

פרויקט ברשותה תקשורת

Project Source Code: [Link to GitHub Repository](#)(<https://github.com/LiroyKai/Computer-Networks-Final-Project>)

מגיסטים:

ליורי לייבל (212289987)

מייכאל וקנין (207973363)

רועי מימון (207972290)

חלק 1: אריזה ולכידת מנוט (Encapsulation & Capture)

1. ייצרת קובץ הנתונים (CSV)

בשלב הראשון של הפרויקט, נדרשנו ליצור תעבורת מדומינית של אפליקציה. לצורך כך, יצרנו קובץ CSV (בשם group212289987_http_input.csv) המכיל רשימה של הודעות ב프וטוקול HTTP, הקובץ נוצר בסיוו AI כדי להבטיח פורמט תקין נתונים מגוונים, והוא כולל את השדות הבאים: מזהה הודעה, פרוטוקול, אפליקציית מקור, אפליקצייתיעד תוכן ההודעה וחותמת זמן.

תהליך אריזת המנות(Encapsulation)

באמצעות מחברת Jupyter ביצענו סימולציה ידנית של תהליך הרכimos (Encapsulation) המתרחש במערכת הפעלה:

1. **שכבת היישום (Application):** חילצנו את המידע הגלומי (ההודעה) מתוך ה-CSV.
2. **שכבת התעבורה (Transport):** לכל הודעה הוספנו כותרת (Header) של פרוטוקול TCP/UDP/HTTP הכוללת את פורט המקור ופורט היעד.
3. **שכבת הרשת (Network):** עטפנו את המזרב בכתובת IP הכוללת את כתובות ה-IP-של המקור והיעד.
4. **שידור:** שלחנו את המנות המוכנות דרך ממשק הרשת, כך שתוכנת Wireshark תוכל לצלות אותן.

ניתוח לכידת התעבורה(Wireshark)

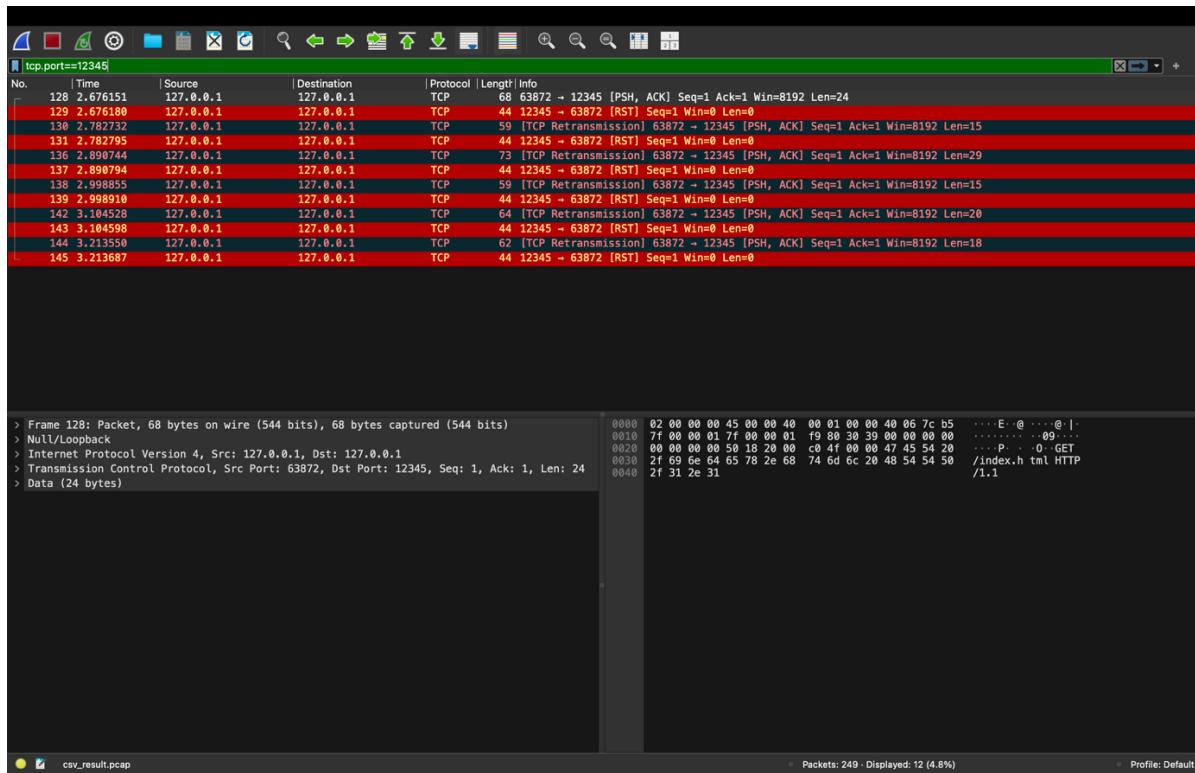
לאחר הרצת המחברת, ניתחנו את קובץ הלכידה csv_result.pcap.

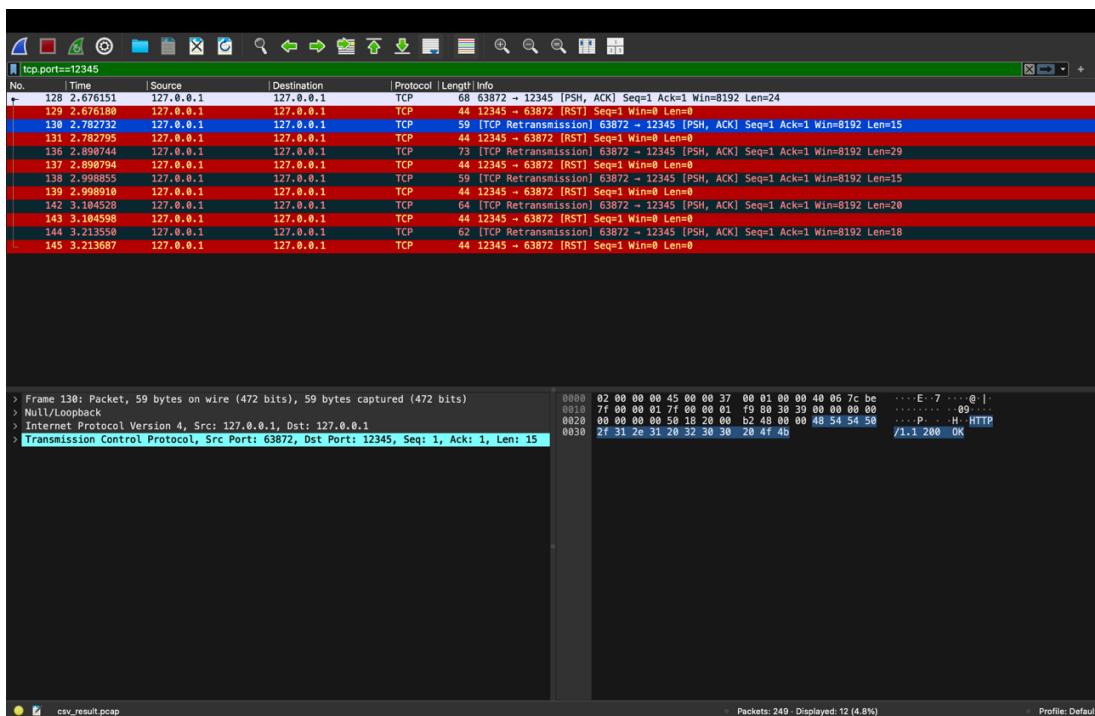
דיהוי ה프וטוקול Wireshark: זיהה את המנות TCP/HTTP-בהתבסס על הפורטים שהוגדרו.

תוכן המנה (Payload): בתוך גופו המנה, ניתן לראות בבירור את הטקסט שנלקח מקובץ ה-CSV.

תובנה: התרגיל המחייב כיצד כל שכבה מוסיפה את המידע החדש לתוכה (Headers) ללא תלות בתוכן המידע עצמו.

לכידת מנות ה-CSV ב-Wireshark





חלק 2: יישום צ'אט וניהוט תעבורת

מבנה המערכת והקוד

פיתחנו מערכת צ'אט מבוססת Sockets ב Python-הפעלת ארכיטקטורת שרת-לקוח-(Client-Server). המערכת אינה משתמשת בספריות ניהול שירותים חיצונית, אלא ממשת את התקשרות באמצעות שיר.

רכיבי המערכת:

1. השרת (server.py)

מנהל את כל התעבורה

ריבוי תהליכי (Multi-threading): עבור כל לקוח שמחבר, השרת פותח Thread נפרד. מנגנון זה מאפשר לשרת לטפל במספר רב של לקוחות במקביל (מעל 5 כנدرש) מבלי לחסום את התקשרות עם לקוחות אחרים.

השרת מנהל רשימת משתמשים מחוברים ומבצע ניתוב (Routing) של הודעות מלוקוט אחד לאחרר.

2. הלקוח (client.py / Gui_Client.py)

פותח חיבור TCP לשרת ומפheid בין תהליכי השידור והקליטה באמצעות Threads, מה שמאפשר קבלת הודעות בזמן אמת.

ממשק גרפי (GUI) כחלק מבונוס, פיתחנו ממשק משתמש באמצעות ספריית Tkinter. המשך השימוש, מציג את ההיסטוריה הצ'אט ומפריד ויזואלית בין הודעות כניסה ליצאות.

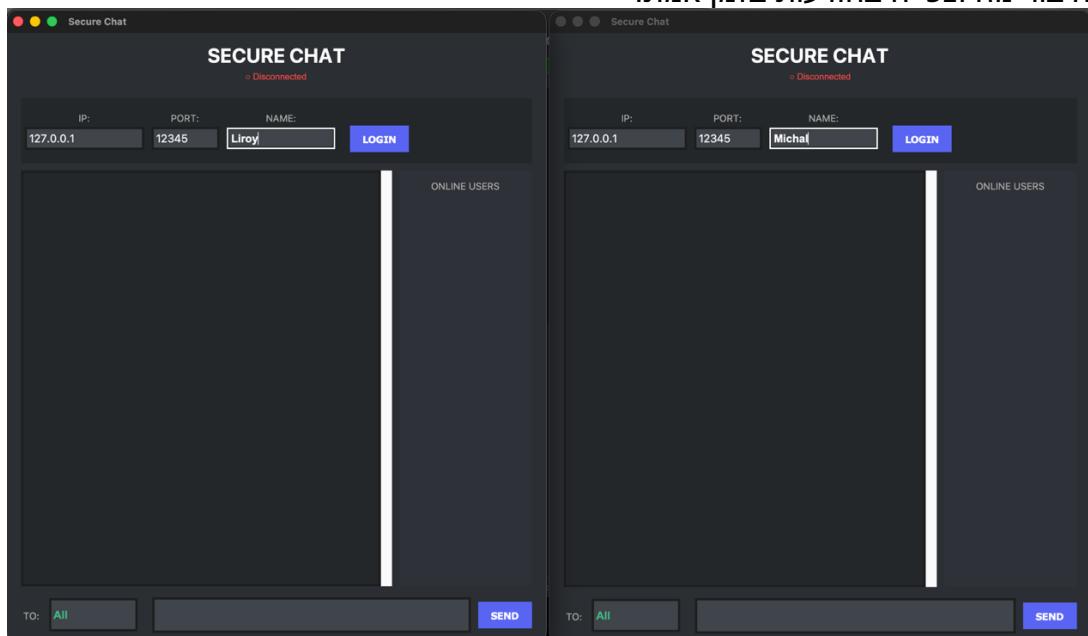
הוראות התקנה והרצה:

המערכת מtabסת על ספריות סטנדרטיות של Python ואינה דורשת התקנות מיוחדות.

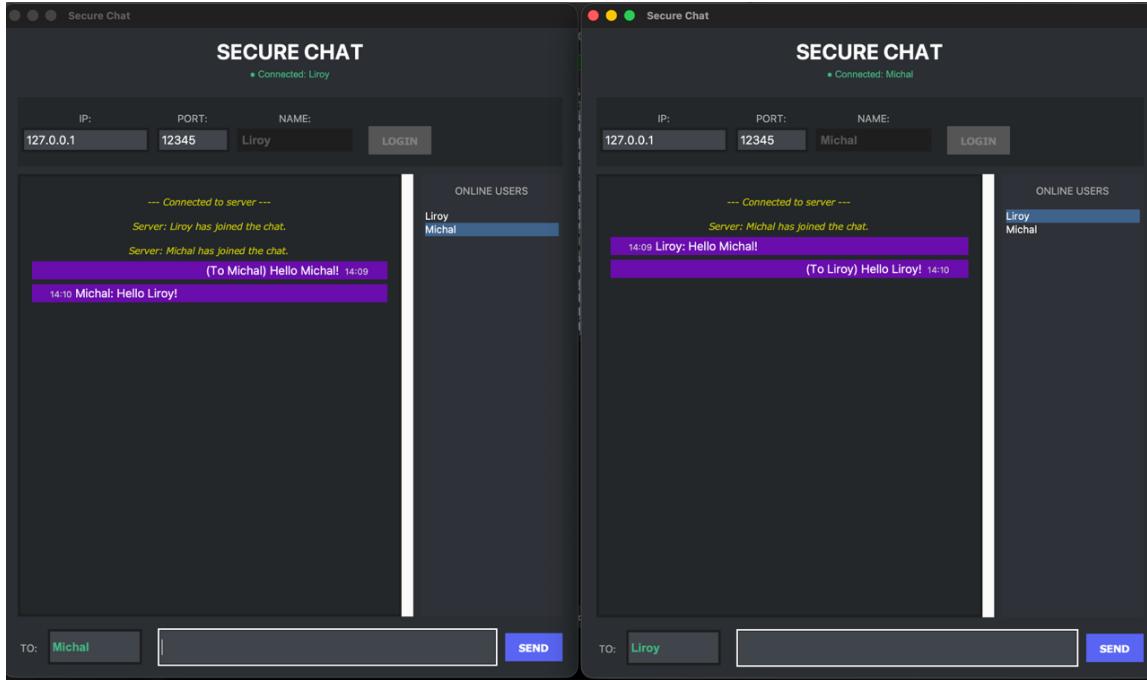
- יש להפעיל את השרת באמצעות הפקודה: `python server.py`: השרת יופיע בכתובת `127.0.0.1:12345`.
- עבור כל משתמש, יש להפעיל את הליקוב `python Gui_Client.py`.
- בחילון שנפתח, יש להזין שם משתמש ולהתחל בשיחת.

דוגמאות קלט ופלט:

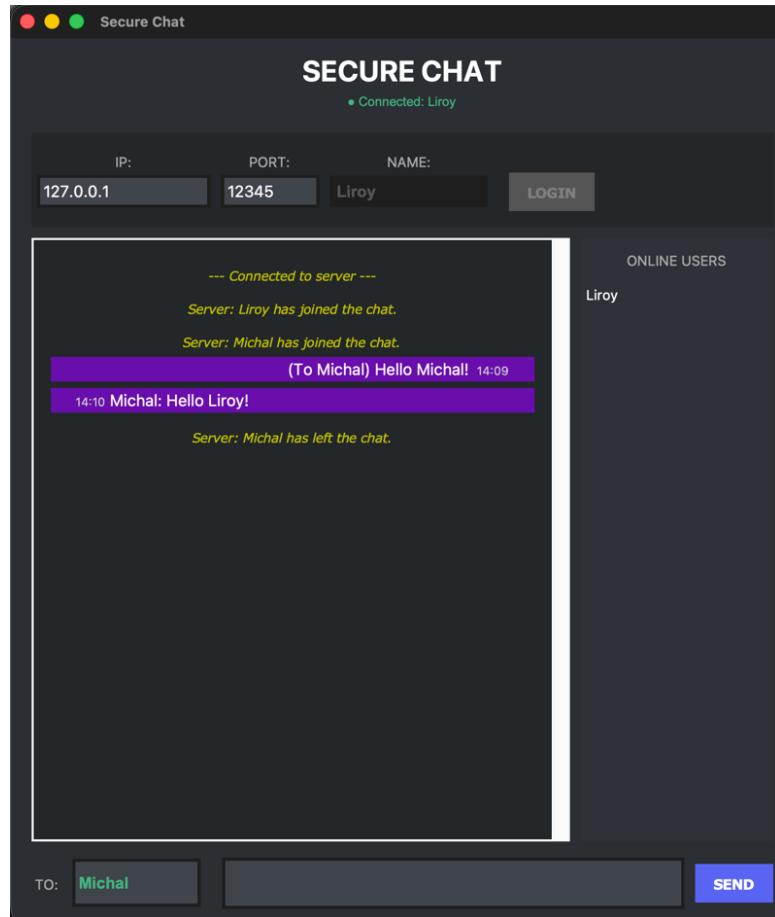
א. ממשק המשתמש (GUI) בתמונה הבאה ניתן לראות את חלון הצ'אט שפיתחנו. הממשק מאפשר חיבור נוח וצפיה בהודעות בזמן אמת.



ב. תרחיש שיחה בין לקוחות בתרחיש זה הפעילו שני לקוחות במקביל. הודעה שנשלחה מלקוח א' התקבלה מידית אצל לקוח ב'.



ג. לוג השרת ה�示 מציג אמת את הסטטוס של המוחברים והודעות מערכת כולל התראה על עדיבה והצטרפות לצ'אט.



ניתוח תüberות היישום

- בניתוח קובץ הלכידה Chat_Analyze.pcap ניתן לראות את שלבי ה프וטוקול:TCP.
- .1. ייצור הקשר: (Handshake) רואים את תהליך SYN, SYN-ACK, ACK.
 - .2. העברת מידע: (Data Transfer) הודעות הצ'אט (כגון שמות המשתמשים והתוכנה) מועברות במנות עם דגל PSH (Push) המורה על העברת מיידית לשכבה האפליקציה. מיד לאחר מכן מקבל אישור ACK.
 - .3. סיום קשר: בעת סגירת החלון, נשלחת מנת FIN/RST לסתירת ה-Socket.

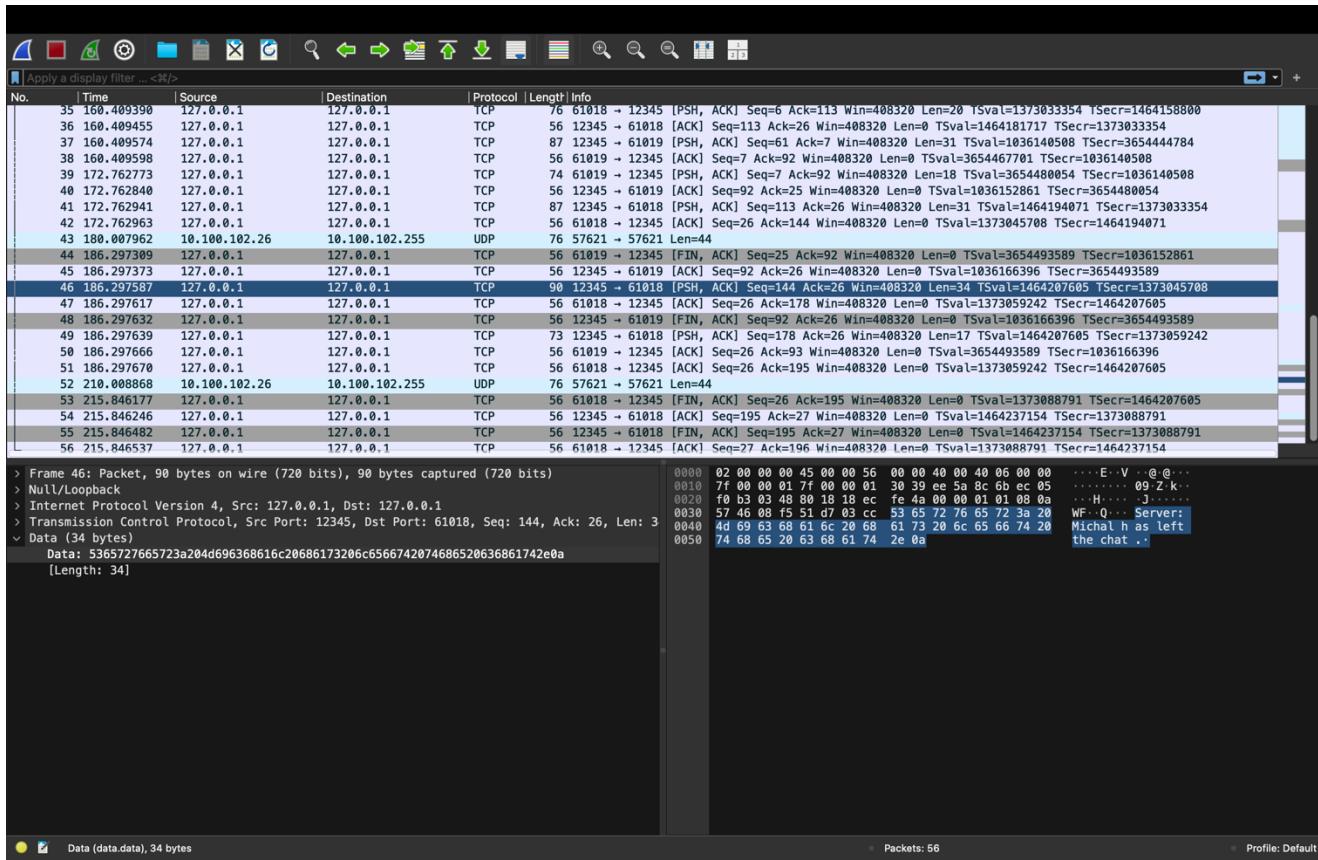
הודעת הצליפות:

הודעות בין משתמשים:

Apply a display filter ... <?>/						
No.	Time	Source	Destination	Protocol	Length	Info
33	137.492006	127.0.0.1	127.0.0.1	TCP	56	61019 → 12345 [ACK] Seq=7 Ack=61 Win=408320 Len=0 TSval=3654444784 TSecr=1036117591
34	150.007138	10.100.102.26	10.100.102.255	UDP	76	57621 → 57621 Len=44
35	160.409390	127.0.0.1	127.0.0.1	TCP	76	61018 → 12345 [PSH, ACK] Seq=6 Ack=113 Win=408320 Len=20 TSval=1373033354 TSecr=1464158800
36	160.409455	127.0.0.1	127.0.0.1	TCP	56	12345 → 61018 [ACK] Seq=113 Ack=26 Win=408320 Len=0 TSval=1464181717 TSecr=1373033354
37	160.409574	127.0.0.1	127.0.0.1	TCP	87	12345 → 61019 [PSH, ACK] Seq=61 Ack=7 Win=408320 Len=31 TSval=1036140508 TSecr=3654444784
38	160.409598	127.0.0.1	127.0.0.1	TCP	56	61019 → 12345 [ACK] Seq=7 Ack=92 Win=408320 Len=0 TSval=3654467701 TSecr=1036140508
39	172.762773	127.0.0.1	127.0.0.1	TCP	74	61019 → 12345 [PSH, ACK] Seq=7 Ack=92 Win=408320 Len=18 TSval=3654480054 TSecr=1036140508
40	172.762840	127.0.0.1	127.0.0.1	TCP	56	12345 → 61019 [ACK] Seq=92 Ack=25 Win=408320 Len=0 TSval=1036152861 TSecr=3654480054
41	172.762941	127.0.0.1	127.0.0.1	TCP	87	12345 → 61018 [PSH, ACK] Seq=113 Ack=26 Win=408320 Len=31 TSval=1464194071 TSecr=1373033354
42	172.762963	127.0.0.1	127.0.0.1	TCP	56	61018 → 12345 [ACK] Seq=26 Ack=144 Win=408320 Len=0 TSval=1373045708 TSecr=1464194071
43	180.007962	10.100.102.26	10.100.102.255	UDP	76	57621 → 57621 Len=44
44	186.297309	127.0.0.1	127.0.0.1	TCP	56	61019 → 12345 [FIN, ACK] Seq=25 Ack=92 Win=408320 Len=0 TSval=3654493589 TSecr=1036152861
45	186.297373	127.0.0.1	127.0.0.1	TCP	56	12345 → 61019 [ACK] Seq=92 Ack=26 Win=408320 Len=0 TSval=1036166396 TSecr=3654493589
46	186.297587	127.0.0.1	127.0.0.1	TCP	90	12345 → 61019 [PSH, ACK] Seq=144 Ack=27 Win=408320 Len=34 TSval=1464207605 TSecr=1373045708
47	186.297617	127.0.0.1	127.0.0.1	TCP	56	61018 → 12345 [ACK] Seq=26 Ack=178 Win=408320 Len=0 TSval=1373059242 TSecr=1464207605
48	186.297632	127.0.0.1	127.0.0.1	TCP	56	12345 → 61019 [FIN, ACK] Seq=92 Ack=26 Win=408320 Len=0 TSval=1036166396 TSecr=3654493589
49	186.297639	127.0.0.1	127.0.0.1	TCP	73	12345 → 61018 [PSH, ACK] Seq=178 Ack=26 Win=408320 Len=17 TSval=1464207605 TSecr=1373059242
50	186.297666	127.0.0.1	127.0.0.1	TCP	56	61019 → 12345 [ACK] Seq=26 Ack=93 Win=408320 Len=0 TSval=3654493589 TSecr=1036166396
51	186.297670	127.0.0.1	127.0.0.1	TCP	56	61018 → 12345 [ACK] Seq=26 Ack=195 Win=408320 Len=0 TSval=1373059242 TSecr=1464207605
52	210.008868	10.100.102.26	10.100.102.255	UDP	76	57621 → 57621 Len=44
53	215.846177	127.0.0.1	127.0.0.1	TCP	56	61018 → 12345 [FIN, ACK] Seq=26 Ack=195 Win=408320 Len=0 TSval=1373088791 TSecr=1464207605
54	215.846246	127.0.0.1	127.0.0.1	TCP	56	12345 → 61018 [ACK] Seq=195 Ack=27 Win=408320 Len=0 TSval=1464237154 TSecr=1373088791

Apply a display filter ... <?>/						
No.	Time	Source	Destination	Protocol	Length	Info
29	137.491991	127.0.0.1	127.0.0.1	TCP	56	61019 → 12345 [ACK] Seq=7 Ack=37 Win=408320 Len=0 TSval=365444784 TSecr=1036117591
30	137.491997	127.0.0.1	127.0.0.1	TCP	80	12345 → 61018 [PSH, ACK] Seq=89 Ack=6 Win=408320 Len=24 TSval=1464158800 TSecr=1373010437
31	137.492001	127.0.0.1	127.0.0.1	TCP	80	12345 → 61019 [PSH, ACK] Seq=37 Ack=7 Win=408320 Len=24 TSval=1036117591 TSecr=3654444784
32	137.492004	127.0.0.1	127.0.0.1	TCP	56	61018 → 12345 [ACK] Seq=6 Ack=113 Win=408320 Len=0 TSval=1373010437 TSecr=1464158800
33	137.492006	127.0.0.1	127.0.0.1	TCP	56	61019 → 12345 [ACK] Seq=7 Ack=61 Win=408320 Len=0 TSval=3654444784 TSecr=1036117591
34	150.007138	10.100.102.26	10.100.102.255	UDP	76	57621 → 57621 Len=44
35	160.409390	127.0.0.1	127.0.0.1	TCP	56	12345 → 61018 [ACK] Seq=113 Ack=26 Win=408320 Len=0 TSval=1373033354 TSecr=1464158800
36	160.409455	127.0.0.1	127.0.0.1	TCP	56	12345 → 61018 [ACK] Seq=113 Ack=26 Win=408320 Len=0 TSval=1373033354
37	160.409574	127.0.0.1	127.0.0.1	TCP	87	12345 → 61019 [PSH, ACK] Seq=61 Ack=7 Win=408320 Len=31 TSval=1036140508 TSecr=3654444784
38	160.409598	127.0.0.1	127.0.0.1	TCP	56	61019 → 12345 [ACK] Seq=7 Ack=92 Win=408320 Len=0 TSval=3654467701 TSecr=1464194071
39	172.762773	127.0.0.1	127.0.0.1	TCP	74	61018 → 12345 [PSH, ACK] Seq=7 Ack=92 Win=408320 Len=18 TSval=3654480054 TSecr=1036140508
40	172.762840	127.0.0.1	127.0.0.1	TCP	56	12345 → 61019 [ACK] Seq=26 Ack=178 Win=408320 Len=0 TSval=1373059242 TSecr=1464207605
41	172.762941	127.0.0.1	127.0.0.1	TCP	87	12345 → 61018 [PSH, ACK] Seq=113 Ack=26 Win=408320 Len=31 TSval=1464194071 TSecr=1373033354
42	172.762963	127.0.0.1	127.0.0.1	TCP	56	61018 → 12345 [ACK] Seq=26 Ack=144 Win=408320 Len=0 TSval=1373045708 TSecr=1464194071
43	180.007962	10.100.102.26	10.100.102.255	UDP	76	57621 → 57621 Len=44
44	186.297309	127.0.0.1	127.0.0.1	TCP	56	61019 → 12345 [FIN, ACK] Seq=25 Ack=92 Win=408320 Len=0 TSval=3654493589 TSecr=1036152861
45	186.297373	127.0.0.1	127.0.0.1	TCP	56	12345 → 61019 [ACK] Seq=92 Ack=26 Win=408320 Len=0 TSval=3654493589
46	186.297587	127.0.0.1	127.0.0.1	TCP	90	12345 → 61018 [PSH, ACK] Seq=144 Ack=26 Win=408320 Len=34 TSval=1464207605 TSecr=1373045708
47	186.297617	127.0.0.1	127.0.0.1	TCP	56	61018 → 12345 [ACK] Seq=26 Ack=178 Win=408320 Len=0 TSval=1373059242 TSecr=1464207605
48	186.297632	127.0.0.1	127.0.0.1	TCP	56	12345 → 61019 [FIN, ACK] Seq=92 Ack=26 Win=408320 Len=0 TSval=1036166396 TSecr=3654493589
49	186.297639	127.0.0.1	127.0.0.1	TCP	73	12345 → 61018 [PSH, ACK] Seq=178 Ack=26 Win=408320 Len=17 TSval=1464207605 TSecr=1373045708
50	186.297666	127.0.0.1	127.0.0.1	TCP	56	61019 → 12345 [ACK] Seq=26 Ack=93 Win=408320 Len=0 TSval=3654493589 TSecr=1036166396

הודעת עצית משתמש מהמצ'אט:



חלק 3: שימוש במבנה מלכוטית (AI)

במסגרת העבודה נעשה שימוש בכל'י הבינה המלאכותית Google Gemini ככלי עזר לפיתוח ולמידה. השימוש בוצע באופן מבוקר תוך בדיקת התוצאות.

מטרות השימוש ודוגמאות לפורמפעטיפ:

1. **יצירת נתונים (חלק 1):** (יצירת קובץ CSV-עם נתוני דמה ריאליים לשימוש).

Prompt Example: "Create a CSV file with columns: msg_id, app_protocol, src_app, dst_app, message, timestamp. Generate 10 rows representing HTTP GET requests."

2. **פיתוח השירות (Server):** סיעע בכתיבה שלד הקוד לטיפול בריבוי לקוחות.

Prompt Example: "Write a Python TCP server using 'socket' and 'threading' to handle multiple clients simultaneously."

3. **עיצוב הממשק (GUI):** עזרה בבניית הממשק הגרפי ב Tkinter וסידור האלמנטים בחלון.

Prompt Example: "Create a GUI chat client in Python using tkinter with a scrollable text area and an input field"