שיש לו שיחה עם "Yossi".

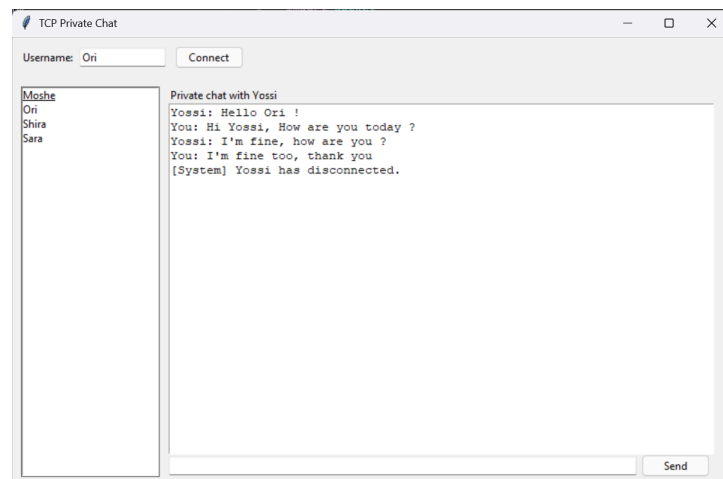


דוגמאות קלט ופלט (המשך)

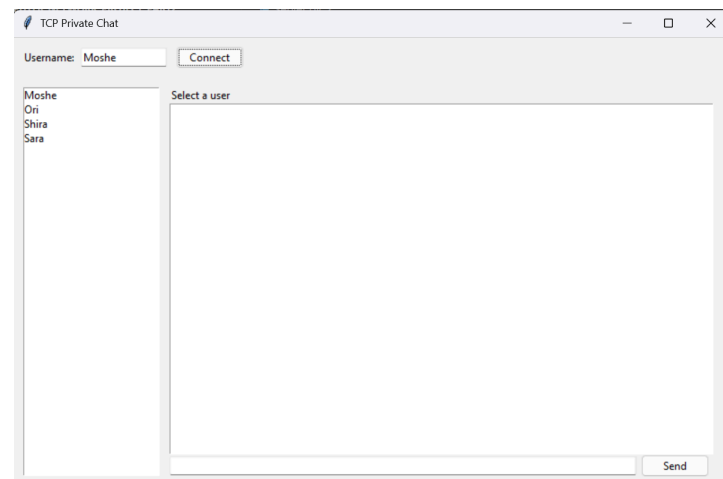
❖ ניתוק משתמש מהמערכת

בהמשך לדוגמא בו הייתה שיחה בין "Yossi" ו "Ori" נבצע ניתוק של "Yossi" מהמערכת.

נראה שמתקבלת הודעה לאורי על ניתוק של "Yossi".
וגם ניתן לראות שהוא ירד בסרגל בצד (הוא ירד אצל כל המשתמשים).



אצל משתמש אחר (למשל "Moshe") ניתן לראות ש "Yossi" לא מופיע גם בסרגל צד לאחר הניתוק שלו.



```
PS C:\Users\user\OneDrive\Desktop\Computer_Network_Project_part2> python server.py
[SERVER] Listening on 0.0.0.0:5000
[SERVER] New connection from 127.0.0.1:63357
[SERVER] User 'Yossi' connected
[SERVER] New connection from 127.0.0.1:63360
[SERVER] User 'Moshe' connected
[SERVER] New connection from 127.0.0.1:63366
[SERVER] User 'Ori' connected
[SERVER] New connection from 127.0.0.1:63369
[SERVER] User 'Shira' connected
[SERVER] New connection from 127.0.0.1:63372
[SERVER] User 'Sara' connected
[SERVER] User 'Yossi' disconnected
```

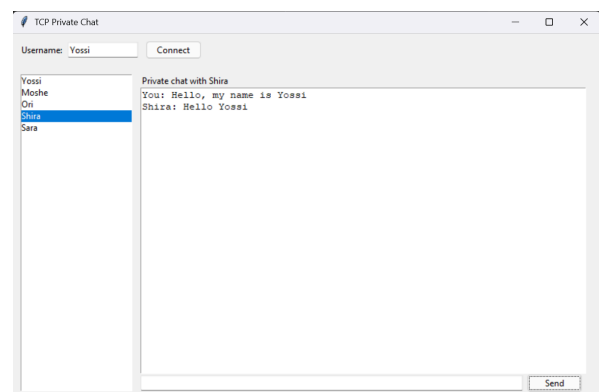

ניתוח תעבורה של היישום עד שכבת הרשת (כולל)

שליחת הודעה ממשתמש ספציפי למשתמש אחר ספציפי:

ראשית, פתחתי את Wireshark על מצב Npcap Loopback Adapter והתחלתי את לכידת התעבורה. לאחר מכן, עברתי VS code והרצתי את המערכת תקין. ניתן לציין לצורך ההדגמה, פתחתי שוב 5 משתמשים שונים בשמות: "Shira", "Ori", "Moshe", "Yossi", "Sara". שלחתי הודעה מהמשתמש "Yossi" למשתמש ספציפי "Shira" והודעה היא: "Hello, my name is Yossi". ו"Shira" שלחה לו הודעה בחזרה "Hello Yossi". לאחר מכן, ביצעתי ניתוק של המשתמש "Yossi" מהמערכת. לבסוף, חזרתי בחזרה ל Wireshark, עצרתי את הלכידה, ופילטרי בפילטר מתאים על מנת לראות את הפקטות המתאימות.

שכבת היישום:

בפרויקט, שכבת היישום ממומשת בצד הלקוח ובצד השרת. בצד הלקוח, נוצר צ'אט בממשק GUI שדרכו המשתמש יכול לשלוח את ההודעות. בצד השרת, מתבצע קבלת ההודעה, ניתוח ההודעה ושליחתה לשאר המשתמשים במערכת. נוצרה הודעה מהמשתמש "Yossi" בצ'אט, כלומר נוצרה הודעה מהמשתמש (הלקוח) לשרת. השרת מפיץ את ההודעה למשתמש הספציפי "Shira" הנבחר ונוצרה הודעה בחזרה גם.



כאשר גם נעבור ל Wireshark, נבחר פקטה של ההודעה. נבחר את פקטה 66 (פקטה של ההודעה) לצורך הדגמה ונוכל לראות את תוכן ההודעה בצורת טקסט וגם בקוד ASCII. כמובן נראה גם את שם המשתמש השולח.

Frame 66: Packet, 109 bytes on wire (872 bits), 109 bytes captured (872 bits) on interface \Device\NPF_{...} Loopback, id 0
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 50330, Dst Port: 5000, Seq: 40, Ack: 315, Len: 65
Data (65 bytes)

```
0000  02 00 00 00 45 00 00 69 64 d1 00 00 06 00 00  ....E..i d@....  
0010  7f 00 00 01 7f 00 00 01 c4 9a 13 88 ab f7 6e b3  ....n.....  
0020  54 68 08 30 50 18 00 fe c3 12 00 00 7b 22 74 79  -h@....."ty  
0030  70 65 22 3a 20 22 64 6d 22 2c 20 22 74 6f 22 3a  pe": "dm ", "to":  
0040  20 22 53 68 69 72 61 22 2c 20 22 74 65 78 74 22  "Shira", "text"  
0050  3a 20 22 48 65 64 6c 6f 2c 20 6d 75 20 6e 61 6d  : "Hello , my nam  
0060  65 20 69 73 20 59 6f 73 73 69 22 7d 0a          e is Yossi";
```

ניתוח תעבורה של היישום עד שכבת הרשת (כולל) (המשך)

שכבת התעבורה:

ההודעה מועברת בפרוטוקול TCP.

כפי שלמדנו בהרצאה, זה פרוטוקול אמין, וגם שומר על סדר ההודעות (חשוב בממשק צ'אט).

נבחר כמו קודם את פקטה Frame 66 (פקטה של ההודעה) לצורך הדגמה.

tcp.port == 5000

No.	Time	Source	Destination	Protocol	Length	Info
59	77.131480	127.0.0.1	127.0.0.1	TCP	44	Seq=40 Ack=273 Win=65024 Len=0 [ACK] 5000 → 50333
60	77.131661	127.0.0.1	127.0.0.1	TCP	119	Seq=147 Ack=38 Win=65280 Len=75 [PSH, ACK] 50336 → 5000
61	77.131705	127.0.0.1	127.0.0.1	TCP	44	Seq=38 Ack=222 Win=65280 Len=0 [ACK] 5000 → 50336
62	77.131767	127.0.0.1	127.0.0.1	TCP	119	Seq=89 Ack=40 Win=65280 Len=75 [PSH, ACK] 50339 → 5000
63	77.131795	127.0.0.1	127.0.0.1	TCP	44	Seq=40 Ack=164 Win=65280 Len=0 [ACK] 5000 → 50339
64	77.131833	127.0.0.1	127.0.0.1	TCP	119	Seq=22 Ack=39 Win=65280 Len=75 [PSH, ACK] 50342 → 5000
65	77.131854	127.0.0.1	127.0.0.1	TCP	44	Seq=39 Ack=97 Win=65280 Len=0 [ACK] 5000 → 50342
66	115.321613	127.0.0.1	127.0.0.1	TCP	109	Seq=40 Ack=315 Win=65024 Len=65 [PSH, ACK] 5000 → 50330
67	115.321664	127.0.0.1	127.0.0.1	TCP	44	Seq=315 Ack=105 Win=65280 Len=0 [ACK] 50330 → 5000
68	115.322480	127.0.0.1	127.0.0.1	TCP	111	Seq=164 Ack=40 Win=65280 Len=67 [PSH, ACK] 50339 → 5000
69	115.322522	127.0.0.1	127.0.0.1	TCP	44	Seq=40 Ack=231 Win=65280 Len=0 [ACK] 5000 → 50339
70	133.504547	127.0.0.1	127.0.0.1	TCP	94	Seq=40 Ack=231 Win=65280 Len=50 [PSH, ACK] 5000 → 50339
71	133.504605	127.0.0.1	127.0.0.1	TCP	44	Seq=231 Ack=90 Win=65280 Len=0 [ACK] 50339 → 5000
72	133.505514	127.0.0.1	127.0.0.1	TCP	96	Seq=315 Ack=105 Win=65280 Len=52 [PSH, ACK] 50330 → 5000
73	133.505551	127.0.0.1	127.0.0.1	TCP	44	Seq=105 Ack=367 Win=65024 Len=0 [ACK] 5000 → 50330

Frame 66: Packet, 109 bytes on wire (872 bits), 109 bytes captured (872 bits) on interface \Device\NPF_{...} id 0 <
Null/Loopback <
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1 <
Transmission Control Protocol, Src Port: 50330, Dst Port: 5000, Seq: 40, Ack: 315, Len: 65 <
Data (65 bytes) <

```
0000 02 00 00 00 45 00 00 69 64 d1 40 00 80 06 00 00  ....E..i d @....
0010 7f 00 00 01 7f 00 00 01 c4 9a 13 88 ab f7 6e b3  ....n
0020 94 68 d8 30 50 18 00 fe c3 12 00 00 7b 22 74 79  ..h-0P...{"ty
0030 70 65 22 3a 20 22 64 6d 22 2c 20 22 74 6f 22 3a  pe": "dm ", "to":
0040 20 22 53 68 69 72 61 22 2c 20 22 74 65 78 74 22  "Shira", "text"
0050 3a 20 22 48 65 6c 6c 6f 2c 20 6d 79 20 6e 61 6d  : "Hello , my nam
0060 65 20 69 73 20 59 6f 73 73 69 22 7d 0a  e is Yos si"}.
```

No.: 66 · Time: 115.321613 · Source: 127.0.0.1 · Destination: 127.0.0.1 · Protocol: TCP · Length: 109 · Info: 50330 → 5000 [PSH, ACK] Seq=40 Ack=315 Win=65024 Len=65

ניתן לראות שמופיע בפרטי פקטה Frame 66:

- ❖ שימוש בפרוטוקול: TCP (Transmission Control Protocol)
- ❖ פורט מקור: 50330, זה הפורט הזמני של המשתמש.
- ❖ פורט יעד: 5000, זה הפורט הקבוע של השרת.
- ❖ דגלי ACK ו: PSH
- ❖ PSH- העברת נתונים.
- ❖ ACK- אישור על קבלת נתונים בצורה תקינה.
- ❖ אורך הנתונים: 65 byte

ניתוח תעבורה של היישום עד שכבת הרשת (כולל) (המשך)

שכבת הרשת:

בשכבת הרשת ניתן לראות בעזרת ה Wireshark את השימוש בפרוטוקול IPv4.

The image shows a Wireshark packet capture interface. The top pane displays a list of captured packets, with packet 66 selected. The middle pane shows the details of packet 66, which is an Internet Protocol Version 4 packet. The bottom pane shows the raw data of the packet, which is an HTTP GET request.

Packet 66: Seq=40 Ack=273 Win=65024 Len=0 [ACK] 5000 → 50333 44 TCP 127.0.0.1 127.0.0.1 77.131480 59

Frame 66: Packet, 109 bytes on wire (872 bits), 109 bytes captured (872 bits) on interface \Device\NPF_{...} id 0 < Null/Loopback < Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1 < Transmission Control Protocol, Src Port: 50330, Dst Port: 5000, Seq: 40, Ack: 315, Len: 65 < Data (65 bytes) <

0000 02 00 00 00 45 00 00 69 64 d1 40 00 00 06 00 00E..i d@....
0010 7f 00 00 01 7f 00 00 01 c4 9a 13 88 ab f7 6e b3n
0020 94 68 d8 30 50 18 00 fe c3 12 00 00 7b 22 74 79 .h-0P.....{ "ty
0030 70 65 22 3a 20 22 64 6d 22 2c 20 22 74 6f 22 3a pe": "dm ", "to":
0040 20 22 53 68 69 72 61 22 2c 20 22 74 65 78 74 22 "Shira", "text"
0050 3a 20 22 48 65 6c 6c 6f 2c 20 6d 79 20 6e 61 6d : "Hello , my nam
0060 65 20 69 73 20 59 6f 73 73 69 22 7d 0a e is Yos si"}.

No.: 66 · Time: 115.321613 · Source: 127.0.0.1 · Destination: 127.0.0.1 · Protocol: TCP · Length: 109 · Info: 50330 → 5000 [PSH, ACK] Seq=40 Ack=315 Win=65024 Len=65

ניתן לראות שאכן מופיע:

❖ כתובת מקור: 127.0.0.1

❖ כתובת יעד: 127.0.0.1

זה תקין שהכתובת המקור זהה לכתובת היעד משום שכל התקשורת בוצעה על localhost. כלומר, כאן הפרויקט בוצע על מחשב יחיד, כל המשתמשים/לקוחות והשרת נמצאים על אותו המחשב.

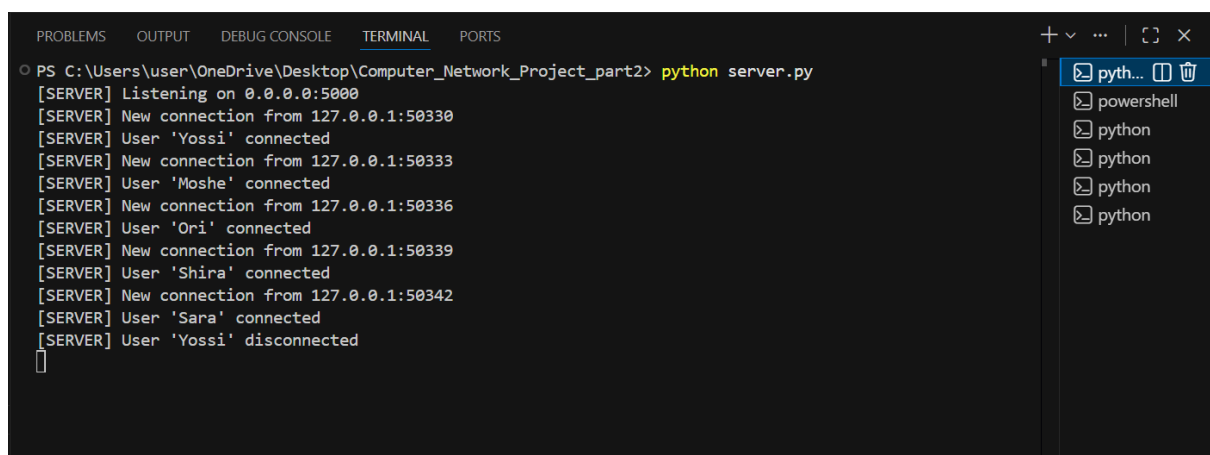
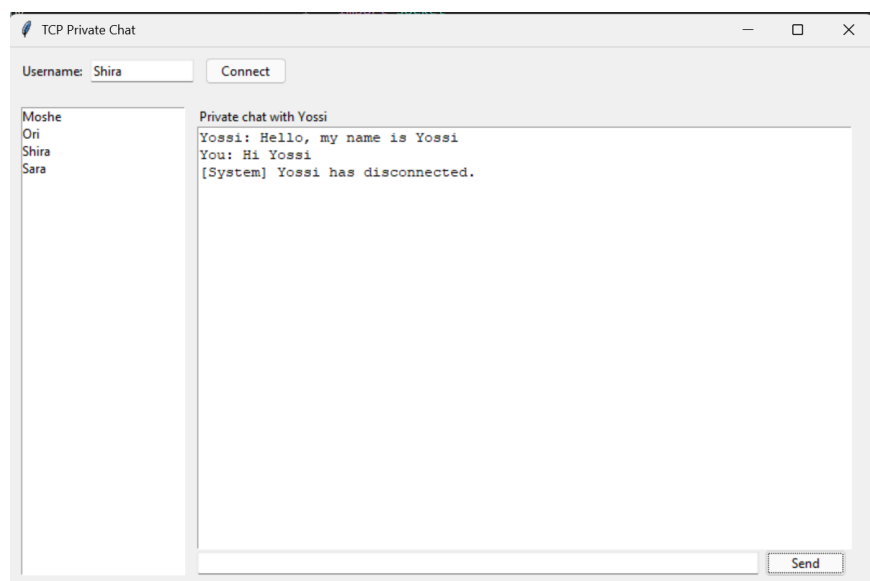
ניתוח תעבורה של היישום עד שכבת הרשת (כולל) (המשך)

ניתוח תעבורה של ניתוק משתמש

המשתמש "Yossi" יצא מהמערכת בניתוק פתאומי מהמערכת.
(ניתן גם לראות שאכן המערכת מטפלת במקרים של ניתוק פתאומי של משתמש או מקרי קצה, היא עדיין מתריאה ומעדכנת, וכמובן השרת ושאר המשתמשים ממשיכים לפעול תקין).
כעת ננתח את התעבורה של הניתוק.

שכבת היישום:

המשתמש "Yossi" יצא מהמערכת במהלך השיחה עם "Shira".
Shira קיבלה עדכון על כך בצ'אט, המשתמש "Yossi" לא נמצא יותר בסרגל הצד.
בנוסף השרת המרכזי עדכן שהמשתמש "Yossi" התנתק מהמערכת.



ניתוח תעבורה של היישום עד שכבת הרשת (כולל) (המשך)

שכבת התעבורה:

נפלט את הפילטר הנכון על מנת למצוא את הפקטה המתאימה (tcp.flags.reset==1).
נראה את הפקטה המתאימה, פקטה 74.

The image shows a Wireshark packet capture. The top pane displays a list of packets, with packet 74 selected. The packet details pane shows the structure of the packet: Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The TCP section is highlighted, showing a Reset (RST) flag set, with sequence number 105, acknowledgment number 367, and window length 0. The bottom pane shows the raw packet data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
74	167.079065	127.0.0.1	127.0.0.1	TCP	44	Seq=105 Ack=367 Win=0 Len=0 [RST, ACK] 5000 → 50330

Frame 74: Packet, 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_{...} id 0

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 50330, Dst Port: 5000, Seq: 105, Ack: 367, Len: 0

0000 02 00 00 00 45 00 00 28 64 d9 40 00 80 06 00 00E..(d@....
0010 7f 00 00 01 7f 00 00 01 c4 9a 13 88 ab f7 6e f4n..
0020 94 68 d8 64 50 14 00 00 51 f2 00 00h-dP...Q...

No.: 74 · Time: 167.079065 · Source: 127.0.0.1 · Destination: 127.0.0.1 · Protocol: TCP · Length: 44 · Info: 50330 → 5000 [RST, ACK] Seq=105 Ack=367 Win=0 Len=0

- ❖ RST סגירה מיידית של החיבור בשל ניתוק פתאומי.
- ❖ ACK אישור קבלת נתונים.
- ❖ Len = 0 תקין כי אין נתונים, זה לבקרה.
- ❖ פורט מקור: 50330, זה הפורט הזמני של המשתמש.
- ❖ פורט יעד: 5000, זה הפורט הקבוע של השרת

ניתוח תעבורה של היישום עד שכבת הרשת (כולל) (המשך)

שכבת הרשת:

גם כאן ניתן לראות כמו קודם בעזרת ה Wireshark את השימוש בפרוטוקול IPv4.

The image shows a Wireshark packet capture. The top pane displays a list of packets, with packet 74 selected. The packet details pane shows the following information:

- Frame 74: Packet, 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_{...} id 0
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- Transmission Control Protocol, Src Port: 50330, Dst Port: 5000, Seq: 105, Ack: 367, Len: 0

The packet bytes pane shows the raw data of the packet, which is a TCP RST, ACK packet with sequence number 105 and acknowledgment number 367.

ניתן לראות שאכן מופיע:

❖ כתובת מקור: 127.0.0.1

❖ כתובת יעד: 127.0.0.1

כפי שרשמתי קודם,

זה תקין שהכתובת המקור זהה לכתובת היעד משום שכל התקשורת בוצעה על localhost.

כלומר, כאן הפרויקט בוצע על מחשב יחיד, כל המשתמשים/לקוחות והשרת נמצאים על אותו המחשב.

סיכום ניתוח תעבורה של היישום עד שכבת הרשת (כולל)

לסיכום, ניתן לראות על ידי הניתוח תעבורה ב Wireshark את המידע עובר דרך שכבת התעבורה ועד שכבת הרשת.

נשלחה הודעה ממשתמש ספציפי למשתמש אחר בצ'אט (או התקבל ניתוק ממשתמש ספציפי), ההודעה נארזת ב TCP בשכבת התעבורה ועוברת דרך IPv4 בשכבת הרשת.

בנוסף, ניתן לומר שאכן המידע עובר דרך תקשורת אמינה ומסודרת כפי שאמור להיות בפרוטוקול TCP.

★ קובץ הלכידה בא Wireshark נשמר בפורמט pcapng ומצורף לפרויקט.

תיאור שימוש בבינה מלאכותית

במהלך העבודה על הפרויקט נעשה שימוש בבינה מלאכותית (GPT) לצורך למידה והבנה של החומר בצורה מעמיקה וטובה יותר. בנוסף היה שימוש בבינה מלאכותית לצורך עזרה טכנית.

מטרות שימוש

- ❖ שאלות מסוימות על החומר התיאורטי על מנת להבין אותו בצורה יותר מעמיקה ונכונה.
- ❖ עזרה טכנית בכתיבת קובץ CSV בחלק הראשון של הפרויקט (כפי שרשום שניתן לעשות בהוראות)
- ❖ עזרה טכנית בכתיבת הקוד (שרת-לקוח) בחלק השני של הפרויקט (לצערי אין לי ידע מעמיק קודם ב python ובכתיבת קוד שרת-לקוח, ולכן נעשה שימוש בבינה מלאכותית לצורך לימוד ועזרה בכתיבה יחד של הקוד והבנה טובה של כל חלק בו).
- ❖ "סיעור מוחות" בנוגע לניתוח נתונים ב Wireshark כדי לוודא את עצמי.
- ❖ הסבר על חלקים מסוימים בקוד לצורך הבנת המערכת.

דוגמאות לפרומפטים

- ❖ "תסביר לי כיצד עובד חיבור שרת-לקוח בפרוטוקול TCP בצורה יותר מעמיקה ומובנת"
- ❖ "תבנה לי קובץ CSV לפי הוראות הפרויקט ולפי הדוגמא"
- ❖ "עזרתך בהכנה יחד את קבצי קוד ב python לצד שרת ולצד לקוח לפי הוראות הפרויקט (אין לי ידע מעמיק וניסיון ב python בכתיבת קוד שרת-לקוח לצערי)"
- ❖ "עכשיו שסיימנו להכין את קבצי הקוד, בוא נעבור חלק חלק בקוד ונבין בדיוק מה הוא עושה ומה תפקידו"
- ❖ "תוודא שאכן הנתון שניתחתי מה Wireshark בנוגע לשכבת התעבורה בחלק הזה נכון, במידה ולא רק תסמן לי מה לא נכון שאבדוק את עצמי"