

מטלה מס' 2 – אילאי בן יהושע ונועם ליבוביץ

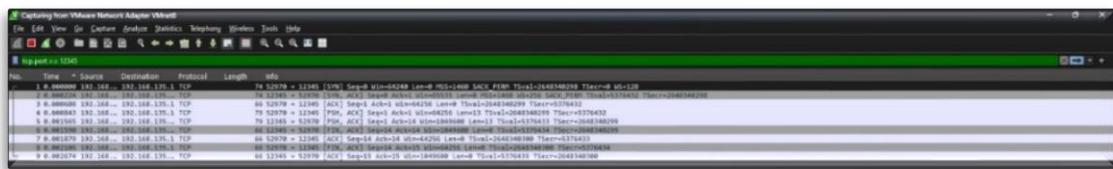
ראשית נתאים ונשנה את הקוד כך שישלוח את שמותינו לשרת, באופן הבא –

```
tcp_server.py > ...
1 import socket
2 server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
3 server.bind(('', 12345))
4 server.listen(5)
5
6 while True:
7     client_socket, client_address = server.accept()
8     print('Connection from: ', client_address)
9     data = client_socket.recv(100)
10    print('Received: ', data)
11    client_socket.send(data.upper())
12    client_socket.close()
13    print('Client disconnected')
```

```
tcp_client.py > ...
1 import socket
2 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
3 s.connect(('192.168.135.1', 12345))
4 s.send(b'Noam and Elay')
5 data = s.recv(100)
6 print("Server sent: ", data)
7 s.close()
```

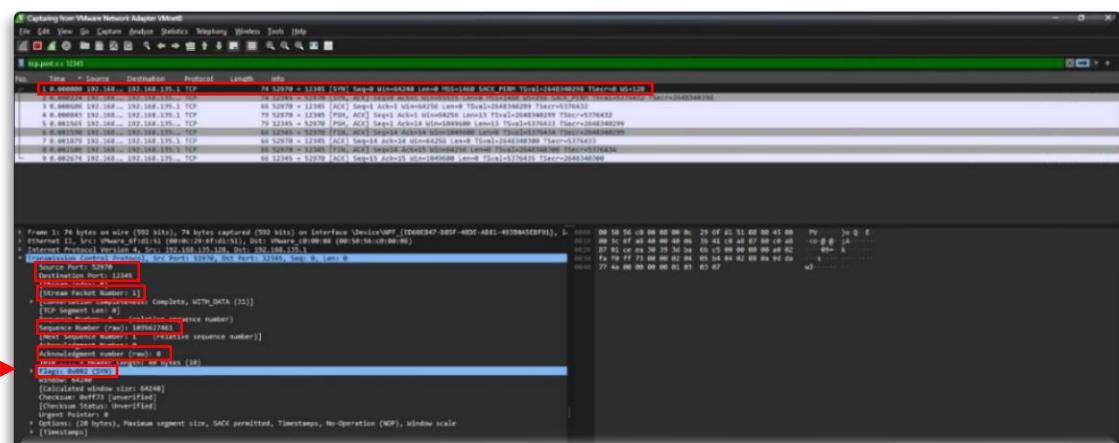
cut נרץ את הקודים המעודכנים הנ"ל, נסניף את המידע באמצעות "כרייש-הכבל" (בלוע – wireshark, מעתה והילך כאשר נכתב 'כרייש' בקיצור נתיחס למינוח זהה), וננתח את מה שקרה פה בהתאם לעקרונות TCP שראינו בהרצאה ובתרגול.

זה מה שמוצג לנו בכרייש לאחר הריצת הקודים -



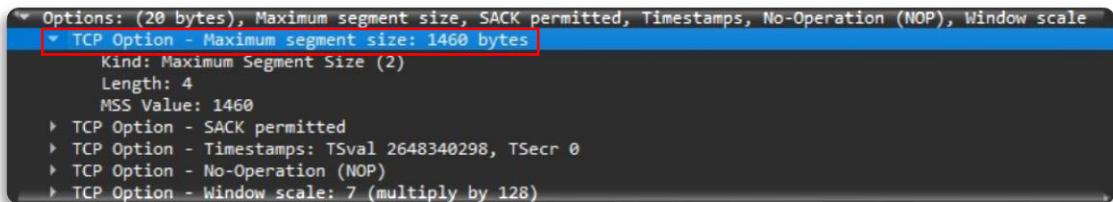
מעבר על כל חבילה וחבילה על מנת לתאר במדויק כל שלב בתקשורת.

חבילה מס' 1 –



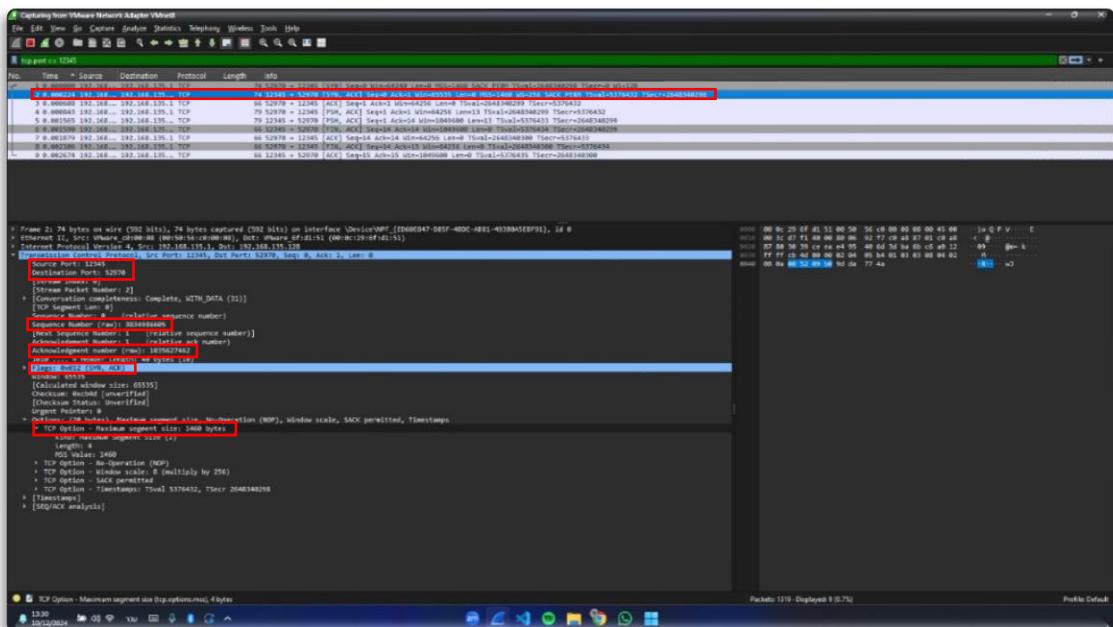
אנו רואים פניה שמתבצעת לPORT 12345 – (אשר אותו נתנו ידנית לשרת), אשר מגיעה מפורט מקור 52979 אשר ניתן דינאמית ללקוח ע"י מה (כיון שלא צמדנו אותו לפורט ספציפי). הפניה היא מסוג chs (ניתן לראות את הדגל הדלוק), ככלומר הלקוח מבקש להסתنصرן עם השרת וזויה תפקידה של החבילה הנ"ל עם דגל החsus, אשר מהוות את השלב הראשון בטקס 'חיצת הידיים המשולשת'. בנוסף לכך אנו רואים את ערך האופטט הגולמי (= raw) של seqnum (seqnum) אשר ניתן לראות בו ערך 0, אשר מציין את תחילת התקשורת – שבחור הלקוח, כאשר ניתן לראות שהוא ack שלו, כיוון שהseqnum שלו, זה eqnum של השרת (אשר טרם נבחר.). כמו כן השהה 0 שחררי אין דאטא בהודעת chs.

דבר מעניין נוסף ניתן לראות כאשר נפתח את הoptions –



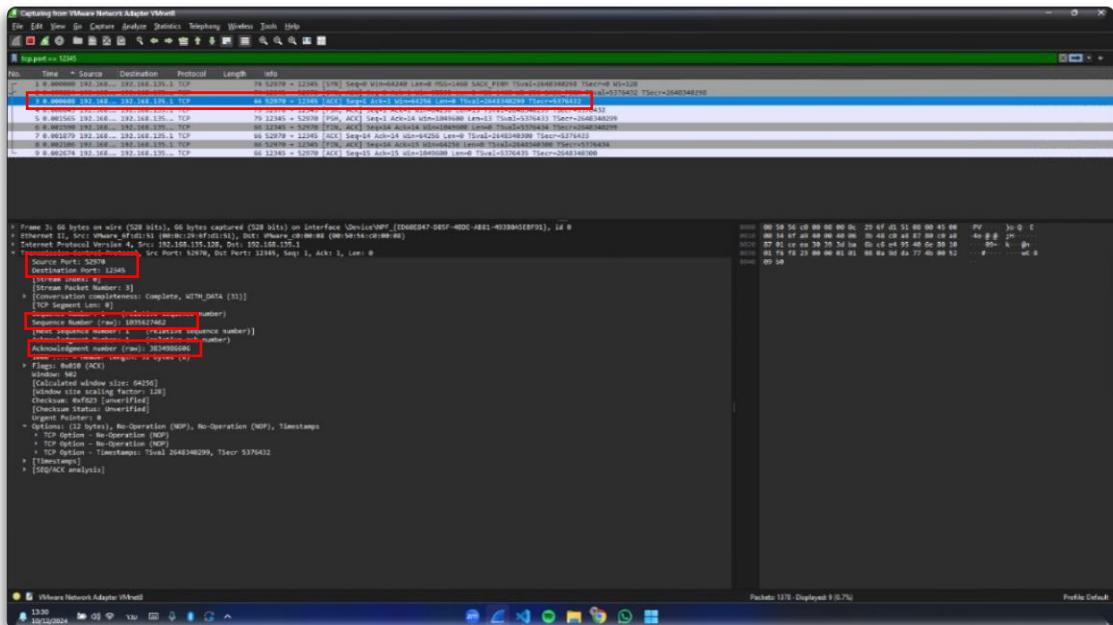
ובו, הלקוח מצין בפני השירות – "רק שתווד, המס שלוי הוא 1460b".

חvíלה מס' 2 –



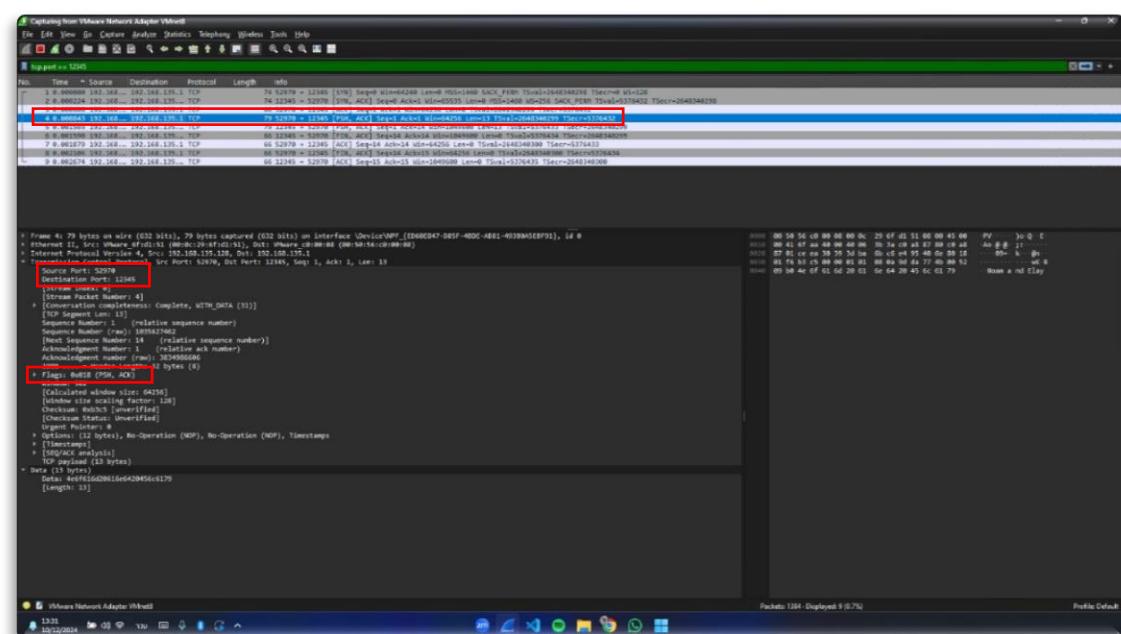
אנו רואים פניה שמתבצעת הפעם בכיוון ההפוך מ-PORT 12345 - PORT 52970 (השירות). הפניה (או יותר נכון 'tagובה') מצד השירות הינה מסוג syn, ack, syn, ack עם הלקוח איתן, פה ניתן לראות את שני הדגלים דלוקים), כולם **השירות ראשית מכיר בסנסרנו של הלקוח איתן**, אך בנוסך הוא מבקש להסתنصرנו גם הוא עם הלקוק. זהה למשזה תפקידה של החvíלה הנ"ל עם דגלי ack, syn, אשר מהוות את השלב השני בטקס 'לחיצת הידיים המשולשת'. ראוי לציין כי בשלב הנ"ל עשוי להתבצע לעתים 2 חבילות שונות (1 ל-ack ואחת לח-syn) אך כמו שאנו רואים כאן, אלו יכולות להישלח גם כן ייחודי בחvíילה אחת. בנוסף לכך אנו רואים את ערך האופסוט הגלומי (= raw) של seq (raw) של 3834986605, ואילו נשים לב שערך ack שלו יהיה התקשורת תחילית מאופסוט – '1305627462', שזה 1 יותר מאשר הסהיה seq של הלקוק בגל ה-bit-bit-phantom. גם כאן הח-raw הוא 0 שחרי אנחנו עדין לא בשלב של העברת נתונים, אך מאוד מתקרבים לשם. אם נפתח את ה-opts option נשים לב שגם השירות מצין בפני הלקוק מהו המס שלוי – 1460b, כדי שזה דעת את הנתון כאשר הוא שולח אליו הודעות.

חvíלה מס' 3 –



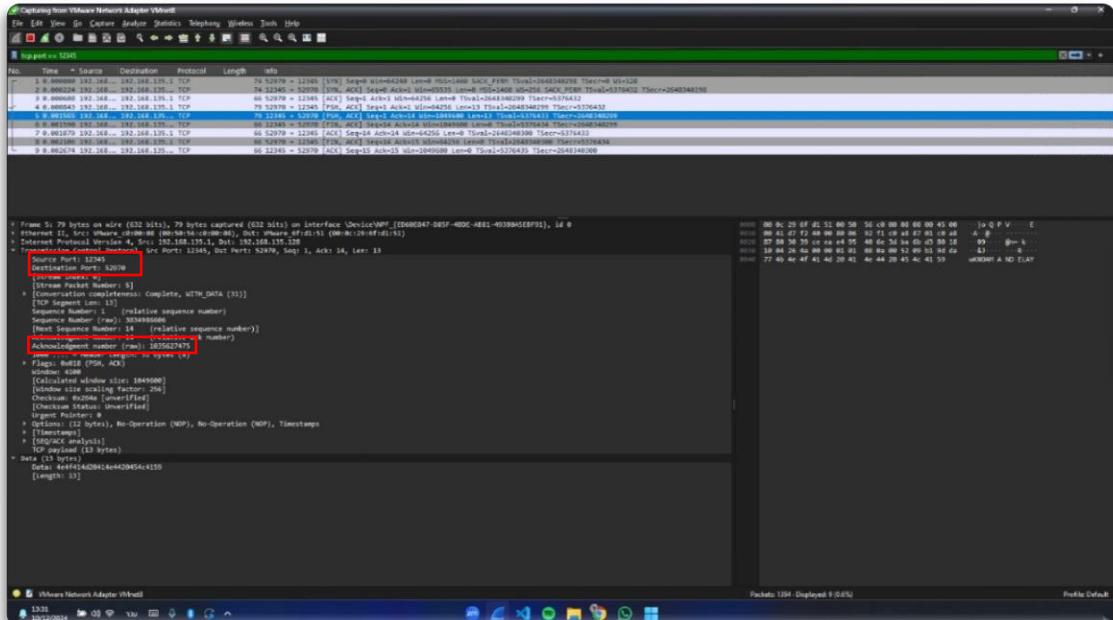
כעת ניתן לראות פניה נוספת מהלך 52970 (הלקוח), אל השרת. הפניה מצד הרשת הינה מסוג ack (שוב הדגל דлок ולמעשה מעתה והוא יהיה דлок עד סוף התקשרות, אז נפסיק לציין זאת...), דהיינו הלקוח גם הוא מכיר בהכרת הרשת בסביבון שלו, ולמעשה תפקידה של החvíלה זו מהוות את השלב השלישי והאחרון בטקס 'לחיצת הידיים המשולשת', שכן שני הצדדים הסתנכרנו אחד עם השני, וכל אחד מהם הכיר בסביבון המשותף, ומכאן והואיל הקשר שරיר ונitin להעביר דרכו הودעות. שוב אנו רואים את ערך האופטט הגלומי (= raw) של ack שהשליח הלקוח, והוא – '063', באופן לא מפתיע זה 1 יותר מהseq שבחר הרשת (שוב אנחנו מגדילים בغال הפאנטום בית של ack). seq לעומת זאת לא השתנה שכן לא נשלח עוד נתונים, וכתוואה מכך שוב החלו הוא 0 (אך לא עוד הרבה זמן). אם נפתח את options נשים לב שההסז שלו כבר לא נשלח, שכן שני הצדדים כבר יודעים ומודיעים למוגבלות כל אחד של השני.

חvíלה מס' 4 –



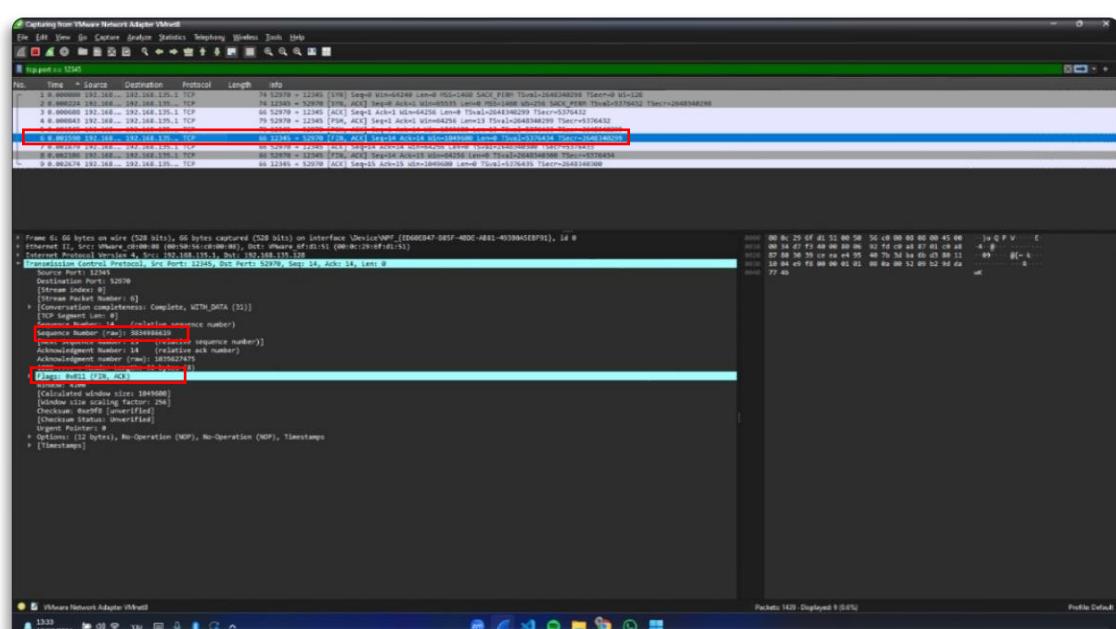
סוף כל סוף הגיענו לחלק המעניין ('פsegת הקורס' לפי חילוק מהשיטות), אנו רואים הودעה מהל��ות (לפי הפורט 52970 שראינו קודם שהוא הפורט של הלוקו), השולח 13 בתים (לפי `len`), שנחננו יודעים שהם הבטים של המחרוזת 'Noam and Elay' (כנדרש במתלה), ובນוסף לדגל `ack` (שאמרנו שמעתה והילר תמיד יהיה דילוק), גם כן דילוק דגל ההשיק שכאיין אומר להעביר בדחיפות לאפליקציה את המידע שהגיע. נשים לב שעכשיו יש לנו גם את שכבת `data` בהודעה (הבטים שלחנו מואפן מוקודד).

חvíלה מס' 5



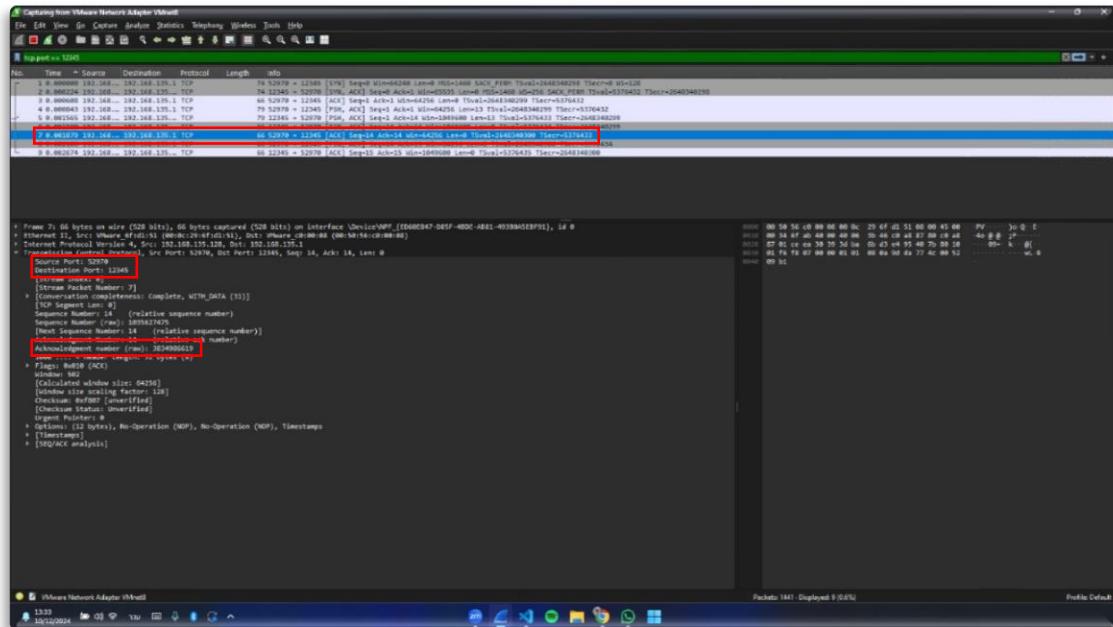
icut נראה את תגובת השירות להודעה שקיבל מהלוקו בחvíלה הקודמת שסקרנו. הרשות הראשית **מעדכן** את המספר בשדה `ack` להיות 13 יותר מאשר - '1035627475' שכן עכשו הוא אומר ללוקו – "שמעו?! קיבלתיכי כבר עד בית מס' 1035627475 תן לי' ממן ולהאה.". אמנם `seq` לא משתנה כיון שהוא טרם שלח נתונים עצמו ללוקו, אך כעת הוא מכרף להודעת `ack` את התוכן שקיבל בזקוקס (ושוב לנו לא מופתעים שההט `len` הוא 13).

חvíלה מס' 6



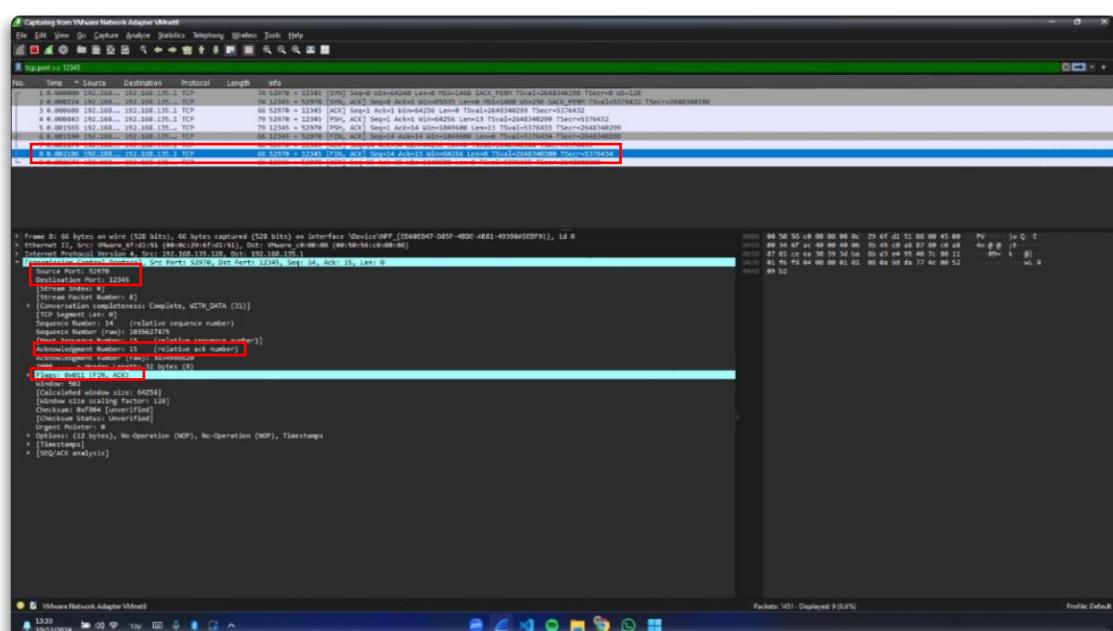
מיד לאחר שהשרת שולח את הודהה **ack** יחד עם התוכן בזקוקס, הוא שולח הודהה **fin** (ניתן לראות דגל דולק), שכן הוא עשה את שלו, ותפקיד ההודהה זה להודיע על כך שהוא מעוניין לסיים את מערכת היחסים עם הלקוח. נשים לב שערך הסeq של השרת גדל ב-13 גם הוא ערך – '3834986619'.

חvíלה מס' 7



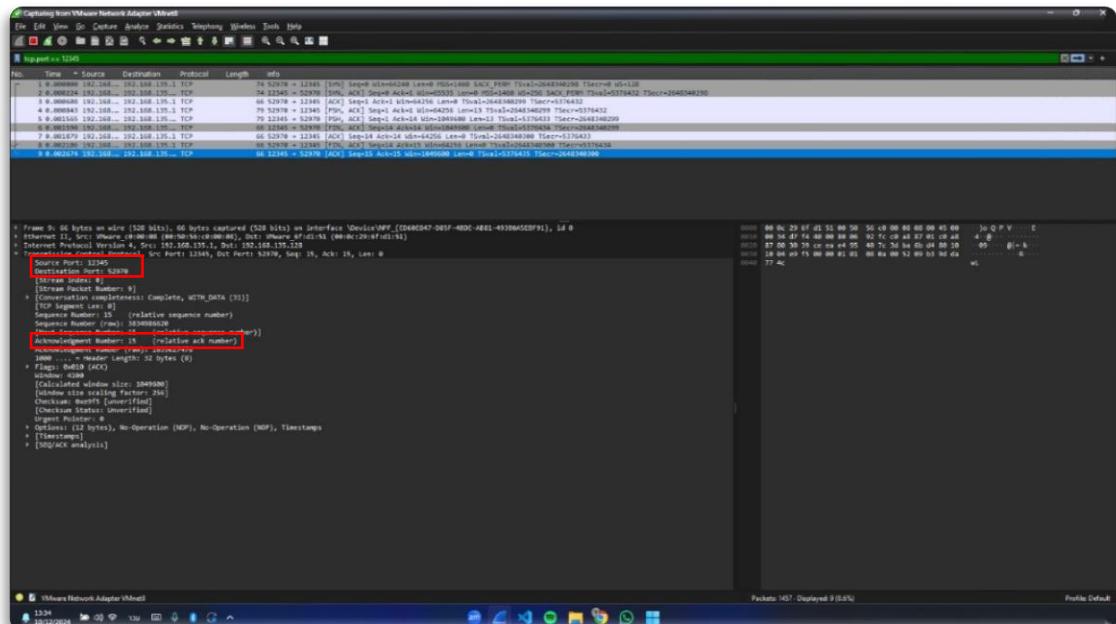
לקוח מקבל את הדטא חזרה מהשרת בזקוקס, בהתאם הוא מגדיל את ערך **acked** ב-13 בתים נוספים לערך – '3834986619'. כמו כן נשים לב שגם הערך **seq** המופיע בהודהה גדול ב-13 מהפעם האחרון בה ראיינו חvíלה נשלחת מהלקוח, וכעת הוא עומד על – '1035627475' (שוקינג זה הערך של השרת..).

חvíלה מס' 8



הלקוח כעת שלוח הודהה ack לשרת, ולמעשה מעדכן שהוא קיבל את ההודהה שלו ברגע לסיום התקשורת עימיו. כיוון שהוא הבין שנגמר הקשר, הוא מסיים אותו גם מבחינתו ומצרף זאת להודהה (דגל החfin דלוק בה). דהיינו **תפקיד הודהה לומר לשרת שהסיום הינו הדדי**. מעבר לזה אין מידע מעניין למעט העובדה שערך ack של הלוקוח גדול ב-1 כתוצאה מהפאנטום ביט של הודהה החfin שמכורמת רק עכשו.

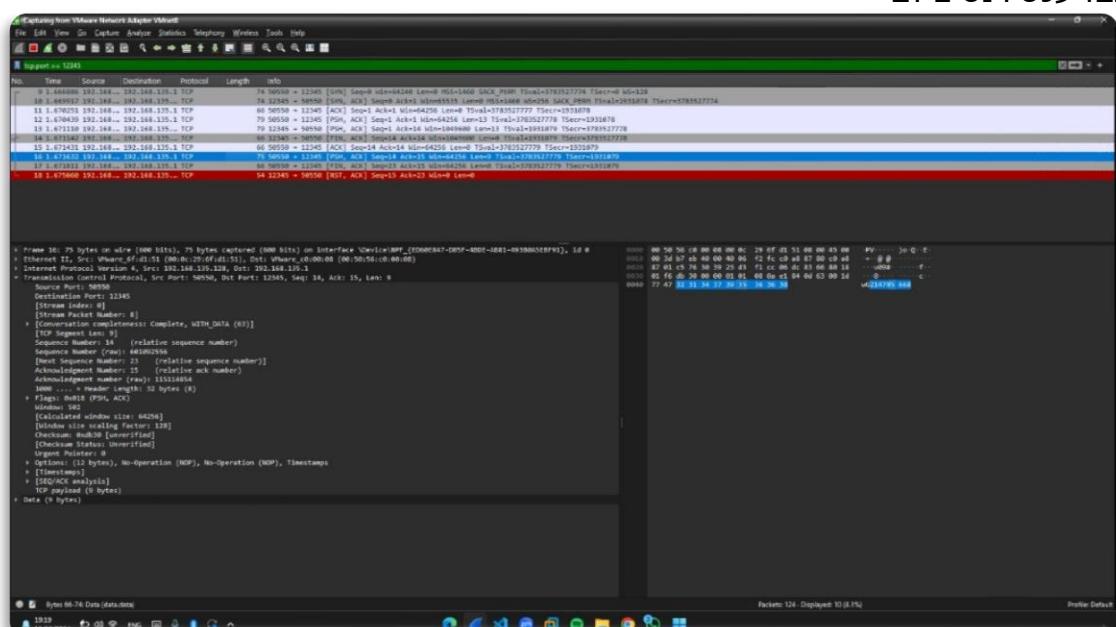
חבילה מס' 9 –



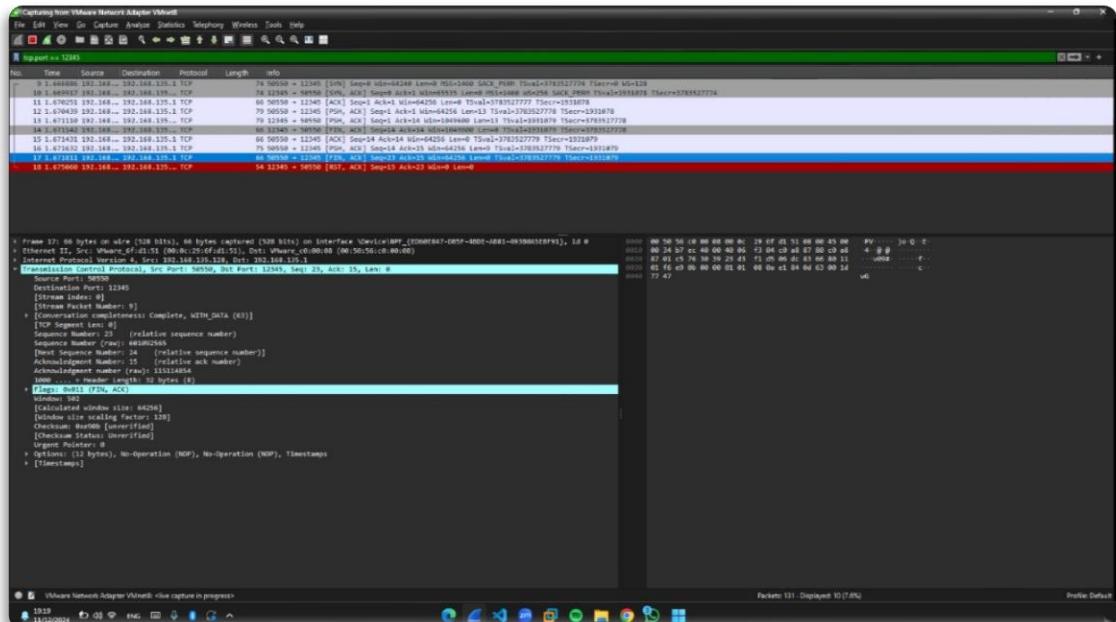
בחבילה الأخيرة (😊) שלנו אנו רואים את השרת מכיר בסיום התקשורת מצד הלוקוח גם הוא בהודעת ack, ובאופן פרקטני **תפקיד הודהה הינו לסייע לחלוטין בהחלה את ערוץ התקשרות שהייתה בינהם**. נשים לב שרגע לפני שהוא אומר "ב'י'", הוא מגדיל גם הוא את ערך ack כתוצאה מהפאנטום ביט של החfin שהגיע מהлокוח, סה"כ ערך ack של השרת בסיום התקשרות עומד על – '1035627476'.

סימנו לנתח את התקשרות בחלק הראשון. נסיף כעת את שליחת ה-tz. של אילאי לאחר שהשרת משיב את תשובתו ונראה את השינויים.

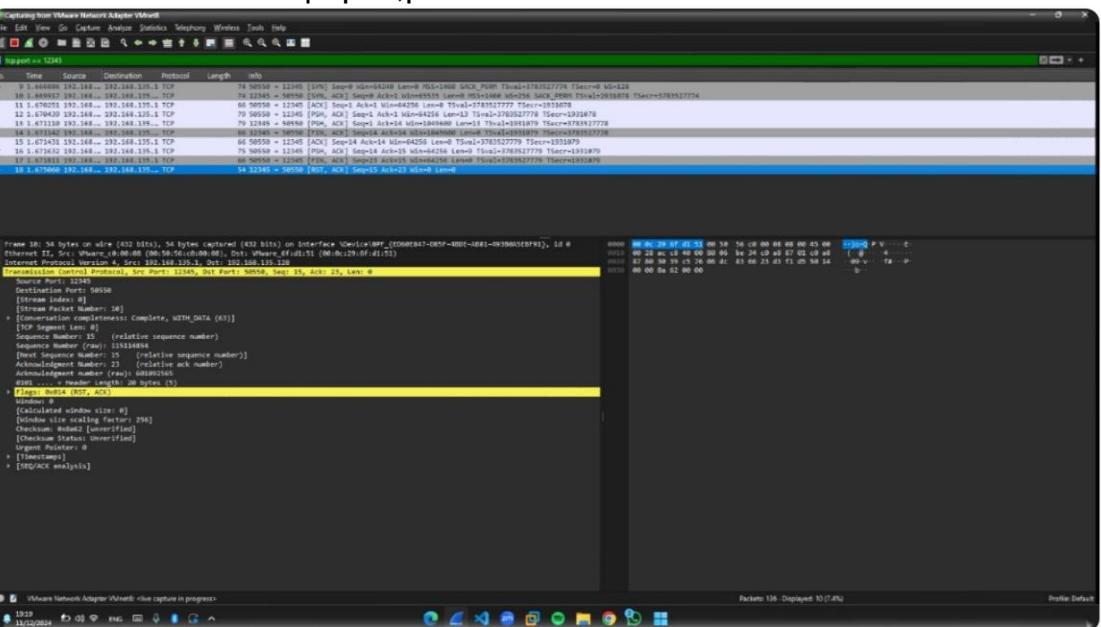
כל ה הודעות בהתחלה תתנהגנה באופן זהה (לכן נחשוך ונתחילה רק מהמקום בו השינוי קורה), אבל עכשו נשים לב –



לאחר שהלkipה החזיר ack על החזיר fin של השרת (פאנטום בית מגדל ב-1 את ה-edp), במקום לשלוח גם הוא fin (כמו שקרה קודם), הוא מנסה לשלוח הודעה נוספת ולבנות לה fin לאפליקציה (הלווא היא הת.z.), ורק לאחר שההודעה הzo נשלחת, הוא שולח גם fin – מצדן –



הseq שלו גדול ב-9, והוא מסכים עכשו לסיים את התקשרות. אלא שבגלל שהשרת לא ידע שעליו לצפות לחבילה נוספת הוא כבר סיים וסגר את העניין, ולכן ממנה הودעת – rst –



כאיו רוצה לומר – "חדש אח שלי", חדש. מה שהוא אני כבר לא שם, ואין מי שישמע ויקבל את מה שיש לך לשלוח. אם אתה רוצה בוא נתחיל מההתחלת, ומשם נשתמע להאה...".

ובכן סיימנו לנתח גם את המקירה בו הودעה נשלחת לאחר שהשרת עושה קלוז קודם לרגע בו היא מגיעה.

סעיף 2:

כעת, ננתח בקצרה את הקודים של הגרסאות השונות ואת התעבורה בהן:

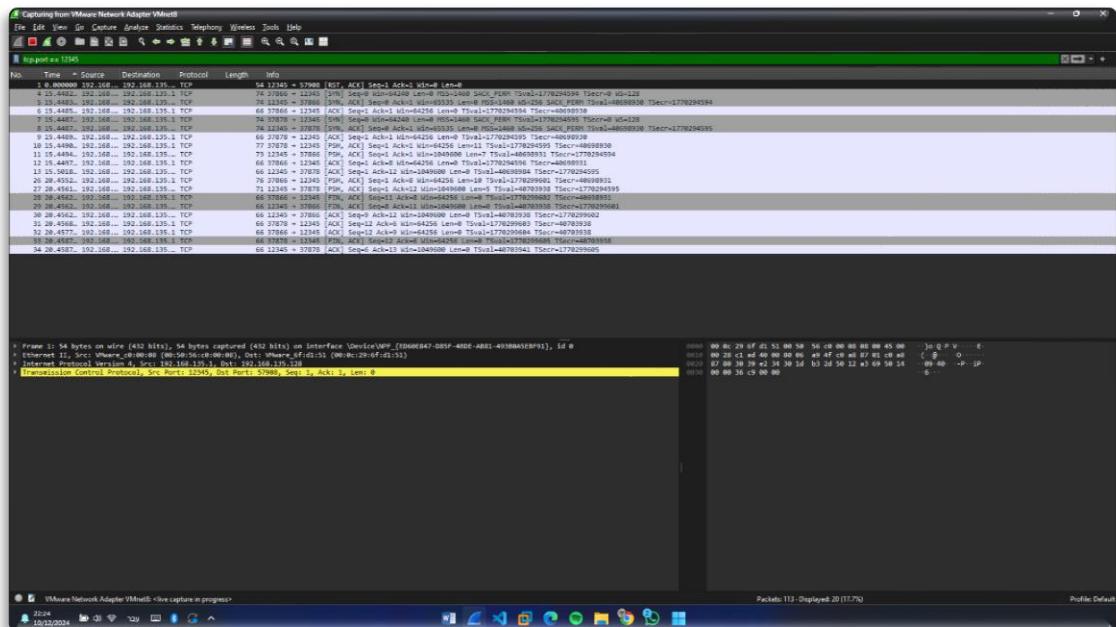
גרסת 1:

```

server.py
versions > versions 2 > v1 > server.py > ...
1 import socket,sys
2
3 TCP_IP = '0.0.0.0'
4 TCP_PORT = int(sys.argv[1])
5 BUFFER_SIZE = 1024
6
7 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8 s.bind((TCP_IP, TCP_PORT))
9 s.listen(0)
10
11 while True:
12     conn, addr = s.accept()
13     print('New connection from:', addr)
14
15     conn1, addr1 = s.accept()
16     print('New connection from:', addr1)
17
18     data = conn1.recv(BUFFER_SIZE)
19     print("received:", data)
20     conn.send(data[0:7])
21
22     data = conn.recv(BUFFER_SIZE)
23     print("received:", data)
24     conn1.send(data[0:5])
25
26     conn.close()
27     conn.close()

client.py
versions > versions 2 > v1 > client.py > ...
1 import socket,sys
2 from time import sleep
3
4 TCP_IP = sys.argv[1]
5 TCP_PORT = int(sys.argv[2])
6 BUFFER_SIZE = 1024
7 MESSAGE = b'Foo, World!'
8 MESSAGE1 = b'bar, Hello'
9
10 s1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
11 s1.connect((TCP_IP, TCP_PORT))
12
13 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
14 s.connect((TCP_IP, TCP_PORT))
15 s.send(MESSAGE)
16
17 sleep(5)
18
19 s1.send(MESSAGE1)
20 data = s1.recv(BUFFER_SIZE)
21 s1.close()
22
23 data = s.recv(BUFFER_SIZE)
24 s.close()

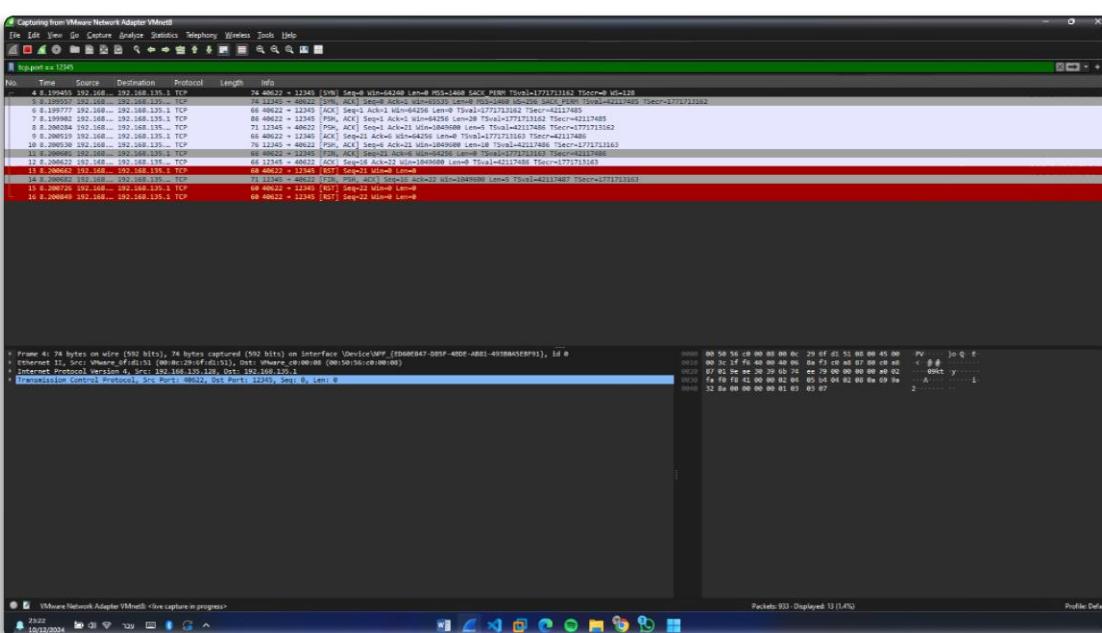
```



הלקוח מקבל את ה-IP והפורט מהארוגמנטים לתוכנית, לאחר מכן הוא מתחבר בשני חיבורים בפרדימט ביעזרת שני סוקיטים נפרדימט לשרת (שורות 4 עד 9). הלקוח תחילת שולח הודעה דרך החיבור השני, מחרכה 5 שניות ושלח הודעה אחרית דרך החיבור הראשון. לבסוף, הלקוח מקבל את שתי ההודעות מהסתוקט בסדר הפוך לסדר בו התבצעה הש寥חה, ולאחר מכן סוגר את החיבורים. צד שני, הרשת מזין לכל החיבורים כיוון שהוא בחר בכתובת 0.0.0.0 ומתקבל פורט קלט. הוא מרים את השרת לאוואר כאשר הוא מוכן שייהי ברגע נתון לכל היותר חיבור 1 בהמתנה (לייטן שווה 0). הוא עושה accept לשני חיבורים (מדפיס מי המ), מקוצר את ההודעה השנייה ל-7 תווים הראשונים שנקלטו ושלח אותה לחיבור הראשון, ואת ההודעה הראשונה ל-5 תווים הראשונים שנקלטו ושלח אותה לחיבור השני. לבסוף, יוזם ניתוק בקעם החיבור הראשון שהוא פתח.

נשים לב שבפועל בכרייש קרה משהו מעניין. לאחר האתחול עד שורה 9, נשלחת ההודעה מהסוקט שנוצר שני בלקוח, והשרת לפניו שהוא מחזיר עליה ack לאוטו סוקט שני, קודם שלוח את ההודעה המקוצרת (7 בתים), לחיבור הראשוני (שורה 11). יתרה מזאת, הוא מקבל מהחיבור הראשוני ack על ההודעה המקוצרת (שורה 12), ורק אז מחזיר ack על עצם זה שהוא קיבל את ההודעה המקורי מהחיבור השני. לאחר מכן השרת מקבל את ההודעה שהוא קיבל את ההודעה המקורי מהחיבור השני. לאחר מכן השרת משלוח fin מהחיבור השני רק עכשו על הצד השני, ועל הדרך נוספת של ack על החום עם הפאנטום בית. החיבור השני מחזיר ack רק עכשו על ההודעה שהוא קיבל (5 הבטים), החיבור הראשוני מחזיר ack על הודעות החום שקיבל מהשרת (שוב פאנטום בית), ולבסוף השרת מקבל fin מהחיבור השני ומשביב על כך בact (הוא לא שולח לו fin שכן בקוד החום ששולח השרת הינו עבור החיבור הראשוני, שכבר נסגר מיזמתו קודם לכך..).

גرسה 2:



```

server.py > ...
1 import socket,sys
2
3 TCP_IP = '0.0.0.0'
4 TCP_PORT = int(sys.argv[1])
5 BUFFER_SIZE = 5
6
7 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8 s.bind((TCP_IP, TCP_PORT))
9 s.listen(1)
10
11 while True:
12     conn, addr = s.accept()
13     print('New connection from:', addr)
14     while True:
15         data = conn.recv(BUFFER_SIZE)
16         if not data: break
17         print("received:", data)
18         conn.send(data.upper())
19     conn.close()
20
21
client.py > ...
1 import socket,sys
2
3 TCP_IP = sys.argv[1]
4 TCP_PORT = int(sys.argv[2])
5 BUFFER_SIZE = 1024
6 MESSAGE = b'World! Hello, World!'
7
8 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9 s.connect((TCP_IP, TCP_PORT))
10 s.send(MESSAGE)
11 data = s.recv(BUFFER_SIZE)
12 s.close()
13
14 print("received data:", data)
15
16

```

כמו בתוכנית הקודמת, הלקוח מקבל את IP והפורט מהארגומנטים לתוכנית, לאחר מכן הוא מתחבר לשרת ושולח לו את ההודעה המצוינת. הוא עושה recv מהשרת, ולאחר מכן מקבל את ההודעה שהגיעה חזרה מהשרת הוא מדפיס אותה. מצד שני, כמו בתוכנית הקודמת, השרת מאזור לכל החיבורים כיון שהוא בחר בכתבota 0.0.0.0 ומתקבל בקלט. הוא פותח חיבור, מקבל את ההודעה בצדדים של בתים 5 (כוגדול הבאför), עד שבתיים מפסיקים להגיע, מדפיס כל צאנק, ושולח אותו חזרה ללקוח באופן של zeros. לבסוף, הוא סגור את החיבור.

כמו התפיסה האחורונה, ניתן לראות את חיצת הידיים בין הלקוח לשרת (שורות 4-6). אך בשונה מהתפיסה האחורונה, השרת קולט את המידע ש מגיע אליו מחיבור אחד בחתיכות של 5 תווים. לכן רואים בכריש את הצאנק הראשון מגיע ומוחזר מהשרת (בזאתם כמובן), ולפניהם רואים את המשך קבלת הצאנקים בשרת, אנו רואים את ack שהלקוח החזיר על הצאנק הראשון. משום מה שרת מוחדר את 2 הצאנקים הבאים ושולח אותם יחדיו ללקוח, כאשר האחרון מוחזר עליהם ack. עם זאת, מכיוון שיש רק recv אחד בלקוח, הוא ממשיר בשלו ולאחר קבלת הצאנק הראשון הוא שולח fin כדי לסיים את ההתקשרות. השרת מקבל את fin ומוחזר עליה ack, אולם מעתה והילך שאר הצאנקים שהשרת שולח זוכים לمعנה של rst כיון שאין אף אחד בצד השני ששמע וזמן לקבל. ככל לחין שהשרת שולח – הכל מקבל rst, שכן הלקוח ניתק בצד השני...

גרסת 3:

```

server.py
1 import socket,sys
2
3 TCP_IP = '0.0.0.0'
4 TCP_PORT = int(sys.argv[1])
5 BUFFER_SIZE = 1024
6
7 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8 s.bind((TCP_IP, TCP_PORT))
9 s.listen(1)
10
11 while True:
12     conn, addr = s.accept()
13     print('New connection from:', addr)
14     while True:
15         data = conn.recv(BUFFER_SIZE)
16         if not data: break
17         print("received:", data)
18         conn.send(data.upper()*1000)
19     conn.close()

client.py > ...
1 import socket,sys
2
3 TCP_IP = sys.argv[1]
4 TCP_PORT = int(sys.argv[2])
5 BUFFER_SIZE = 1024
6 MESSAGE = b'Hello, World!'
7
8 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9 s.connect((TCP_IP, TCP_PORT))
10 s.send(MESSAGE)
11 data = s.recv(BUFFER_SIZE)
12 print("received data:", data)
13 data = s.recv(BUFFER_SIZE)
14 print("received data:", data)
15 s.close()
16
17
18
19

```

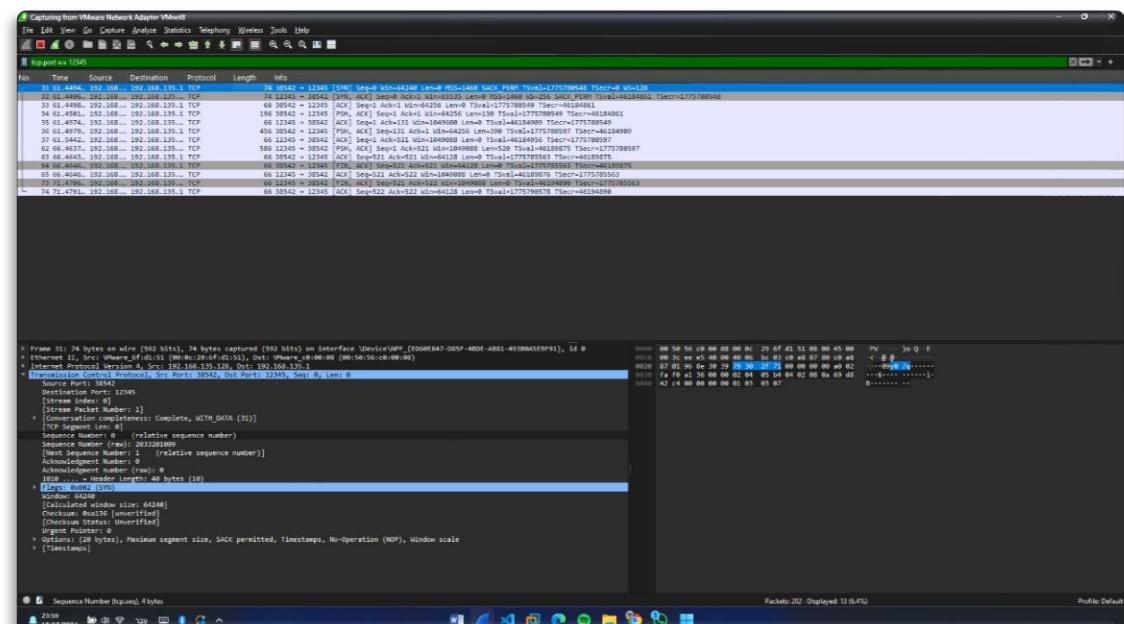
כמו בתוכנית הקודמת, הקוד לא השתנה המון – הלקוח מקבל את התוכנית, לאחר מכן הוא מתחבר לשרת ושולח לו הודעה, אך בשונה מהתוכנית הקודמת, לאחר שקיבל את הודעה חזירה מהשרת הוא מושך ומדפיס את 1024 בתים הראשונים ולאחר מכן שוב הוא מנסה למשוך עוד 1024 בתים. כמו כן בצד שני, כמו בתוכנית הקודמת, השרת מאיין לכל החיבורים כיוון שהוא בחר בכטובת 0.0.0.0 ומקבל פורט נקלט. הוא פותח חיבור, מקבל את 1024 בתים הראשונים ושולח אותם חזירה ללקוח באoitיות גדולות בהכפלה של 1000 פעמים הלקוח. לבסוף, סוגר את החיבור.

כמו בתפיסה האחורונה, ניתן לראות את לחיצת הידיים בין הלקוח לשרת. מיד לאחר מכן את 13 הבטים שהלקוח שולח, ואז השרת קולט את כל המידע שמגיע אליו, ושולח אותו באoitיות גדולות כפול 1000 פעמים, لكن יש הרבה שליחות של 12345 (השרת) ל-44550 (הלקוח). השרת שולח את החבילה הראשונה, ולאחר מכן עוד מלא חבילות. הלקוח מאשר את קבלת החבילה הראשונה (1024 הבטים הראשונים) וכן את השניה (אלו שבאים אחריו), מחזיר ack לאחר מכן מואחד על 13000 הבטים שהגיעו, וסגור את החיבור. אבל יש עוד חבילות בבאפר שלא נקבעו ע"י האפליקציה כאשר היא נסגרת, לכן נשלחת הודעה stz לשרת שידע שהלקוח ניתק מבלי לקרוא עד הסוף.

גרסה 4:

```
server.py > ...
1 import socket,sys,time
2
3 TCP_IP = '0.0.0.0'
4 TCP_PORT = int(sys.argv[1])
5 BUFFER_SIZE = 1024
6
7 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8 s.bind((TCP_IP, TCP_PORT))
9 s.listen(1)
10
11 while True:
12     conn, addr = s.accept()
13     print('New connection from:', addr)
14     while True:
15         time.sleep(5)
16         data = conn.recv(BUFFER_SIZE)
17         if not data: break
18         print("received:", data)
19         conn.send(data.upper())
20     conn.close()
```

```
client.py > ...
1 import socket,sys
2
3 TCP_IP = sys.argv[1]
4 TCP_PORT = int(sys.argv[2])
5 BUFFER_SIZE = 1024
6 MESSAGE = b'Hello, World!'
7
8 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9 s.connect((TCP_IP, TCP_PORT))
10 s.send(MESSAGE*10)
11 s.send(MESSAGE*10)
12 s.send(MESSAGE*10)
13 s.send(MESSAGE*10)
14 data = s.recv(BUFFER_SIZE)
15 s.close()
16
17 print("received data:", data)
```



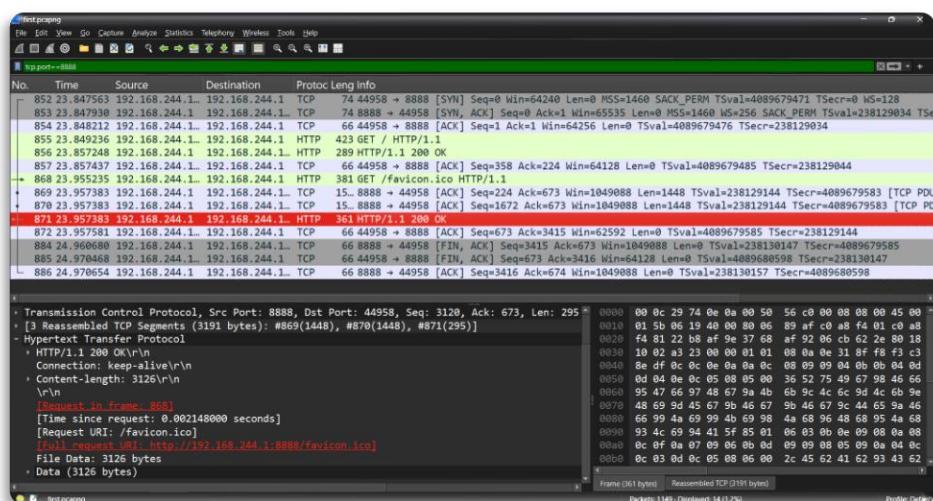
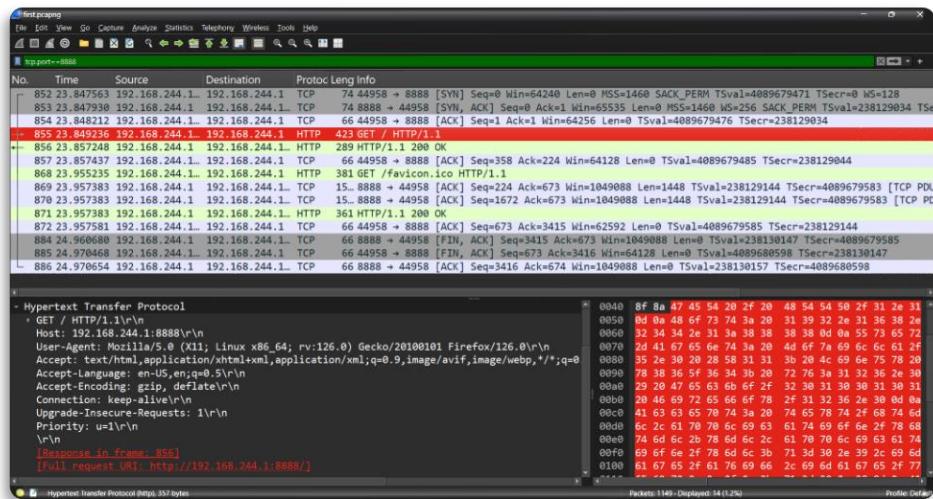
כרגיל הלוקו מקבל את ה-IP והפורט מהארגוני לתוכנית, לאחר מכן הוא מתחבר לשרת ושולח לו 4 הודעות שכל הודעה היא 10 פעמים "Hello, World", השוני הפעם מהתוכנית הקודמת, הוא שלאחר שקיבל את ההודעה חוזרת מהשרת הוא מדפיס את 1024 בתים הראשונים. בצד שני, כרגיל השירות מażין לכל החיבורים כיון שהוא בחר בכתובת 0.0.0.0 ומקבל פорт קקלט. הוא פותח חיבור מכחכה 5 שניות, מקבל את 1024 הבטים הראשונים ושולח אותם חוזרת ללקוח באוטיות גדולות. הוא חוזר על התהיליך עד שהוא מפסיק לקבל מידע מה לקוח ולבסוף סגור את החיבור.

בכרייש אנו רואים תחילת את לחיצת הידיים בין הלקוח לשרת. אך בשונה מהתפיסה האחורונה, השירות מכחכה 5 שניות ולכן התגובה שלו מתעכבות. ניתן לראות כי השילחה (של הלקוח) והתגובה (של השירות) קורית פה 2 פעמים (כנראה 3 השילוחות האחרונות אוחדו בעט שליחתן). בכל פעם השירות (שנמצא בדעתן timeout ב一封יקציה ולן לא קורא מהבאפר), מוחזיר ack שהוא קיבל את ההודעה. השירות שולח את כל ה-520 בתים בחזרה בפעם אחת (מאוחדת) באוטיות גדולות, ומתקבל ack לעלייה מהשרת. מכאן והילך טקס הפרידה הסטנדרטי של-fin-ack בין השירות ללקוח, כאשר כל אחד מעלה את הפאנטום בית, ונפרדים לשלום. כאן לא הייתה בעיה שלrst בשל העבודה שכל המידע נקרא מהבאפר לפני האפליקציה סיימה את הריצה.

חלק ב'

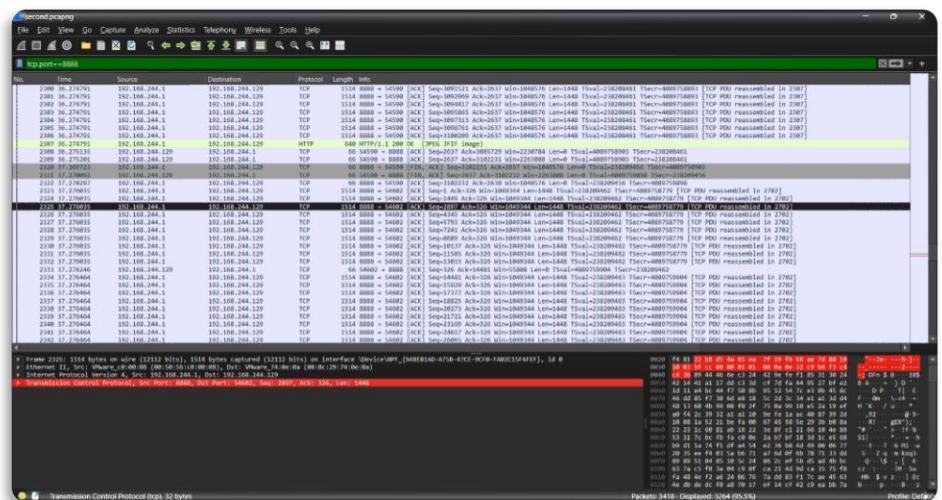
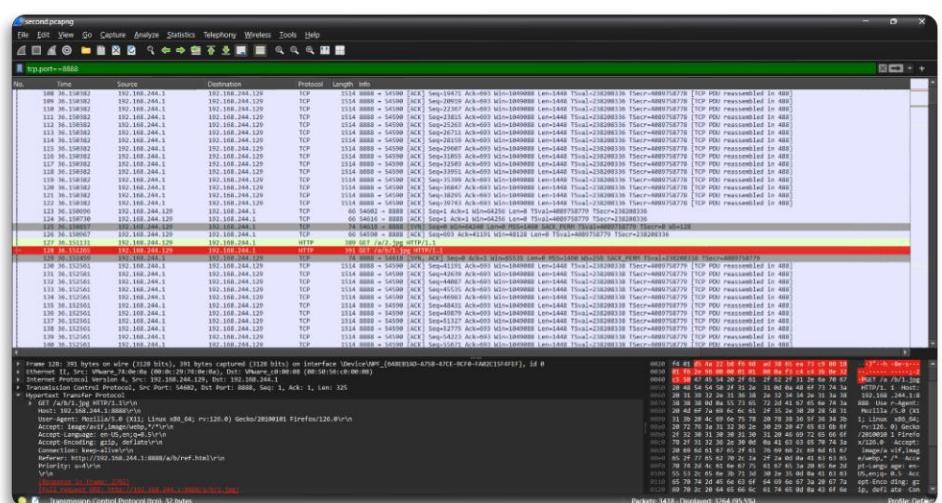
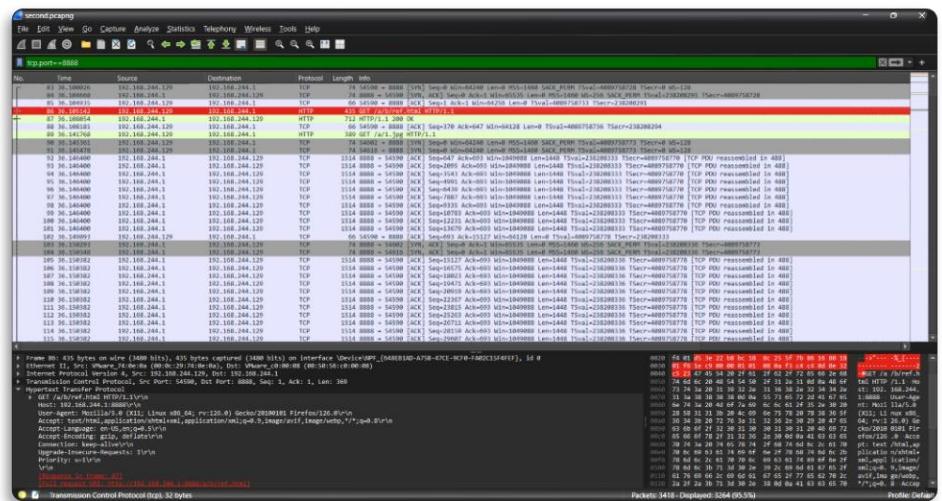
נסים חלק זה של התרגיל בהציג הכריש בזמן ריצת קוד השירות שלנו, כאשר האחרון מקבל בקשות מהדפסן (אשר רץ בVM).

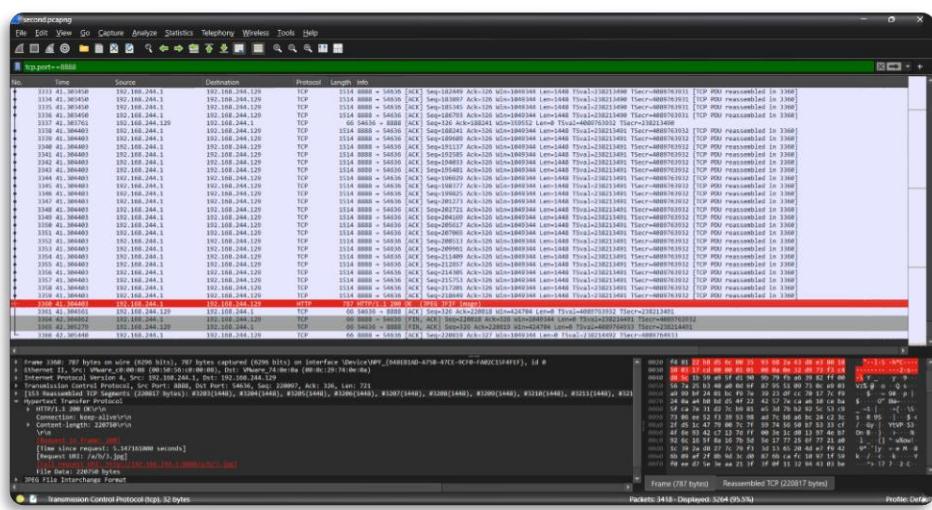
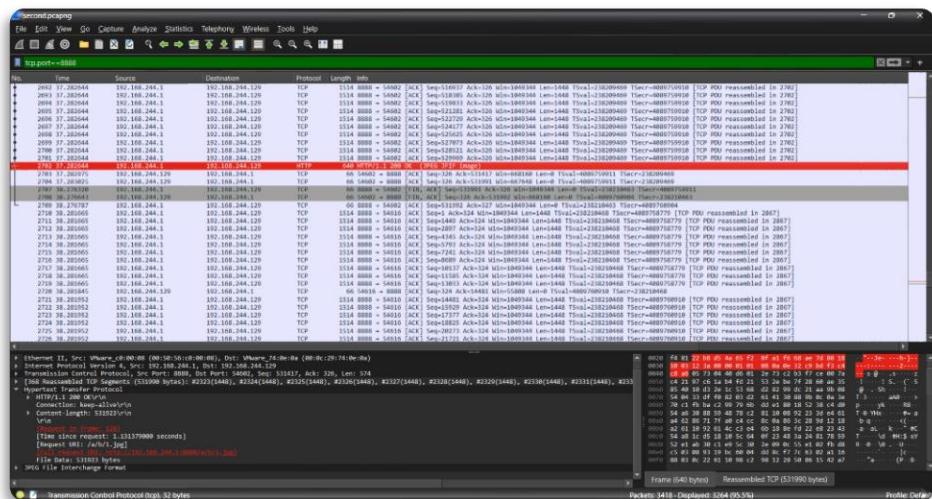
<http://192.168.244.1:8888/>



הכתובת פונה וambilת מהשרת את דף הבית, והוא מוחזיר בתמורה את – index.html. ניתן לראות בתמונה הראשונה שדף עושה מס' פניות מסווגו פורט (44958), כאשר תחילתו שלlox את הבקשה של הקובץ, ומתקבל אליו חזרה את ה ok 200, מיד לאחר שהם מחליפים ack ביןיהם אוזות מעבר הקובץ באופן תקין, הדפסן שולח בקשה get נוספת עבור הקובץ favicon.ico הלוא הוא האיקון של "הarter" שלו שאננו רואים בקצת הרכטיסיה בדף, גם על בקשה זו מתקבלת הודעה ok 200 (תמונה שנייה – רואים את הקובץ עליו הוחזר ok למטה body), והתקשרות נסגרת באופן הדדי בחו"ם משני הצדדים. ניתן לראות שנשלח המונן מידע עם הודעה get מדף, כמו סוג הדפסן, הגרסה שלו, סטטוס connection, עדיפות ועוד. מעבר לget, ההחזרה של המידע מהשרת – ok 200, אז הבקשה לאיקון אין מיוחד, ואין בקשה נוספת פורט.

<http://192.168.244.1:8888/a/b/ref.html>

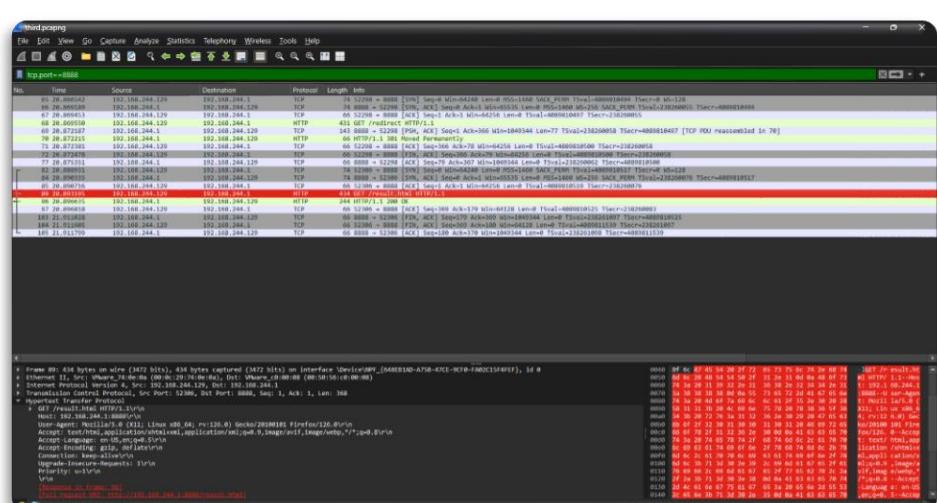
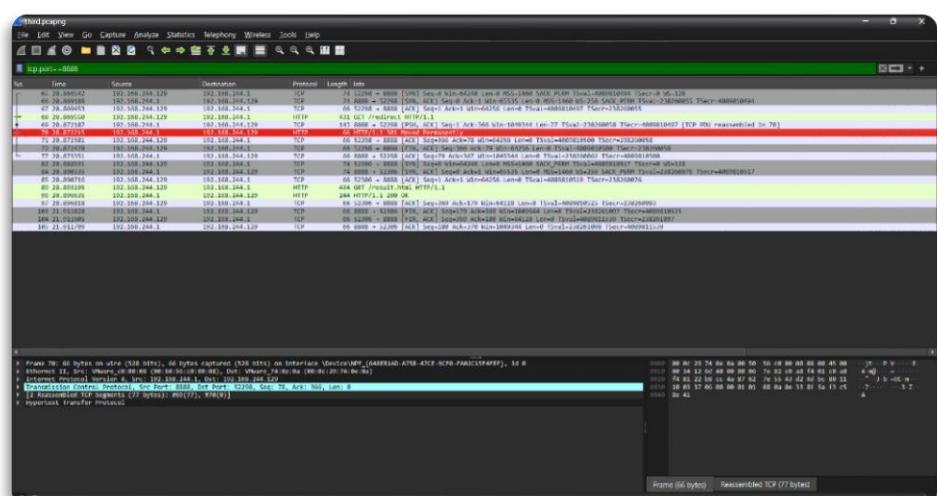
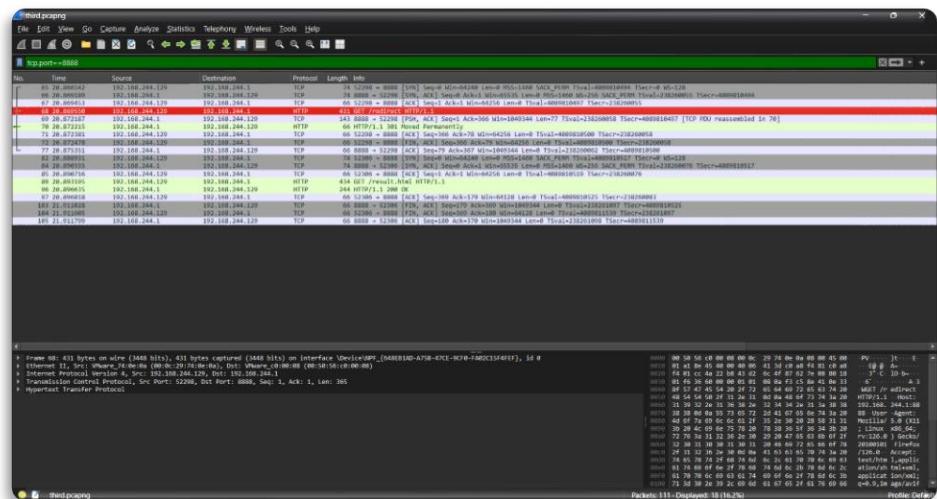




כעת מבקש הדפסן את קובץ `a/b/ref.html` אשר מכיל בתוכו מס' קבצים (תמונה) נוספים. ניתן לראות שהפעם הוא פותח מס' חיבורים (כ-שלושה) דרך כמה פורטים (תמונה ראשונה), אבל בפועל בשלב הראשון רק דרך החיבור הראשון עוברת התקשרות של הבקשה מהשרת. לאחר שהוא מבקש את הדף הראשון (זה שביקשנו בכתב), הוא מקבל אותו ומחילה ברגע בקשות כבדות מאוד של התמונות המופיעות באוטו קובץ. נשים לב שההתמונה מועברת בהמון צאנקים, אך לאחר כמה זמן הוא מנסה לבקש דרך אחד הפורטים האחרים שהוא פתח תמונה נוספת מזו שביקש דרך הפורט הראשי (תמונה שנייה), אך למרות זאת עדין התקשרות נספתחת שונה מזו שביקש דרך הפורט הראשי – הוא נסגר (תמונה השלישייה), ואז אנו רואים כיצד התמונה שנຕבקשה בפורט השונה (45602) מתחילה להגיע גם שלישיית), ואז אנו רואים כיצד התמונה שנຕבקשה בפורט השונה (45602) מתחילה להגיע גם היא. אוטו דפוש פועלה חוזר על עצמו (בתמונה הרבעית אנו רואים את `get` על התמונה שהגיעה דרך הפורט 2), כאשר מס' תמונות מגיעות דרך כמה פורטים שונים, אך אין ערבות והכל מתבצע בצדדים, דהיינו רק לאחר שפורט קיבל את התמונות שלו ונסגר (`close` הడק'), אנחנו רואים שההתמונות שביקש פорт אחר מתחילה להגיע (זה ועוד נובע מהמחסור במקביליות באופן שבו בנוינו את השרת, ובשל כך שאר הפורטים תקווים בחומרה) כאשר האחד מקבל שירותים מהשרת). התמונות שהן די גדולות מועברות בהמון צאנקים (כנראה בגלל שאנחנו עובדים מחשב-VM), ואחת לכך זמן אנו רואים במהלך התקשרות של ack שחודר עליו מהלך לו לשרת, כאשר בסיסים העברה של כל קובץ תמונה נראה שמדובר "get" מאוסף". שוב בקשות `get` מצליחות באוטו מידע ייש על הדפסן ושאר התשתיות. כך זה מתבצע באופן

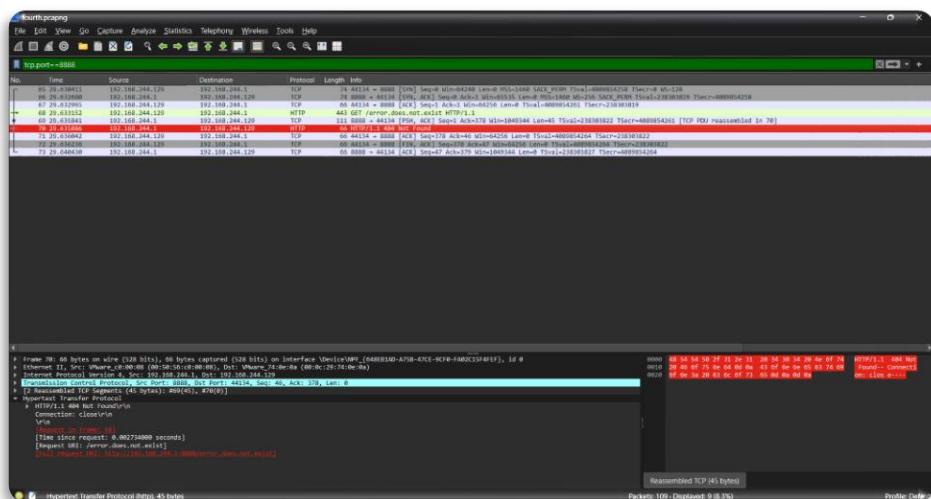
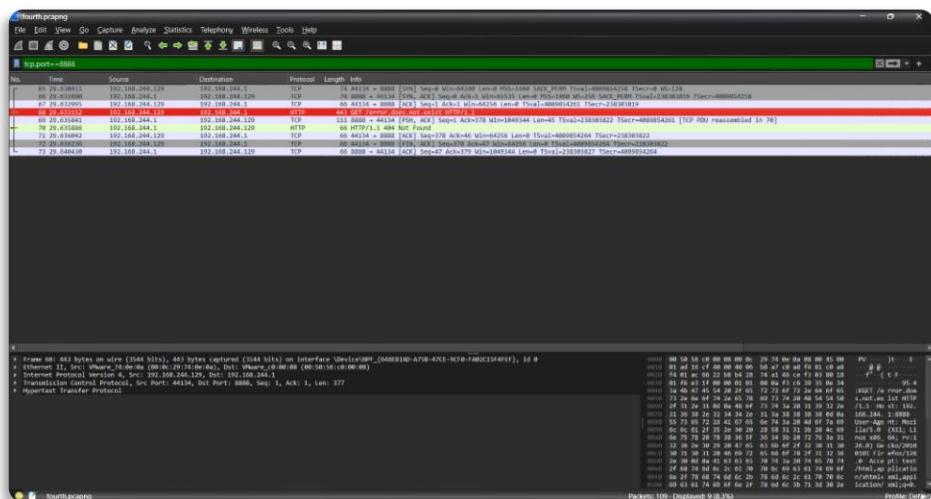
איטרטיבי עד שבסופו של דבר הפורט האחרון שביקש את התמונה האחרונות נסגר גם הוא, והדפסן מציג את הדף בשלמותו (תמונה חמישית).

<http://192.168.244.1:8888/redirect>



הדף פונה כתע אל הנטייב /redirect של השרת שלנו. ניתן לראות שהפעם הוא פותח תחילת חיבור יחיד של פורט אחד, אשר דרכו הוא שולח את הפניה (תמונה ראשונה). השרת שולח לו תשובה חוזרת של 'moved permanently', מctrפ' לשובה שלו את הפניה/הכתובת של מקום החדש, והחיבור נסגר (תמונה שנייה). מיד לאחר מכן מוקם חיבור חדש מצד הדף, אשר דרכו מתבצעת הבקשה לאוטו location-micromatch חדש שהציג השרת לדף, בבקשת הקודמת (תמונה שלישי). השרת מקבל את הבקשה השנייה, מחזיר את הקובץ שביקש בדף – result.html, ולאחר מכן רואים את הסגירה של התקשרות בחוף שגיאי משני הכוונים. ברמת המטא-דאטא אין שוני יוצא דופן מאשרינו כבר.

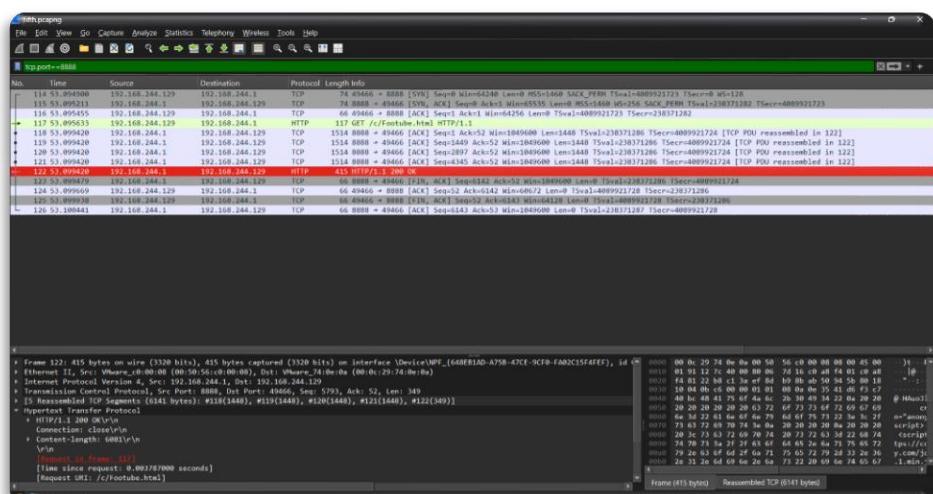
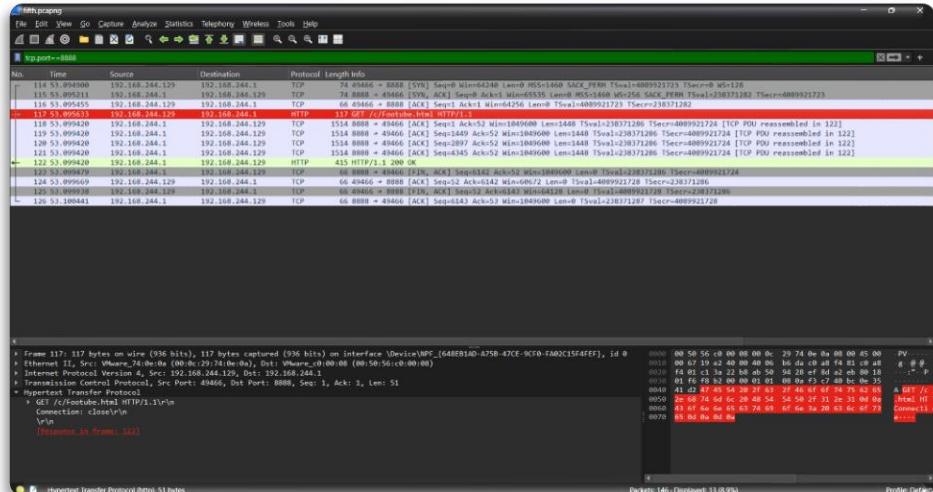
<http://192.168.244.1:8888/error.does.not.exist>



אנו רואים כתע ניסיון של הדף לפנהות לכתובת אשר לא קיימת מבחינת השרת שלנו. אנו רואים שנפתח שוב חיבור יחיד של פורט אחד, אשר דרכו נשלחת הבקשה. השרת אשר שולח ack על קר שקיבל את הבקשה (תמונה ראשונה), מיד מחזיר לדף תשובה חוזרת של status_code אינדיקציה לדף, שהדף לא נמצא – '404 not found' (תמונה שנייה). לאחר שהתשובה מוחזרת, התקשרות נסגרת בחוף. ברמת המטא-דאטא שוב אין שוני יוצא דופן מאשרינו כבר.

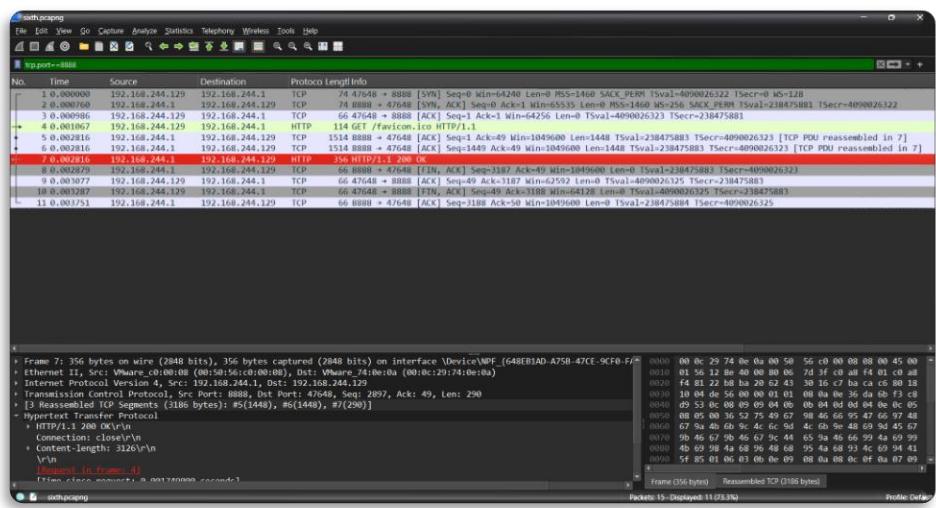
נסים בכך ש נראה מס' בקשות בכירש אשר הטענו אל הרשות אך הפעם לא מהדפסן אלא
מיקוד הקליינט שאנו חנו כתבנו –

/c/Footube.html



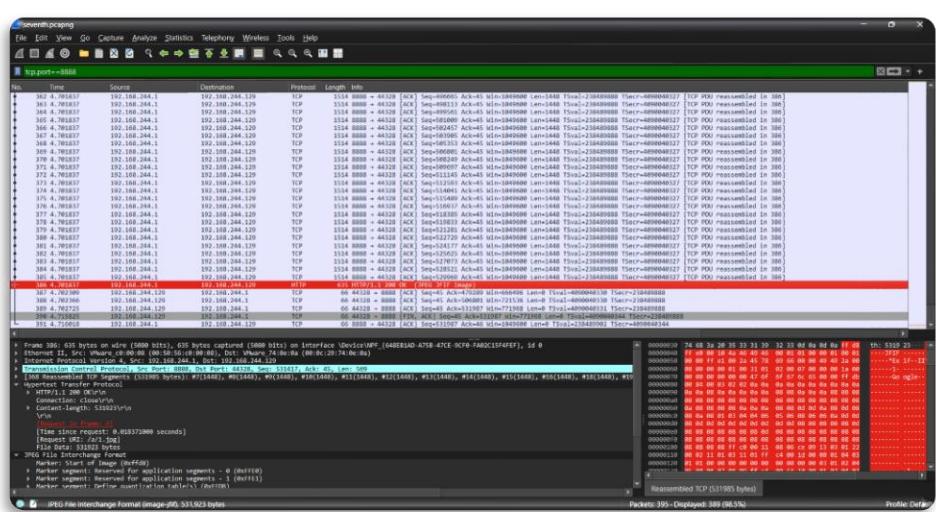
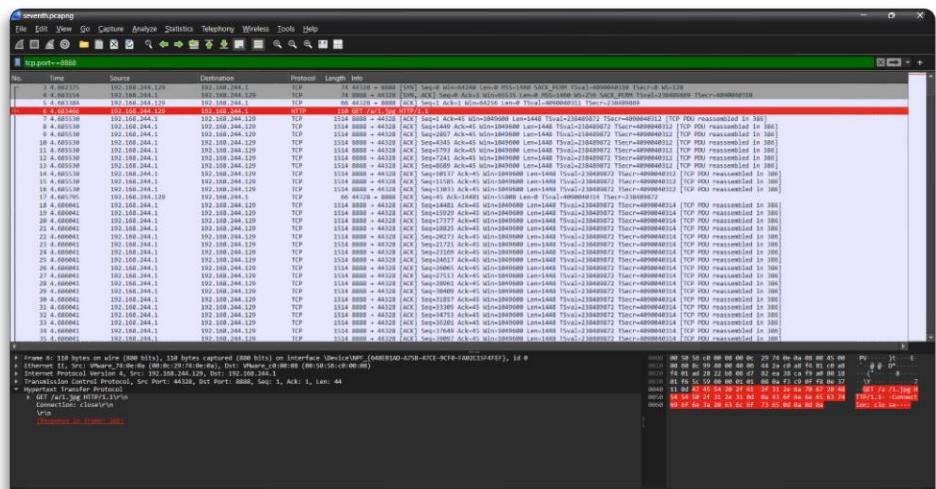
הלקוח מבקש את הקובץ [/c/Footube.html](#) עם מכל "עמוד אינטרנט" עם שלל תמונות וקבצים המוכלים בו, אולם אכן אנו רואים פורט יחיד אשר נפתח ע"י הלקוח ופונה לרשות, ויתרה מזאת לאחר שמתබל הקובץ אנו לא רואים שרשת של בקשות בדומה למה שראינו כאשר הפניה הגיעה מהדפסן, כיוון שפה הלקוח רק מבקש את הקובץ ולא מנסה להציגו, אולם לו היה מנסה היה נתקל בכל מיני קבצים ומשאבים אשר אין, ובאמת הינו רואים שרשת בקשות המנסה ליבא את כל המשאים המוחזקים באופן צזה או אחר אצל הרשות, אשר אלו דרישם באופן חינוי על מנת להציג את הקובץ במלואו. לאחר שהקובץ מגיע לתקשורת נסגרת בחfine הדדי. יש לציין שניתן לראות שהזאת הרבה יותר דليل מזה שראינו בבקשת שהגינו מהדפסן, שכן שאנו חנו במיקוד הקליינט כתבנו בפניה רק את – סוג הפניה, שוב הקובץ המבוקש, הפרוטוקול בו אנו משתמשים, וכן סטיוט החסияן.

/favicon.ico



דוג' פשוטה נוספת בה הלקוח מבקש את הקובץ של האיקון של הרשת (התמונה הקטנה המופיעה בראש הלקוח), אנו רואים שהשרת מחזיר את התוכן ב2 חבילות שונות (מפתחת גודלו), ומאסף זאת בהודעתה נשלם, אנו רואים את הסגירה של התקשרות.

/a/1.jpg



דוג' אחרונה אשר בה הלקוח מבקש את קובץ של תמונה מהשרת. מכיוון שהתמונה מאד גדולה עבור השירות (וכן אנו עושים פה מעבר מהמחשב לוח שMRI'יןוקס), אנו רואים שהשרת מחזיר את התוכן בהמון המונח חבילות שונות, ורק בסופו של דבר לאחר מעבר של חמיש מאות אלף בתיים, נשלחת הודעת הסיכום של ה- ok 200. לאחר שזו מגיעה נסגרת התקשרות באופן תקין. ניתן לשים לב אגב שאם אנו פותחים את הליבל של הקונטנט כאשר נשלחת תמונה, ניתן לראות כל מיני פרטיים עליה ועל סוגה על מנת שייהי ניתן לפתח ולפענה אותה מצד שני (מידע זה נשלח בודאי גם בדוג' הקודמות כאשר שלחנו תמונה בגין ההודעה).