

## מקרה מס' 2 – אלאי בין יושע ונועם ל'יבויץ'

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')

```

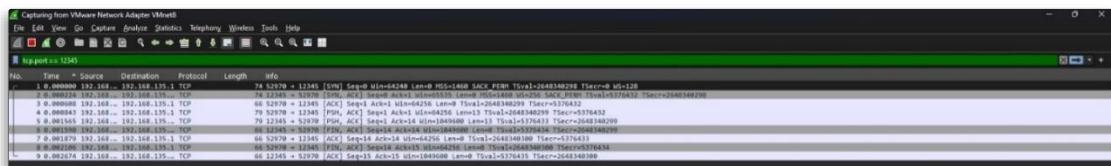
```

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()

```

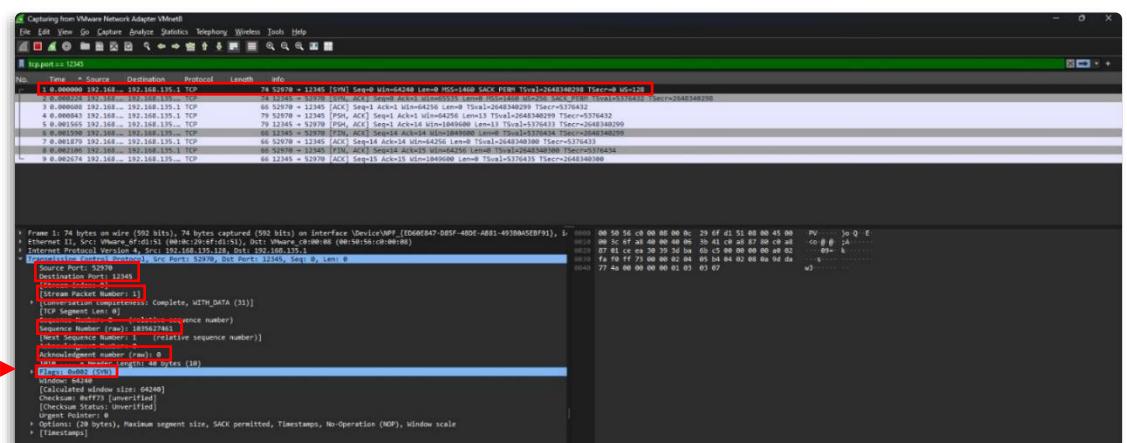
cut נרץ את הקודים המעודכנים הנ"ל, ונסיף את המידע בערת "כרייש-הכבל" (בלוע – wireshark, מעתה והילך כאשר נכתב 'כרייש' בקיצור נתיחס למינוח זהה), וננתח את מה שקרה פה בהתאם לעקרונות TCP שראינו בהרצאה ובתרגול.

זה מה שמוצג לנו בכריש לאחר הרצת הקודים -



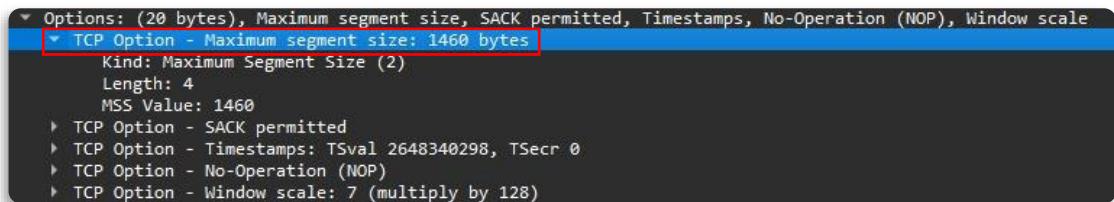
מעבר על כל חבילה וחבילה על מנת לתאר במדויק כל שלב בתקשורת.

## חבילה מס' 1 –



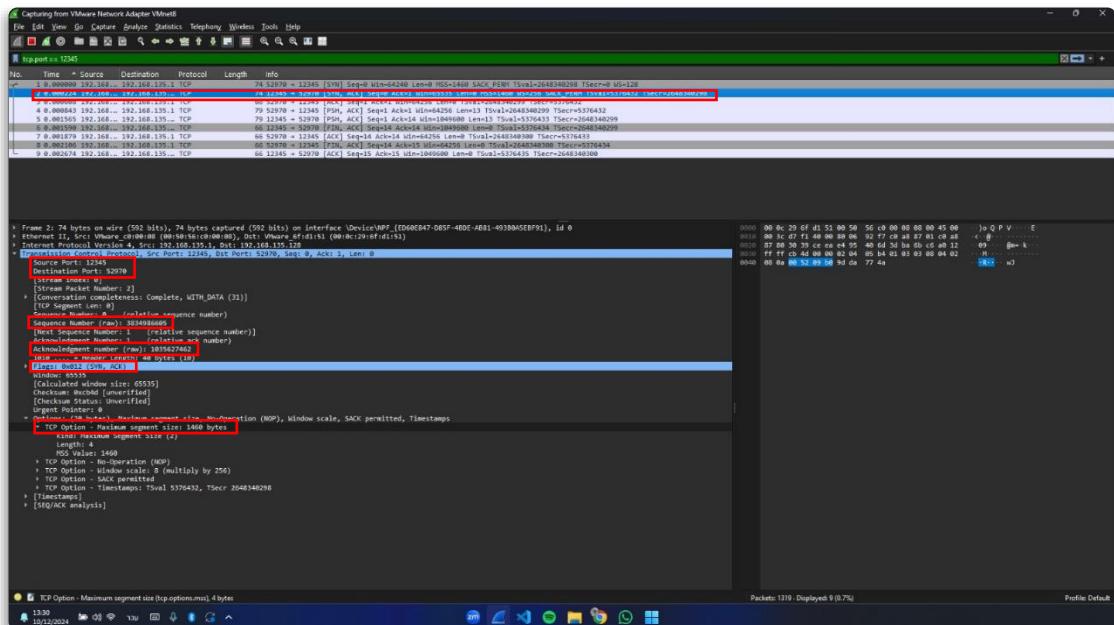
אנו רואים פניה שמתבצעת ל'PORT 12345 - (אשר אותו נתנו ידנית לשרת), אשר מגיעה מפורט מקור 52970 אשר ניתן דינאמית ללקוח ע"י מ"ה (כיון שלא צמדנו אותו לפורט ספציפי). הפניה היא מסוג chs (ניתן לראות את הדגל הדלוק), ככלומר הלקוח מבקש להסתنصرן עם השרת וזו הפקידה של החבילה הנ"ל עם דגל החsus, אשר מהוות את השלב הראשון בטקס 'לחיצת הידיים המשולשת'. בנוסף לכך אנו רואים את ערך האופטט הגולמי (= raw) של seq (raw) שהחומר הלקוח, כאשר ניתן לראות שהוא החלטת תחילת התקשורת תחילה מאופטט – שבחר הלקוח, כאשר לנו שערך ack שלו כרגע יהיה '0', משום שהseq שלו, זה eqp של השרת (אשר טרם נבחר.). כמובן שהseq הוא 0 שהרי אין DATA בהודעת chs.

דבר מעניין נוסף ניתן לראות כאשר נפתח את הoptions –



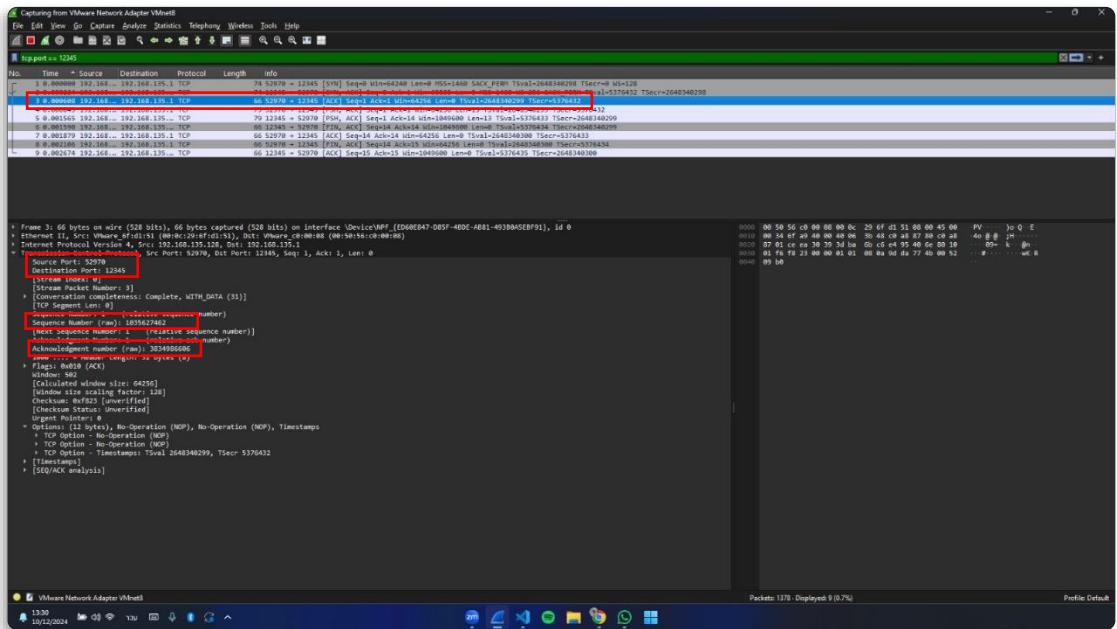
ובו, הלקוח מציין בפני השירות – "רק שטdue, התסס שלו הוא 1460b".

## חvíלה מס' 2 –



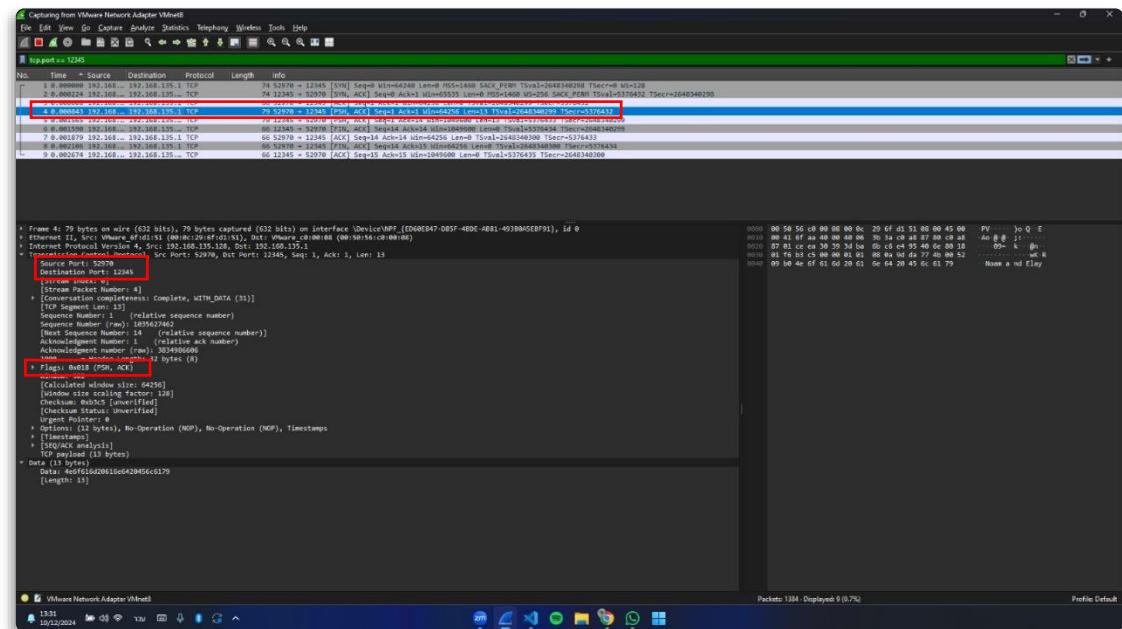
אנו רואים פניה שמתבצעת הפעם בכיוון הפהוף מ-PORT 12345 (השירות), אשר מיודעת לפורט 52970 (הלקוח). הפניה (או יותר נכון 'tagובה') מצד השירות הינה מסוג syn, ack (גם פה ניתן לראות את שני הדגלים דלוקים), כלומר **השירות ראשית מכיר בסנסרנו של הלקוח איתנו**, אך בנוסך הוא מבקש להסתنصرנו גם הוא עם הלקוק. זהה למשזה תפקידה של החvíלה הנ"ל עם דגלי ack, syn, אשר מהוות את השלב השני בטקס 'לחיצת הידיים המשולשת'. ראיו לציין כי השלב הנ"ל עשוי להתבצע לעתים ב-2 חבלות שונות (1 ל-ack ואחת לח-syn) אך כמו שאנו רואים כאן, אלו יכולות להישלח גם כן ייחודי בחvíלה אחת. בנוסף לכך אנו רואים את ערך האופסוט הגולמי (= raw) של seq (raw) של 3834986605, ואילו נשים לב שערך ack שלו יהיה הักษורת תחיל מואפסט – '1305627462', שזה 1 יותר מאשר הסהיה seq של הלקוק בגל ה-bit-bit phantom. גם כאן הchen הוא 0 שחיי אנחנו עדין לא בשלב של העברת>Data, אך מאוד מתקרבים לשם. אם נפתח את ה-optiosn נשים לב שגם השירות מציין בפני הלקוק מהו התסס שלו – 1460b, כדי שזה דעת את הנתון אשר הוא שולח אליו הודעות.

## חvíלה מס' – 3



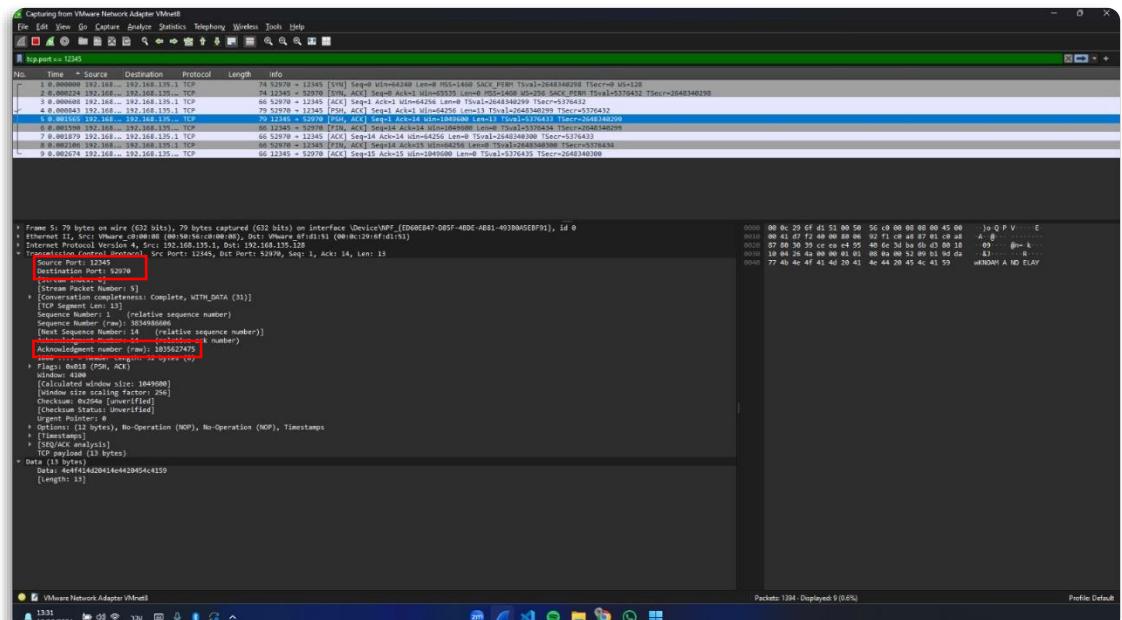
כעת ניתן לראות פניה נוספת מהלך 52970 (הлокח), אל השרת. הפניה מצד הרשת הינה מסוג ack (שוב הדגל דлок ולמעשה מעתה והוא יהיה דлок עד סוף התקשרות, אז נפסיק לציין זאת...), דהיינו הлокח גם הוא מכיר בהכרת הרשת בסמכון שלו, ולמעשה תפקידה של החvíלה זו מהוות את השלב השלישי והאחרון בטקס 'לחיצת הידיים המשולשת', שכן שני הצדדים הסתנכרנו אחד עם השני, וכל אחד מהם הכיר בסמכון המשותף, ומכאן והואיל הקשר שරיר ונitin להעביר דרכו הודעות. שוב אנו רואים את ערך האופטט הגלומי (= raw) של ack שהשלוח הлокח, והוא – '06'3', באופן לא מפתיע זה 1 יותר מהseq שבחר הרשת (שוב אנחנו מגדילים בגל הפאנטום ביט של ack). הseq 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 שהseq שלו כבר לא נשלח, שכן שני הצדדים כבר יודעים ומודיעים למוגבלות כל אחד של השני.

## חvíלה מס' – 4



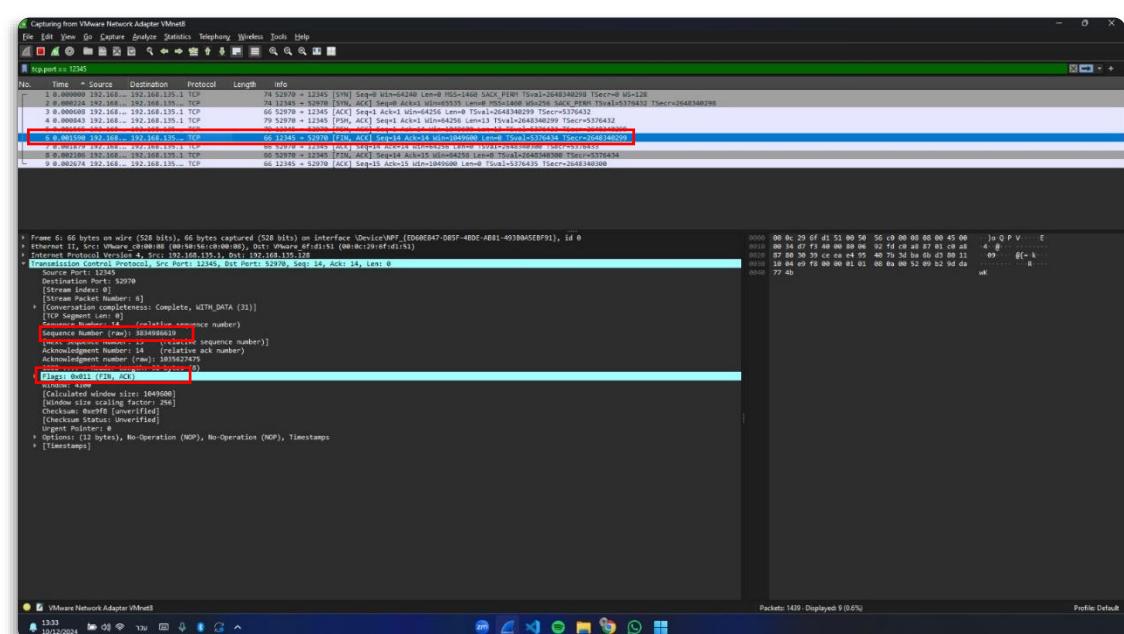
סוף כל סוף הגיענו לחלק המעניין ('פsegת הקורס' לפי חלק מהשיעור), אנו רואים הودעה מהלוקו (לפי הפורט 52970 שראינו קודם שהוא הפורט של הלוקו), השולח 13 בתים (לפי *len*), שנחננו יודעים שהם הבטים של המחרוזת 'Noam and Elay' (כנדרש במתלה), ובנוסף לדגל *ack* (שאמרנו שמעתה והילך תמיד היה דילוק), גם כן דילוק דגל ההשיק שכאילו אומר להעביר בדחיפות לאפליקציה את המידע שהגיע. נשים לב שעכשיו יש לנו גם את שכבת *data* בהודעה (הבטים שלחנו מואפן מוקודד).

## – 5 'חבילה מס'



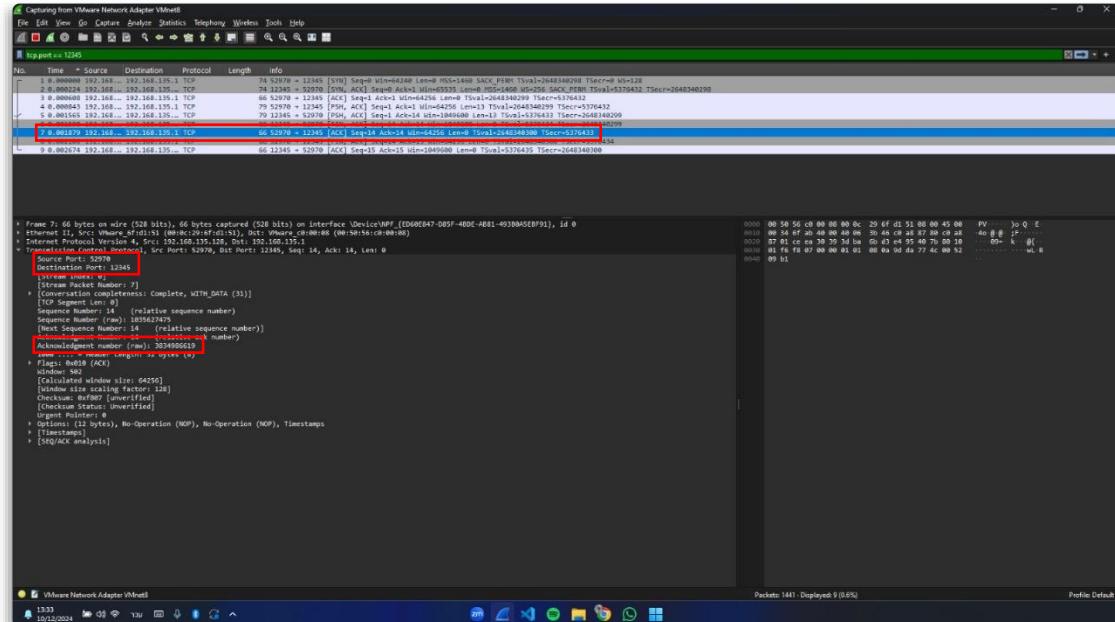
כעת נראה את תגובת השירות להודעה שקיבל מהלוקו בחבילה הקודמת שסקרנו. הרשות הראשית **מעדכן** את המספר בשדה *ack* להיות 13 יותר מאשר – '1035627475' שכן עכשו הוא אומר ללוקו – "שמעו?! קיבלתיכי כבר עד בית מס' 1035627475 תן לי ממוני והלאה.". אמןם *seq* לא משתנה כיון שהוא טרם שלח נתונים משל עצמו, אך כעת הוא מכרף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

## – 6 'חבילה מס'



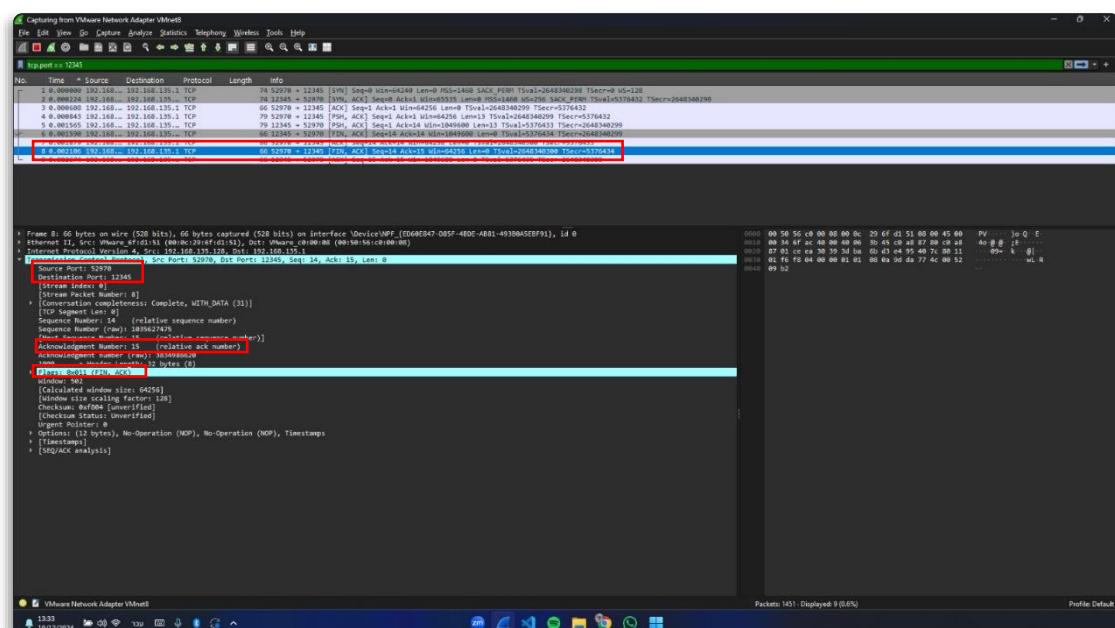
מיד לאחר שהשרת שולח את הודהה **ack** יחד עם התוכן בזקוקס, הוא שולח הודעה **fin** (ניתן לראות דגל דולק), שכן הוא עשה את שלו, ותפקיד ההודהה זה להודיע על כך שהוא מעוניין לסיים את מערכת היחסים עם הלקוח. נשים לב שעכשו הseq של השרת גדל ב-13 גם הוא ערך – '3834986619'.

## – 7' חבילה מס'



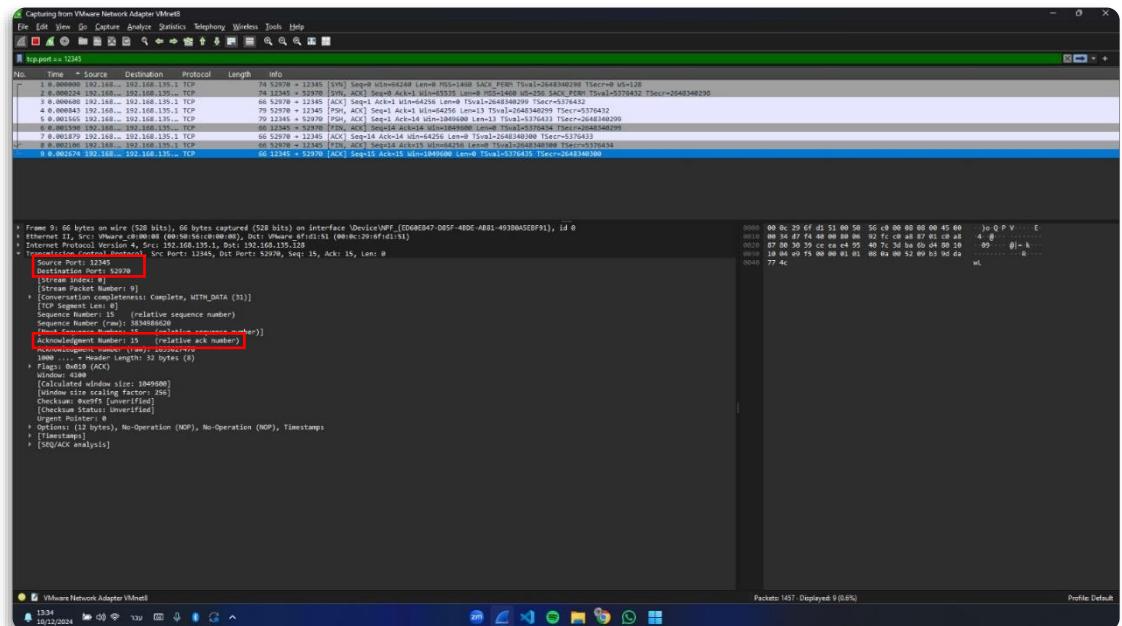
לקוח מקבל את הדטא חזרה מהשרת בזקוקס, בהתאם הוא מגדיל את ערך **acked** ב-13 בתים נוספים לערך – '3834986619'. כמו כן נשים לב שגם הseq המזמין בהודהה גדול ב-13 מהפעם האחרון בה ראיינו חבילה נשלחת מהלקוח, וצעת הוא עומד על – '1035627475' (שוקינג זה הseq של השרת..).

## – 8' חבילה מס'



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

## חבילה מס' 9 –



בחבילה الأخيرة (😃) שלנו אנו רואים את השרת מכיר בסיום התקשורת מצד הלוקוח גם הוא בהודעת ack, ובאופן פרקטטי **תפקיד ההודעה הינו לסייע לחלטין בהחלט את ערך התקשורת שהייתה בינהם**. נשים לב שרגע לפני שהוא אומר "ב'י'", הוא מגיד גם הוא את ערך ack כתוצאה מהפאנטום בית של החfin מהлокוח, סה"כ ערך ack של השרת בסיום התקשורת עומד על – '1035627476'.

סימנו לנתח את התקשורת בחלק הראשון. נוסיף כעת את שליחת ה-ת.ז. של אילאי לאחר שהשרת משיב את תשובתו ונראה את השינויים.

```

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')

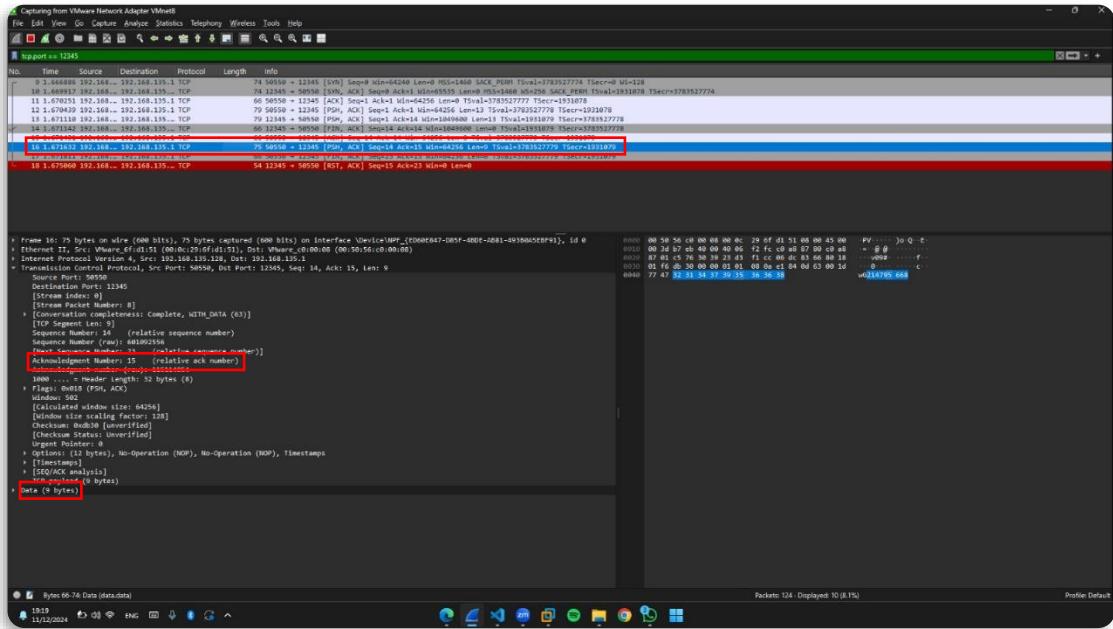
```

```

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.send(b'214795668')
8 s.close()

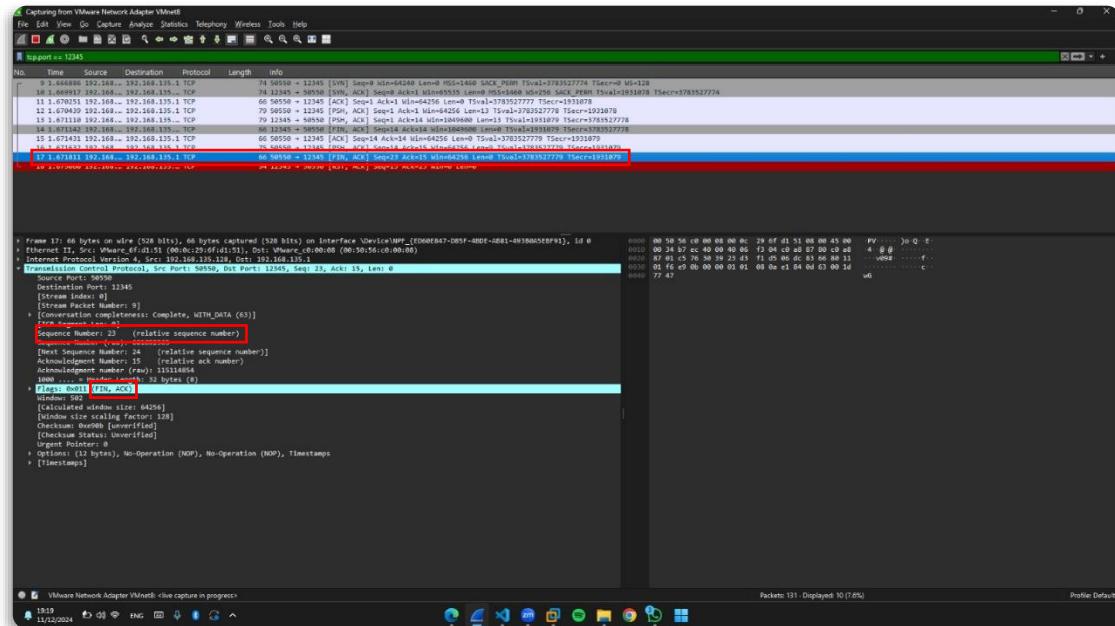
```

כל ההודעות בהתחילה תתנהגנה באופן זהה (לכן נחשוך ונתחיל רק מהמקום בו השינוי קורה), אבל עכשו נשים לב –

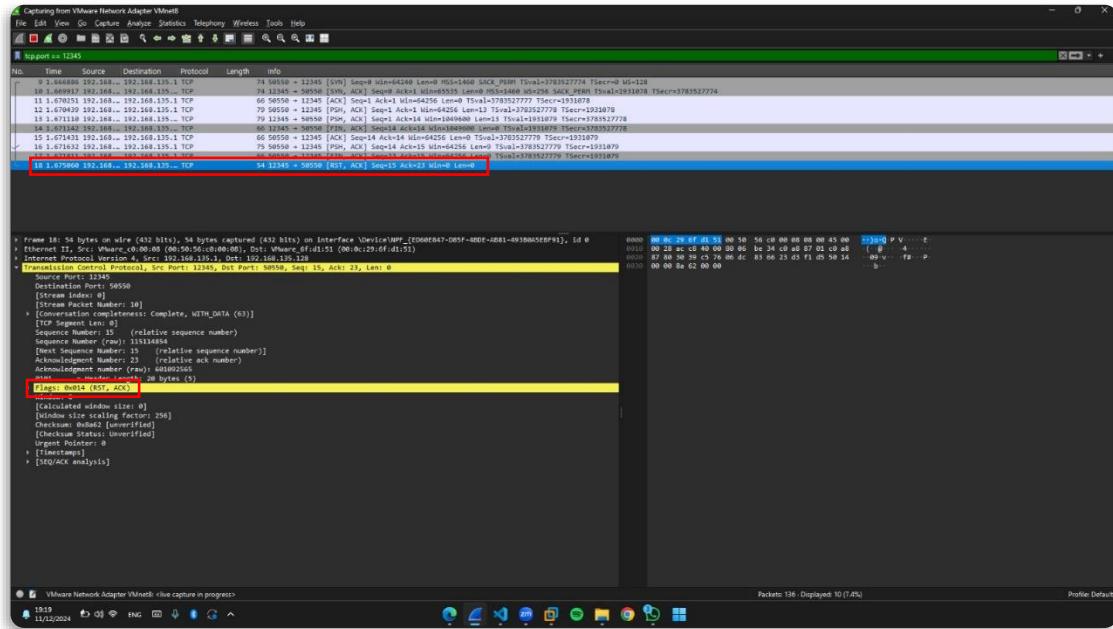


לאחר שהליך החזר ack על החין של הרשות (פאנטום ביט מגיל ב1 את ה-acked), הוא מנסה לשלווח גם fin (כמו שקרה קודם), והוא מנסה לשלווח נספח ולבסוף לה sksh לאפליקציה (הלווא היא התז). ורק לאחר שההודעה fin נשלחת, הוא שולח גם fin

מצידן –



הqed שלו גדל ב-9, והוא מסכום עכשווי לסיים את התקשורת. אלא שבגלל שהשרות לא ידע שעליו לצפות לחבילה נוספת הוא כבר סיים וסגר את העניין, ולכן קיבל ממנו הודעה – `rst`



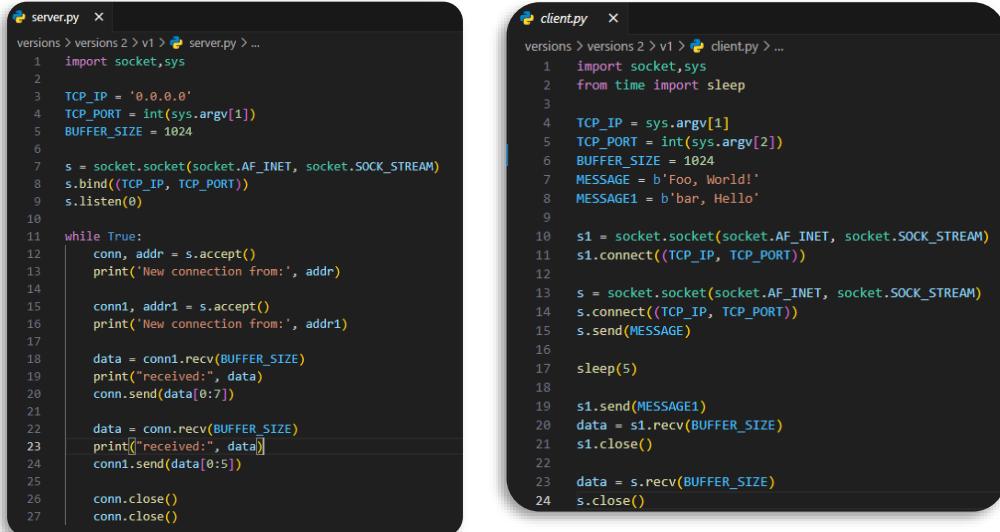
כאילו רוצה לומר – "חדש את שלו", חדש. מה שהיא אני כבר לא שם, ואני מי שישמע ויקבל את מה שיש לך לשולח. אם אתה רוצה בוא נתחיל מההתחלת, ומשם נשתמע הלאה..".

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

## סעיף 2:

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

### גרסת 1:

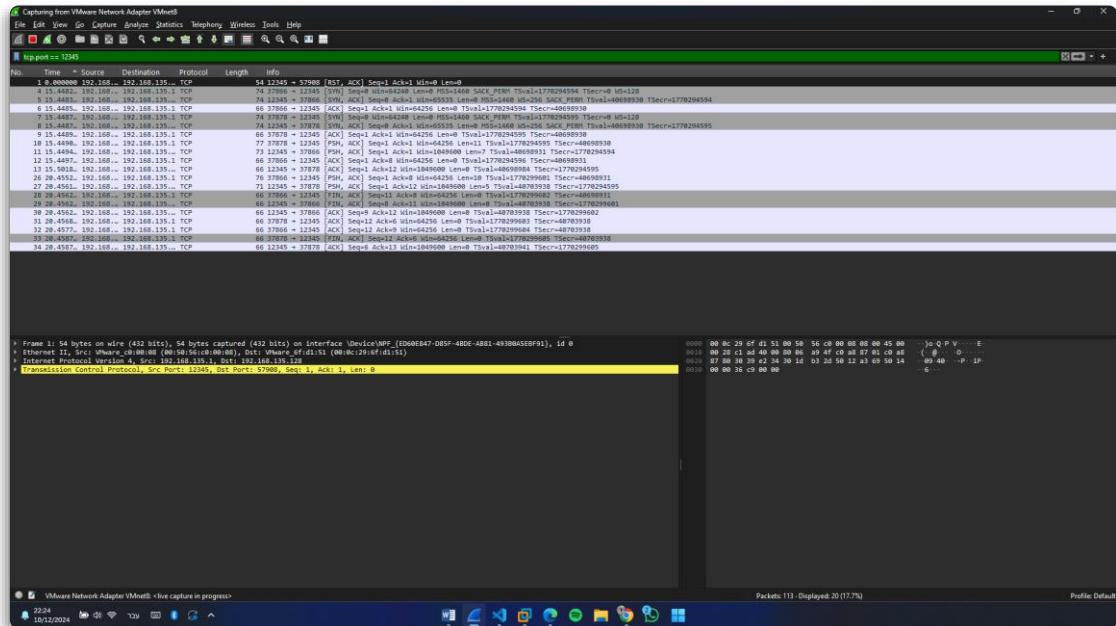


```

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 מהארוגומנטים לתוכנית, לאחר מכן הוא מתחבר בשני חיבורים בפדרטם בعزيزת שני סוקרים נפרדים לשרת (שורות 4 עד 9). הלקוח תחילת שולח הודעה דריך החיבור שני, ממחכה 5 שניות ושלח הודעה אחרית דריך החיבור הראשון. לבסוף, הלקוח מקבל את שתי ההודעות מהסתוקט בסדר הפוך לסדר בו הבצעה הש寥חה, ולאחר מכן סגור 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), מקצר את הודעה השנייה ל-7 תווים הראשונים שנקלטו ושלח אותה לחיבור הראשון, ואת הודעה הראשונה ל-5 תווים הראשונים שנקלטו ושלח אותה לחיבור שני. לבסוף, יוזם ניתוק בקע 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 מחייב עלייה ack לאותו 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 מחייב 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 מחייב ack על עצם זה מהסוקט שנוצר ראשון בקלינט, ובאופן דומה לפניו שהוא מחייב ack על עצם זה, הוא מעביר את 5 הבטים הראשונים מההודעה לחיבור סוקט השני. בשלב זה מגיעה הודעה fin מהחיבור הראשוני, אשר נוענית בחין מצד השרת, ועל הדרך הוא גם נותן לו ack (רק בשלב זהה) על החבילה שנשלחה על ידו, ומיד הודעה נוספת נסotta של ack על החין עם הפאנטום ביט. החיבור השני מחייב ack רק עכשו על ההודעה שהוא קיבל (5 הבטים), החיבור ראשון מחייב ack על הודעות החין שקיבל מהשרת (שוב פאנטום ביט), ולבסוף השרת מקבל fin מהחיבור השני ומשביב על כר ack (הוא לא שלוח לו fin שכן בקוד החין ששולח השרת הינו עבור החיבור הראשוני, שכבר נסגר מיזמתו קודם לכך..).

## גזרה 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
```

Capturing from VMware Network Adapter VMnet8  
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help  
tcp port == 12345  
No. Time Source Destination Protocol Length Info  
1 0.199455 192.168.1.35.1 TCP 74 48622 -> 12345 [SYN] Seq=0 Win=64248 Len=0 MSS=1468 SACK\_PERM TS=0.1-1771713162 TSecr=<0 WS=128  
2 0.199455 192.168.1.35.1 TCP 66 48622 -> 12345 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1771713162 TSecr=<0  
3 0.199777 192.168.1.35.1 TCP 66 48622 -> 12345 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1771713162 TSecr=<0  
4 0.200324 192.168.1.35.1 TCP 71 12345 -> 48622 [PSH, ACK] Seq=1 Ack=2 Win=64256 Len=10 TSval=1771713162 TSecr=<0  
5 0.200324 192.168.1.35.1 TCP 66 48622 -> 12345 [ACK] Seq=2 Ack=1 Win=64256 Len=0 TSval=1771713162 TSecr=<0  
6 0.200324 192.168.1.35.1 TCP 71 12345 -> 48622 [PSH, ACK] Seq=2 Ack=3 Win=64256 Len=10 TSval=1771713162 TSecr=<0  
7 0.200324 192.168.1.35.1 TCP 66 48622 -> 12345 [ACK] Seq=3 Ack=2 Win=64256 Len=0 TSval=1771713162 TSecr=<0  
8 0.200324 192.168.1.35.1 TCP 70 12345 -> 48622 [PSH, ACK] Seq=21 Ack=21 Win=64256 Len=0 TSval=1771713162 TSecr=<0  
9 0.200324 192.168.1.35.1 TCP 66 48622 -> 12345 [ACK] Seq=21 Ack=21 Win=64256 Len=0 TSval=1771713162 TSecr=<0  
10 0.200324 192.168.1.35.1 TCP 70 12345 -> 48622 [PSH, ACK] Seq=22 Ack=22 Win=64256 Len=0 TSval=1771713162 TSecr=<0  
11 0.200324 192.168.1.35.1 TCP 66 48622 -> 12345 [ACK] Seq=22 Ack=22 Win=64256 Len=0 TSval=1771713162 TSecr=<0  
12 0.200324 192.168.1.35.1 TCP 70 12345 -> 48622 [PSH, ACK] Seq=23 Ack=23 Win=64256 Len=0 TSval=1771713162 TSecr=<0  
13 0.200324 192.168.1.35.1 TCP 66 48622 -> 12345 [ACK] Seq=23 Ack=23 Win=64256 Len=0 TSval=1771713162 TSecr=<0  
14 0.200324 192.168.1.35.1 TCP 71 12345 -> 48622 [PSH, ACK] Seq=24 Ack=24 Win=64256 Len=0 TSval=1771713162 TSecr=<0  
15 0.200324 192.168.1.35.1 TCP 66 48622 -> 12345 [FIN] Seq=24 Win=0 Len=0  
16 0.200324 192.168.1.35.1 TCP 68 48622 -> 12345 [STP] Seq=25 Win=0 Len=0

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 לחיצת הידיים בין הלקוח לשרת (שורות 4 ו-5). 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 (בזאתםן 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 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()

```

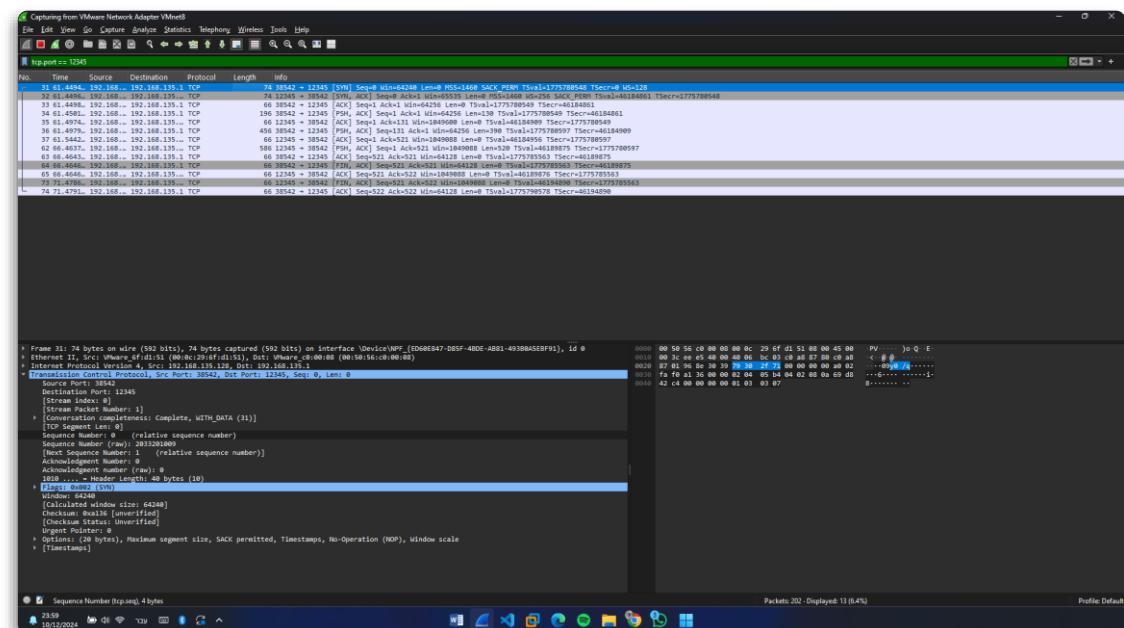
כמו בתוכנית הקודמת, הקוד לא השתנה המון – הלקוח מקבל את התוכנית, לאחר מכן הוא מתחבר לשרת ושולח לו הודעה, אך בשונה מהתוכנית הקודמת, לאחר שקיבל את הודעה חזירה מהשרת הוא מושך ומדפיס את 1024 בתים הראשונים ולאחר מכן שוב הוא מנסה למשוך עוד 1024 בתים. כמו כן בצד שני, כמו בתוכנית הקודמת, השרת מאיין לכל החיבורים כיוון שהוא בחר בכטובת 0.0.0.0 ומקבל פורט נקלט. הוא פותח חיבור, מקבל את 1024 הבטים 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 ש מגיע 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 שהגעו, 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)
```



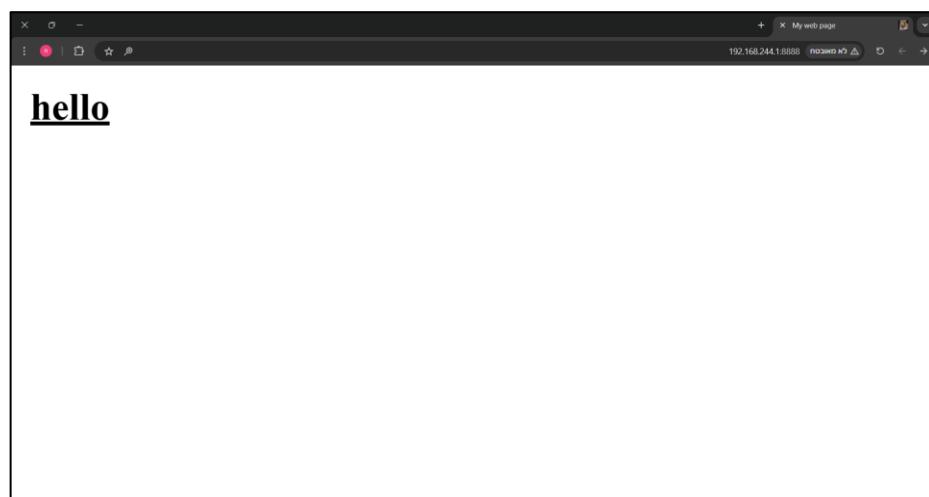
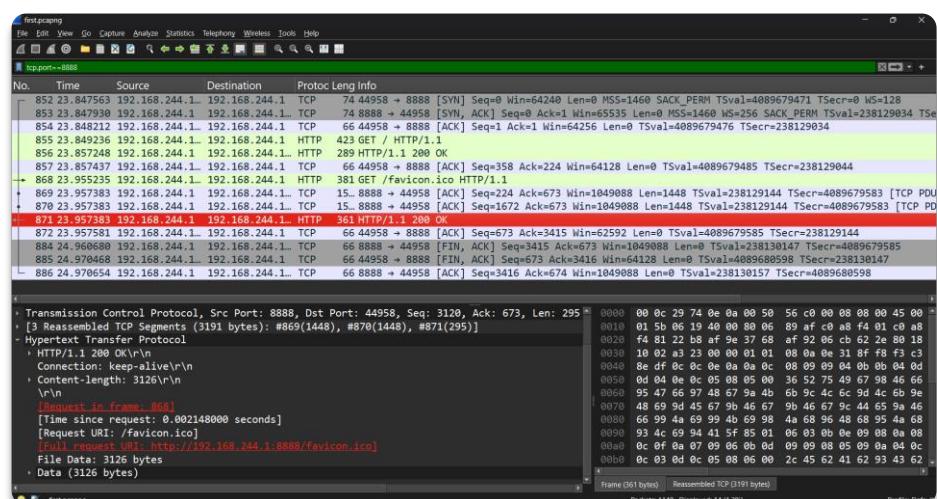
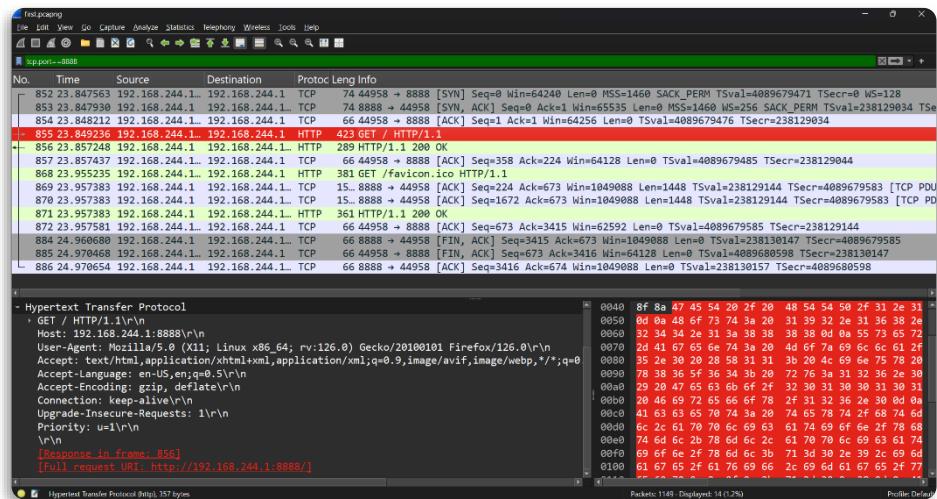
כרגיל הלקוק מקבל את IPו והפורט מהארגוןנים לתוכנית, לאחר מכן הוא מתחבר לשרת ושולח לו 4 הודעות שכל הודעה היא 10 פעמים "Hello, World", השוני הפעם מהתוכנית הקודמת, הוא שלאחר שקיבל את הודעה חזירה מהשרת הוא מדפיס את 1024 בתים הראשונים. בצד שני, כרגיל השירות מażין לכל החיבורים כיון שהוא בחר בכתובת 0.0.0.0 ומקבל פורט קקלט. הוא פותח חיבור מכחכה 5 שניות, מקבל את 1024 הבטים הראשונים ושולח אותם חזירה ללקוק באוטיות גדולות. הוא חוזר על התהיליך עד שהוא מפסיק לקבל מידע מהלקוק ולבסוף סגור את החיבור.

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

## תיק ב'

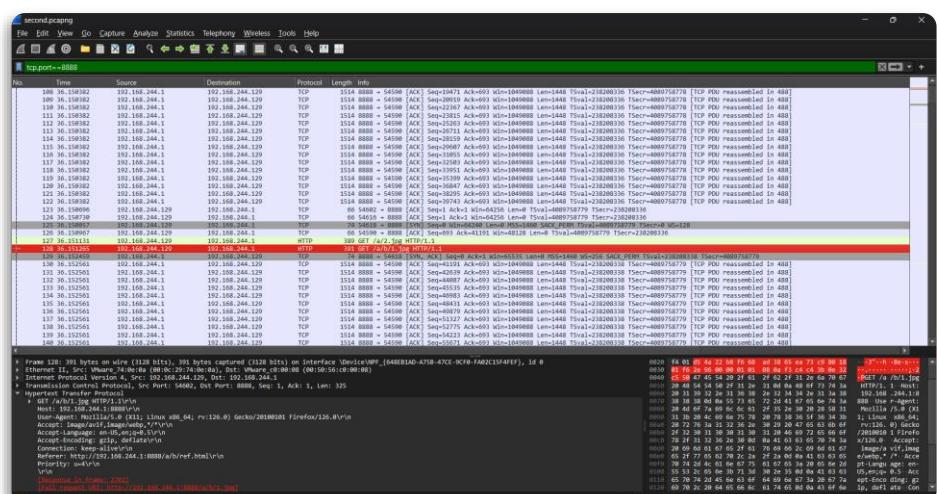
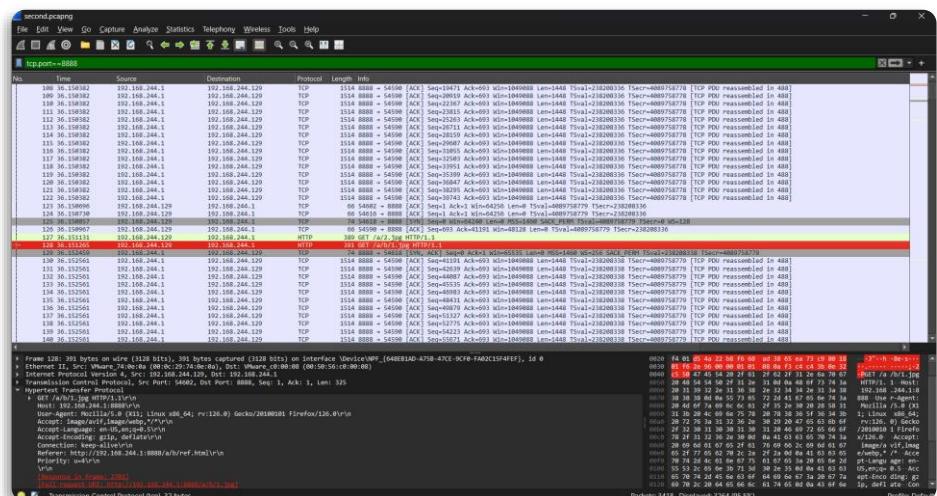
נסים חלק זה של התרגיל בהציג הכריש בזמן ריצת קוד השרת שלנו, כאשר האחרון מקבל בקשות מהדפסן (אשר רץ בVM).

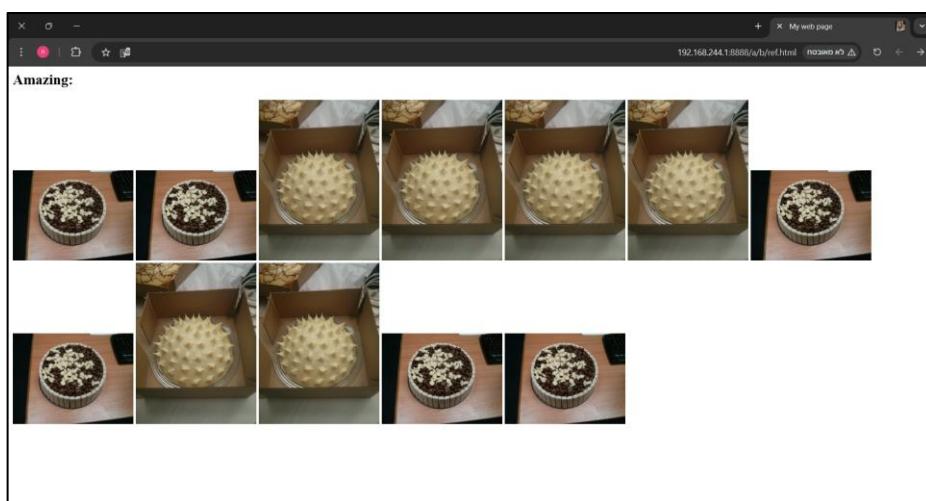
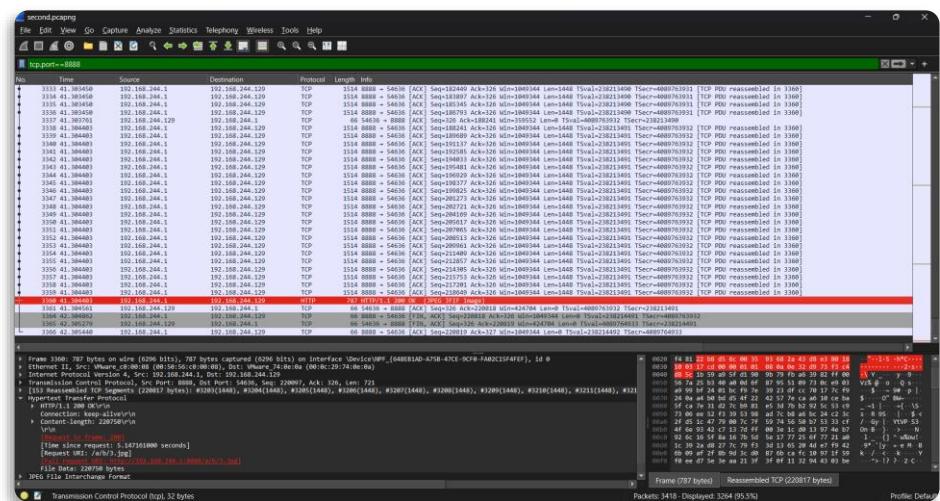
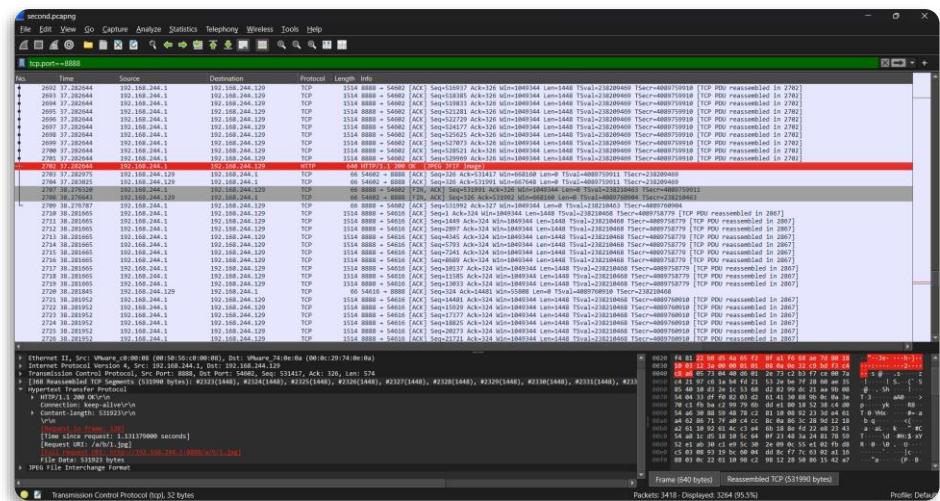
<http://192.168.244.1:8888/>



הכתובת פונה ומבקשת מהשרת את דף הבית, והוא מוחזיר בתמורה את – index.html. ניתן לראות בתמונה הראשונה שדף עושה מס' פניות מסווגו פורט (44958), כאשר תחילת הוא שלוח את הבקשה של הקובץ, ומתקבל אליו חזרה את ה ok 200, מיד לאחר שהם מחליפים ack ביןיהם אודות מעבר הקובץ באופן תקין, הדף שולח בקשה נוספת עבור הקובץ favicon.ico הלווא הוא האיקון של "הarter" שלו שאננו רואים בקצתו הגרפיה בדף, גם על בקשה זו מתקבלת הודעה ok 200 (תמונה שנייה – רואים את הקובץ עליו והוזר ok למטה body), והתקשרות נסגרת באופן הדדי בחומר משני הצדדים. ניתן לראות שנשלחו המונן מידע עמו הודעה get מדף, כמו סוג הדף, הגרסה שלו, connection, סטטוס ווד. מעבר לget, ההזרה של המידע מהשרת – ok 200, אז הבקשה לאיקון אין מיוחד, ואין בקשה נוספת פורט.

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

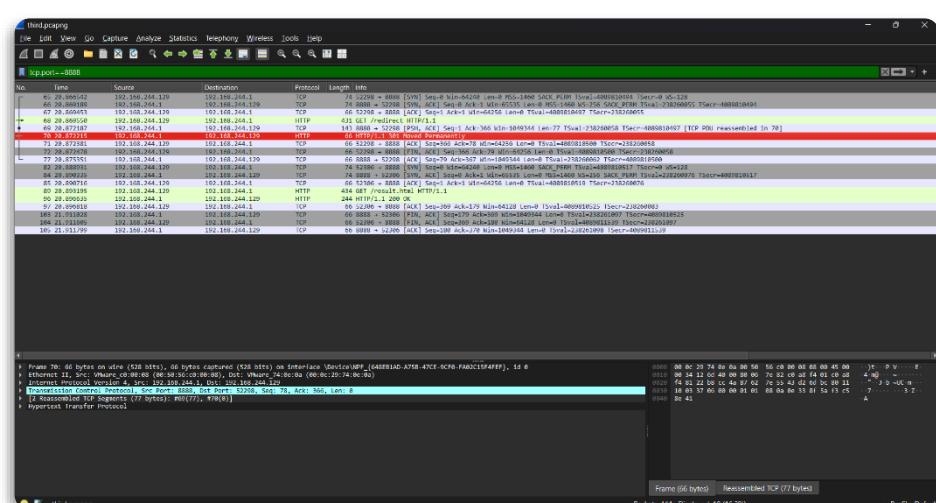
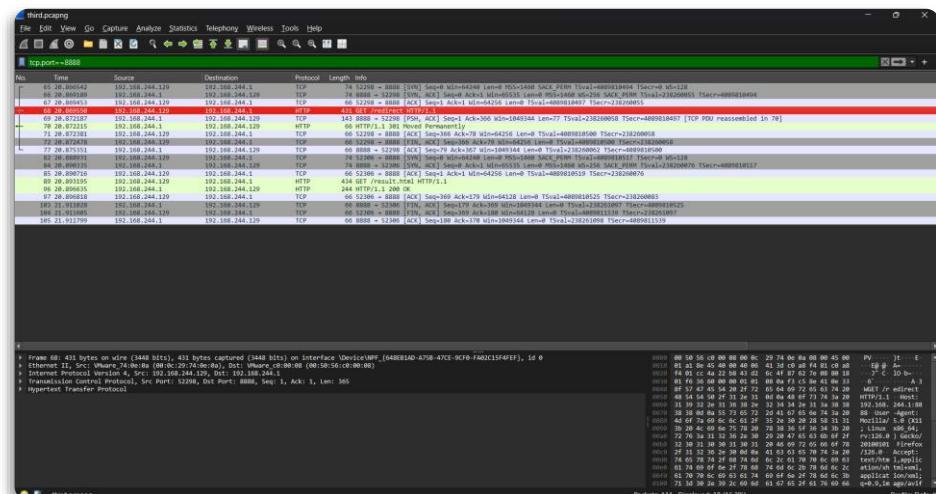


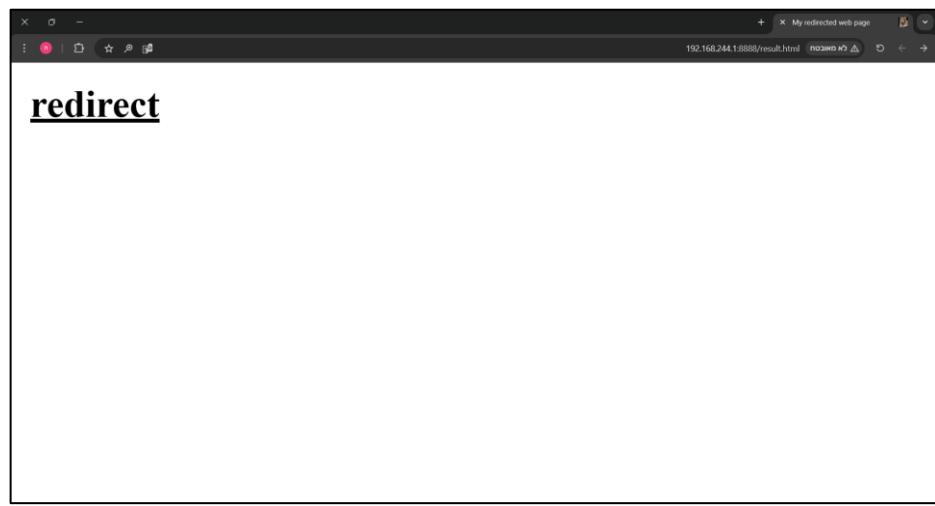
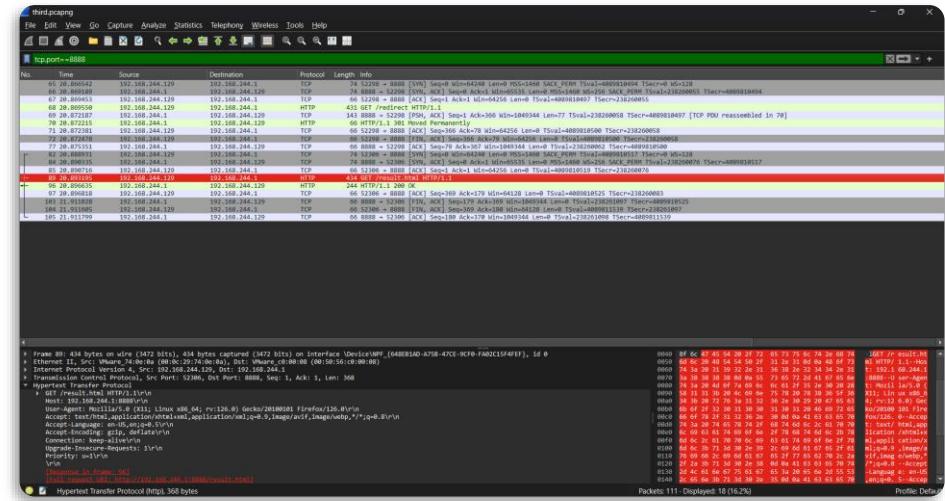


cutet mbaeksh ddpfen at kovz a/b/ref.html / a'er mciil b'toco ms' kbzim (tmonot) nosfim. nitn lrato shafum ha potch ms' chibrim (c-shlosha) drk cmha portim (tmona rasona). abel b'poval b'shlb hrason rk drk hchbor hrason uverb tchkosrt sl b'ksha mhesrat. la'achr shao mbaeksh at dd' hrason (za shbiksho b'ctobet), hoa mkbil otu mthail bratz b'kshot cbdot mao dl tmonot mofifut b'otu kovz. nshim lb shahmmona moubart b'mon zanekim, ar la'achr cmha zm ha mnsa b'ksh drk achd portim a'hriim shao pth tmona

נוסף שונה מזו שביקש דרך הפורט הראשי (תמונה שנייה), אך למרות זאת עדין התקשרות ממשיכה להגעה דרכו. לאחר שמס' תמונות הגינו מהפורט הראשי – הוא נסגר (תמונה שלישית), ואז אנו רואים כיצד התמונה שנטבקשה בפורט השונה (45602) מתחילה להגיע גם היא. אוטו דפוס פועל חזר על עצמו (בתמונה הרביעית אנו רואים את get על התמונה שהגעה דרך הפורט (45602), כאשר מס' תמונות מגיעות דרך כמה פורטים שונים, אך אין ערבוב והכל מתבצע בצדדים, דהיינו רק לאחר שפורט קיבל את התמונות שלו ונסגר (fin הדדי), אנחנו רואים שהתמונות שבקשה פорт אחר מתחילה להגעה (זה ודאי נובע מהמחסום במקבילויות באופן שבו בנוינו את השירות, ובשל כך שאר הפורטים תקועים בחוזר האחד מהלך שירותים מהשרת). התמונות שהן די גדולות מועברות בהמון צדדים (כנראה בגלן מקבל שירותים מחשב-VM), ואחת לכמה זמן אנו רואים במהלך התקשרות ack ש חוזר עליהם שאנו עובדים מחשב-VM), ואחת לכמה העברה של כל קובץ תמונה נראה שmagiu "מאסף". שוב מהלך לשרת, כאשר בסיסים העברה של כל קובץ תמונה נראית get "מאסף". שוב בקשנות get מכלות באותו מידע ישב על הדפדפן ושאר התשתיות. כך זה מתבצע באופן איטרטיבי עד שבסתוף של דבר הפורט האחרון שביקש את התמונה האחרונות נסגר גם הוא, והדפדפן מציג את הדף בשלמותו (תמונה חמישית).

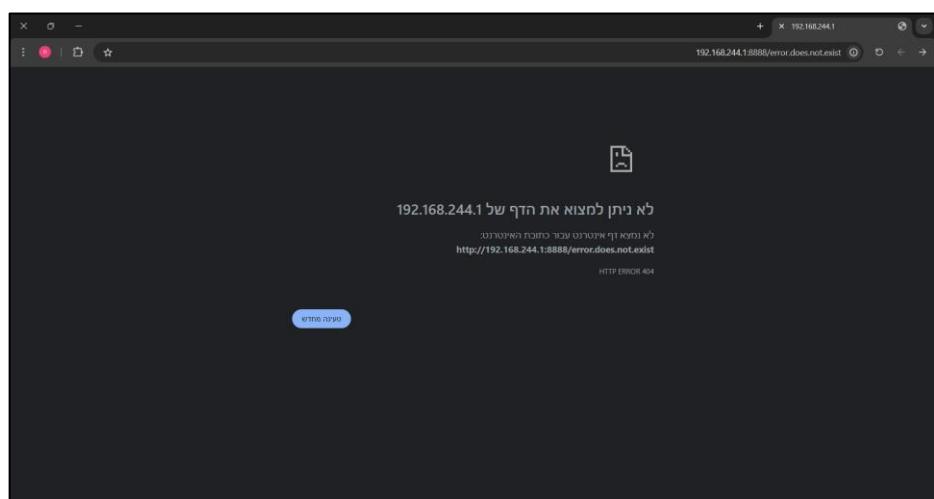
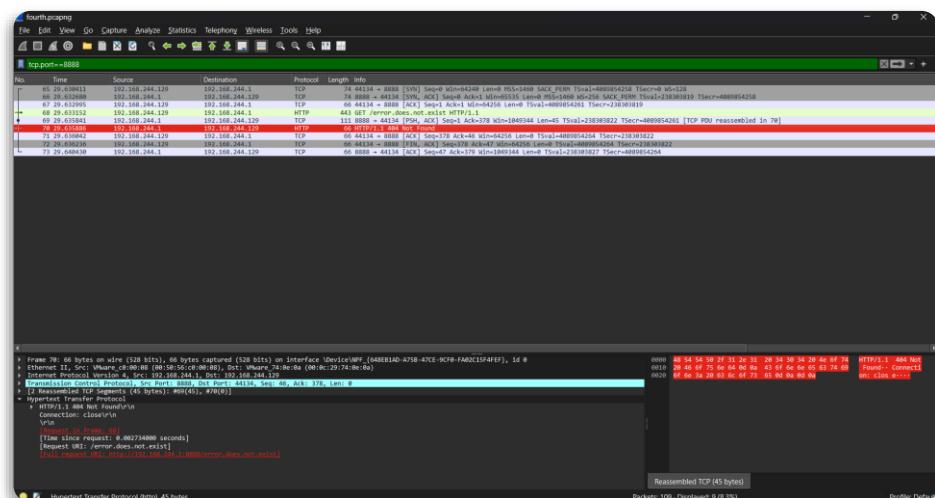
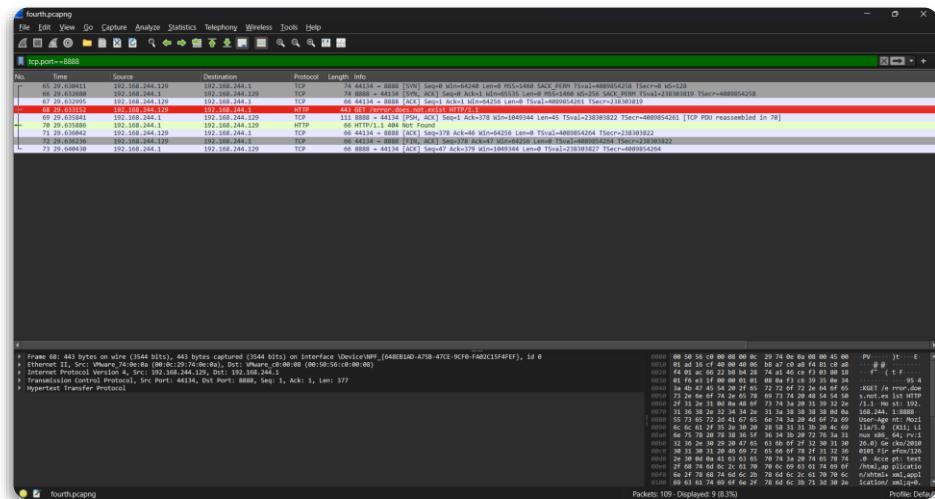
<http://192.168.244.1:8888/redirect>





הדף פונה כתע אל הנטיב /redirect של השרת שלנו. ניתן לראות שהפעם הוא פותח תחילת חיבור ייחד של פורט אחד, אשר דרכו הוא שולח את הפניה (תמונה ראשונה). השרת שולח לו תשובה חזירה של 'moved permanently', מכרך לתשובה שלו את ההפניה/הctaות של מקום החדש, והחיבור נסגר (תמונה שנייה). מיד לאחר מכן מוקם חיבור חדש מצד הדףן, אשר דרכו מתבצעת הבקשה לאוטו `location`-`location` חדש שהציגו השרת לדףן בבקשת הקודמת (תמונה שלישיית). השרת מקבל את הבקשה השנייה, מחזיר את הקובץ שבקש לדףן – `result.html`, ואחר כך שוב פעם אנו רואים את הסגירה של התקשרות בחfine שmagus משבי הכוונים. ברמת המטא-דאטה אין שוני יוצא דופן משליאינו כבר.

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

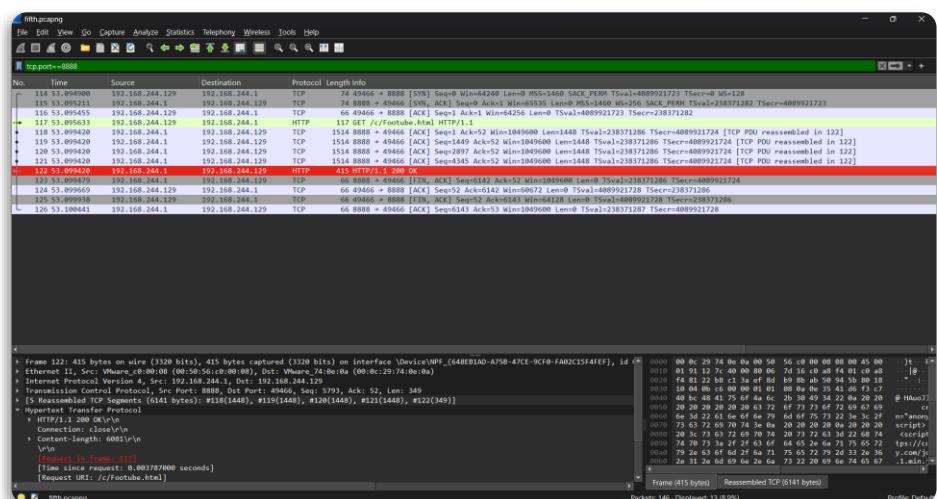
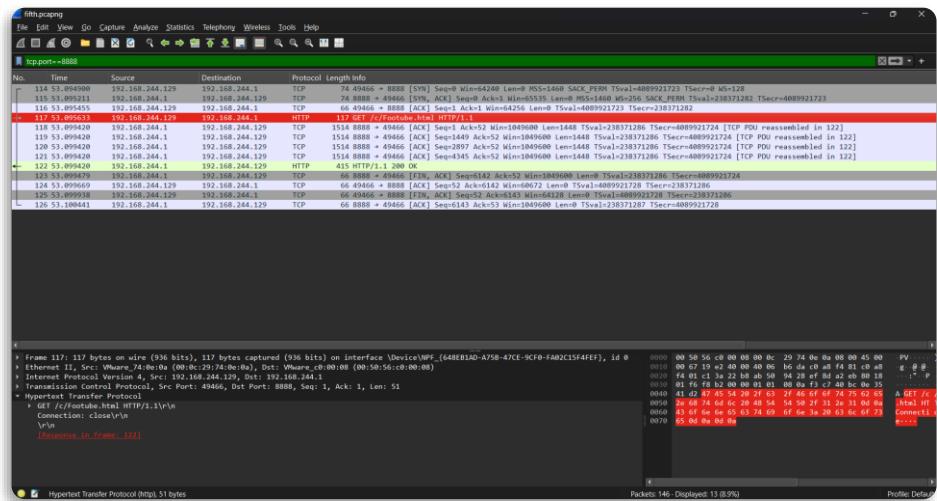


אנו רואים כתוב ניסיון של הדפסן לפנות לכתובת אשר לא קיימת מבחינת השרת שלנו. אנו רואים שנפתח שוב חיבור יחיד של פורט אחד, אשר דרכו נשלחת הבקשה. השרת אשר שולח על כך קיבל את הבקשה (תמונה ראשונה), מיד מוחזר לדפסן תשובה חזרה של שנותנת אינדיקציה לדפסן שהדף לא נמצא – '404 not found' (תמונה שנייה). לאחר

שהתשובה מוחזרת, התקשרות נסגרת בחוף. בرمת המטא-דאטה שוב אין שוני יוצא דופן  
משראיינו כבר.

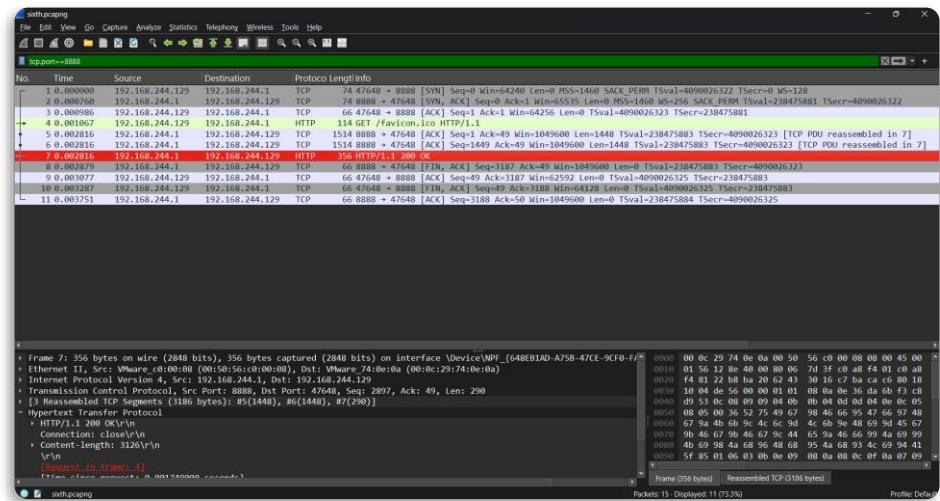
נסים בכר שנראה מס' בקשות בכריש אשר הטענו אל הרשות אך הפעם לא מהדפסן אלא  
מקוד הקליינט שאנו חנו כתבנו –

[/c/Footube.html](#)



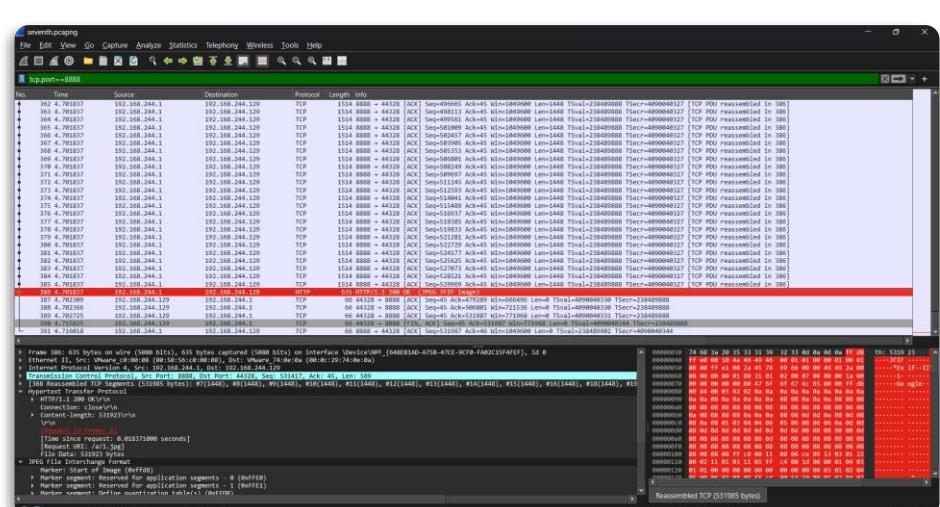
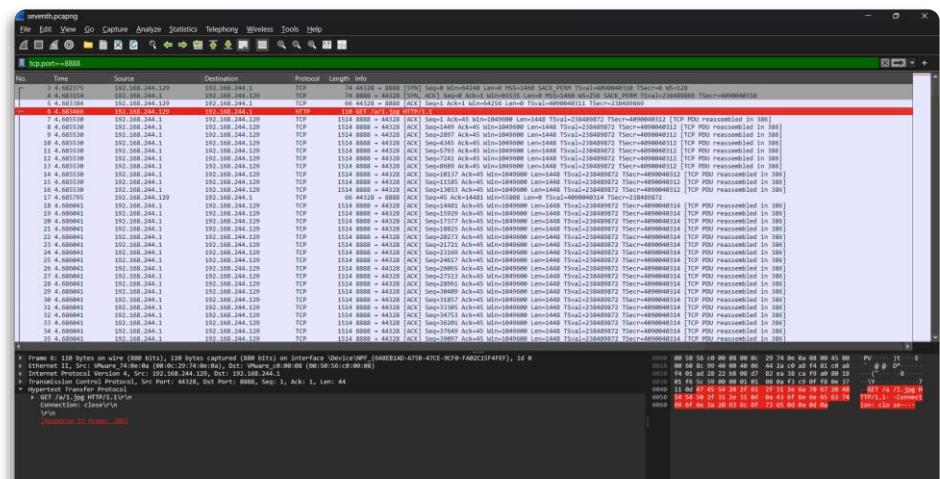
הלקוח מבקש את הקובץ [/c/Footube.html](#) אשר מכיל "עמוד אינטרנט" עם שלל תמונות וקבצים המוכלים בו, אולם כאן אנו רואים פורט יחיד אשר נפתח ע"י הלקוח ופונה לשרת, ויתרתו מזאת לאחר שמתיקל הקובץ אנו לא רואים שרשראת של בקשות בדومة למה שראינו כאשר הפניה הגיעה מהדפסן, כיון שפה הלקוח רק מבקש את הקובץ ולא מנסה להציגו, אולם לו היה מנסה היה נתקל בכל מיין קבצים ומשאים אשר אין, ובאמת הינו רואים שרשראת בקשות המנסה לייבא את כל המשאים המוחזקים באופן צזה או אחר אצל השרת, אשר אלו דרישים באופן חיווי על מנת להציג את הקובץ במלואו. לאחר שהקובץ מגיע התקשרות נסגרת בחוף הדדי. יש לציין שניותן לראות שהזאים header יותר דليل מזה שראינו בבקשת שהגינו מהדפסן, שכן שאנו חנו בקוד הקליינט כתבנו בפניה רק את – סוג הפניה, שוב הקובץ המבוקש, הפרוטוקול בו אנו משתמשים, וכן סטטוס החוסט.

/favicon.ico



דוג' פשוטה נוספת בה הלקוח מבקש את הקובץ של האיקון של הרשת (התמונה הקטנה המופיעה בראש הכרטיסיה), אנו רואים שהשרת מছזר את התוכן ב2 חבילות שונות (מפתחת גודל), 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 הודעה הסיכום של ה- ok. 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 בצד השני (מידע זה נשלח מן הסתם גם בדוג' הקודמות כאשר שלחנו תמונה בגוף ההודעה).