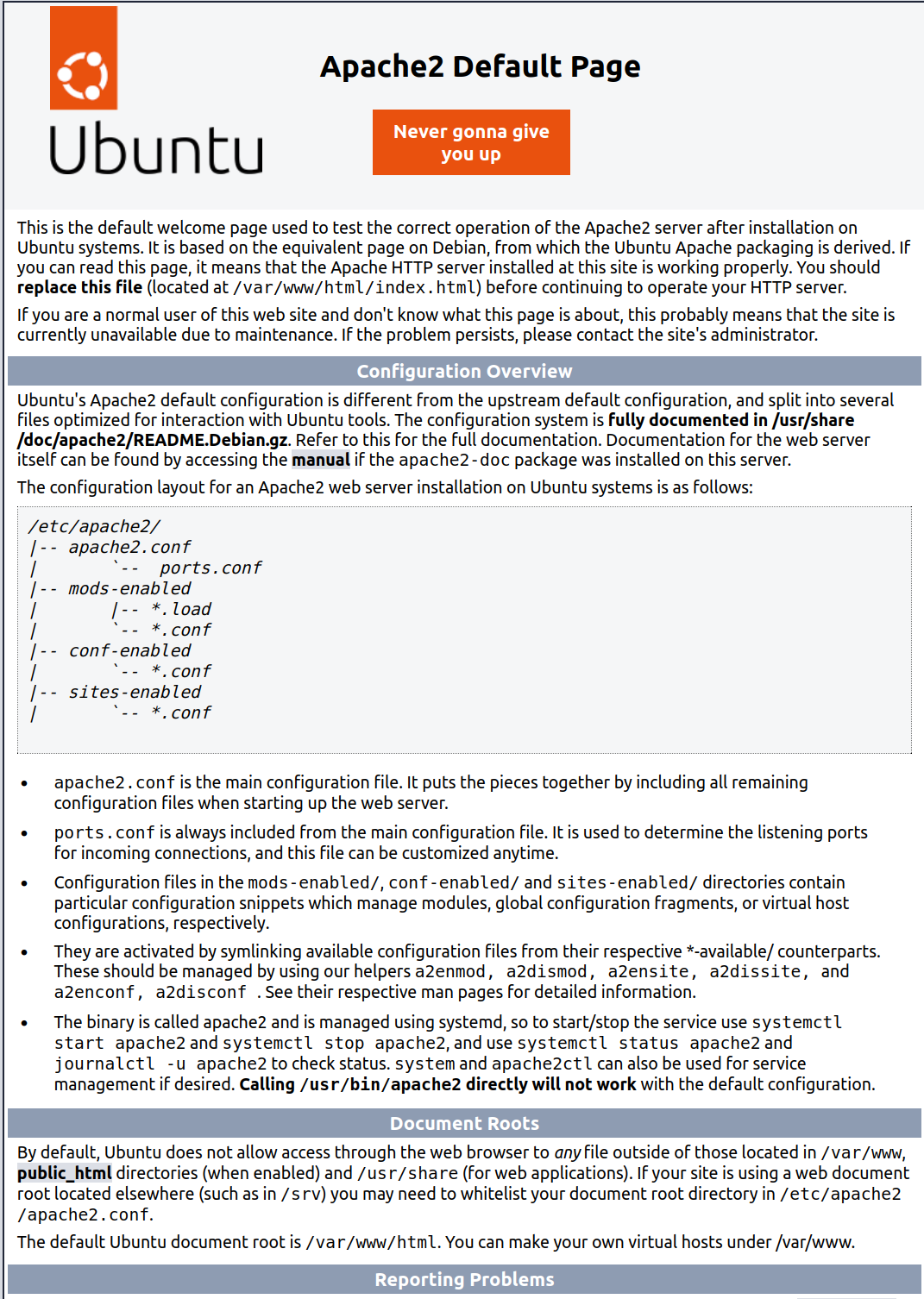
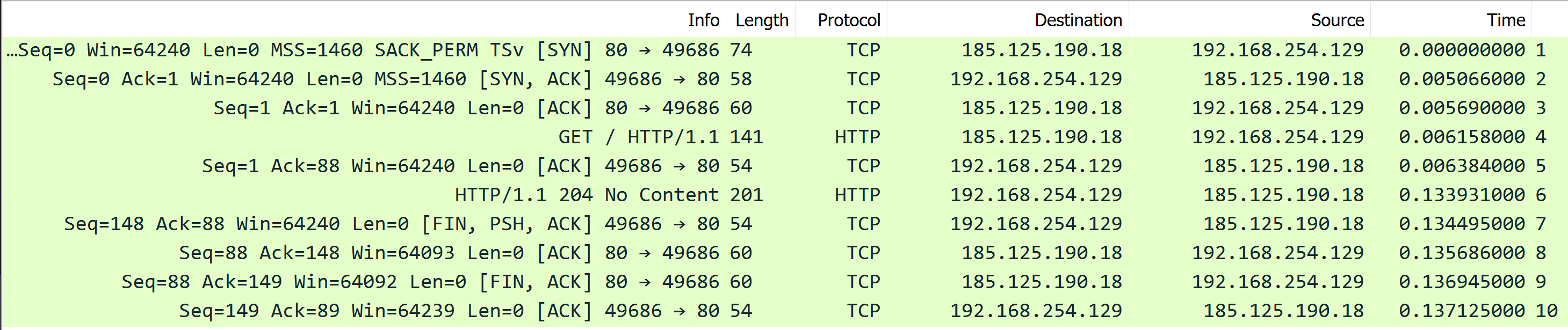
**רשתות תרגיל 3**

**חלק א:**ראשית, התקנתי את הדפדפן, פתחתי את הקובץ /var/www/index.html מהטרמינל וכתבתי בו בכותרת "Never gonna give you up" במקום ה"it works".

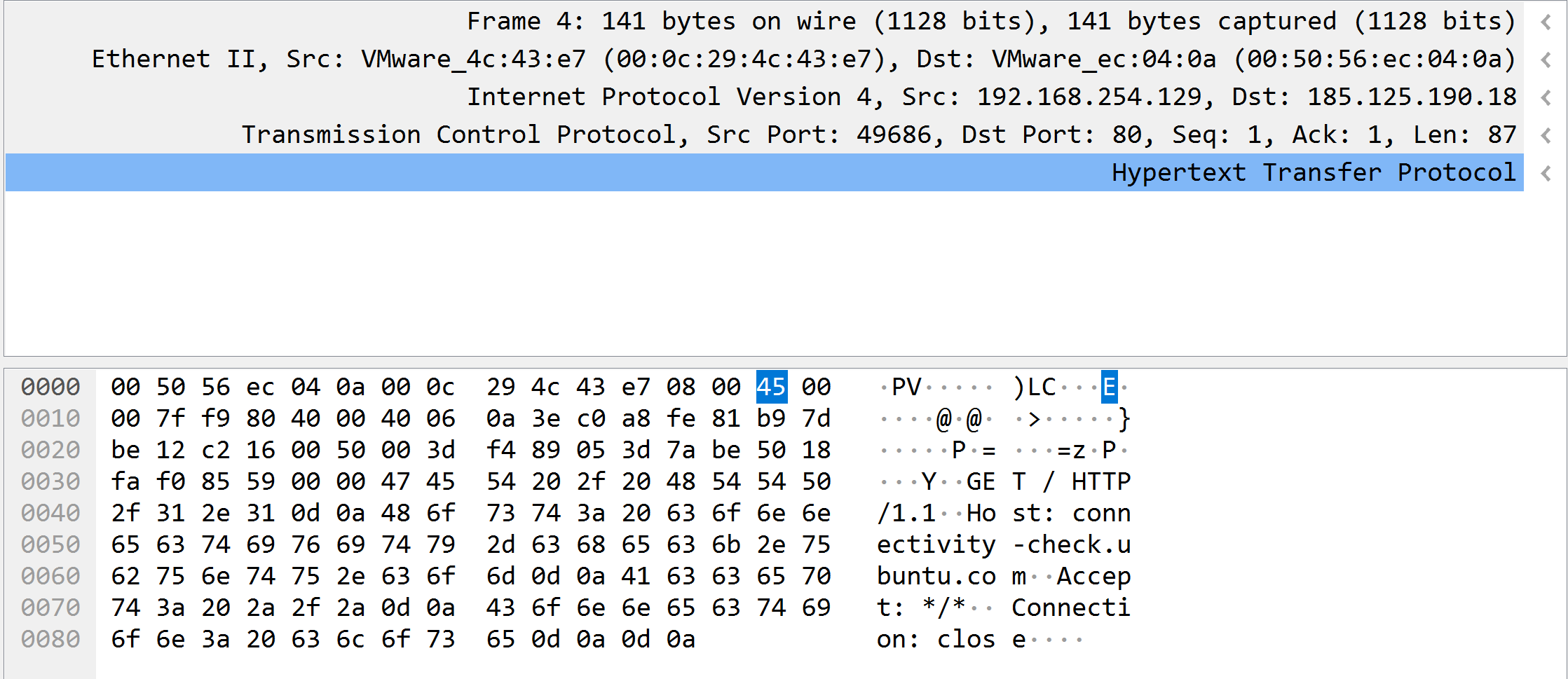
והרי התוצאה  


כעת, הרצתי מאותו המחשב לקבלת והסנפתי את התעבורה, ספיצפתי (מלשון ספציפי) עם tcp.port==80, וקיבלתי:  


(החץ שמופיע בין הפורטים מוצג אצלי הפוף כאשר ראש החץ הוא המקור וזנב החץ הוא היעד).

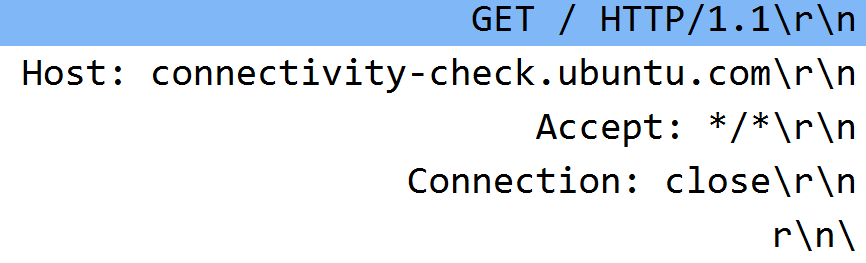
אסביר כעת את מה שקורה כאן:  
ניתן לראות שנפתח לנו חיבור אחד, שכן השרת (בעל port = 80) מתקשר כל הזמן עם port=49686 (הלקוח). (הערה: בתרגיל הזה שיניתי חזרה שהseq, ack יהיו יחסיים ולא במספרים האמיתיים שלהם משום שכבר פחות צריך להתמקד בהם).

שלוש השורות הראשונות- טקס ההיכרות הידוע- טקס לחיצת הידיים, syn, syn ack, ack- קובעים על סיקוונס נאמבר של כל אחד והשני שולח לו אישור (הלקוח מתחיל עם הודעה לשרת, השרת שולח לו אישור עם הסיקוונס נאמבר שלו והלקוח שולח לו אישור.

כאן מגיע תורו של פרוטוקול הHTTP עם ההודעה הבאה (שורה 4):  


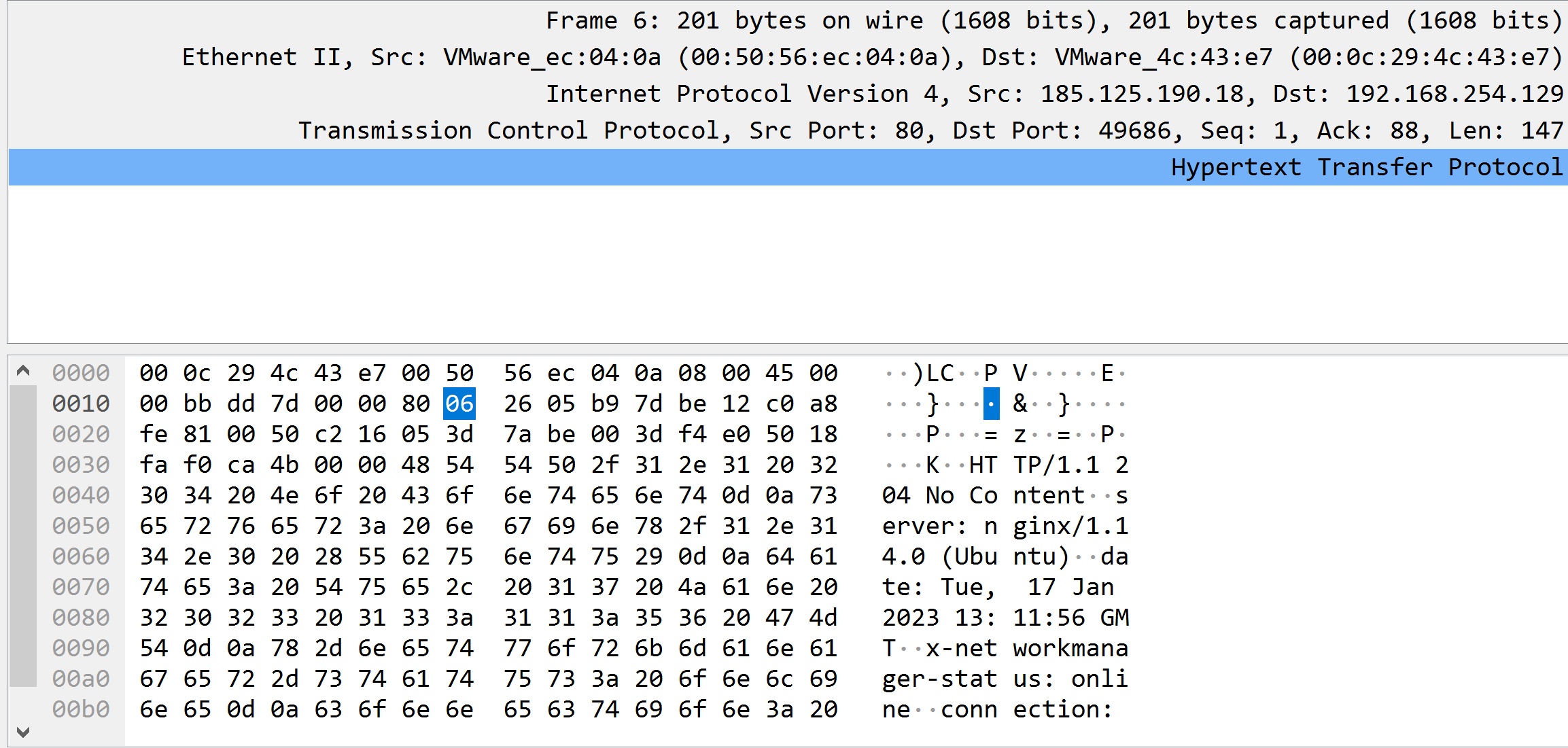
ניתן לראות שהיעד הוא השרת, והמקור הוא הלקוח (לפי הפורטים והIPים),

תוכן ההודעה עצמה:

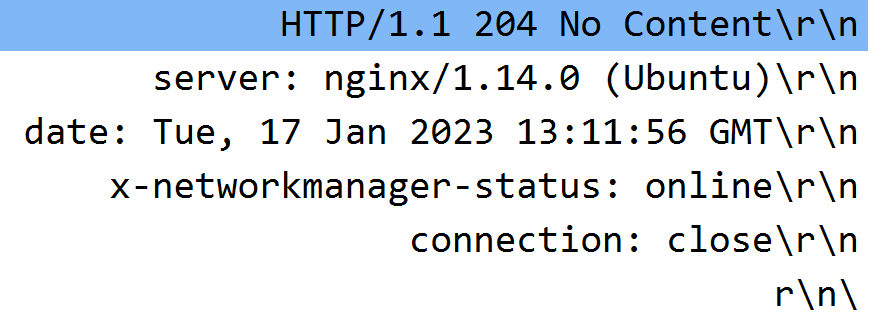


"GET /"- כאשר הבקשה ריקה ורק מקלידים localhost ומחפשים את הקובץ index.html, אותו שיניתי ולכן הוא נפתח לנו.

ו"Connection: close" בשביל לסגור את החיבור עם השרת לאחר פתיחת הקובץ, וסיום עם שתי שורות חדשות לציון סיום ההודעה כידוע לנו (על בשרינו) מהתרגיל הקודם.

בשורה הבאה זה בפרוטוקול הTCP שוב להחזרת האישור על קבלת ההודעה ומיד לאחריה שוב בפרוטוקול הHTTP מהשרת ללקוח בחזרה עם ההודעה הבאה:

כאן אפשר לראות שהשרת הוא המקור והלקוח הוא היעד, ותוכן ההודעה זה:

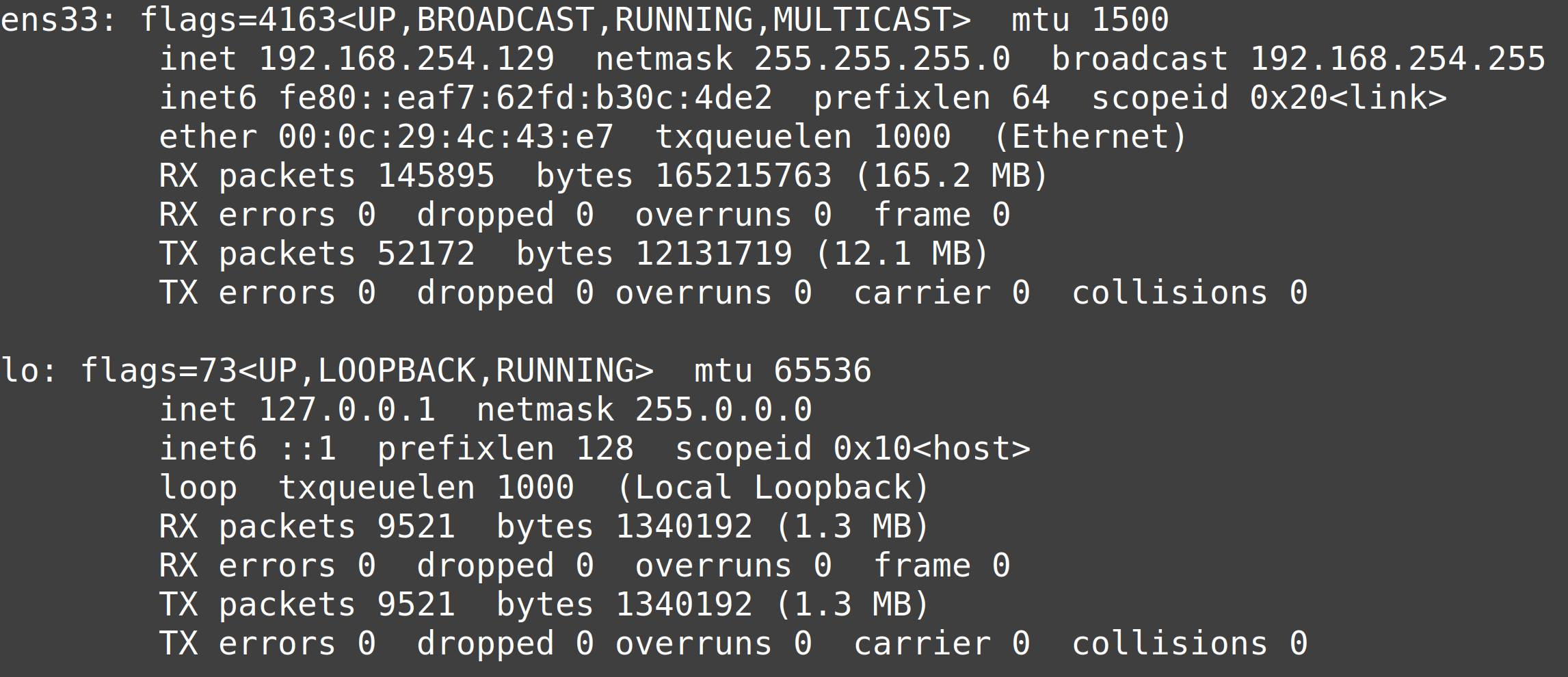


אין תוכן כביכול בהודעה הראשונה ואין תוכן קובץ שנשלח (והרי ידוע לנו מהתרגיל הקודם שאמור להיות), אז איך בכל זאת זה עובד?

שעשיתי ריפרש וזה הופיע, הדפדפן ידע שכבר יש לו את זה שמור לכן הוא ניגש להיסטוריה, והתאריך זה הפעם הקודמת שהוא פתח את זה ולכן הוא לא צריך את תוכן הקובץ משום שזה כבר שמור אצלו.

ומיד לאחר מכן בפרוטוקול הTCP השרת שולח FIN (עם בקשה להיפרד), הלקוח בתגובה שולח ראשית ACK על ההודעה עצמה, ואז ACK על הבקשה לסיום עם FIN שגם הוא מסיים, והשרת מסיים את הכל עם ACK על הFIN.

(הסיום אכן קורה כראוי משום שהconnection mode בהודעות בפרוטוקול הHTTP הוא אכן close).

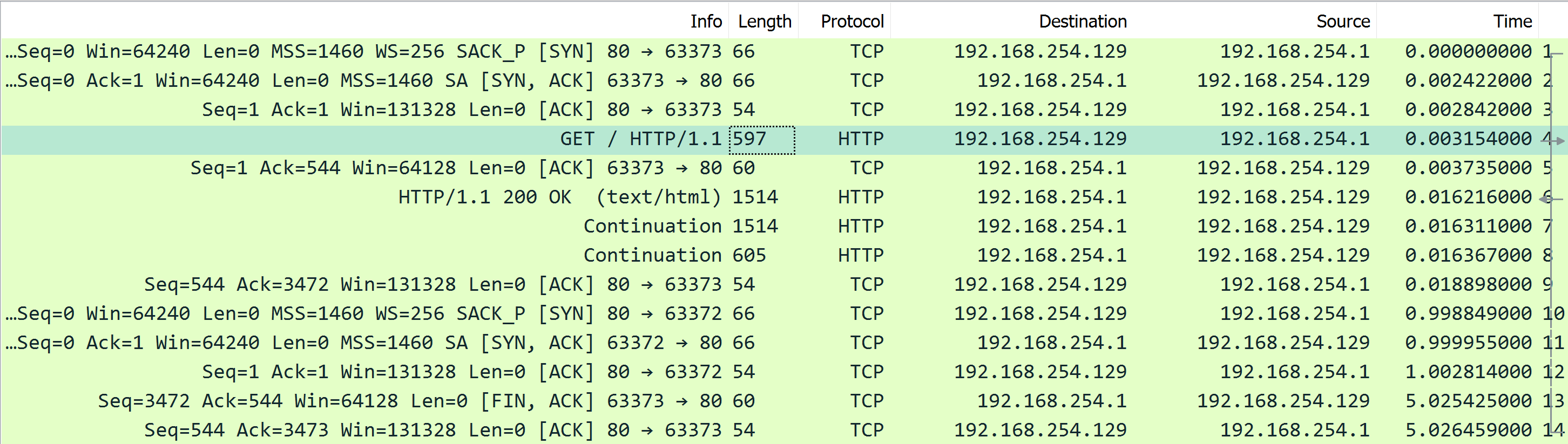
כעת עם ifconfig נקבל מידע:  


(שם כרטיס: ens33, IP: 192.168.254.129)

לכן נריץ,



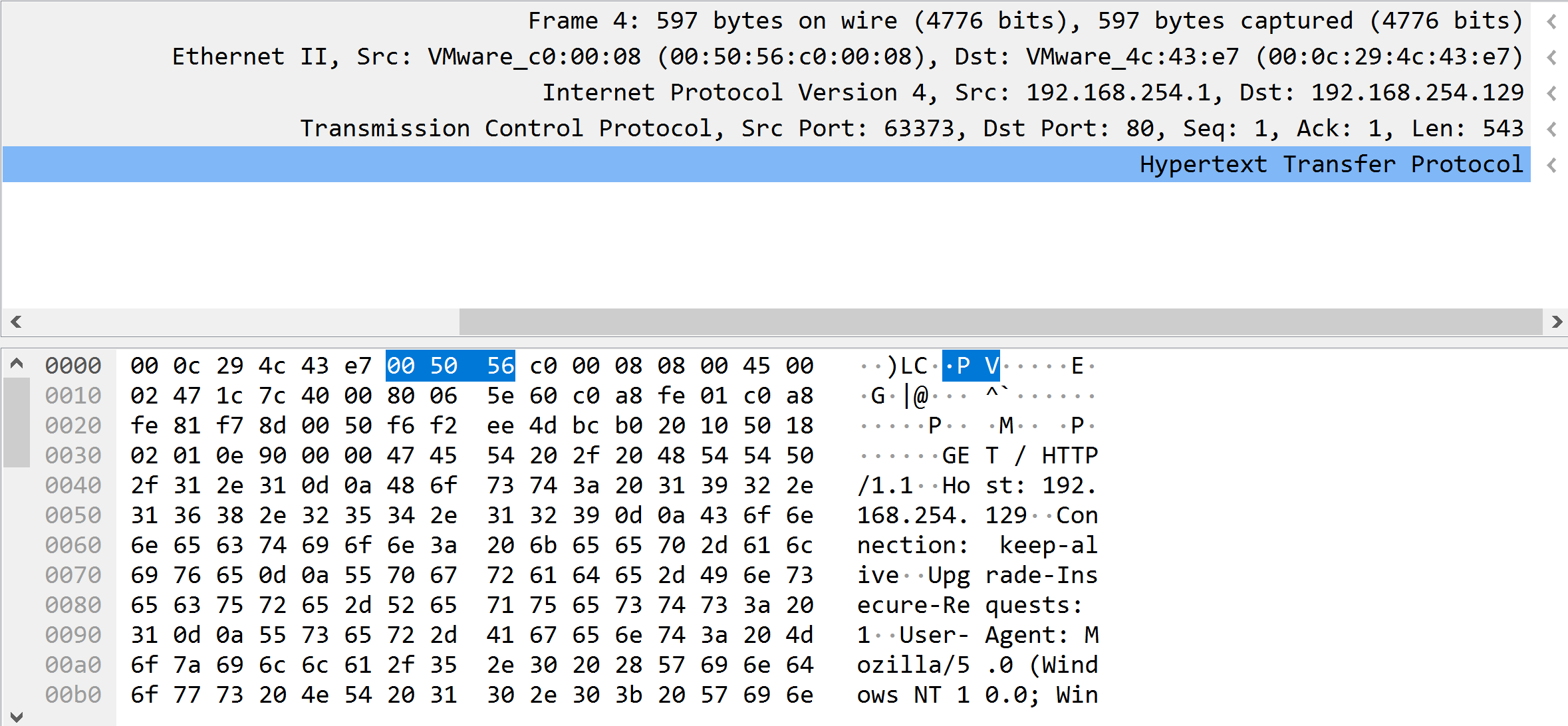
על מנת לבטל את מנגנון הsegmentation offload.

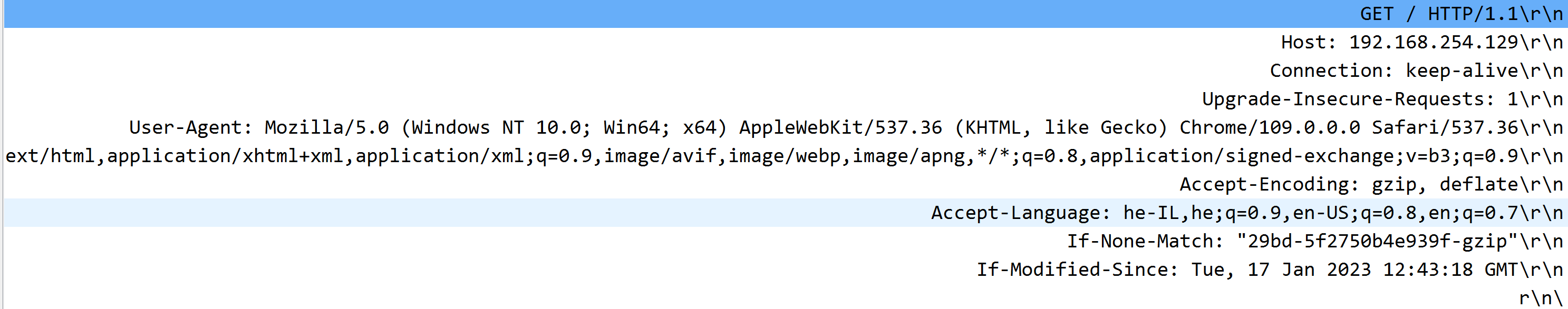
ועכשיו, נתחבר לדפדפן מהמחשב השני ונרשום את כתובת הip 192.168.254.129 על מנת לקבל את קובץ הindex.html מהמחשב השני, ואכן הוא רץ לנו. נסניף (כמובן) את התעבורה בוויירשארק ונספצף (מלשון ספציפי, שוב) לקבלת:  


(IP לקוח (דפדפן המבקש) 192.168.254.1, IP שרת (המחשב עם הindex.html) 192.168.254.129, כמוזכר לעיל, port לקוח: 63373, port שרת:80, שוב. וport לקוח: 63373)

שוב. שלוש השורות הראשונות לטקס ההיכרות הידוע.

שורה ארבע, פרוטוקול HTTP עם תוכן הבקשה מהדפדפן-

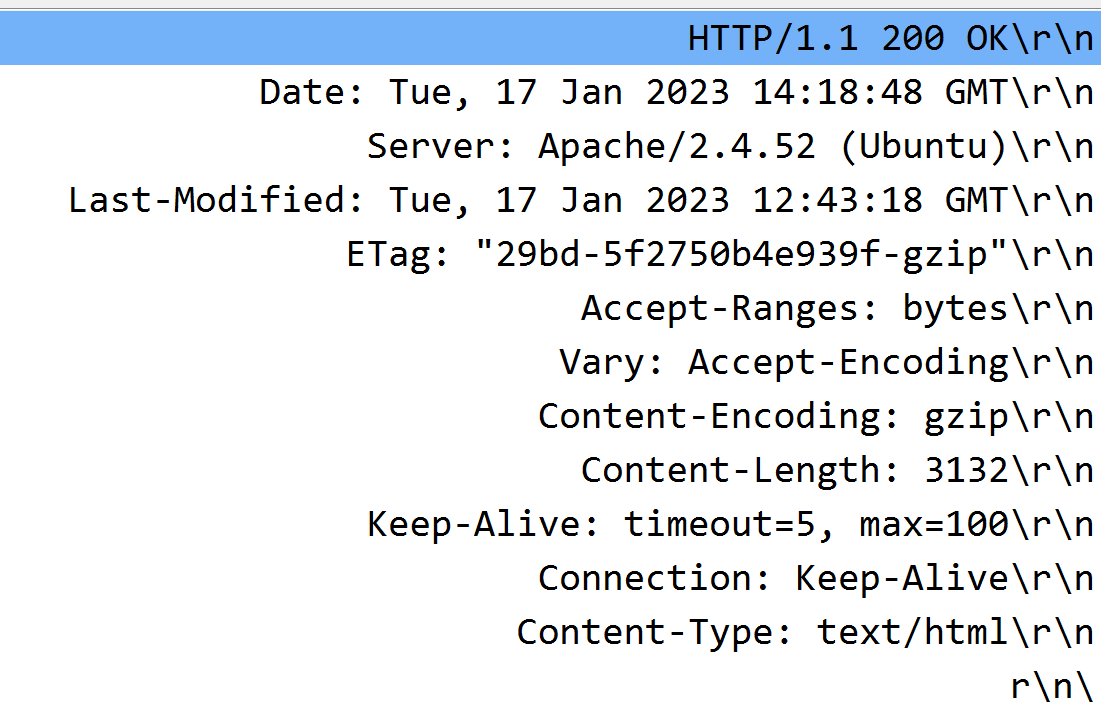


כאשר התוכן עצמו- 

שוב- get / לבקשה עם path= '/' וזה מתפרש ל"index.html", והפעם הconnection mode- keep alive, (ושוב בסיום ההודעה שתי שורות חדשות ריקות).

חבילה הבאה (5) עם אישור ACK מהשרת ללקוח, ובתגובה השרת שולח ללקוח שלוש חבילות (בהמשכים) עם תוכן התשובה בפרוטוקול HTTP-

ההתחלה, המופיעה בחבילה הראשונה מבין השלוש, אותה אנחנו עוד יכולים להבין-



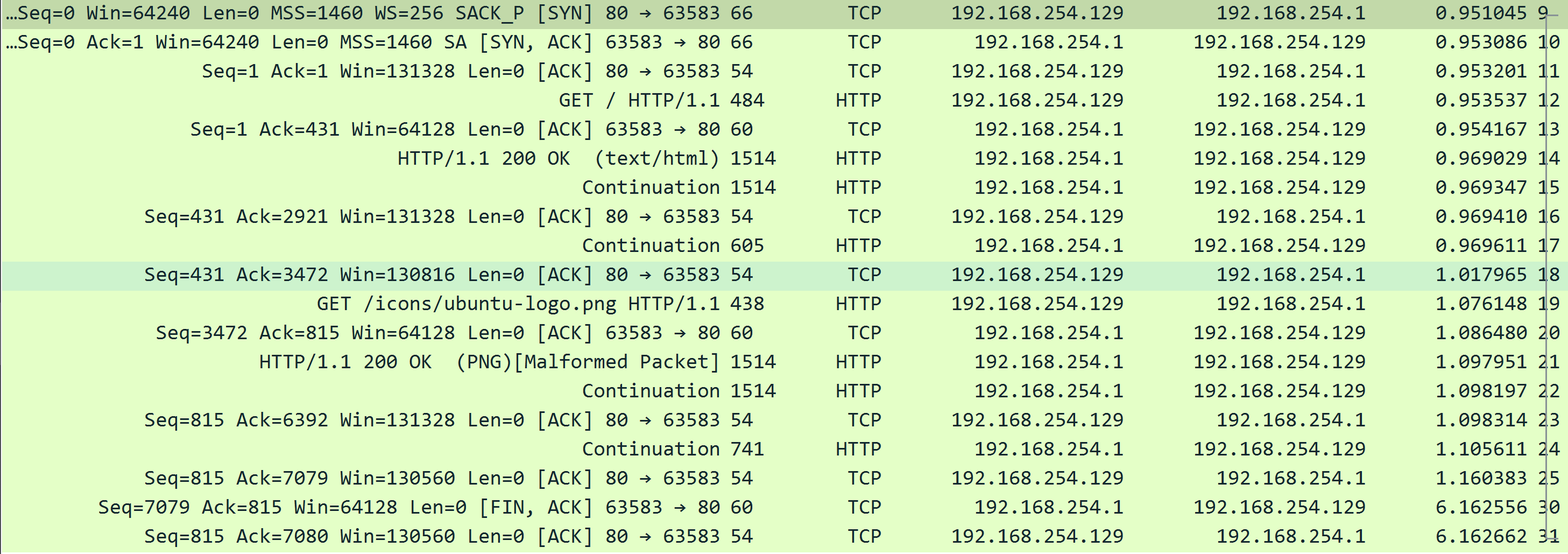
התוכן פותח באישור לבקשה עם "OK", connection: Keep-Alive עם timeout=5 לKeep-Alive (עליו נדבר בהמשך) וContent-Length: 3132, שזה אורך הקובץ.

ושאר התוכן הוא הביטים של הקובץ index.html אותו שולחים לדפדפן שיציג.

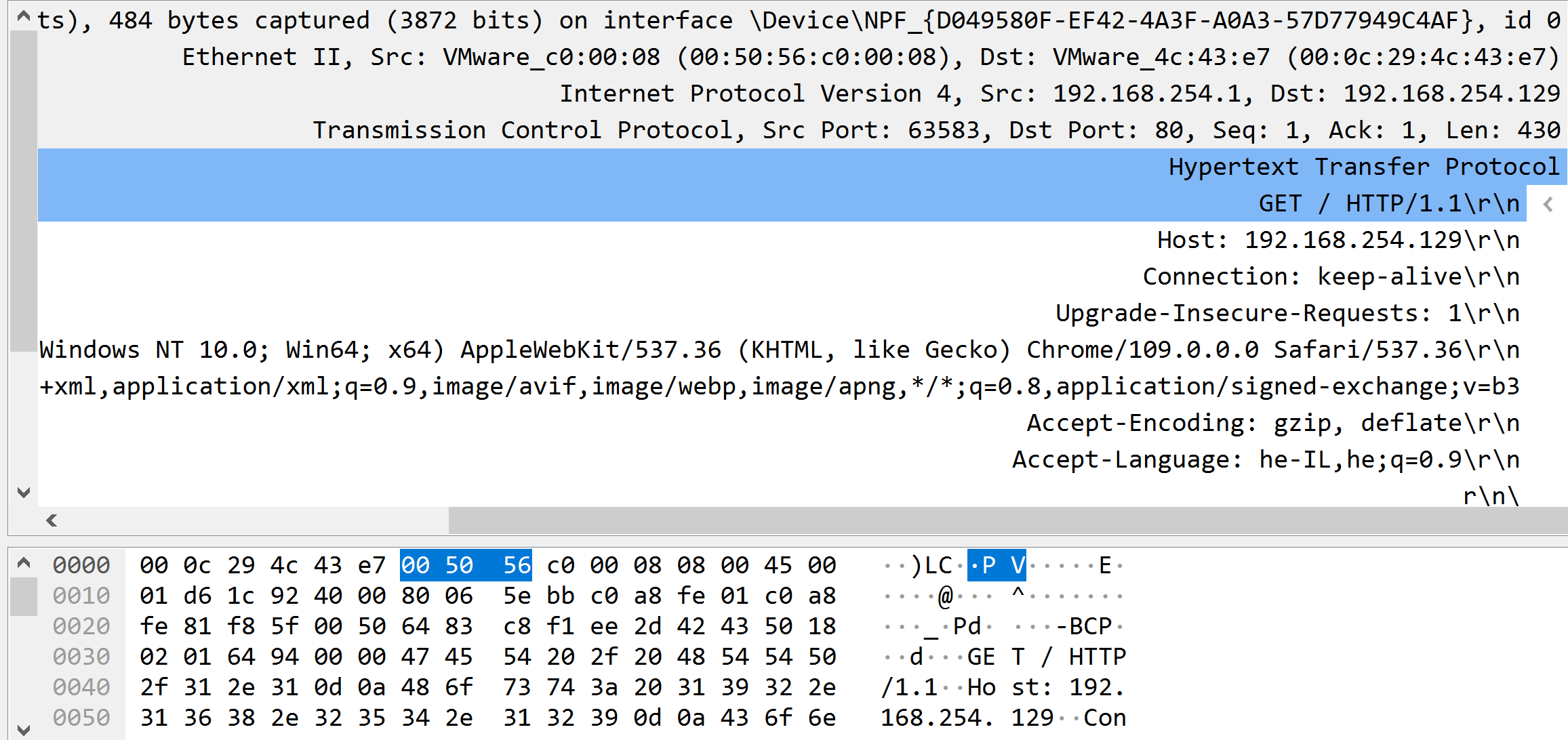
נספרתי את אורך הטקסט שהוא לא ביטים של הקובץ שנשלח בכל חבילה, בראשון יש 393, בשני 54, ובשלישי 54, סהכ 501 בתים שנשלחו שהם לא הקובץ מקודד, וניתן לראות שאכן סכום האורכים של החבילות (3633) פחות סכום הטקסט שהוא לא הקובץ (501) שווה לcontent length המופיע בהתחלה (3132), כידוע מהתרגיל הקודם, שמה שמופיע בcontent length זה אורך הקובץ המקודד.

כאן יש לנו אישור מהלקוח לשרת שהוא קיבל, ונפתח לנו חיבור נוסף עם השרת, עם טקס ההיכרות.

בשורה 13 ניתן לראות שהשרת שולח בקשה לסיום החיבור, ונסתכל על הזמן ונראה שזה קורה ב5.025, כחמש שניות אחרי שליחת החבילה הראשונה מהשרת של הHTTP עם הtimeout=5 לKeep-alive כפי שראינו קודם, ואכן אחרי 5 שניות השרת מבקש לסגור את החיבור עם FIN המעיד שאין לו עוד מה לשלוח אך הוא עדיין מקשיב (הלקוח רק שולח אישור על קבלת ההודעה אך החיבור עדיין לא נסגר).

נריץ מגלישה בסתר לקבלת:  


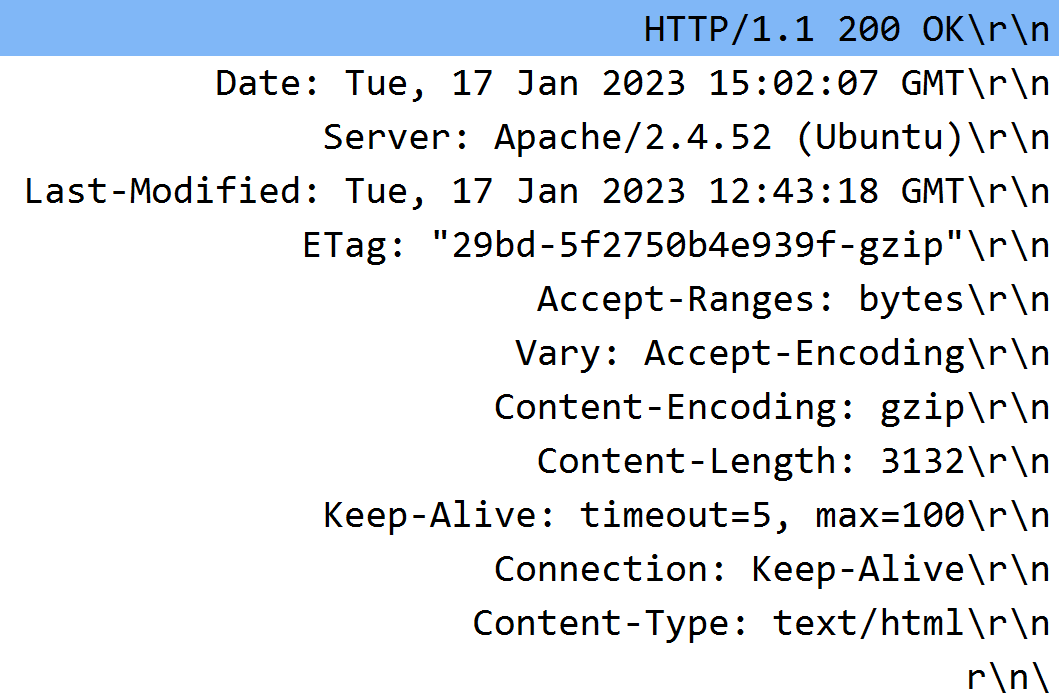
(IP לקוח (דפדפן המבקש) 192.168.254.1, IP שרת (המחשב עם הindex.html) 192.168.254.129, כמוזכר לעיל, port לקוח: 63373, port שרת:80, שוב. וport לקוח: 63583)

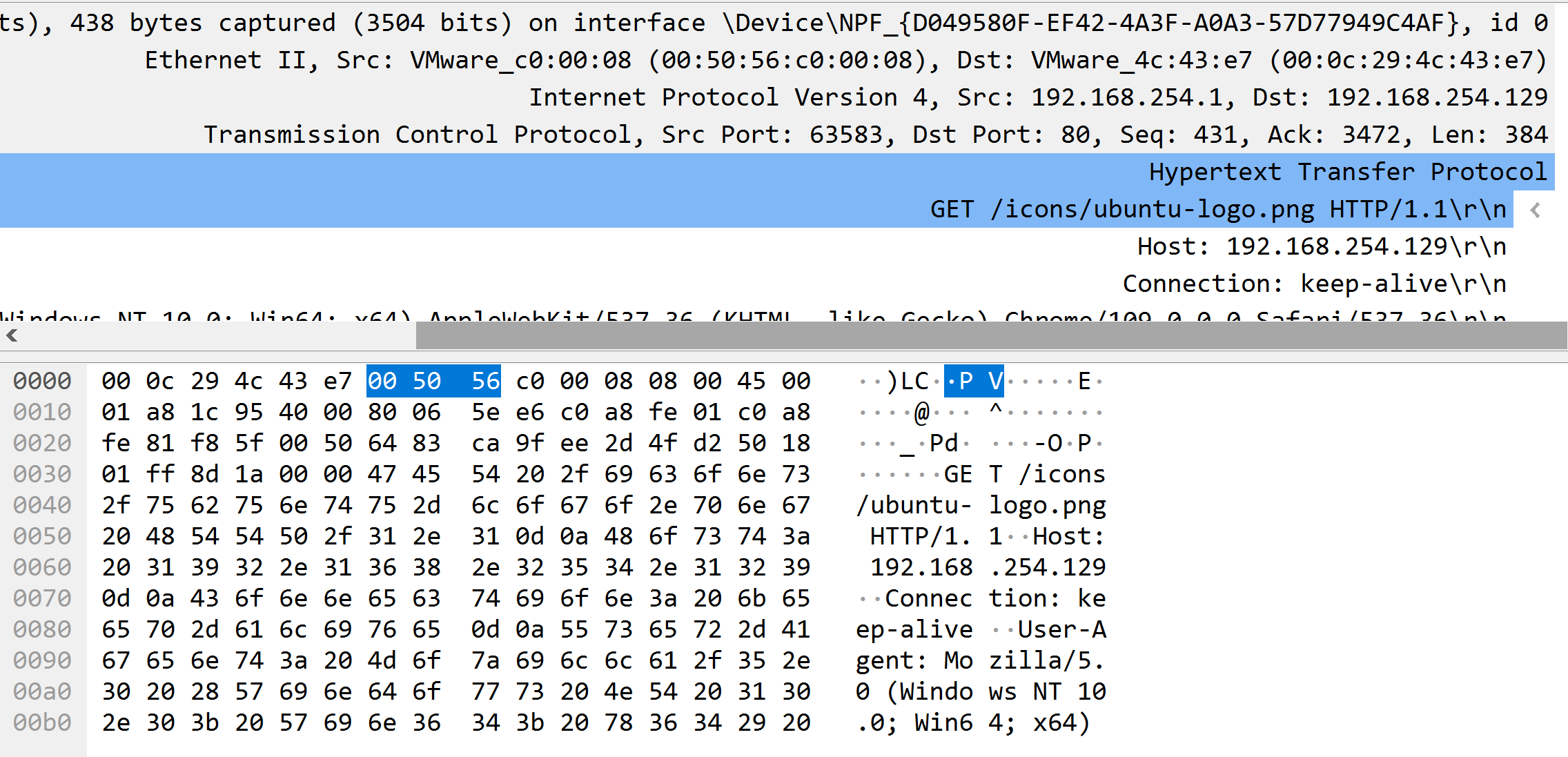
בשורה הרביעית החבילה המוכרת לנו:  


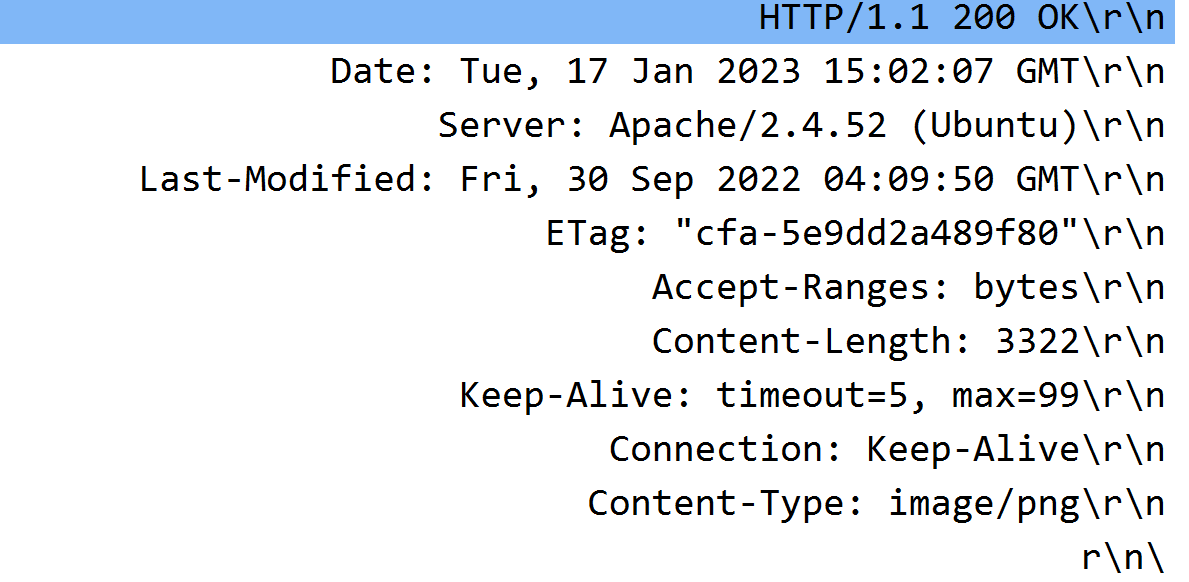
עם connection: keep-alive כמו בהרצה הקודמת.

חבילה לאחר מכן עם אישור על קבלת הבקשה, ומיד לאחר מכן רצף של שלוש חבילות עם תוכן הקובץ (רק הפעם עם עוד אישור קבלה באמצע, בשורה 18, לא באמת משנה לפעמים זה ככה ולפעמים ככה)

והפעם אראה שוב את תוכן החבילה הראשונה לפני הקובץ המקודד-



שזה כמו קודם, ולאחר שהלקוח מקבל את הקובץ index.html הדפדפן שולח עוד בקשה בשורה 19:  


"GET /icons/ubuntu/logo.png"- מבקש את הקובץ בנתיב שליד הGET ניתן לנחש שזה הלוגו של אובונטו וכנראה שלאחר קריאה של הindex.html הדפדפן ראה במהלך הביצוע שצריך את הקובץ הזה ולכן מבקש אותו מהשרת, השרת בתגובה שוב לאורך שלוש חבילות שולח את הקובץ logo.png עם התוכן הזה בחבילה הראשונה:  


ניתן לראות שכאן הcontent type הוא image/png של תמונה (בניגוד לקודמים שהיו html/text).

ושוב הkeep-alive:timeout=5 שישפיע בשורה 30 שלאחר חמש שניות השרת ישלח FIN.

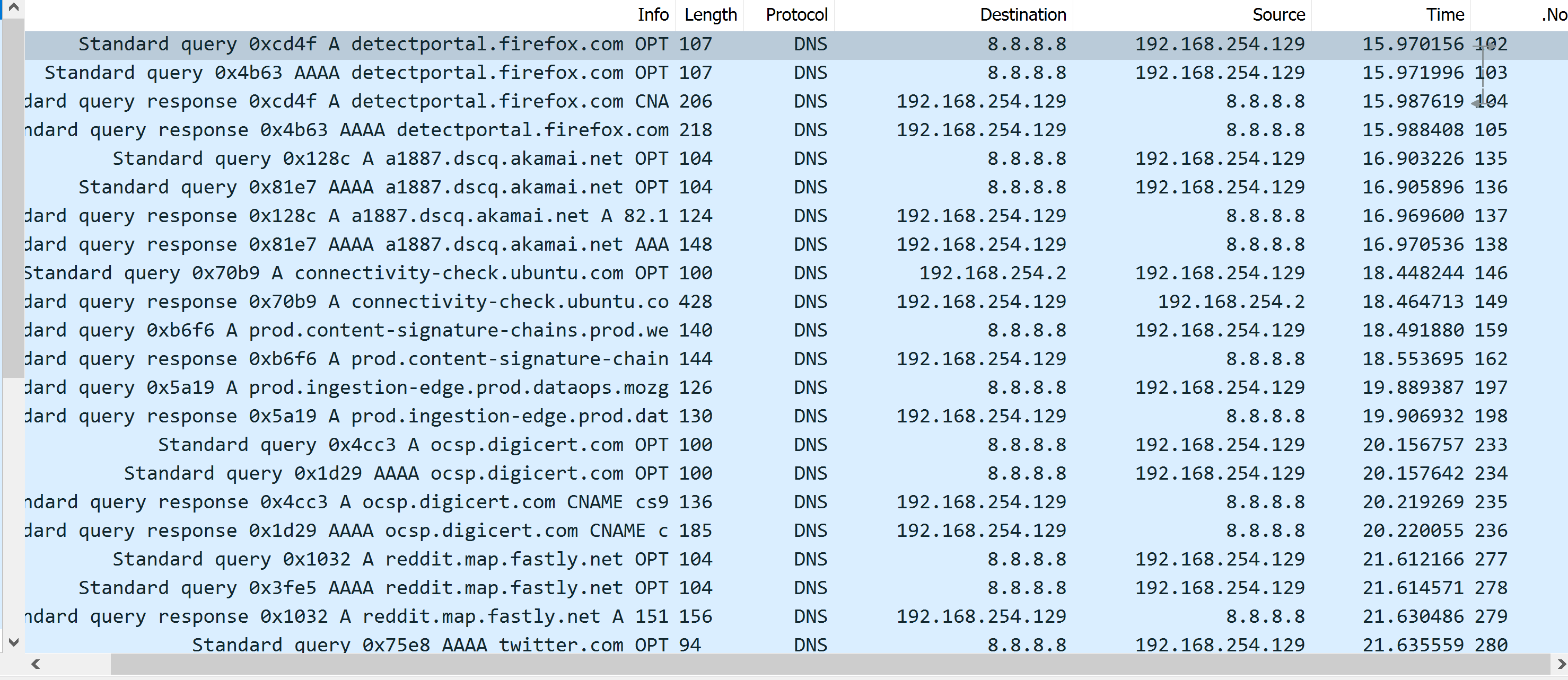
ההבדל בין כאן לפעמים הקודמות, הוא שבגלל שגלשנו מגלישה בסתר, הדפדפן לא מקושר לפעמים הקודמות ולכן התמונה לא שמורה אצלו (בניגוד לדפדפן למעלה, אצלו כבר הרצתי את הבקשה קודם לכן וכבר היה לו את התמונה אז הוא לא היה צריך לבקש אותה) ולכן הוא ביקש אותה.

כל זה קרה בחיבור אחד ולאחר אישורים על קבלות נגיע לשורה 30 עליה הסברתי ובשורה 31 הלקוח מחזיר אישור על קבלת החבילה עם הFIN אך עדיין לא מחזיר FIN משום שהדפדפן מבחינתו בconnection: Keep-Alive.

***חלק ב:****לאחר הורדת שרת* DNS *למחשב, ושינוי הforwarders לשרתי הDNS של גוגל (8.8.8.8, 8.8.4.4) ושינוי הIP של שרת הDNS לשרת שלנו, פשוט פתחתי דפדפן חדש של פיירפוקס, והסנפתי את התעבורה.*

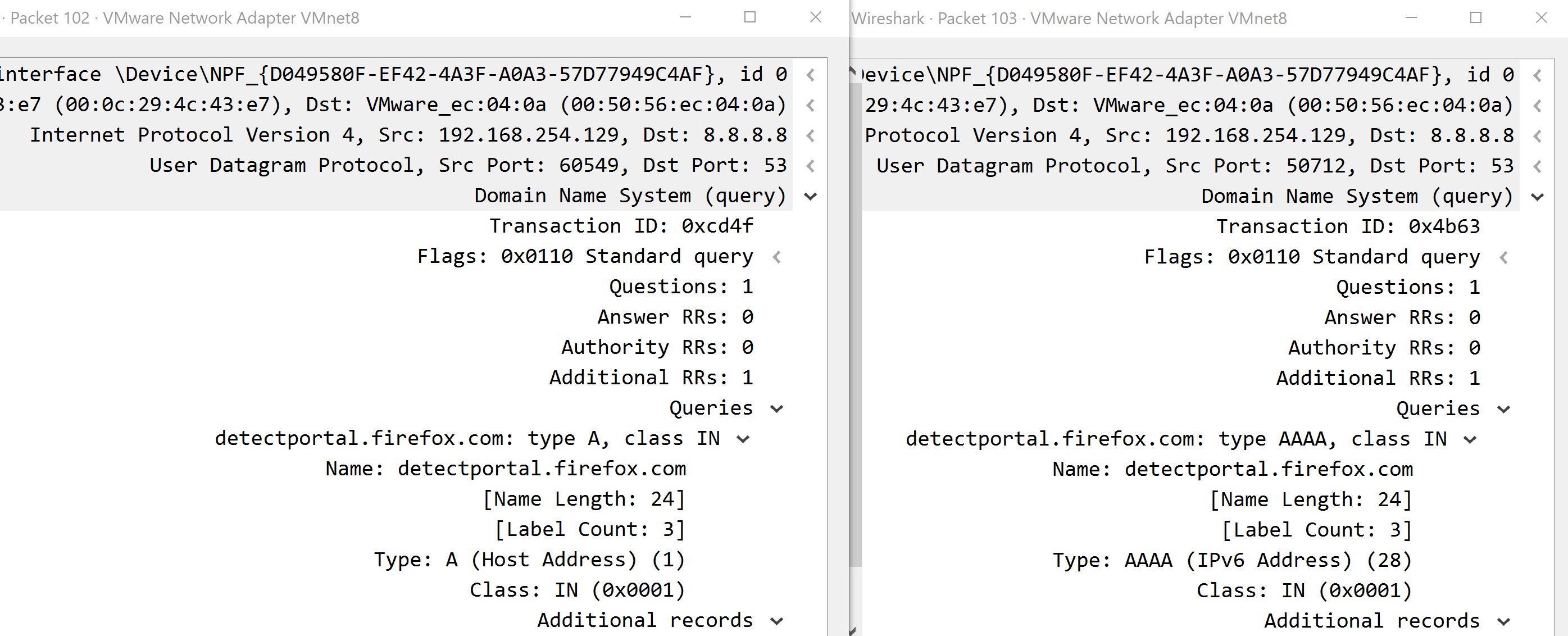
*היו הרבה חבילות, העיקריות היו בתקשורת בתעבורות TCP,DNS וNTP אך בתמונה כאן ובקובץ הpcap אציג רק את החבילות מסוג DNS וכשיהיה צורך ארחיב במילים על תעבורת הTCP.*

*אז לאחר ספיצוף:*

**

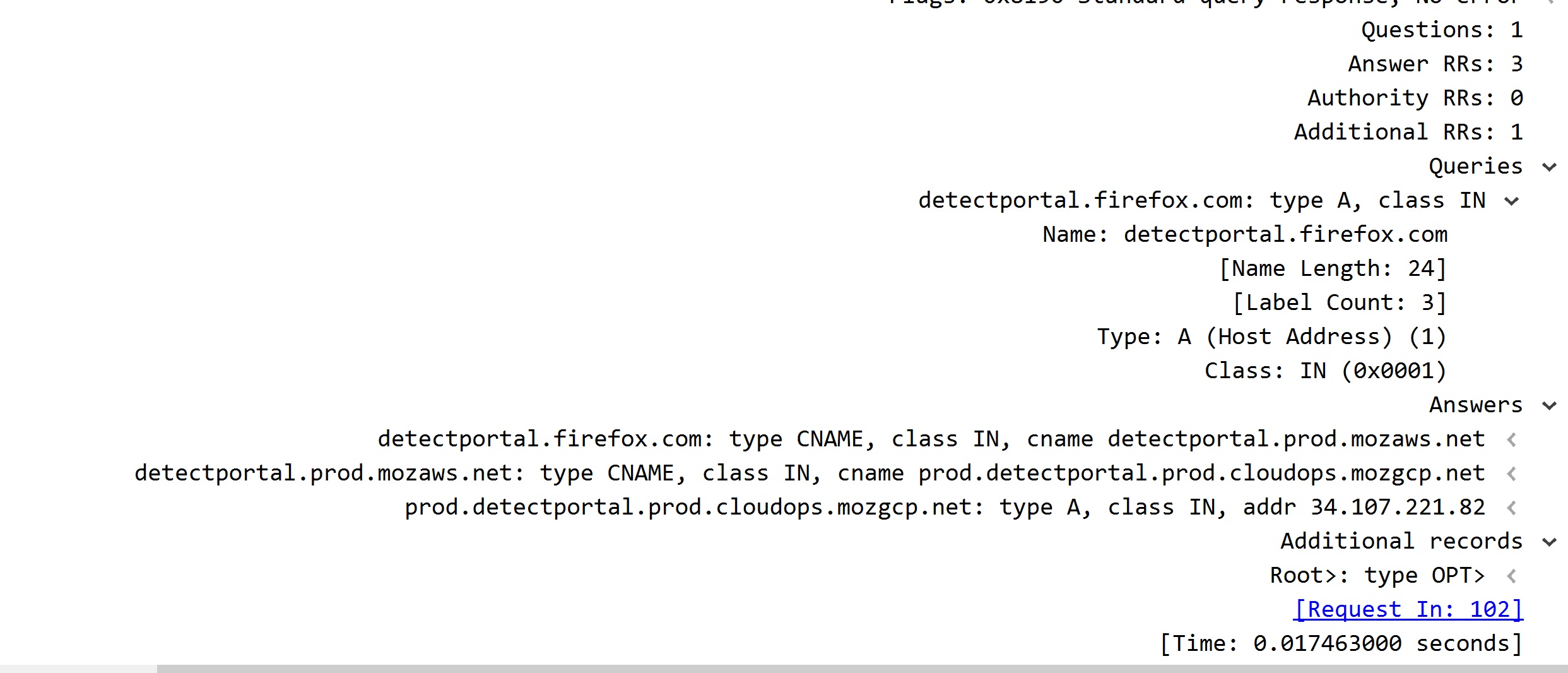
*בתעבורת הTCP שקורית קודם, הדפדפן פותח קשר עם השרת (בעל הIP של המחשב שלנו) ושולח לו בקשות לכתובות של אתרים מסוימים (על החלק הזה הסברתי בחלק א לכן כאן זה בקצרה) וכאן מגיע החלק שלנו:*

*שרת הDNS שלנו (בעל ip=192.168.254.129, כשל המחשב כולו) שולח לשרת של גוגל כפי שהגדרנו לו (עם Ip=8.8.8.8) ומבקש ממנו כל מיני כתובות לפי הדומיינים שלהם (כדוגמת השורה הראשונה בתמונה, חבילה 102, "detectportal.firefox.com") והשרת בתשובה מחזיר לו את כתובת הIP שלהם כפי שראינו בהרצאה.*

*אדגים כאן על הדו שיח הראשון:  
בקשת השרת המקומי:  
*

*שתי החבילות הראשונות (הראשונה משמאל והשניה מימין), ניתן לראות שבראשונה השרת מבקש בtype A ובשניה בtype AAAA (IPv6), והשרת מנסה את שתי הגרסאות משום שעדיין קיימים דומיינים כאלה ודומיינים כאלה.*

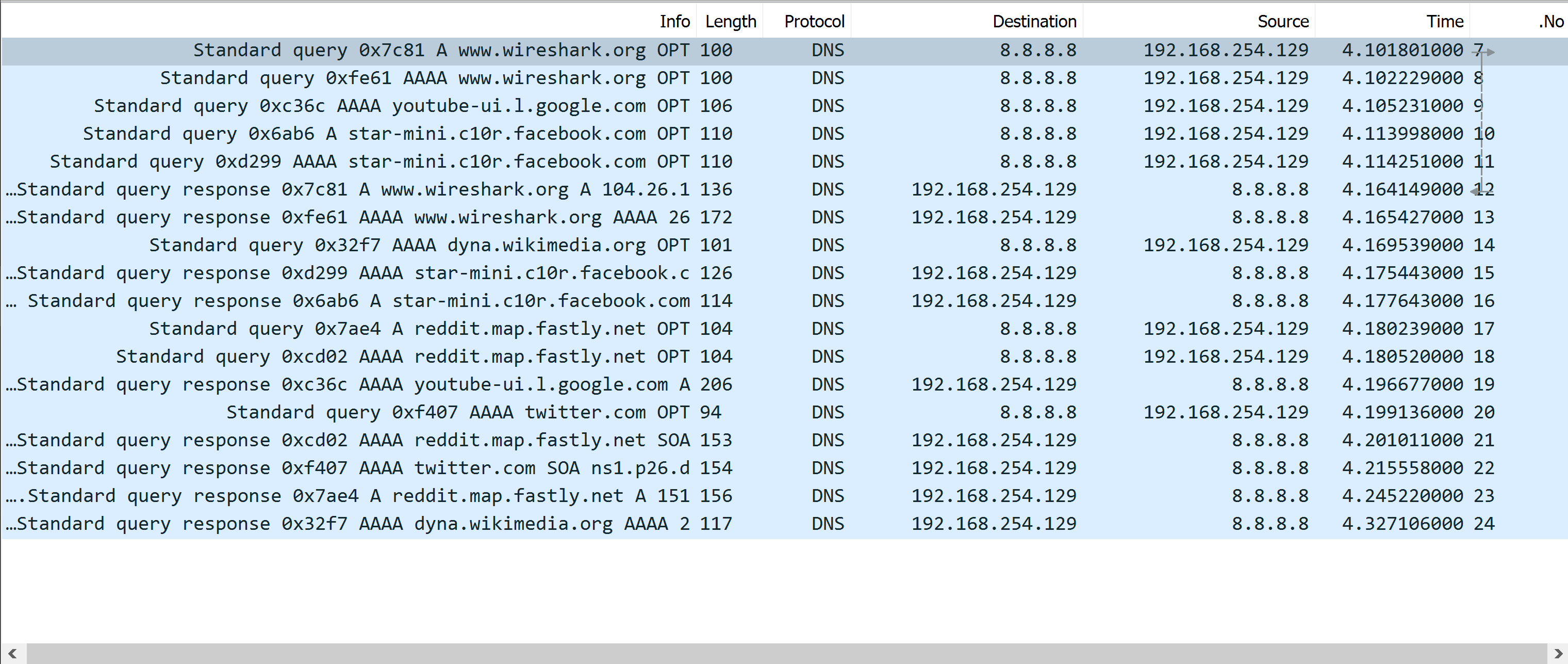
*והשדות המוכרים לנו מההרצאה והתרגול: Class: IN וName: detectportal.firefox.com (כדומיין).*

*ותשובת השרת של גוגל (לאחר חיפוש אצלו לפי שיטת הroot שראינו בהרצאה):  
השרת שולח לו שתי תשובות (בשתי החבילות 104-105, כאשר הראשונה עם IPv4 השניה עם IPv6, אראה כאן את הראשונה בעלת IPv4- חבילה 104:  
*

*ובאמת כידוע לנו, השרת מחזיר עם התשובה את הבקשה בQueries ואז עם תשובות בAnswers בצורה הבאה:  
כל פעם השרת מפנה אותו לכתובת הבאה אך גם בודק לו אותה עד שמגיע לip ובמקרה שלנו מסוג IPv4:*

*detectportal.firefox.com (המבוקש מהשרת) -> detectportal.prod.mozaws.net -> prod.detectportal.prod.cloudops.mozgcp.net = 34.107.221.82*

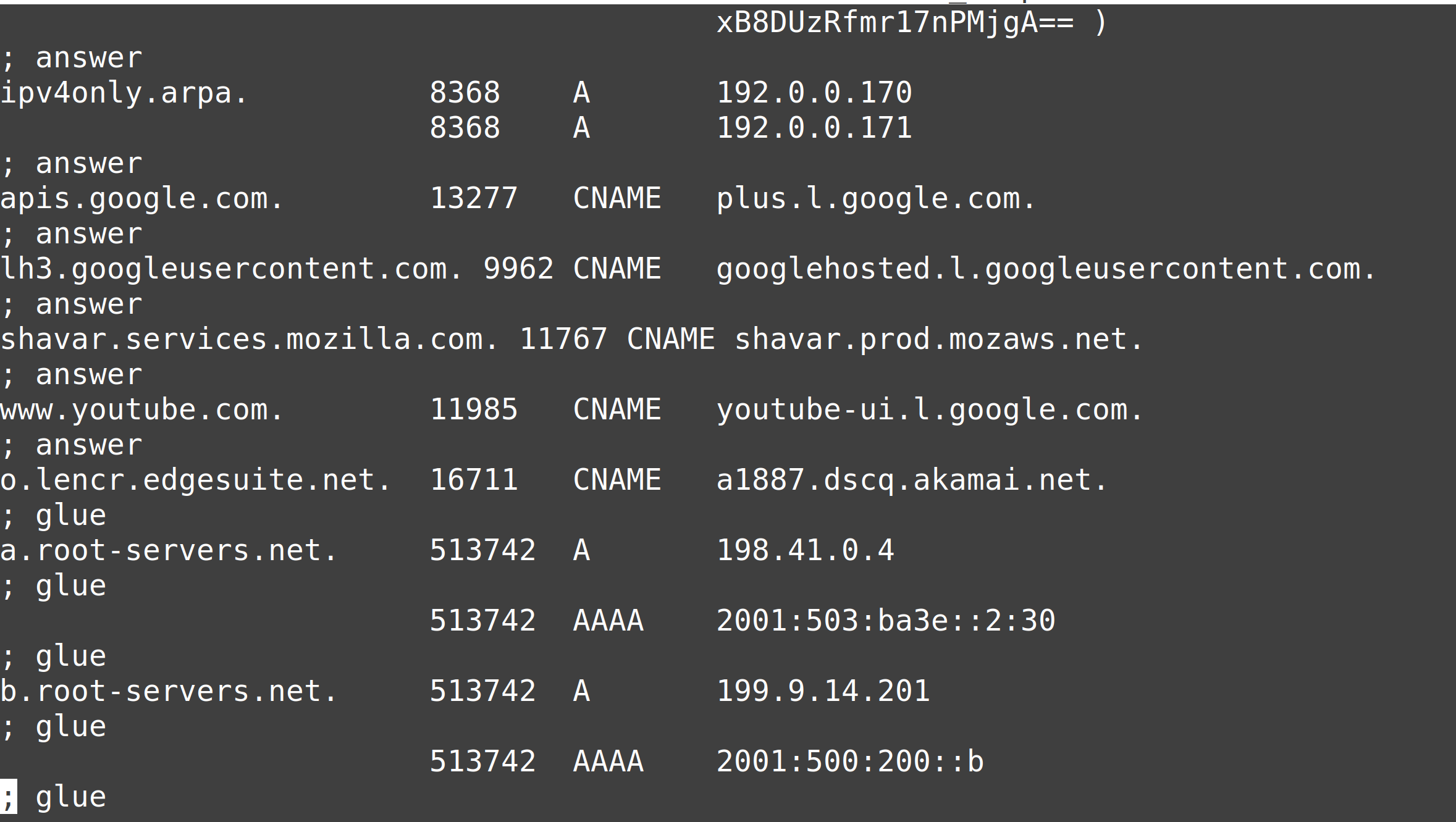
*ניתן לשער שזה איזה דומיין של הדפדפן firefox שצריך על מנת לפתוח אותו, מכאן הדפדפן ממשיך ונתקל בעוד דומיינים שהוא צריך ומבקש גם אותם וקורה בדומה למה שהצגתי כאן.*

*ניתן לראות לאחר הרצה נוספת וסינון שוב:  
*

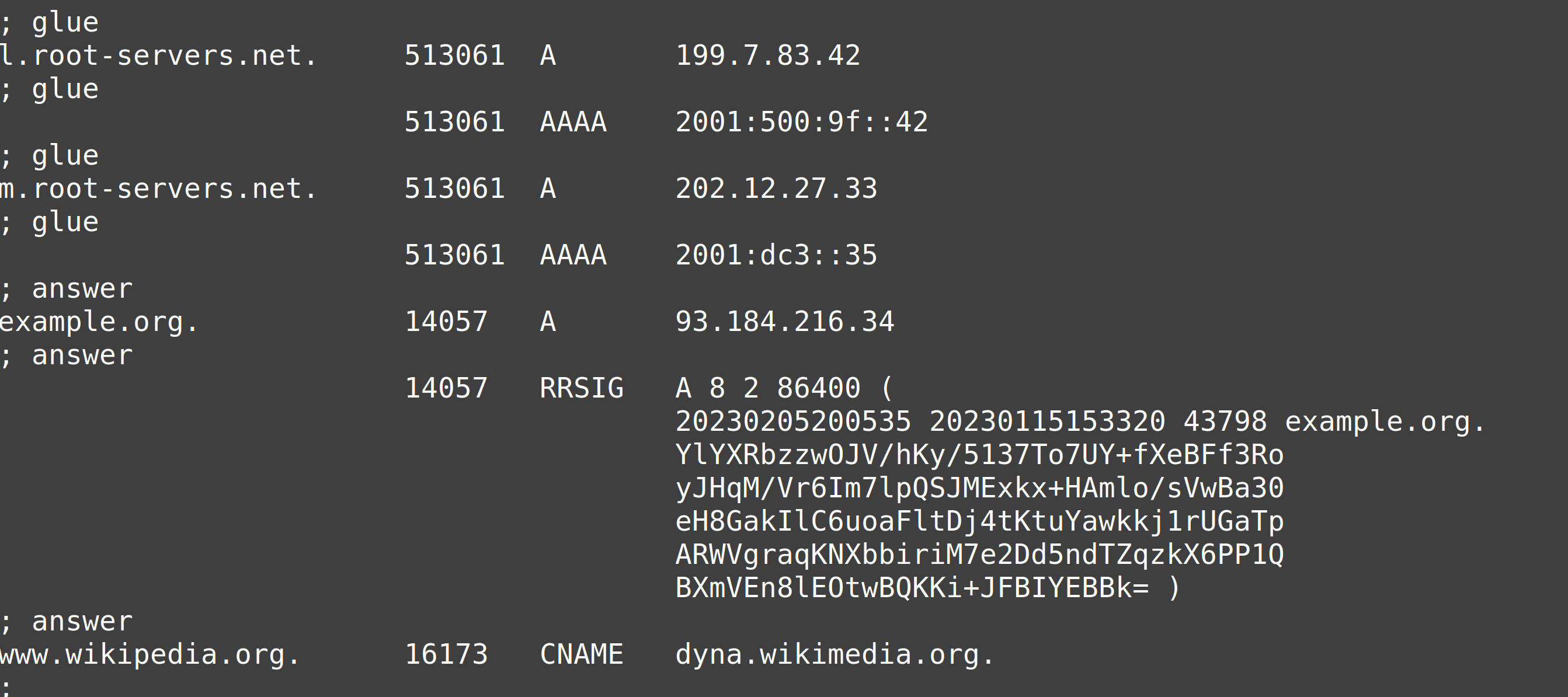
*ישנן הרבה פחות חבילות מתקשורת הDNS וכל התקשורת שקרתה בהרצה הראשונה (בתמונה למעלה) בכלל לא קורית כאן משום שהשרת שמר בcache ולכן הוא לא פונה לשרת של גוגל משום שהדומיינים והIP שלהם שמורים אצלו.*

*הערה: בכל זאת הוא כן פונה ואני לא יודע בדיוק למה, אלה כתובות שהוא לא שמר בcache שלו, כנראה בגלל שהוא לא היה צריך אותן בפעם הקודמת, או שהוא לא ראה לנכון לשמור אותן וראה לנכון יותר לשמור את הראשונות, חיפשתי בהרצה הראשונה והכתובות האלו לא הופיעו שם אבל אולי זה בגל שעצרתי את ההרצה קודם, לא יודע.*

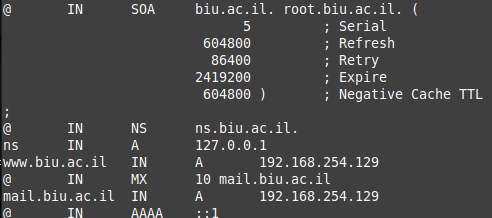
*הערה 2: ניתן לראות בזמנים שהתהליך כולו קורה הרבה יותר מהר משום שלדפדפן ולשרת המקומי יש פחות עבודה לבצע.*

*כעת נכנסתי ליוטיוב (דומיין-* [*www.youtube.com*](http://www.youtube.com)*) ולאחר הרצת הפקודה* sudo rndc dumpdb -cache *נראה את הכתובות השמורות בcache:  
*

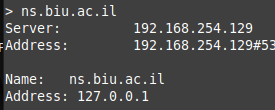
*ואכן יש כאן את הדומיין שמור.*

*כעת נגלוש לויקיפדיה שאכן לא מופיע בcache ונייצא שוב ואכן:  
*

*כעת אשנה את הדברים המבוקשים וככה נראה הקובץ שלי*

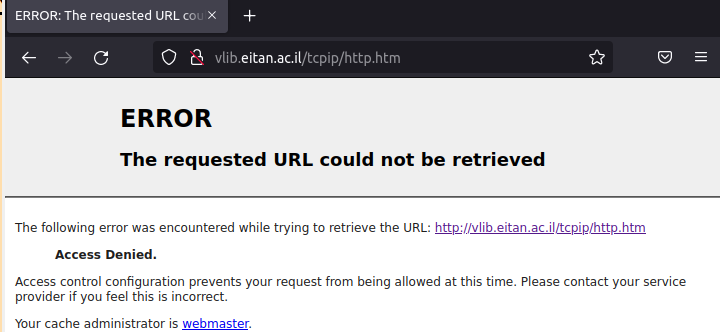
**

*כעת נבצע הרצה של nslookup לבקשת ns.biu.ac.il*

**

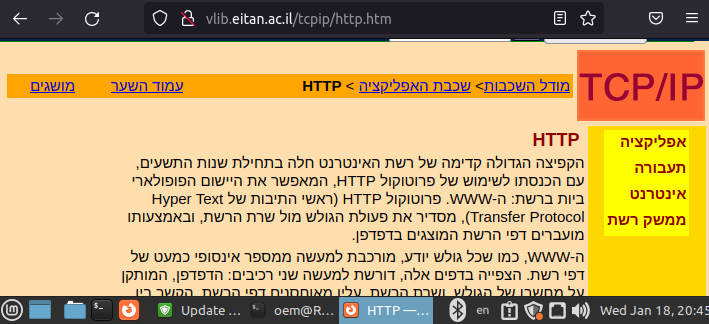
*אך כאן הwireshark משום מה לא הסניף לי כלום (ניסיתי מלא פעמים ועכשיו אני כבר על סף ייאוש) לכן איני יכול לצרף את זה.*

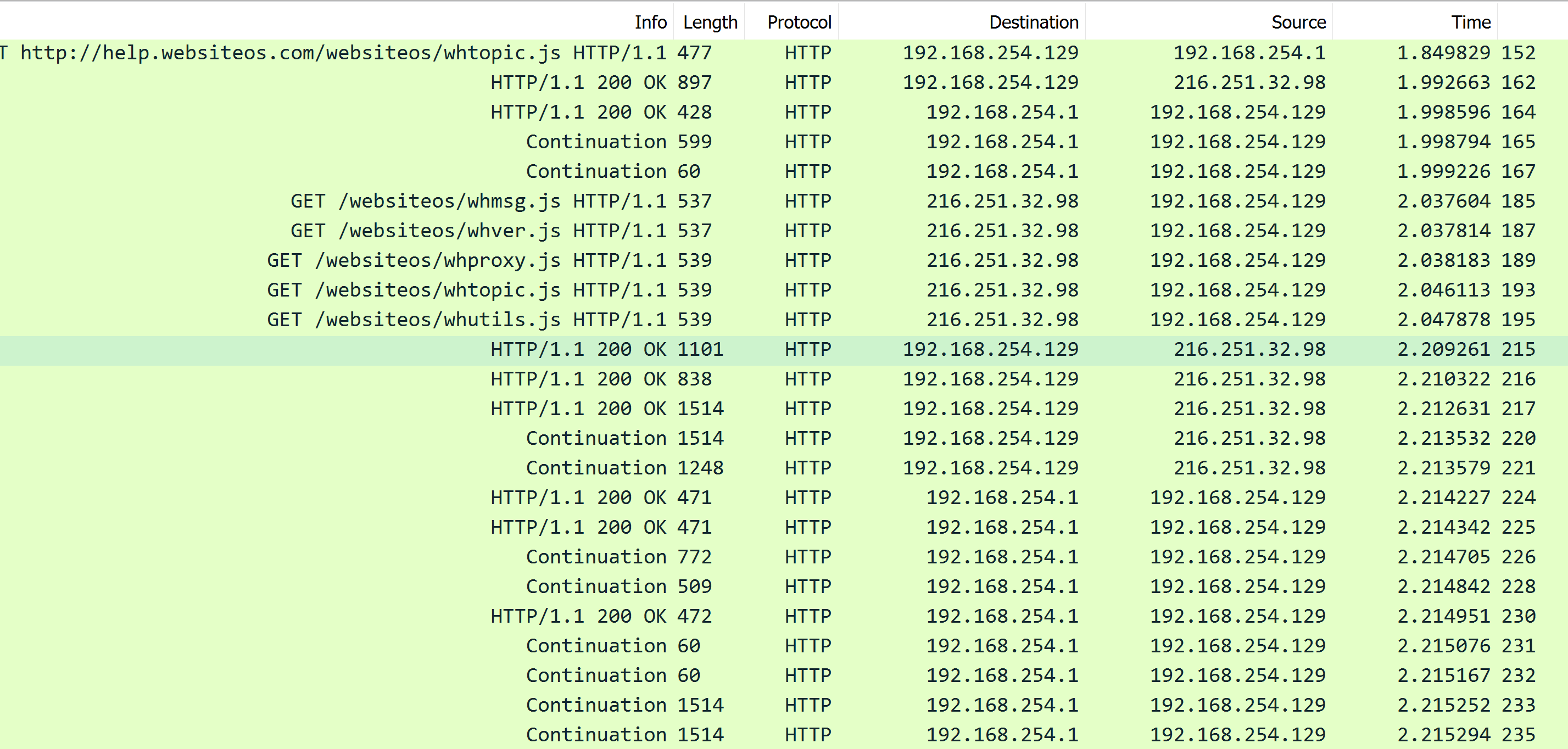
***חלק ג:****לאחר התקנת שרת הproxy נתחבר לכתובת http://vlib.eitan.ac.il/tcpip/http.htm ואכן:*

**

*Access denied*

*לאחר השינויים המבוקשים (http\_access allow all) אכן עובד:*

**

*כעת גלשתי לאתר http://help.websiteos.com/websiteos/example\_of\_a\_simple\_html\_page.htm משום שלאתר הקודם כבר גלשתי וכשהסנפתי זה לא הראה את כל הדברים שקורים, לכן גלשתי אליו והסנפתי:* 

*(הIP של המחשב עם השרת פרוקסי הוא 192.168.254.129 והIP של המחשב השני המבקש הוא 192.168.254.1)*

*אכן ניתן לראות ראשית שכל בקשה לא קורית בחיבור אחד אלא בשניים למשל כאן:*

*בשורות 215-221 זה השרת אליו פונה השרת שלנו שולח לשרת עם הפרוקסי, והשרת עם הפרוקסי שולח למחשב עם הדפדפן בשורות 224-228 וניתן לראות שזה קורה בשני חיבורים שונים.*

*בנוסף נראה כי אכן השרת שולח לדפדפן את הקובץ כולו (לאחר שהוא מוריד אותו אליו מהחיבור שקיבל ממנו) לדוגמה בחבילה 226 (הcontinuation) מבין חבילות 224-228:  
*

*נשלחת דאטה בבתים של הקובץ עצמו ואכן זה הקובץ כולו מקודד שהשרת מעביר הלאה.*

*ובנוסף, אכן הDNS מתבצע משרת הproxy כפי שניתן לראות כל התקשורת עוברת דרכו.*

***הערה לבודק: הרבה דברים קרו לבסוף לאחר הרבה ניסיונות שלא הבנתי למה הם שגו לי, על אף שאני יודע את החומר, אך המחשב שלי משום מה לפעמים מריץ דברים אחרת ואילו במחשבים אחרים זה עבד לי בניסיונות שלי. אני לא יודע אם קבצי הpcap מהwireshark וצילומים שלהם יהיו מדויקים בעקבות כך אך ניתן לראות אכן שהשקעתי בתרגיל הזה רבות ובאמת התאמצתי הרבה והשקעתי בו. אשמח להתחשבותך בנושא. תודה רבה מראש ובשעה טובה חסל סדר תרגיל.***