

## דוח מסכם - פרויקט ברשותן תקשורת מחשבים

### מגישים: אלון דולב ושרון קנדוב

#### חלק 1 - אריזת נתונים ולכידת מנות

##### 1. יצירתקובץ-CSV

לצורך הפרויקט יצרנו קובץ CSV שמכיל הודעות בשכבה היישום (Application Layer) בחרנו להשתמש בפרוטוקול HTTP משומש שהוא פשוט, ברור וקל לזהות ב-Wireshark.

שם הקובץ:

group01\_http\_input.csv

המודדות בקובץ:

msg\_id,app\_protocol,src\_app,src\_port,dst\_app,dst\_port,message,timestamp

הקובץ כולל סדרת בקשות ותשובות HTTP כמו GET, POST, 200 OK, Not Found 404 ועוד. כדי לדמות תקשורת בין דפן לשרת.

##### 2. הרצת המחברת Jupyter והדמית אריזת הנתונים

הפעלנו את המחברת SSOPKA בקורס מקומי דרך VSCode. זה מאפשר ליצור חבילות TCP אמיתיות שנשלחות דרך מערכת הפעלה.

במהלך העבודה ביצעו השלבים הבאים:

- טעינת קובץ CSV באמצעות pandas.

- בדיקת תקינות מבנה CSV.

- בניית כותרות של שכבות TCP/IP.

coturte IP (גרסה, כתובות מקור ויעד, אורך חבילה, checksum).

coturte TCP (src\_port, dst\_port, sequence, flags).

Application -> TCP -> IP -> ARP.

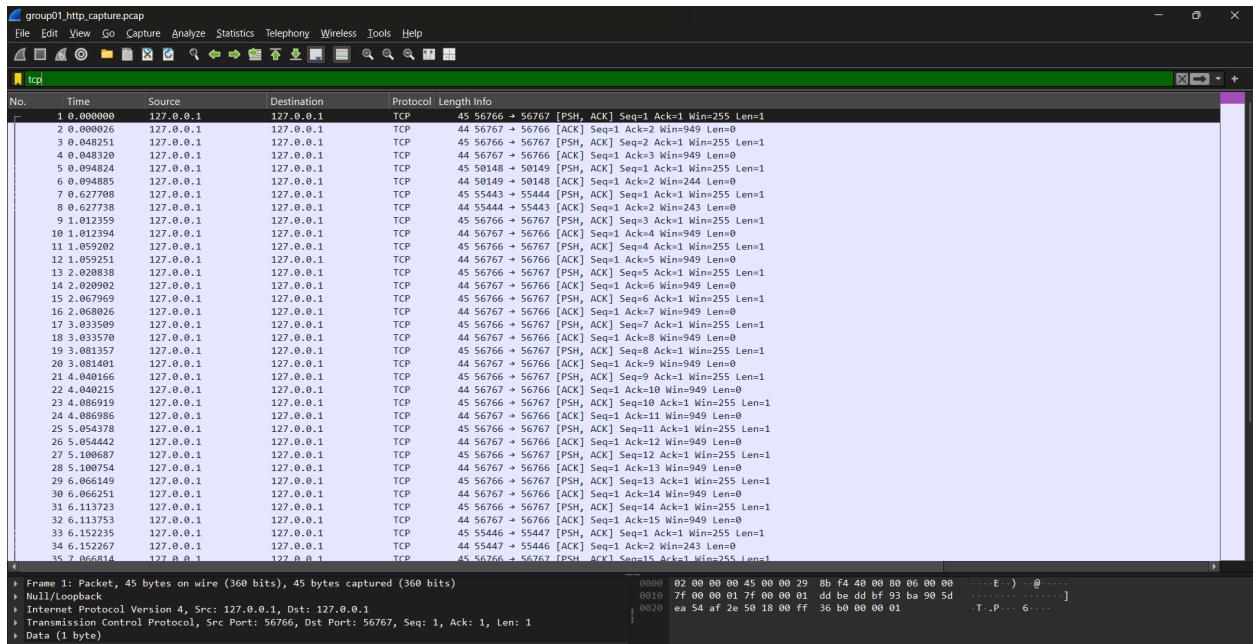
שליחת החבילות בפועל לכתובת 127.0.0.1 באמצעות Scapy Raw sockets או בהתקם למערכת.

המחברת שלחה בהצלחה את כל ההודעות מה-CSV לחבילות TCP ל-loopback.

The screenshot shows a Jupyter Notebook cell with Python code. The code uses the scapy library to find network interfaces, build TCP packets, and send them. The output shows the generated packet structure and its hex dump.

```
#!/usr/bin/python3
# Find interface name for Windows
if IS_WINDOWS and HAVE_SCAPY:
    try:
        print('\n'.join(get_if_list()))
    except Exception as e:
        print('Could not list interfaces:', e)

# Preview packet structure
src_ip = '127.0.0.1'
dst_ip = '127.0.0.1'
src_port = random.randint(1024, 65535)
dst_port = 12345
payload = b'Hello Packet (preview)'
pkt_preview = build_ip_header(src_ip, dst_ip, 20 + len(payload)) + build_tcp_header(src_ip, dst_ip, src_port, dst_port, payload) + payload
hexdump(pkt_preview)
```



### 3. לכידת התעבורה ב-Wireshark

הPLICIDA בוצעה באמצעות Adapter for loopback traffic capture על ממשק 127.0.0.1, לאחר והתקשרות בוצעה לכתובת 127.0.0.1. בוצע סינון תצוגה באמצעות הפילטר: .tcp.

בסקרינישוט ניתן לראות סדרת חבילות TCP שנשלחו בין כתובת המקור והיעד 127.0.0.1, כאשר כל חבילה מייצגת הودעה שנשלחה מקובץ CSV-Encapsulation (אריזה).

בדוגמה המוצגת נבחרה חבילת TCP הכוללת Payload באורך 1 byte, המתאימה להודעתה מה-CSV.

בכותרת TCP ניתן לראות:

פורט מקור ופורט יעד תואמים לערכיהם שהוגדרו בקובץ CSV-V-

daglimim mosgov PSH ו-ACK ha-muidim ul shelihat ntoni aflikziah,

mosfari Sequence ו-Acknowledgment mishkafim at mangnon Beckeret hzrimah shel TCP.

בכותרת IP מופיעות כתובת המקור והיעד 127.0.0.1, דבר המאשר שהשליחה בוצעה דרך ממשק loopback. Payload של החבילה מופיעים נתונים ההודעה משכבה Application, כפי שהוגדרו בשורת CSV.

לכידה זו מדגימה את תהליך האrizah המלא:

הודעת Application, עטיפה TCP, עטיפה IP, שליחה חבילת רשות שנקלטה ב-Wireshark.

ניתן לקשר בין שורות קובץ CSV לבין החבילות שנמצאו ב-Wireshark באמצעות שדה id msg. לדוגמה, שורה עם msg\_id=2 בקובץ CSV, המכילה בקשה GET, מופיעה ב-Wireshark חבילת TCP עם Payload תואם.

התאמאה זו מאשרת שכל הודעת Application מה-CSV נארזה, נשלחה ונלכדה כמתואר.

לאחר הPLICIDA שמרת את הקובץ group01\_http\_capture.pcap:

## **חלק 2 – מימוש מערכת צ'אט מבוססת TCP וניהוח תעבורת**

### **1. מבנה כללי של המערכת**

במסגרת חלק זה מימשנו מערכת צ'אט מבוססת TCP הכוללת שרת (server.py) ושני לקוחות או יותר עם משק גרפי (client\_gui.py).  
השרת מORIZן לחיבורים נכנים, שומר כל לקוח בשם ייחודי, ומשמש כמתווך בין לקוחות המעוניינים לשוחח זה עם זה.  
כל התקשרות מוצעת מעל TCP תוך שימוש ב-sockets בלבד, ללא שימוש בספריות מוכנות לניהול שירותים או לקוחות.

### **2. השרת – server.py**

השרת יוצר socket מסוג TCP, מזמן בפורט 5000 ומטפל במספר לקוחות במקביל באמצעות threads.  
בעת התחברות, כל לקוח שלוח לשרת את שמו הייחודי.  
השרת תומך בפקודות טקסטואליות פשוטות כגון HELLO, LIST, CHAT, MSG ו-QUIT, ומנהל מיפוי בין לקוחות מחוברים.  
כאשר לקוח מבקש לפתח צ'אט עם לקוח אחר, השרת בודק אם הלוקו קיים ומתווך את ההודעות ביניהם.  
כל הודעה מועברת דרך השרת ולא ישירות בין הלקוחות. השרת כולל טיפול בניטוקים לא צפויים של לקוחות.

### **3. הלוקו – client\_gui.py**

הлокו יוצר חיבור TCP לשרת בפורט 5000 ומספק ממשק גרפי מבוסס על Tkinter.  
המשק מאפשר הזנת שם משתמש, התחברות לשרת, צפיה ברשימה משתמשים מחוברים, בחירת משתמש לצ'אט ושליחת הודעות טקסט בזמן אמת.  
קליטת הודעות מתבצעת באמצעות thread "יעוד", בעוד שה-thread הראשי אחראי על המשק הגרפי ושליחת הודעות.

### **4. אופן הרצה**

1. מפעלים את השרת: `python server.py`

2. מפעלים מספר לקוחות: `python client_gui.py`

3. כל לקוח מזין שם ייחודי ומתחבר לשרת.

4. הлокו בוחר משתמש אחר מרשימה המשתמשים ופותח צ'אט.

5. לאחר החיבור, מתבצעת שיחה דו-כיוונית בזמן אמת.

המערכת תומכת במספר לקוחות בו-זמנית.

## 5. דוגמאות קלט ופלט

להלן דוגמאות לתקשורת טקסטואלית בין ללקוח לשרת ב프וטוקול ה-**TCP**. הדוגמאות מייצגות את ההודעות שנשלחות בפועל דרך חיבור ה-**socket**, ללא תלות במכשיר הגרפי.

תרחיש 1 – התחברות ורישימת משתמשים

Client → HELLO Alon  
Server → SYSTEM Hello Alon, you are connected.  
Client → LIST  
Server → USERS Alon, Shiran

תרחיש 2 – פתיחת שיחה ושליחת הודעה

Client → CHAT Shiran  
Server → SYSTEM You are now chatting with Shiran  
Client → MSG Hello, how are you?  
Server → CHAT Alon: Hello, how are you?

תרחיש 3 – ניתוק

Client → QUIT  
Server → SYSTEM Disconnected

הודעות אלו מועברות על גבי חיבור TCP, מקודדות כמחוזות טקסט, ומנותחות לצד השרת לצורך ניהול חיבורם, ניתוב שיחות והפצת הודעות בין משתמשים.

## 6. לכידת תעבורת ב-Wireshark

לכידת התעבורת בוצעה באמצעות Wireshark על ממשק **Adapter for loopback traffic capture** על כתובת IP 127.0.0.1, לאחר והתקשרות בוצעה לכתובת 127.0.0.1.

שלבי הלכידה:

1. פתיחת Wireshark.
2. בחירת **Adapter for loopback traffic capture**.
3. שימוש בפילטר: **tcp.port == 5000**.
4. התחלה **Capture**.
5. הפעלת השרת ושני לקוחות והרצת צ'אט.
6. יצירת הלכידה ושמירה לקובץ: **group01\_chat\_capture.pcap**.

## 7. ניתוח תעבורת עד שכבה הרשת (IP + TCP)

ב-Wireshark ניתן לראות בבירור:

- פתיחה חיבור באמצעות 3-way handshake (TCP (SYN, SYN-ACK, ACK) של side-to-side).
- שימוש בפורט 5000 מצד השרת וב포רטים זמינים מצד הלקוחות.
- חבילות PSH ו-ACK בעט שליחת הודעות טקסט בין הלקוחות.
- ב-Stream TCP ניתן לראות את פקודות הפרוטוקול והודעות שהוחלפו (HELLO, CHAT, MSG).
- בשכבה ה-IP נראות כתובות המקור והיעד 127.0.0.1.
- בשכבה ה-TCP נראים מספרי רצף, חלונות ו-checksum.

התעבורת יציבה, דו-כיוונית ועומדת בתקן TCP.

Pawtocol TCP Chat

Server IP: 127.0.0.1 Port: 5000 Connect

Name: Alon

Online cats: Alon, Shiran

[SYSTEM] Meow! Connected as Alon

[SYSTEM] Welcome.

[SYSTEM] Hello Alon. Choose your fellow feline from list to start.

[SYSTEM] Connected to Shiran. Use MSG <text>.

Alon: Hey!

Alon: How are you?

Refresh Start Chat

Pawtocol TCP Chat

Server IP: 127.0.0.1 Port: 5000 Connect

Name: Shiran

Online cats: Alon, Shiran

[SYSTEM] Meow! Connected as Shiran

[SYSTEM] Welcome.

[SYSTEM] Hello Shiran. Choose your fellow feline from list to start.

[SYSTEM] Alon connected to you. Use MSG <text>.

Shiran: Hey!

Shiran: How are you?

Refresh Start Chat

Capturing from Adapter for loopback traffic capture

tcp.port == 5000

No.	Time	Source	Destination	Protocol	Length	Info
98	130.219364	127.0.0.1	127.0.0.1	TCP	62	5000 → 63900 [PSH, ACK] Seq=14
99	130.219376	127.0.0.1	127.0.0.1	TCP	44	63900 → 5000 [ACK] Seq=14
100	130.222684	127.0.0.1	127.0.0.1	TCP	49	63900 → 5000 [PSH, ACK] Seq=15
101	130.222705	127.0.0.1	127.0.0.1	TCP	44	5000 → 63900 [ACK] Seq=16
102	130.222802	127.0.0.1	127.0.0.1	TCP	62	5000 → 63900 [PSH, ACK] Seq=17
103	130.222837	127.0.0.1	127.0.0.1	TCP	44	63900 → 5000 [ACK] Seq=18
106	136.817727	127.0.0.1	127.0.0.1	TCP	56	63899 → 5000 [PSH, ACK] Seq=19
107	136.817781	127.0.0.1	127.0.0.1	TCP	44	5000 → 63899 [ACK] Seq=20
108	136.817965	127.0.0.1	127.0.0.1	TCP	88	5000 → 63899 [PSH, ACK] Seq=21
109	136.817995	127.0.0.1	127.0.0.1	TCP	44	63899 → 5000 [ACK] Seq=22
110	136.818011	127.0.0.1	127.0.0.1	TCP	98	5000 → 63900 [PSH, ACK] Seq=23
111	136.818027	127.0.0.1	127.0.0.1	TCP	44	63900 → 5000 [ACK] Seq=24
116	144.665121	127.0.0.1	127.0.0.1	TCP	53	63899 → 5000 [PSH, ACK] Seq=25
117	144.665145	127.0.0.1	127.0.0.1	TCP	44	5000 → 63899 [ACK] Seq=26
118	144.665297	127.0.0.1	127.0.0.1	TCP	60	5000 → 63900 [PSH, ACK] Seq=27
119	144.665319	127.0.0.1	127.0.0.1	TCP	44	63900 → 5000 [ACK] Seq=28
125	149.290533	127.0.0.1	127.0.0.1	TCP	61	63899 → 5000 [PSH, ACK] Seq=29
126	149.290562	127.0.0.1	127.0.0.1	TCP	44	5000 → 63899 [ACK] Seq=30
127	149.290701	127.0.0.1	127.0.0.1	TCP	68	5000 → 63900 [PSH, ACK] Seq=31
128	149.290729	127.0.0.1	127.0.0.1	TCP	44	63900 → 5000 [ACK] Seq=32

## 8. שימוש ב-**AI** - מטרות ודוגמאות פרומפטים

במהלך פיתוח הפרויקט נעשה שימוש בכל'י בינה מלאכותית (AI) לצורך סיוע בתכנון, מימוש ותיעוד המערכת. השימוש ב-**AI** לא יהיה חלק מהריצה של המערכת עצמה אלא כל'י עזר למפתח.

מטרות השימוש ב-**AI**:

- הבנת דרישות הפרויקט ופרויקט למשימות פיתוח
- תכנון פרוטוקול תקשורת פשוט מעלה TCP
- שיפור מבנה הקוד והטיפול במקרי קצה
- ניסוח ורחיבת תיעוד טכני לדוח

דוגמאות לפרומפטים שנעשה בהם שימוש:

- *"Design a simple text-based protocol for a TCP chat server with multiple clients"*  
התקבל מבנה פקודות בסיסי (HELLO, LIST, CHAT, MSG, QUIT)
- *"?How should a TCP server handle unexpected client disconnections in Python"*  
סיעם בהבנת טיפול ב-exceptions וניקוי חיבורים
- *"Explain TCP packet flow for a simple client-server chat application"*  
סיעם בכתיבה סעיף ניתוח התעבורה בדוח
- *"Improve the clarity of this technical explanation for a networking course report"*  
שיפור ניסוחים בדוח המסכם

השימוש ב-**AI** תרם ליעילות הפיתוח ולדיקת התיעוד, אך כל הקוד נכתב ונבדק על ידיינו.