

חלק א

הIP של השרת: 192.168.1.30
הIP של הקליינט: 192.168.1.44

נשאל בפורום: "האם מותר לשנות גם את קוד השרת כדי שהיא אפשר לשמר על החיבור פתוח בזמן שליחת שתי הודעהות?"

נענה: "שלום, זה מה שתתרגש מבקש. תודה"
על כן, שינוינו את קוד השרת נוספת על קוד הלקוח, ככללו.
קוד השרת:

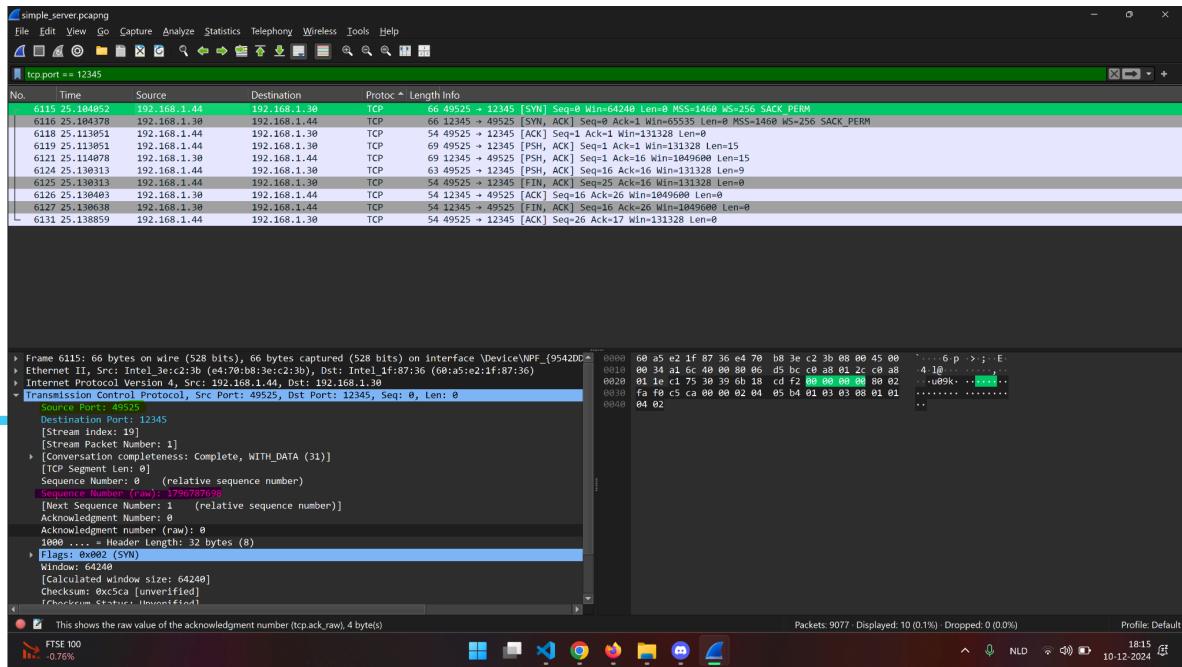
```
tcp_server.py > ...
import socket
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('', 12345))
server.listen(5)

while True:
    client_socket, client_address = server.accept()
    print('Connection from: ', client_address)
    data = client_socket.recv(100)
    print('Received: ', data)
    client_socket.send(data.upper())
    data = client_socket.recv(100)
    print('Received: ', data)
    client_socket.close()
    print('Client disconnected')
```

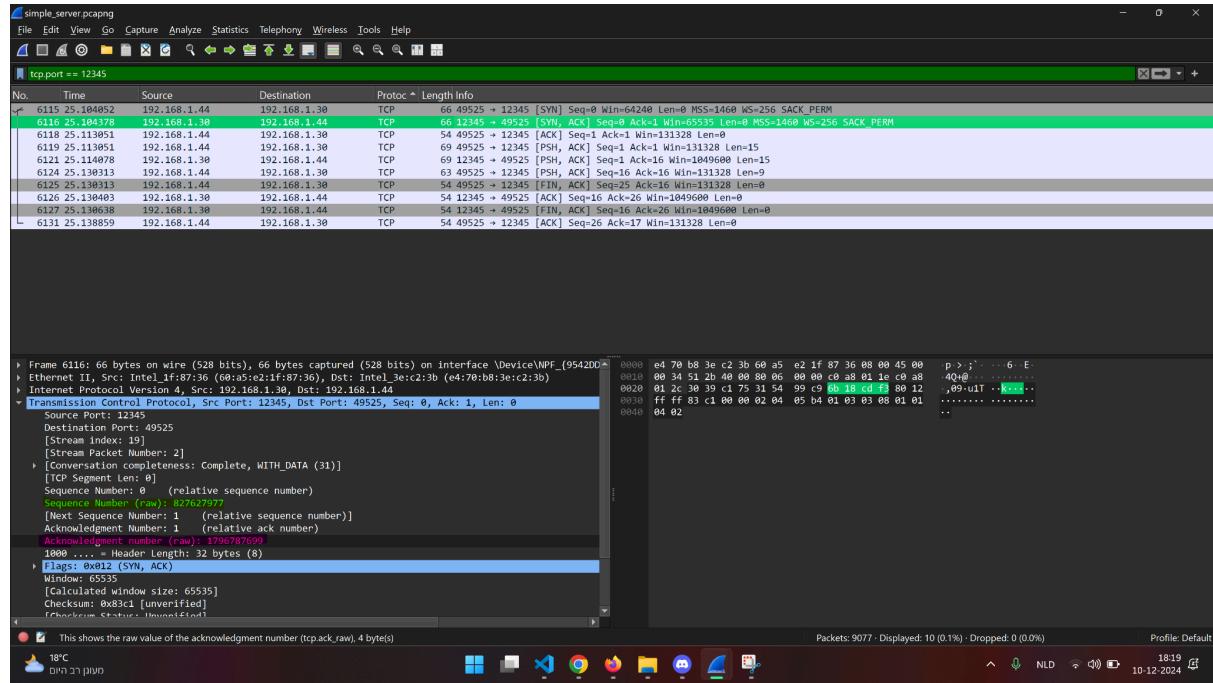
קוד הלקוח:

```
tcp_client.py > ...
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.1.30', 12345))
s.send(b'Gideon and Ayal')
data = s.recv(100)
print("Server sent: ", data)
s.send(b'329924567')
s.close()
```

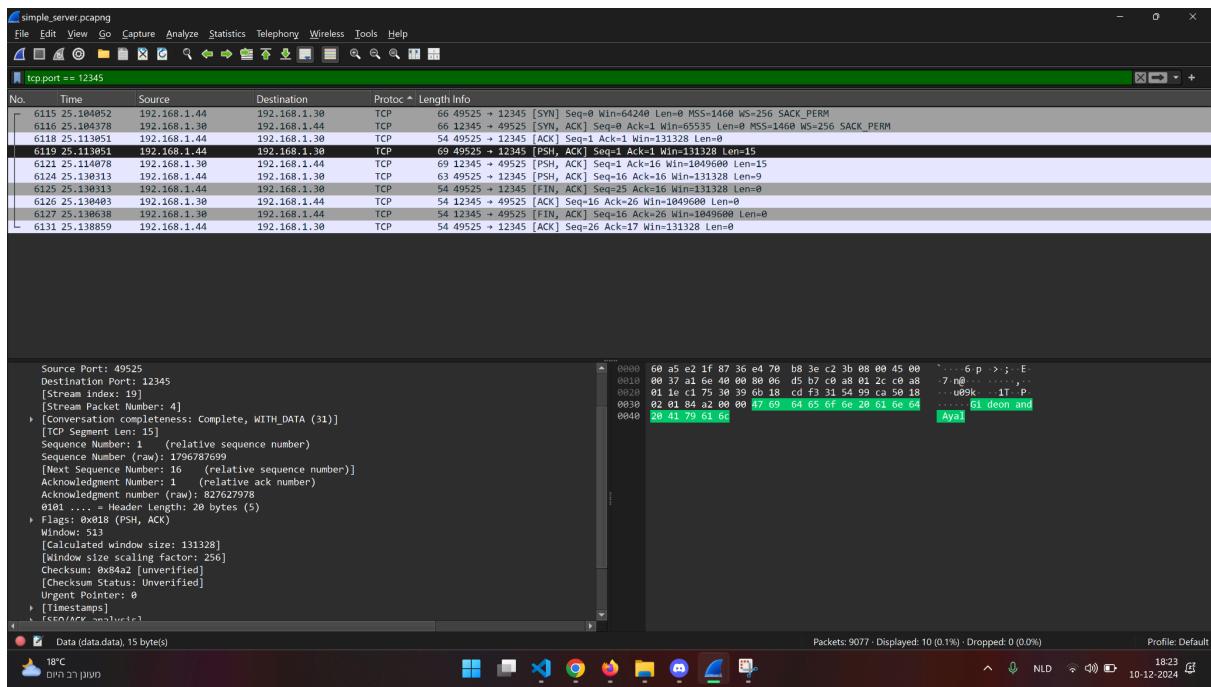
כפי שניתן לראות, בתחילת החיבור הלקוח פונה לשרת בבקשת SYN, המספר הסידורי המקורי הוא הדבר הנחמד הבא: 1796787698. והמספר של ACKNN הוא 0.



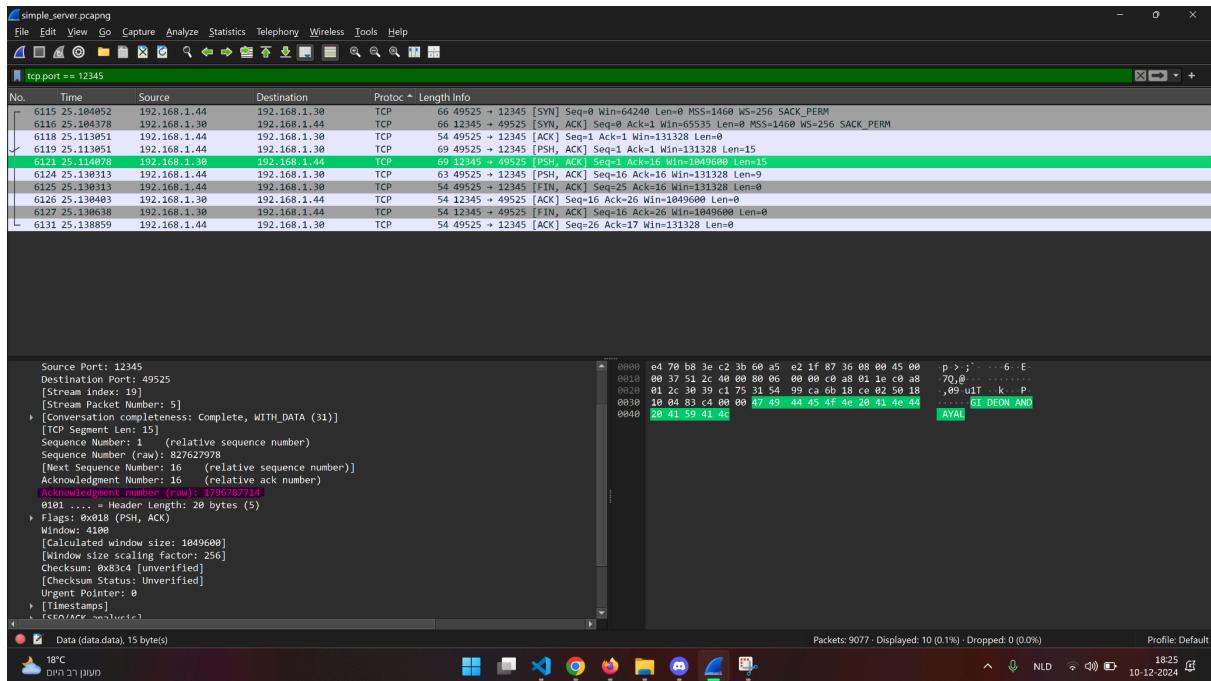
השרת מניב ב-ACK SYN ומבקש להסתنصرן. אם כך, המספר של ה-ACKNN CUT נדב ב-1 מהSEQ. השהיה מקודם, שחריו הוא מחזיר ACK. וכך המספר הוא 1796787699. המספר של SEQ של ה-ACK הוא .827627977.



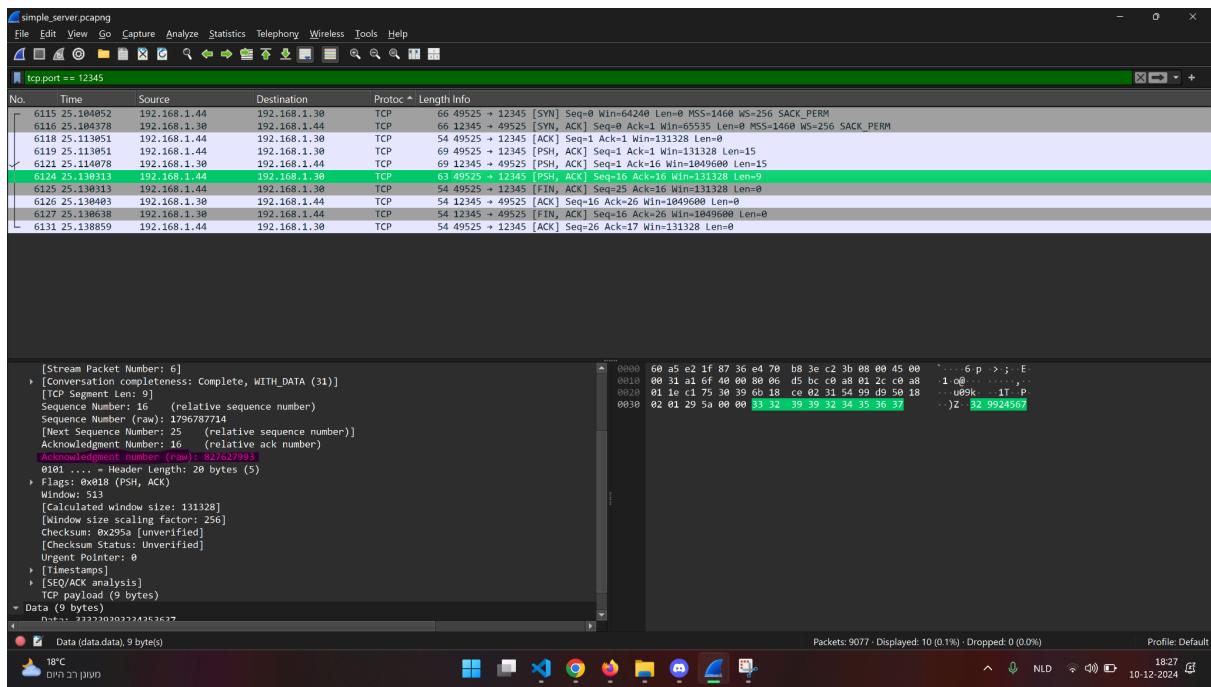
CUT לא נשים צילום מסך בכל רגע כדי לא להזכיר. הלקוח מחזיר ACK על ה-SYN של השרת, מעלה את ACKNN ב-1 (של השרת) ולאחר מכן הוא .827627978. לאחר מכן, הלקוח שלוח את השם (במקרה שלנו, השמות שלנו). לא התקדמנו עדין בSEQ, ומספר ACKNN עלה ב-1.



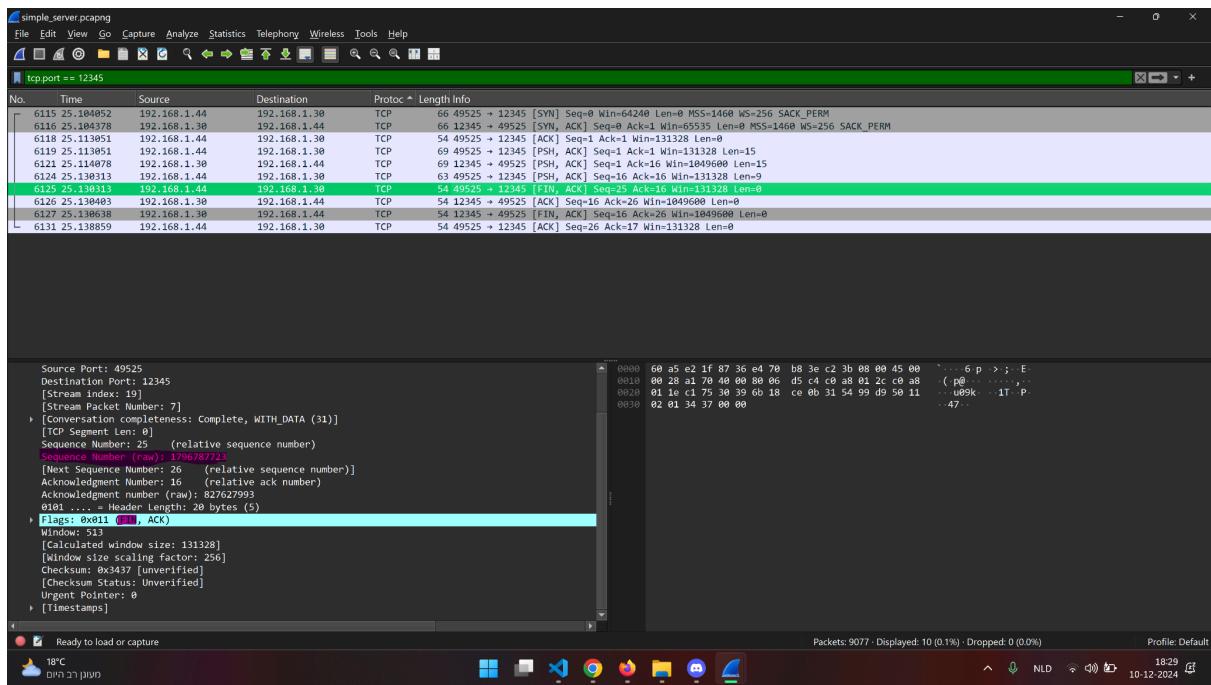
השרת מוחזר ACK כשהΝΑΚ נדל ב15 בתים, 1796787714 על כך שקיבל את השמות, ושולח אותם באמצעות גוזלות.



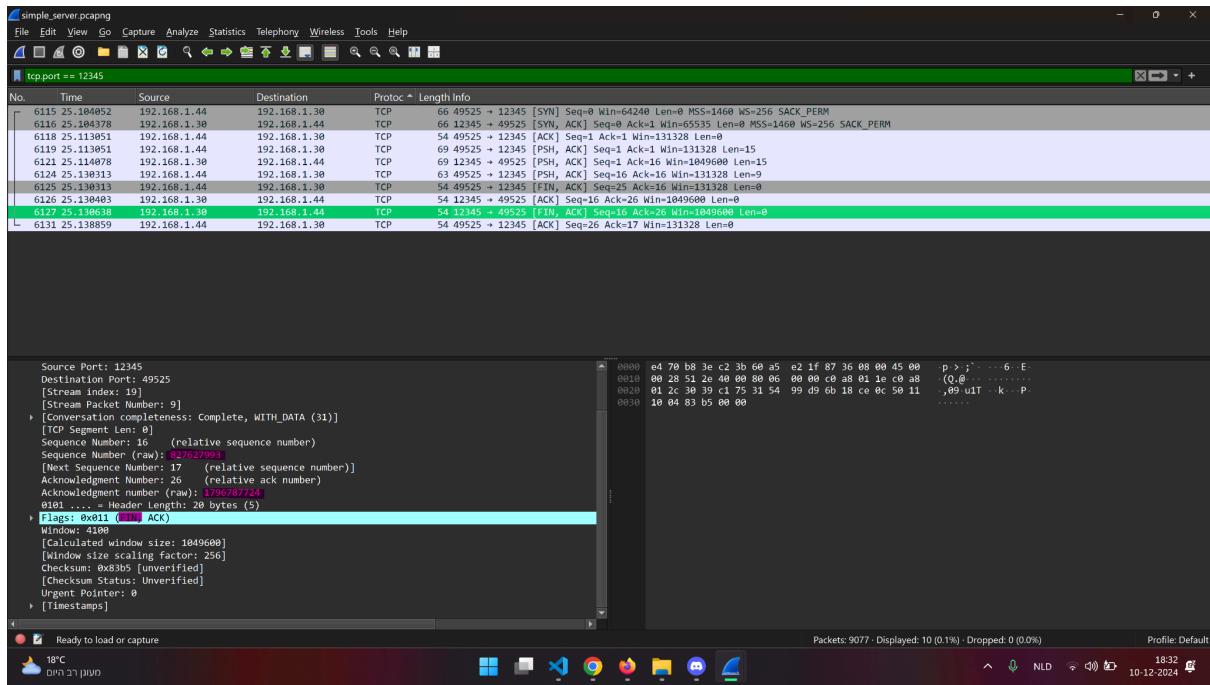
כעת הלוקה מאשר שקיבל את השמות, ושולח את התז, ועל כן הΝΑΚ נדל ב15 ל-827627993.



כעת הלקוח מודיע שסימן לשלוח, ככלומר FIN, וمعدכן ב+9 עברו התז, וכן SEQn עלה .1796787723.



כעת שוב, השרת מודיע שקובל, הנקוד, נמדד ב1 בעבור ACKn, ולאחר מכן, מעדכן סיסים בעזרת FIN, הלקוח מעדכן שקיבל, ומעליה ב1 את ACKn אל 1796787724



וכמוון הלקוח מעדכן שקיבל את הסגירה.

cutת לחץ של ה17 וכוכו.

- בעבר 17:

- השרת

■

מקבל חיבור, מדפס שקיבל אותו, מקבל עוד חיבור וככל. עברו החיבור השני, מקבל 1024 בתים, מודיע שקיבל ומחזיר 7 בתים ראשונים. עברו החיבור הראשון, מקבל 1024 בתים, מודיע שקיבל, ומחזיר 5 בתים ראשונים. לאחר מכן, סוגרים את החיבור הראשון, ומנסים לבצע פעולה זו פעמיים.

- הלקוח

■

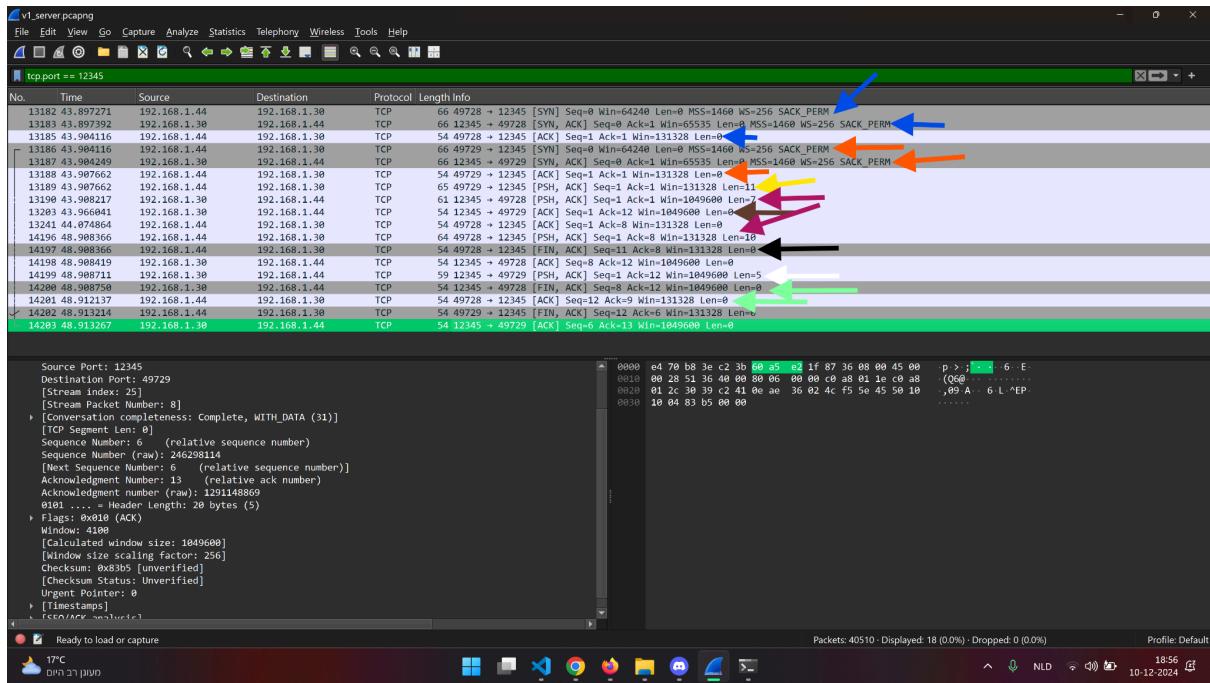
יש לו שתי הודעות, נסמן M וM1. יוצר סוקט S ומתחבר אל השרת. יוצר סוקט נסף, S, ומחברו אל השרת. שלוח בזה האחרון את M. ישן 5 שניות, שלוח בS את M1, מקבל בחזרה מההשרת החזר באוטו הסוקט. סוגר את S. מקבל מההשרת שלח בS, וסגור את S.

- cutת לwireshark

■

הלקוח פותח סוקט (נסמן בקצרה) 8 בכחול, ומבקש להסתנכרן עם השרת. השרת מאשר את הקמת החיבור, ו8 מאשר שקיבל. cutת, הלקוח פותח סוקט נסף (נסמן בקצרה) 9 ובכחותם, ושוב מבקש להסתנכרן, השרת מאשר ו9 מאשר שקיבל. cutת 9 שלוח לו את M בכחוב, השרת מחזיר אל 8 חלק מההודעה בוורוד, "so, Foo". מודיע 9 שקיבל ממנו את ההודעה. cutת 8 מודיע שקיבל את הורוד, ומבקש לשולח את M1. בנסף, הוא שלוח לשרת שישים בשחור. השרת מודיע שהוא קיבל, ושלוח אל 9 חלק מההודעה בלבד, (בר ורמות, כל מיני שיטויות של RTL אז לא יצא טוב

באנגלית) ומודיע ל8 שהוא סגור, 8 מאשר שקיבל (בירוק). 9 מבקש לסגור, והשרת מודיע שקיבל.



בעבור 27:

השרת:

- השרת פותח סוקט ורץ בוללא אינסופית, מאשר חיבור חדש, וכל עוד יש בבראף מידע שהוא יכול לעלות לשכבות האפליקציה הוא מעלה 5 bytes
- ושולח אותם בחזרה ללקוח ב-**uppercase**. לאחר מכן אין עוד מידע שהוא מקבל, והוא סגור את החיבור וממשיר לחכotta להתחברויות נוספות.

להלן:

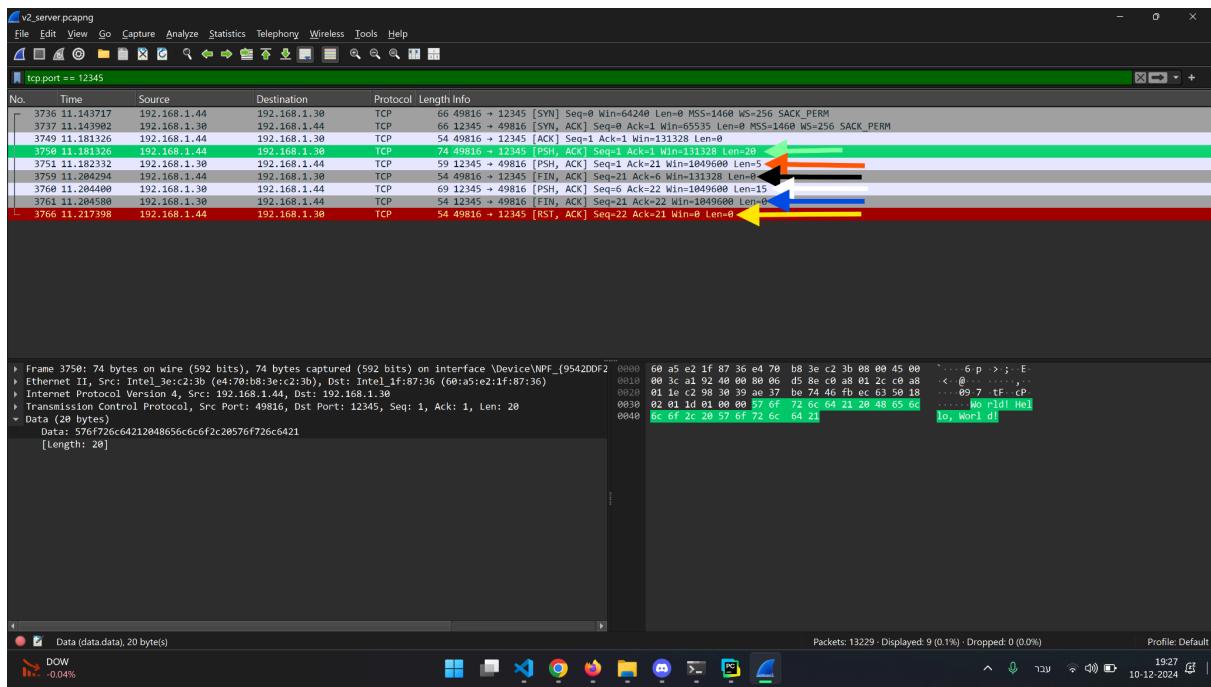
■

- הלקוח פותח סוקט שולח הودעה 'Hello, World! Hello, World!' מעלה לשכבות האפליקציה את התשובה מהשרת, סגור את הסוקט ומדפיס את התשובה למסך.

cut let wireshark

○

- שוב, הלוקה פותח סוקט, מבקש להתחבר, השרת מאשר והכל כרגע.
- הלקוח שולח את ההודעה (ירוק). השרת מחזיר ACK על הכל ושולח את ההודעה WORLD
- 5 בתים לפיה האמור בקוד. הבארף בגודל 5 - لكن מועלה לשכבות TCP הבודדות בגודל 5 אר-TCP בשכבות הרשות מחזיר ACK על כל ההודעה שהתקבלה (כתום). הלקוח סגור ומודיע שישים לשולוח (שחור).
- השרת מחזיר cut בפעם אחת אבל שלוש הודעות שכל אחת בגודל 5, ומודיע שישים (לבן+כחול). cut מקבל RST כי הלוקה כבר סגר עצמו (צהוב).



• בעבר נז:

- השרת:

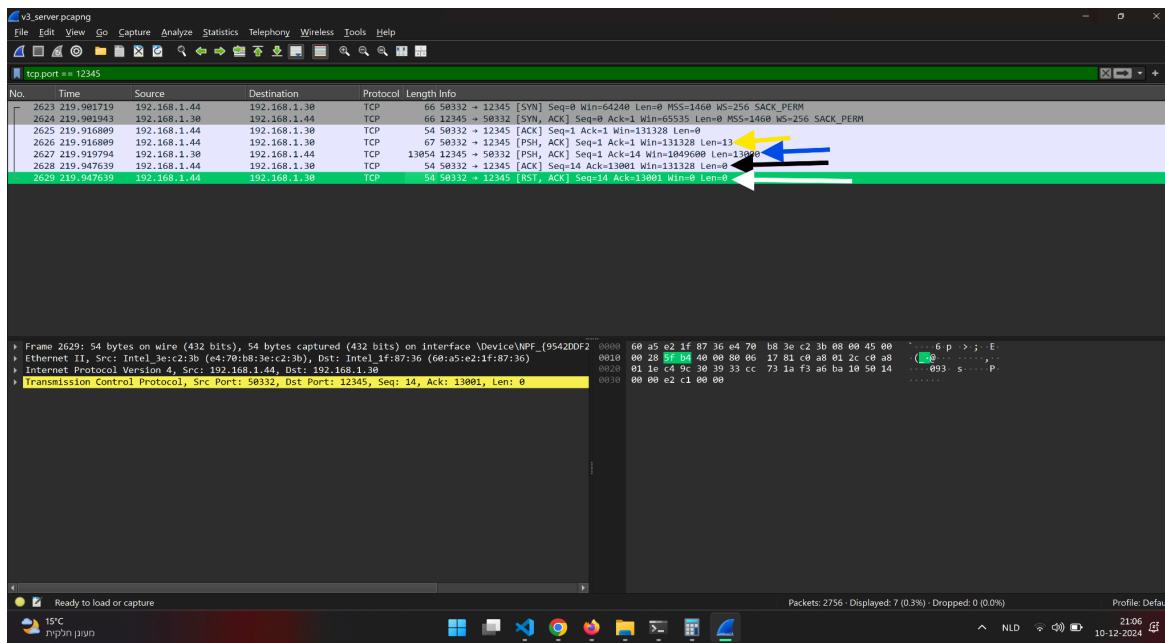
השרת פותח סוקט ומתחילה לróż בולולה אינסופית. הוא מאפשר חיבור של לקוחות וכל עוד יש הודעות שניתן להעלות לשכבות האפליקציה הוא מעלה עד 1024 bytes ומחזיר ללקוח את ההודעה כפول uppercase 1000 bytes. לאחר סיום שליחת ההודעות הוא סגור את הסוקט של הלוקו ומסHIR לחכות להתחברויות נוספות.

- הלוקו:

הלוקו פותח סוקט ושולח לשרת 'Hello world'! ולאחר מכן מבצע פעמיים receive של 1024 בתים לשכבות האפליקציה ומדפיס אותם ולאחר מכן סגור את הסוקט.

- כתעת לwireshark

מתחילה קרניל עם ack - syn-ack על מנת להקים חיבור TCP. בצד הימני ניתן לראות את ההודעה באורך 13 בתים שהלוקו שולח. לאחר מכן בצד הימני השרת מוחזירה ACK 14 על ההודעה של הלוקו ומחזיר לו בונסף הודעה באורך 13000 זהה בדיקת WORLD HELLO!. לאחר מכן שכתוב בקוד השרת. בשחור ניתן לראות שהלוקו מוחזיר ACK 10000 כמו שכתוב בקוד השרת. בצד ימין ניתן לראות שהלוקו קיבל את ההודעה בגודל 13000. לאחר מכן הלוקו נסגר בקוד ולכן שולח לשרת בשורה בלבד ACK RST. כמובן בלקוח קיבלו פעמיים 1024 בתים שהודפסו בגודל הבארט המוגדר בקוד ועל כן ישנו חלק גודל שהאפליקציה בחרה שלא לקרוא.



• בעבור 4:

- ## השראת:

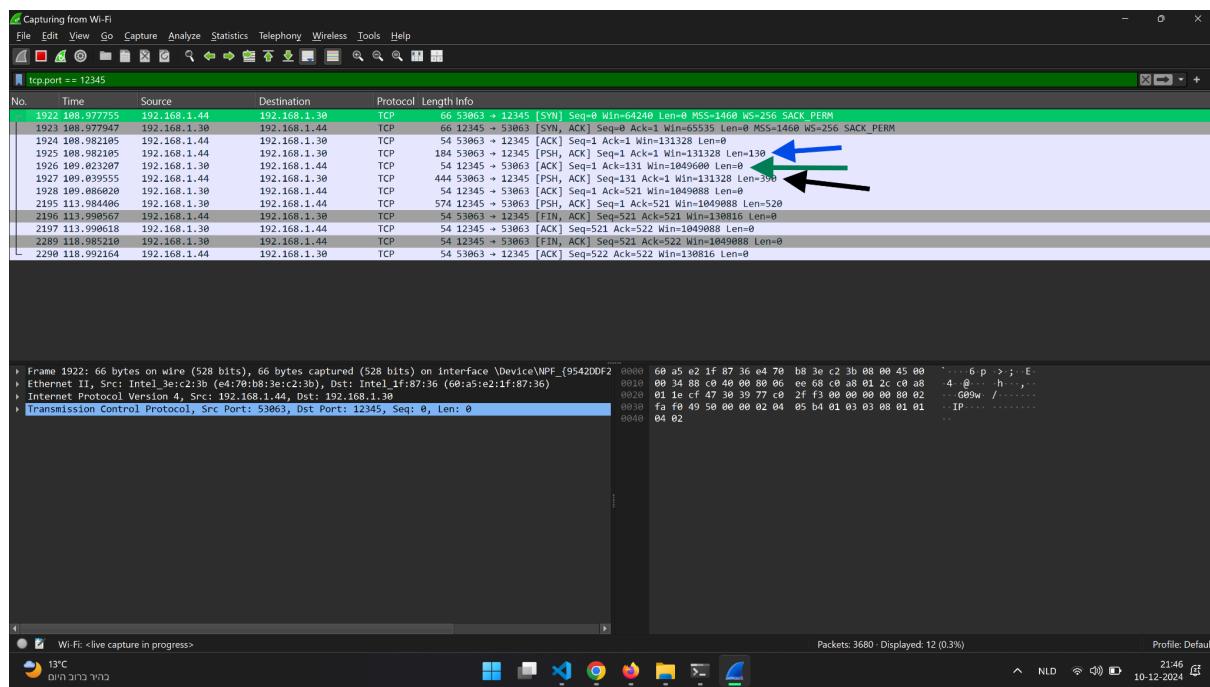
הشرط פותח סוקט ראשי ורץ בלולאה אינסופית. מאשר חיבור של לקוב
ולאחר מכן ישן 5 שניות. מעלה לשכבת האפליקציה לכל היותר 1024
בתים מדפיס אותם ומחייב ללקוח את ההודעה ב upper case . וחוזר לישון
ולהעלות הודעות לשכבת האפליקציה. לאחר שאין יותר הודעות להעלות
לשכבת האפליקציה השרת סוגר את החיבור עם הלקוב ומacha להתחברות
של לקוב נספ.

הלקוח:

הלקות פותח סוקט כאשר הודעה היא HELLO WORLD! הוא מגדיר HELLO WORLD*10 ושולח את הודעה ה-n'ל ב 4 SEND כלומר זה שולח לשרת 40 פעמים HELLO WORLD! לאחר מכן הוא מעלה לשכבה האפליקציה את התשובה של השרת וסגור את הסוקט.

וכעת ל wireshark

השרות הראשונות מיצגנות את הריטוֹאֵל הקבוע `ack - ack` או `syn - syn` על מנת ליצור חיבור TCP. הלוקה שולח את ההודעה כפולה 10 בפעם הראשונה והיא נשלחת לשרת (כחול). השרת מחזיר ACK בירוק שקיבל את ההודעה. לאחר מכן נשלחים כל שאר ה-SEND בבת אחת לשרת (בשחור). בשורה הבאה השרת מחזיר ACK שקיבל גם את כל ההודעות הנ"ל. בשורה הבאה השרת שולח את ההודעות בבת אחת בחזרה על כולם בupper case (לבן). ומסיים את החיבור ע"י כך שהלוקה שולח דגל FIN והשרת עונה לו ACK. והפוך ממשיכים על FIN של השרת. הסבר אפשרי הוא שבהתחלת אין פקודות YL ON لكن השרת מיד שולח את ההודעה הראשונה. ולאחר מכן כיוון שיש פקודות YL ON לא נשלח ישר כל הודעה ולכן רואים אותן בסוף נשלחים בבת אחת. השרת לא ישר עונה על ההודעה הראשונה שקיבול ונכנס ל-5 שניות של שינוי ולאחר מכן כבר כל שאר ההודעה הגיעה וכך הוא שולח בת אחת את התשובה שלו על כל מה שהגיע.

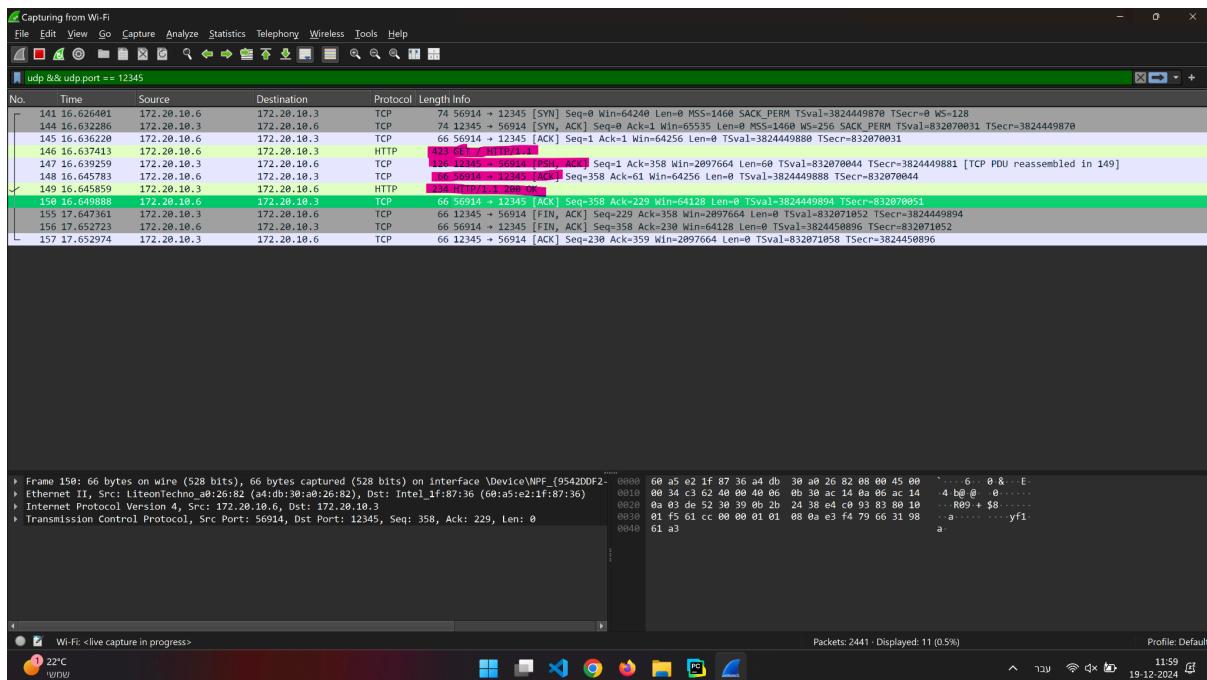


הערה לקראת הסוף: הרצינו את כל הקודים כמה פעמים אך לאחר הריצה אחת
 עיצרנו את hark.h wireshark שכן אחרת הינו הולכים לאיבוד בתוך כל ההודעות בין אותם
 מחשבים. חשוב לציין שלא היה אצלנו ספציפיות שינוי בהתנהנות בין הריצות של
 אותו קוד.

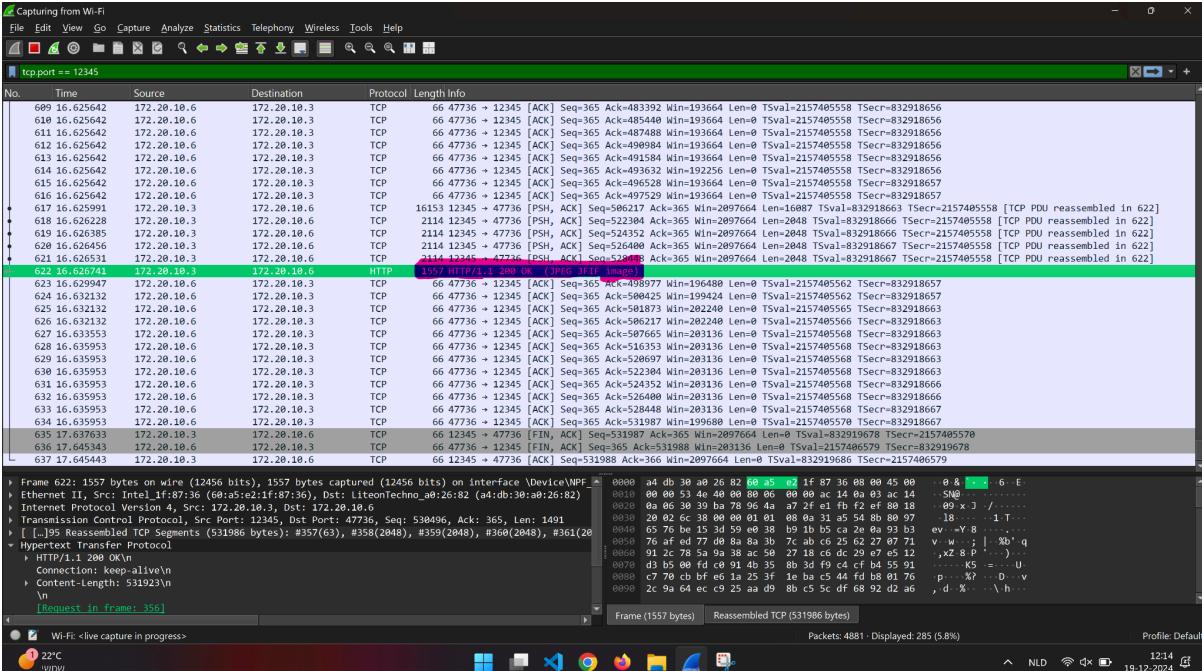
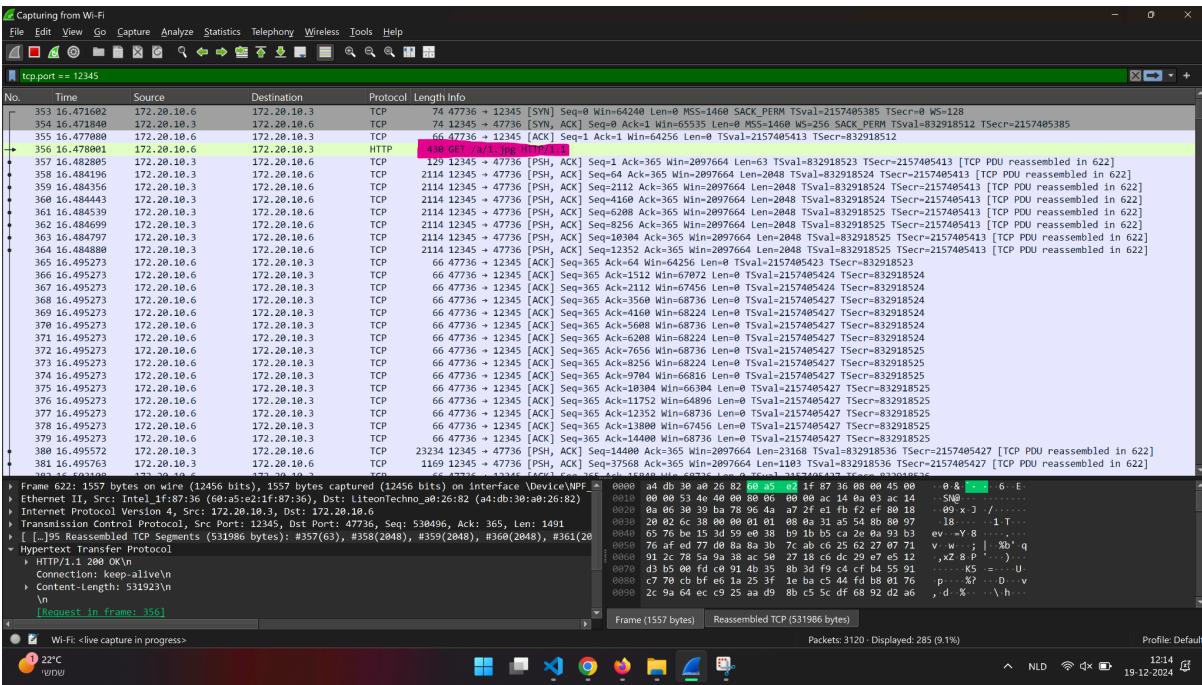
חלק ב - Wireshark שרת ולקוח

הלקוח: 172.20.10.6, השרת: 172.20.10.3

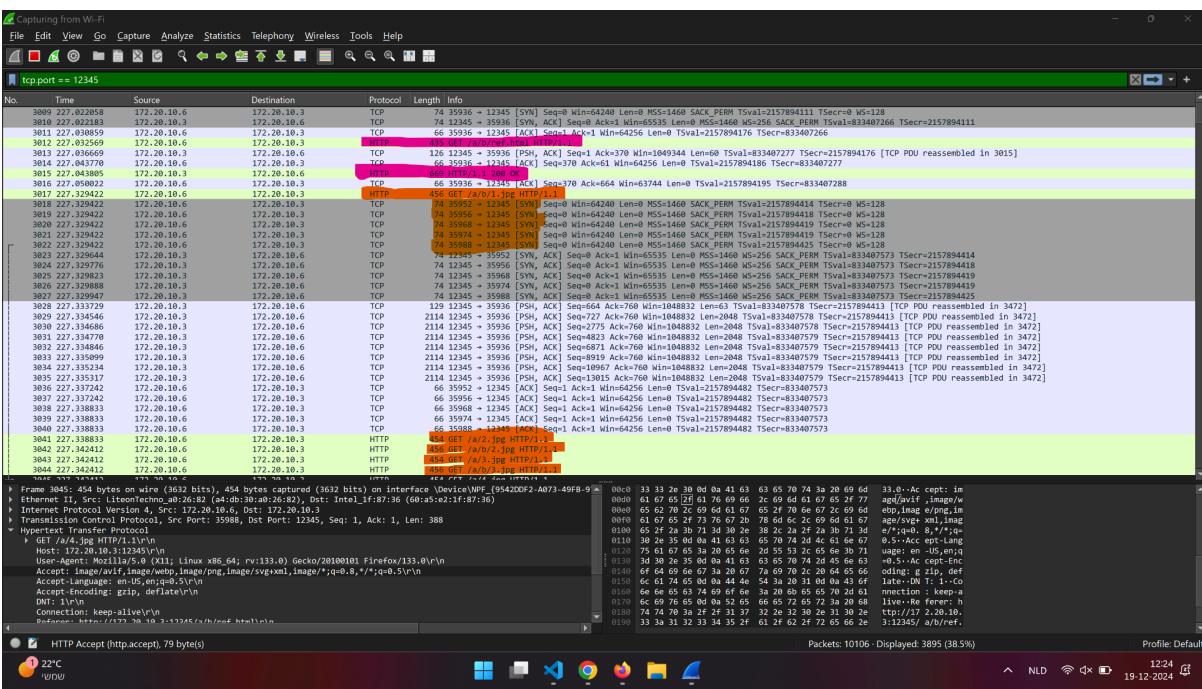
1. ניגשנו אל העמוד הראשי, index.html בערצת /. כפי שניתן לראות בתמונה, התרבצע סyncron בין הלקוח לשרת. לאחר מכן, הלקוח בקש את / בשכבה האפליקציה, השרת החזיר לו חלק מהמידע בשכבה התעבורה, הלקוח אישר שקיבל, והשרת החזיר לו 200 בשכבה האפליקציה באמצעות פרוטוקול HTTP בלבד עם שאר המידע, ולאחר מכן הסוגרים את החיבור. שליחת תוכן מסומן בורוד.



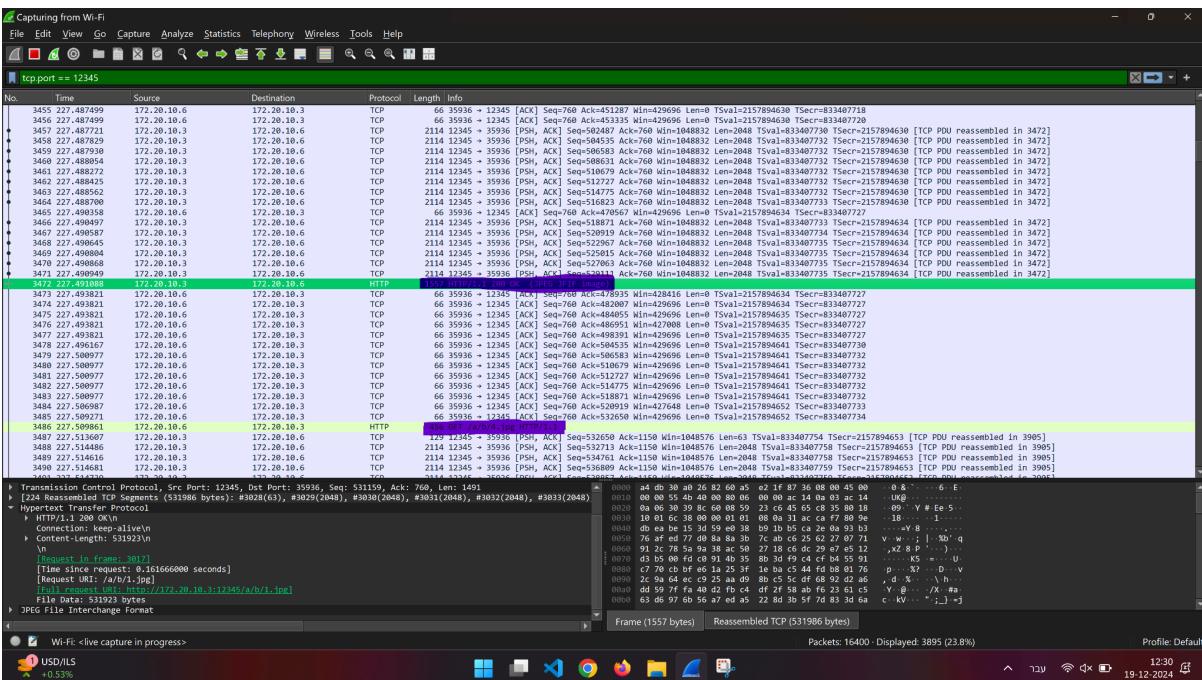
2. הלקוח ניגש אל <http://172.20.10.3:12345/a/1.jpg>, בפרוטוקול HTTP מבקש את התמונה. לאחר מכן השרת מתחילה להחזיר בחתיות (הרבה כללו) את התמונה, הלקוח מאשר שקיביל וכך הרבה פעמים.שוב, כמשמעותם, השרת שולח 200 בשכבה האפליקציה באמצעות HTTP ISOGRIM את החיבור.



3. הלקוח ניגש אל <http://172.20.10.3:12345/a/b/ref.html>, שולחים 200 כרגע על זה שנשלחו, ואז קורה משוה שונה (עד כאן (הקפה ראשונה) בווורד). הלקוח מבקש את אחת התמונות, נפתחים 5 חיבורים חדשים (בכთום), כי הס"כ יש 6 תמונות שונות בדף, על אף שכל תמונה מופיעה פעמיים. מסתנכרנים אותם. אחריו זה שלוחים את המידע של התמונה הראשונה, ואז עם אותם חיבורי TCP החדשם, הלקוח מבקש עוד אלמנטים, ושולחים אותם בנפרד. חיבורים חדשים, סוקטים חדשים וכו'.

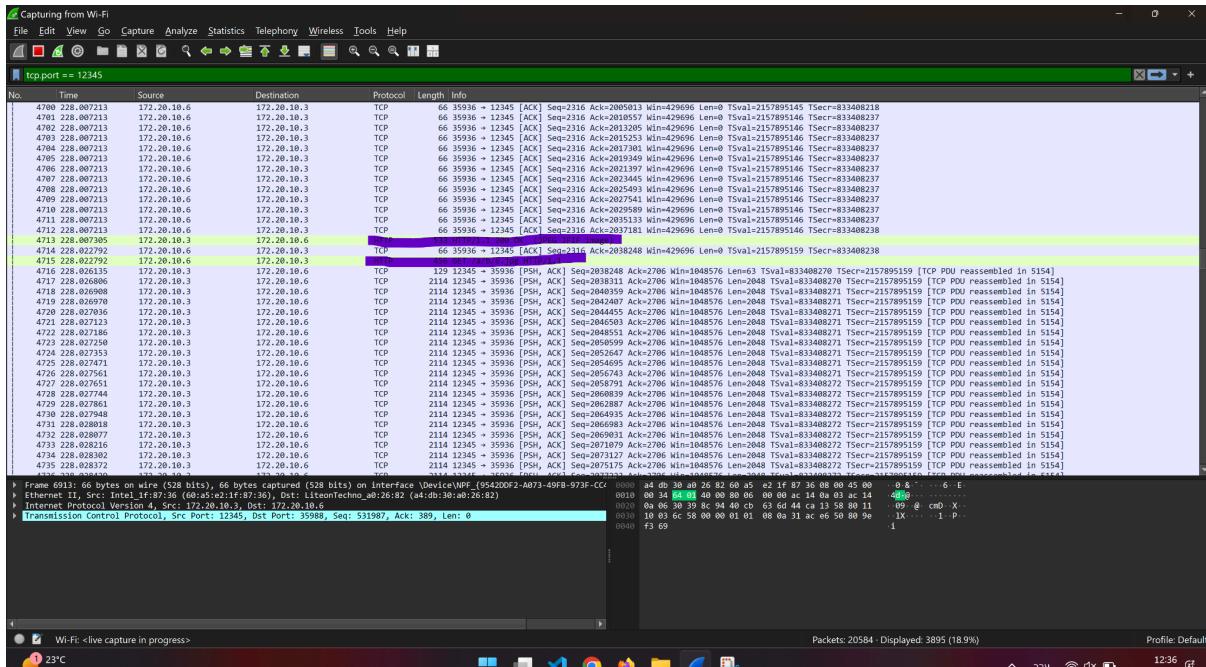


לאחר מכן נשלח המידע לשאר התמונות כפי שראינו קודם בפקtotות רבות. בשלב מסוים השרת מאשר שסימן עם התמונה הראשונה, וcutת הלוח יכול לבקש תמונה נוספת.

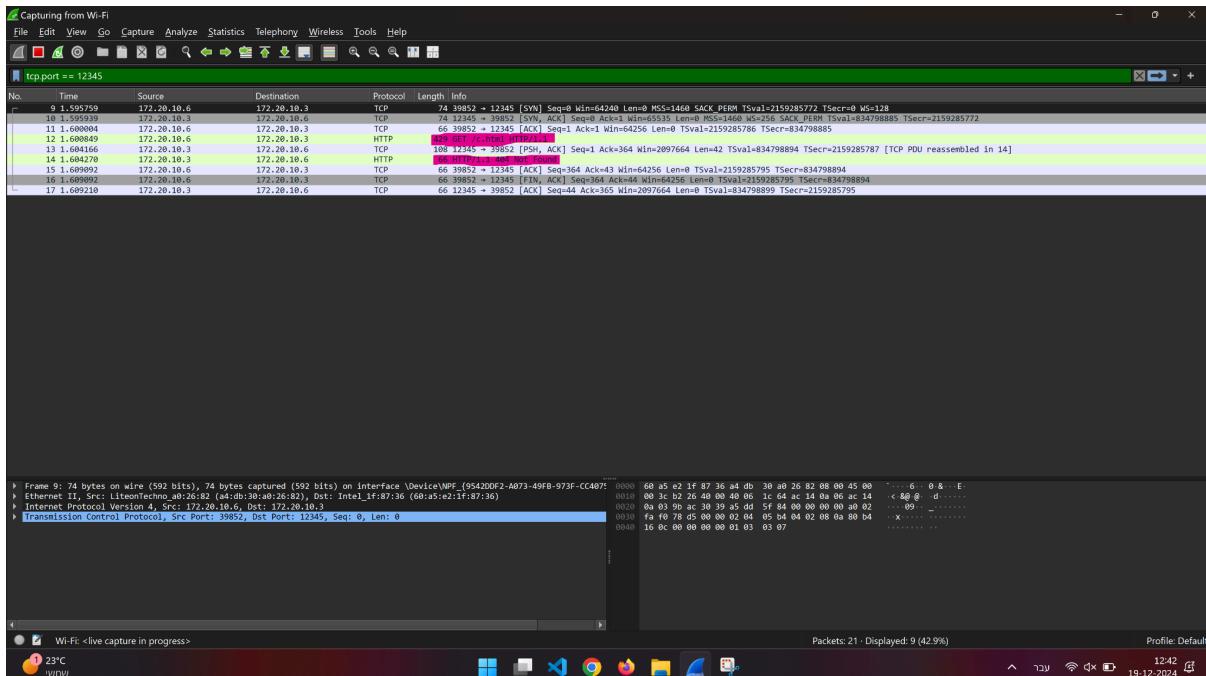


משיכים באופן אופני, תמונה מסת'ימת לשולחן, cutת מבקשים חדשה וכו'. נשים לב שאין יצירה של

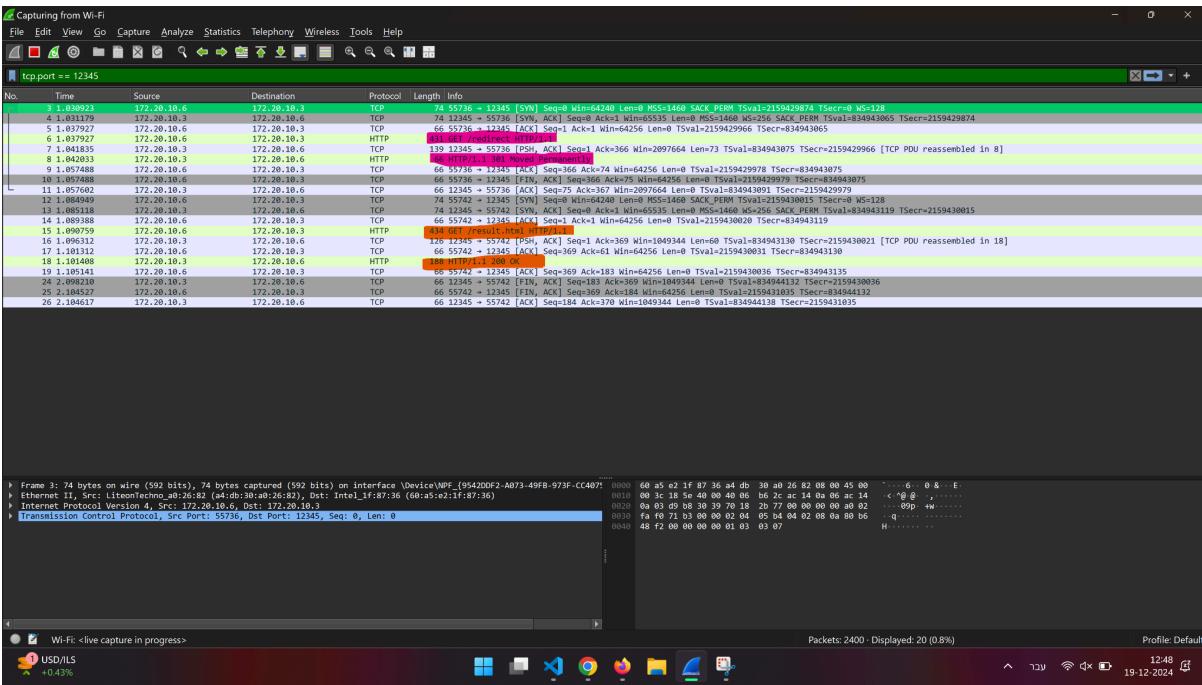
5. מוקדים וחיבורים חדשים כאמור לעיל. חלק מהמחיבורים נסגרים תוך כדי אבל ממשיכים בגדוֹל.

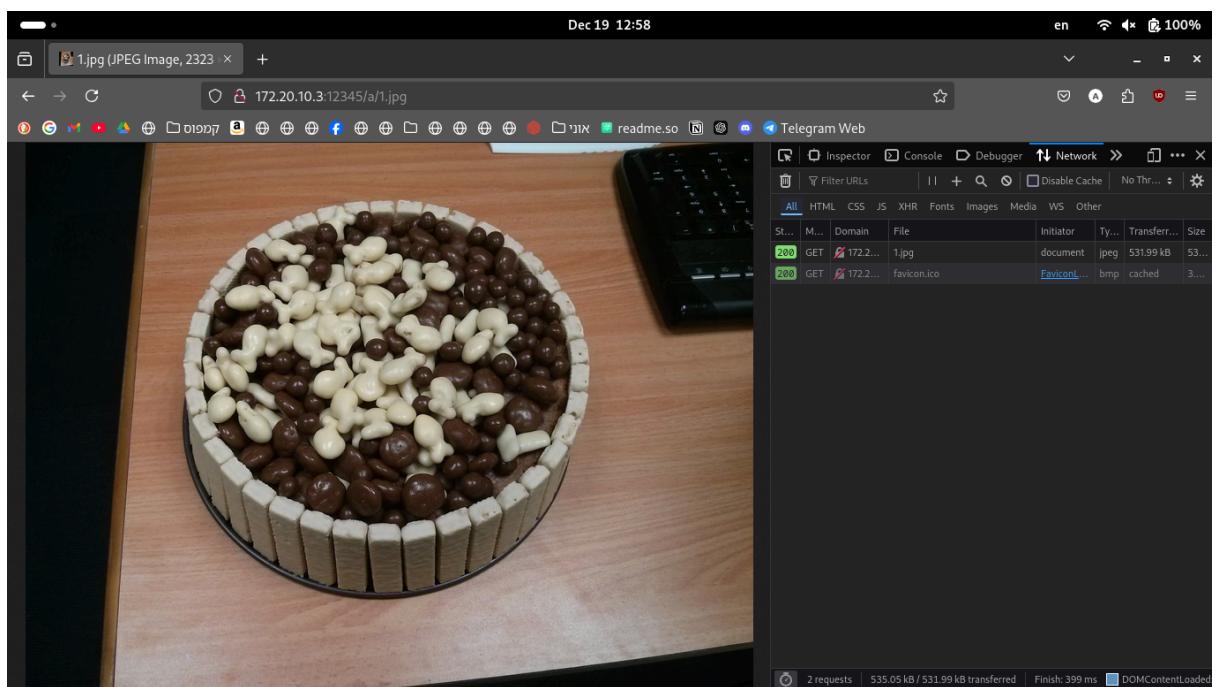
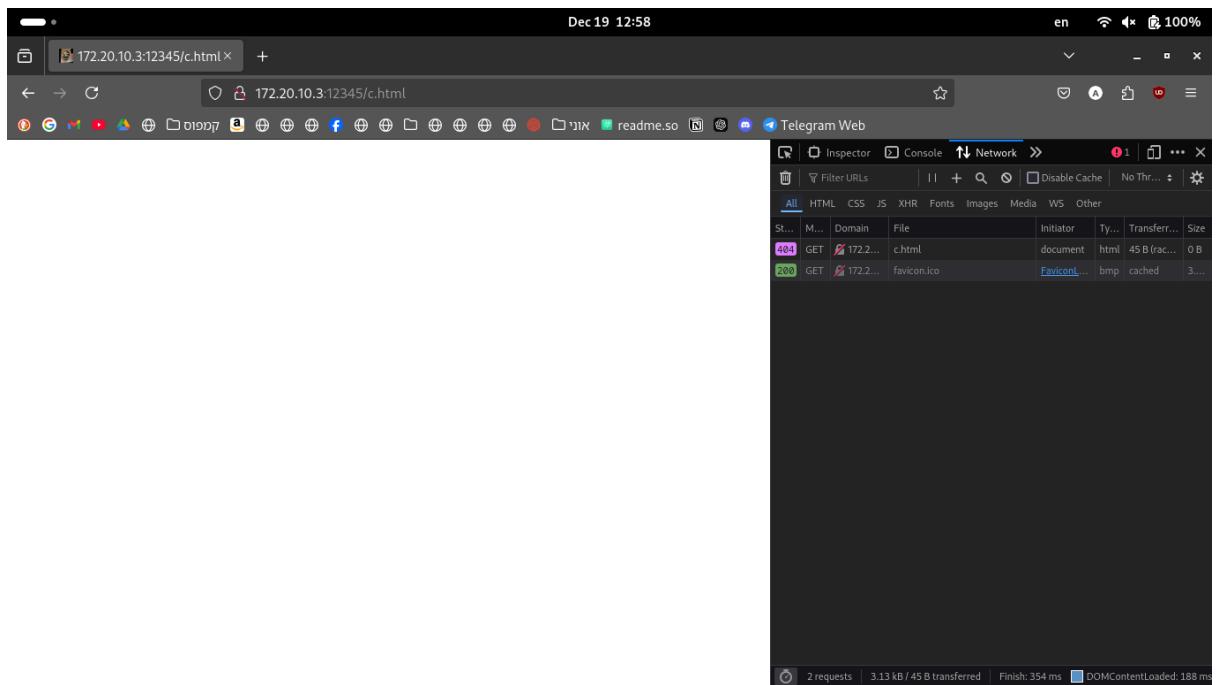


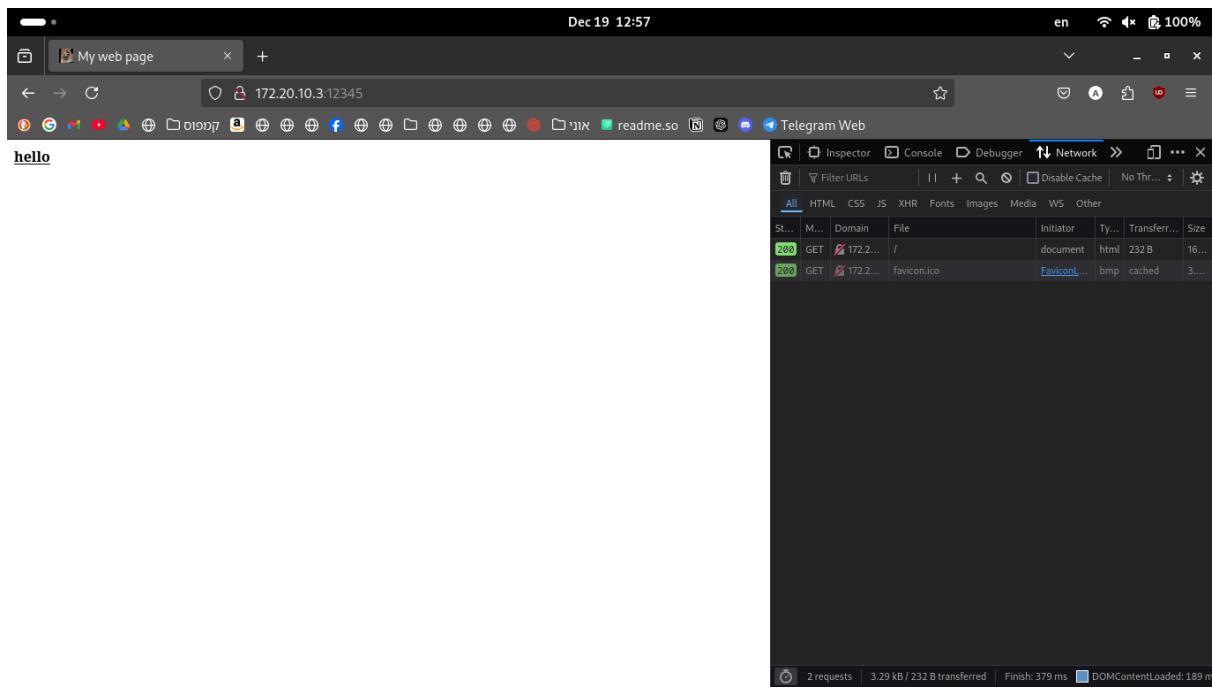
4. הלוח ניגש אל <http://172.20.10.3:12345/c.html>. סנכרון כרגע, מבקשים את הדף ב프וטוקול HTTP, ומתקבלים 404 ויכוון שהוא אינו קיים?



5. הלוח ניגש אל <http://172.20.10.3:12345/redirect>, סנכרון וכו', מבקש HTTP redirect בשבכט redirect, מוחזר 301, redirect result.html, OK 200, result.html ומכבר נסגר כרגע ומופנה לבסוף אל result.html בכתובם.







The screenshot shows a browser window with the title "My redirected web page". The address bar displays the URL `172.20.10.3:12345/result.html`. The page content is a simple "redirect" message. The browser's developer tools Network tab is open, showing the following requests:

St...	M...	Domain	File	Initiator	Ty...	Transfer...	Size
301	GET	172.2...	redirect	document	html	cached	12...
200	GET	172.2...	result.html	document	html	186 B	12...
200	GET	172.2...	favicon.ico	Favicon...	bmp	cached	3...

At the bottom of the browser, there are several tabs: Home, Google, YouTube, etc., and a Telegram Web tab.