

## דוח פרויקט

מגישים:

עמית אלמליח 213339054

אמיר מחתין 211388889

### חלק 1

שלב 1: הכנת קובץ CSV

ביקשנו מהAI שיצור את הקובץ CSV

על ידי שימוש במבנה מלאכותית עם הפורמט "צור לי קובץ CSV עם 5 שורות של הודעות HTTP שכולל את שדות החובה msg\_id, app\_protocol, src\_app, dst\_app, message ו timestamp".

msg_id	app_protocol	src_app	dst_app	src_port	dst_port	message	timestamp
1	HTTP	client_browser	web_server	54321	80	GET /index.html HTTP/1.1	0.010
2	HTTP	web_server	client_browser	80	54321	HTTP/1.1 200 OK	0.020
3	HTTP	client_browser	web_server	54321	80	GET /style.css HTTP/1.1	0.030
4	HTTP	web_server	client_browser	80	54321	HTTP/1.1 200 OK (CSS Content)	0.045
5	HTTP	client_browser	web_server	54321	80	GET /images/logo.png HTTP/1.1	0.050
6	HTTP	web_server	client_browser	80	54321	HTTP/1.1 200 OK (Image Content)	0.080
7	HTTP	client_browser	web_server	54321	80	POST /login_form HTTP/1.1	0.100
8	HTTP	web_server	client_browser	80	54321	HTTP/1.1 302 Redirect (Welcome)	0.120

שלב 2: שימוש במחברת Jupyter

### חלק 1 – סימולציה אנקפסולציה (Encapsulation)

#### 1. מבוא ומטרת הניסוי

ב חלק זה של הפרויקט, המטרה הייתה לבצע סימולציה של תהליכי האנקפסולציה (Encapsulation) המתרחש ברשתות תקשורת. במקרה להשתמש בדף או בשרת אמיתי, השתמשנו בסקריפט Python (בסביבת Notebook Jupyter) שקורא הודעות אפליקציה מתוך קובץ CSV, "אוחז" אותן ידנית בחבילות TCP/IP, ושולח אותן למסך הרשות המקומי (Loopback).

## 2. תיאור התהליך (Encapsulation Process)

שכבה האפליקציה (Application Layer): הסקריפט קרא את תוכן הודעה (Message) מתוך קובץ CSV.

דוגמה: עברו השורה הראשונה בקובץ, המידע הוא: GET /index.html HTTP/1.1 TCP (TCP - Transport Layer). המידע משכבה האפליקציה נטען בcotract TCP (Header).

הוגדר פורט מקור (Source Port) אקראי (למשל 29085 כפ' שנראה ב-Wireshark).

הוגדר פורטיעד (Destination Port) קבוע: 12345 (שהוגדר בסקריפט הסימולציה).

הוגדרו דגלים (Flags): הדגל PSH ו-ACK הופעלן כדי לדמות העברת מידע פעילה.

שכבה הרשת (Network Layer - IP): חבילת TCP נועטה בcotract IP (IP Header).

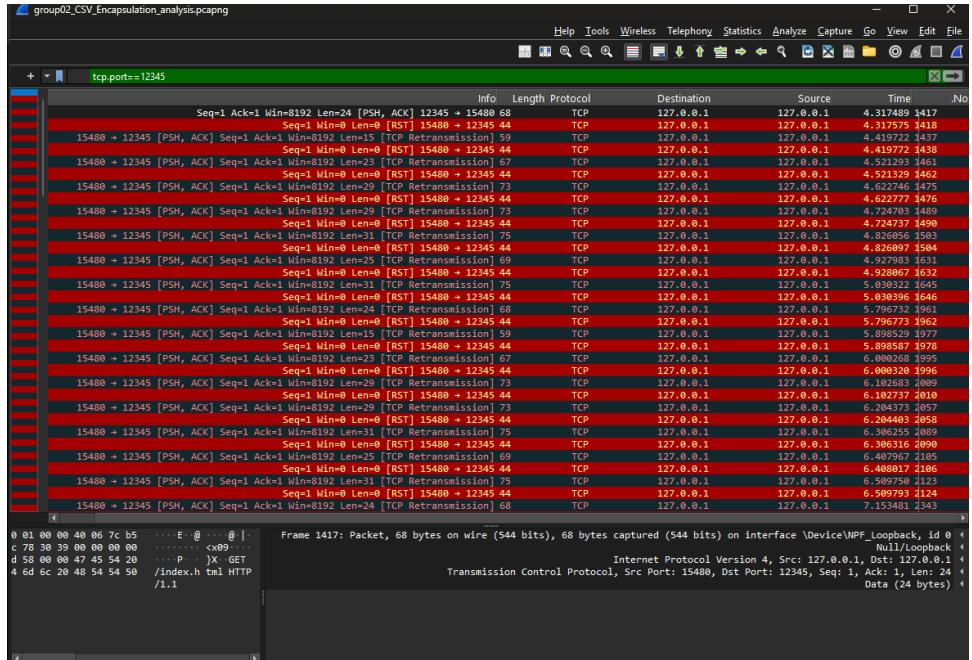
כתובת מקור ויעד: 127.0.0.1 (localhost).

פרוטוקול: 6 (TCP).

### 3. ניתוח תעבורת (Wireshark Analysis)

במהלך הריצת הסקריפט, ביצענו האזנה (Capture) על ממשק ה-Loopback באמצעות CSV.

להלן ניתוח של חבילה בוודת המייצגת את ההודעה הראשונה מה-CSV.



#### פרטי החבילה שנלכדה:

1417 :Packet Number

4.317489 :Time

.Source / Destination: 127.0.0.1 (תקשורת פנימית).

.Protocol: TCP

.(האורק עשי להשתנות בהתאם להודעה). Info: 12345 -> [15480] [PSH, ACK] Len=24

## חלק 2

המערכת שפותחה היא אפליקציית צ'אט (Instant Messaging) הפעלת בארכיטקטורת שרת-לקוח (Client-Server) וmbased על פרוטוקול TCP. המערכת מאפשרת למספר רב של משתמשים להתחבר במקביל, לנוהל שיחות בחדר כללי (Broadcast), וליקויים שיחות פרטיות (Unicast) בין משתמשים.

### מבנה השירות – server.py

השירות הוא רב-תהליכי ומימוש תקשורת באמצעות sockets בלבד  
ניהול חיבורים: השירות מזין בפורט קבוע 1000 עבור כל לקוח שמתחבר, השירות יוצר thread ייעודי המטפל בתקשורת מולו, כך שהשרות אינו נחסם ויכול לעבור באופן שוטף עבור כמה משתמשים במקביל.

פרוטוקול תקשורת מותאם: השירות מפענה את ההודעות המתקבלות ומחליט איך לנטרן אותן

- הודעות שמתחלילות בתו @ מזוהות אוטומטית כהודעות פרטיות למשל @Yossi:Hello.
- נשלחות ל socket של משתמש היעד.
- שאר ההודעות נשלחות לכל המשתמשים והמחוברים broadcast

### 1. מבנה הלקוח client.py

- הלקוח בניו בעזרת ספריית Tkinter (לממשק הגרפי) ומשתמש בreading-CDIMultithreading כדי לאפשר קבלה ושליחה של הודעות בו-זמןית מבליל "לתקוע" את הממשק.
- ניהול מצב (State Management) הלקוח שומר היסטוריה הודעות נפרדת לכל משתמש (self.history) ווסיף הודעות שלא נקרו.
- ממשק הצד (Sidebar): רשימה דינמית שמציגה מי מחובר (●) ומי מנוטק (●).
- פרוטוקול הודעות \*: הודעה רגילה נשלחת כתקסט פשוט לשרת (מודפס לכולם).
- הודעה פרטית נשלחת בפורמט @TargetName: Message.

**תהליך האזנה:** פונקציית receive\_loop רצה ברקע, מזינה להודעות כניסה ומעדכנת את הממשק הגרפי בזמן אמיתי ללא תקיעת המסך.

## 2. איך הקוד עובד

- א. התחברות: הלקוח מתחבר לשרת ושולח את שם המשתמש שלו. השרת מוסיף אותו למילון הלקוחות הפעילים.(clients)
- ב. עדכוני משתמשים: בכל פעם שמשהו מתחבר/מתנתק, השרת שולח לכל הלקוחות הודעה מערכת שמהquila ב: :UPDATE\_USERS-הלקוח מפרק את הודעה זו ועדכן את רשימת אנשי הקשר בצד.
- ג. ניהול הודעות בשרת:
  - ג.1. אם הודעה מתחלילה ב,@-השרת מזיהה את שם היעד ושולח את הודעה רק לו (Private Message).
  - ג.2. אם לא, השרת שולח את הודעה לכל המוחברים.(Broadcast)
- ד. פענו בלקוח: הלקוח בודק אם בהודעה שהתקבל מופיע הכיתוב .(Private)אם כן, הוא מנtab אותה לחילון הצ'אט האישי עם אותו משתמש.

### הוראות התקנה והריצה

- Python: וודא שמותקן 3 Python במחשב.
- ספריות: הקוד משתמש בספריות מובנות בלבד (socket, threading, tkinter), כך שאין צורך להתקין כלום

### שלבי הריצה:

#### 2. הפעלת השירות:

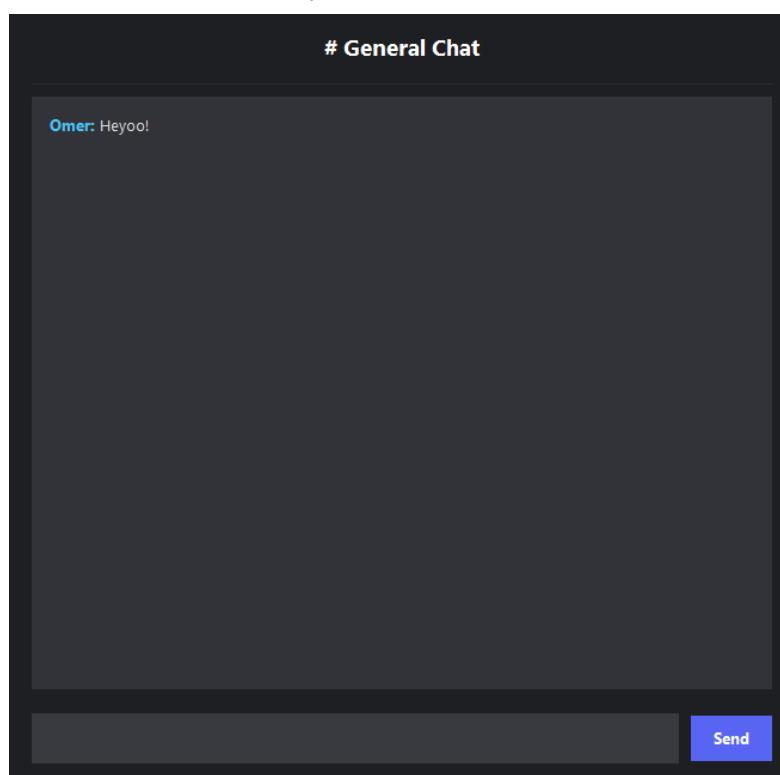
- פותח מסוף (Terminal) בתיקיית הפרויקט והרץ server.py.
- וודא ש毛病עה הודעה על האזנה לפורט 10000.

#### 3. הפעלת הלקוחות:

- פותח מסוף חדש (Terminal) והרץ client\_gui.py.
- בחלון ההתחברות, הזן IP למשל 127.0.0.1 לבדיקה מקומית) ושם משתמש ייחודי. למשל Omer, Yossi.
- לחץ על Join Chat.
- ניתן לפתוח מספר חלונות במקביל כדי לדמות משתמשים שונים.

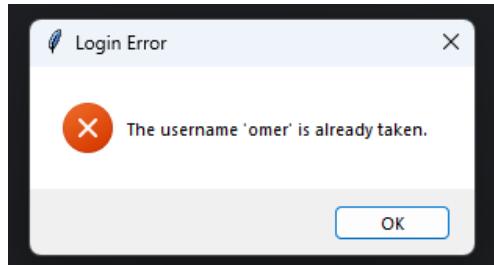
## דוגמאות קלט ופלט (תרחישי שימוש)

- **תרחיש 1: התחברות וצ'אט כללי**
- **קלט) ל��וח:** "Omer" - A מתחבר למערכת.
- **פלט) ל��וח:** "Yossi" - B בראשימת המשתמשים בצד, השם "Omer" מופיע עם סימן יירוק. : Omer
- **פעולה** Omer: שולח הודעה ב-General Chat
- **תוצאה:** כל המחברים רואים את ההודעה תחת החדר הכללי.



## תרחיש 2: שם תפוס

כשמיישהו מנסה להיכנס עם שם משתמש שכבר מחובר יקבל את השגיאה הבאה

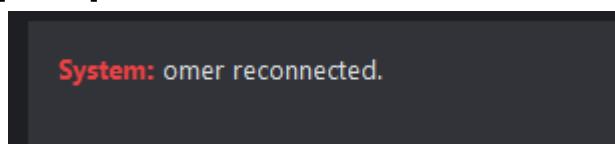


מה שקרה מצד השרת:

1. קיבלת שם המשתמש ובדיקה זמינות
  - ברגע שנוצר חיבור חדש, השרת ממitten לקבל את המחרוזת של שם המשתמש. הוא מבצע בדיקה כנגד מיליון המשתמשים הפעילים (active\_clients):
  - אם השם כבר קיים במילון: השרת שולח בחזרה ללקוח את המחרוזת "Username taken" מ-socket.

```
try:  
    self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
    self.sock.settimeout(3)  
    self.sock.connect((ip, DEFAULT_PORT))  
  
    self.sock.sendall(user.encode('utf-8'))  
    response = self.sock.recv(1024).decode('utf-8')  
  
    if response == "Username taken":  
        messagebox.showerror(title="Login Error", message=f"The username '{user}' is already taken.")  
        self.sock.close()  
        return
```

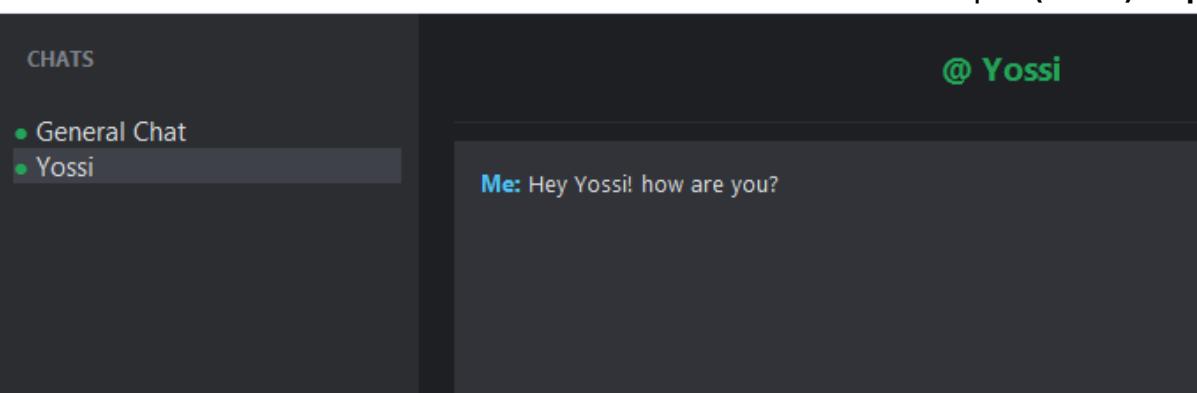
- אם השם פנוי: השרת ממשיך לשלב הרישום
2. הבחנה בין משתמש חדש לחוזר - reconnect
  - השרת משתמש בסט (Set) בשם registered\_users כדי לזכור שמות שהתחברו בעבר:
  - משתמש חוזר: אם השם נמצא בסט, השרת שולח הודעה ברוך הבא ומופיע : "System: [Name] reconnected."



- משתמש חדש: אם השם לא מוכר, הוא מתווסף לסט והשרת מפיץ הודעה על הצטרפות המשתמש חדש.
- שימוש ב lock
- כדי למנוע שני משתמשים יתחברו עם אותו שם משתמש באותה השניה. השרת נעל את הגישה למשתנים המשותפים.

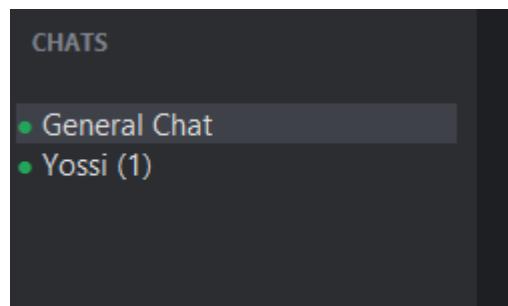
### תרחיש 3: הودעה פרטית והתראות

קלט: (Omer) לוחץ על השם Yossi ונכנס לשיחה פרטית אליו ○

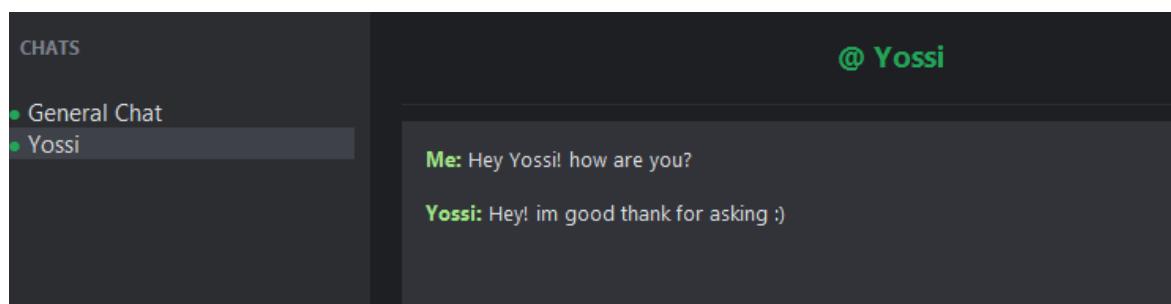


עיבוד: הלקוח שלוח פרוטוקול: Yossi@שלום לשרת. השרת מעביר את ההודעה רק ל-Yossi. ○

פלט: Omer יוסי נמצא כרגע בשיחה אחרת. בראשימת המשתמשים שלו, ליד השם של Omer מופיע (1) שמראה שיש הודעה 1 חדשה שלא נקרה ממנו



פעולה: יוסי לוחץ על השם של עומר. המסר מתנתקה ומוצגת בשיחה הפרטית ביניהם בלבד. מונה ההודעות מתאפס. ○

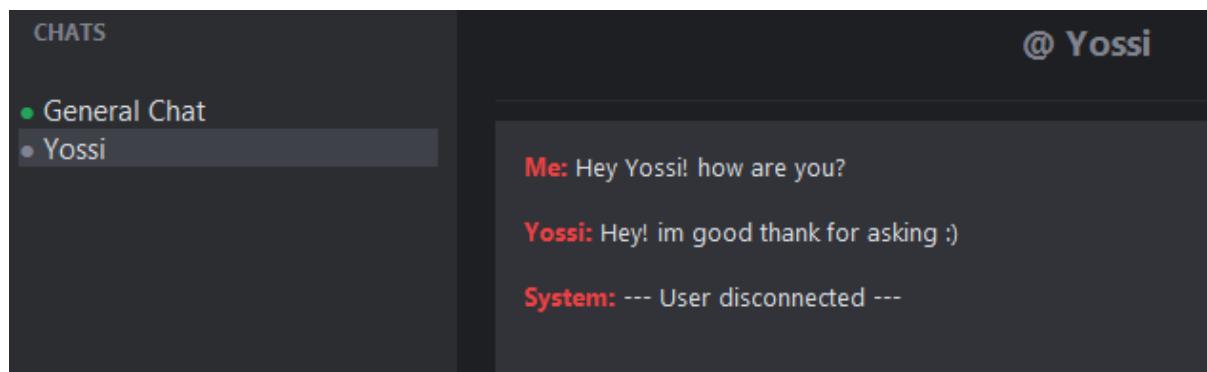


ב wire shark מה שקרה כשבועיים

```
Omer
USERS_UPDATE:Yossi,Omer
@Yossi:Hey Yossi! how are you?
(Private) Yossi: Hey omer im good thanks for asking! :)USERS_UPDATE:Omer
```

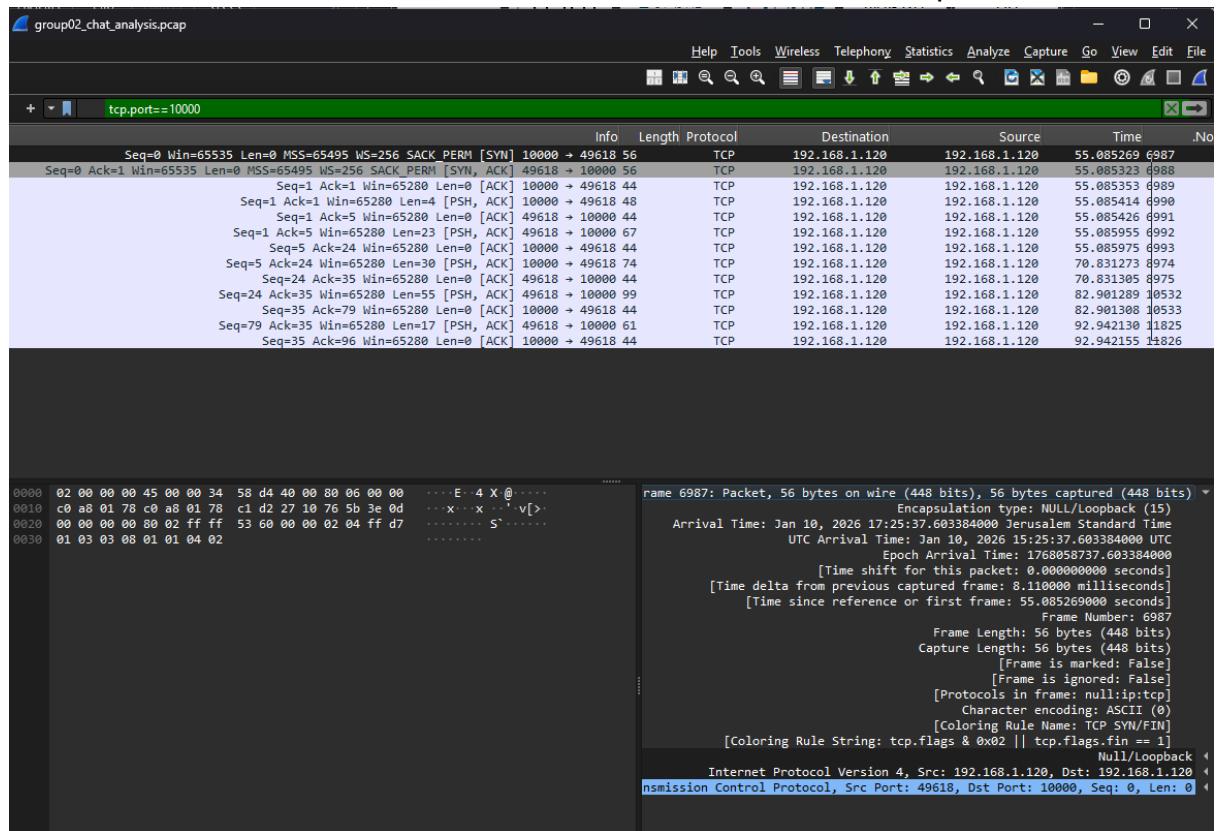
#### תרחיש 4: התנתקות משתמש

- **קלט:** המשתמש "Omer" סגור את החלון.
- **עיבוד:** השרת מזיהה ניתוק ושולח רשיית משתמשים מעודכנת.
- **פלט:(Yossi)**
- הליקוח של Omer מזיהה ש Yossi-חסר ברשימה החדשה.
- הסטטוס ברשימה משתנה לאפור ושולח הודעה ניתוק בתוך חלון היצוא הפרט.
- השם שלו הופך לאדום בתחום היצוא



## ניתוח תעבורת היפר טרנספורם עד שכבה הרשת

הפעלו wireshark בזמן הרצת האפליקציה על מנת לנתח את חבילות המידע שמייצגת הودעה פרטית שנשלחה מהלוקו לשרת



## שכבה פיזית

Frame 1335: Packet, 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface \Device\NPF\_Loopback, id 0

ה(Frame 1335) זה השכבה הפיזית הראשונה. לאחר שהתקשרות בפרויקט זה בוצעה בסביבת loopback השכבה הפיזית היא וירטואלית בפועל

## שכבה הרשת: (Network - IP)

127.0.0.1. כתובות המחשב השולח (192.168.1.120) או Source IP: •

Destination IP: • כתובות הרשת.

IPv4 Address . . . . . : 192.168.1.120

## שכבה התעבורה: (Transport - TCP)

- פורט רנדומלי של הלקוח למשל. (12345) **Source Port**
- פורט השירות.(10000) **Destination Port**
- החיבור הוא אמין (Reliable) ו מבטיח שההודעות יגיעו בסדר הנכון ולא איבוד נתונים שזה חשוב לאפליקציית צ'אט.

```
Transmission Control Protocol, Src Port: 10000, Dst Port: 59183, Seq: 1, Ack: 1, Len: 23
    Source Port: 10000
    Destination Port: 59183
        [Stream index: 13]
        [Stream Packet Number: 1]
        [Conversation completeness: Incomplete (12)]
            [TCP Segment Len: 23]
        Sequence Number: 1      (relative sequence number)
            Sequence Number (raw): 1320241759
        [Next Sequence Number: 24      (relative sequence number)]
        Acknowledgment Number: 1      (relative ack number)
            Acknowledgment number (raw): 41376092
        Header Length: 20 bytes (5) = .... 0101
            Flags: 0x018 (PSH, ACK)
            Window: 255
                [Calculated window size: 255]
                [Window size scaling factor: -1 (unknown)]
                Checksum: 0xf99b [unverified]
                [Checksum Status: Unverified]
                Urgent Pointer: 0
                    [Timestamps]
                [Client Contiguous Streams: 1]
                [Server Contiguous Streams: 1]
            TCP payload (23 bytes)
```

## שכבה ה (Data): היישום

- התוכן שנשלח הוא? Hey Yossi! How are you?
- השרת מפרש את התו @ כפוקודה לניטוב הודעה פרטית, את המחרוזת עד לנקודותים כשם היעד, (Yossi) ואת השאר כתוכן ההודעה.

אפשר לראות שההודעה היא 30 בתים

Data (30 bytes)

02 00 00 00 45 00 00 46 58 db 40 00 80 06 00 00	... E F X @ ...
c0 a8 01 78 c0 a8 01 78 c1 d2 27 10 76 5b 3e 12	... x x ' v[>.
da 44 a7 3b 50 18 00 ff 8f 7e 00 00 40 59 6f 73	D ;P ~ @Yos
73 69 3a 48 65 79 20 59 6f 73 73 69 21 20 68 6f	si:Hey Y ossi! ho
77 20 61 72 65 20 79 6f 75 3f	w are yo u?

**מודל השירות:** התעבורה מראה חיבור מתמשך (Keep-Alive). (TCP) בኒגוד לHTTP-שבו כל בקשה פתוחה וסוגרת חיבור (לרוב), כאן ה-Socket-נשאר פתוח כל עוד המשתמש מחובר, מה שמאפשר קבלת הודעות "דחיפה" (Push) "מהשירות" (Push) (cgon) USERS\_UPDATE באופן מיידי.

Sequence Number	Source IP	Destination IP	Source Port	Destination Port	Protocol	Timestamp	Size
Seq=79 Ack=35 Win=65280 Len=17 [PSH, ACK]	192.168.1.120	192.168.1.120	49618	10000	TCP	Jan 10, 2026 17:26:15.460245000	61
Seq=35 Ack=96 Win=65280 Len=0 [ACK]	192.168.1.120	192.168.1.120	10000	49618	TCP	Jan 10, 2026 17:26:15.460245000	44

0000 02 00 00 00 45 00 00 39 58 df 40 00 80 06 00 00 ... E 9 X @ ...  
0010 c0 a8 01 78 c0 a8 01 78 27 10 c1 d2 da 44 a7 72 ... x x ' D r  
0020 76 5b 3e 30 50 18 00 ff 15 c5 00 00 55 53 45 52 v[>0p ... USER  
0030 53 5f 55 50 44 41 54 45 3a 4f 6d 65 72 S\_UPDATE :Omer

Name 11825: Packet, 61 bytes on wire (488 bits), 61 bytes captured (488 bits)  
Encapsulation type: NULL/Loopback (15)  
Arrival Time: Jan 10, 2026 17:26:15.460245000 Jerusalem Standard Time  
UTC Arrival Time: Jan 10, 2026 15:26:15.460245000 UTC  
Epoch Arrival Time: 1768058775.460245000  
[Time shift for this packet: 0.000000000 seconds]  
[Time delta from previous captured frame: 13.817000 milliseconds]

## שימוש באו:

במהלך הפיתוח השתמשנו באו למטרות הבאות:

שיפור ממשק המשתמש : GUI סיעודי בכתיבת קוד tkinter ליצירת הצעט ב colors וחדירים Direct messages של כל משתמש חדש שנוצר.

הבנת פרוטוקולים: הסבר של ההבדלים בין דגלי TCP ב Wireshark

Debugging - איתור בעיה שבה הממשק הגרפי לא עדכן את רשימת המשתמשים כשנוצר משתמש חדש

פרומפטים לדוגמה:

"צורך ל GUI ב Dark mode שיכיל צאט לכולם וחדירים Direct message לאנשים שמחוברים".

"תסביר לי ההבדלים בין דגלי TCP ב Wireshark ולמה אני רואה חבילות עם Len = 0".

"תקן את סדר הפעולות בשרת כך שמיד לאחר אישור החיבור, השרת ישדר את רשימת כל המשתמשים המחוברים לכל הלקוחות, כדי להבטיח שהמצטרף החדש יראה את כל המשתמשים הקיימים".