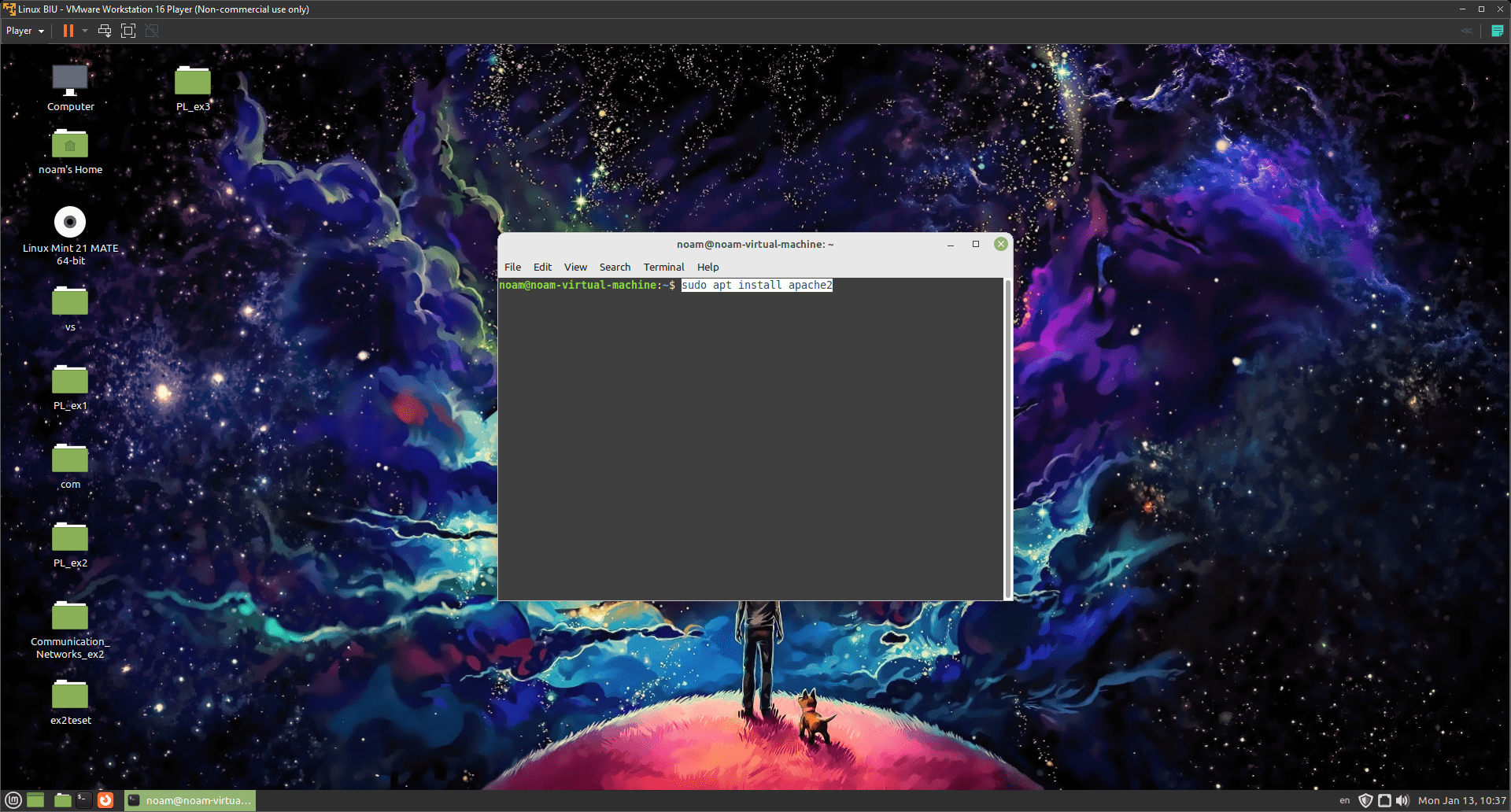
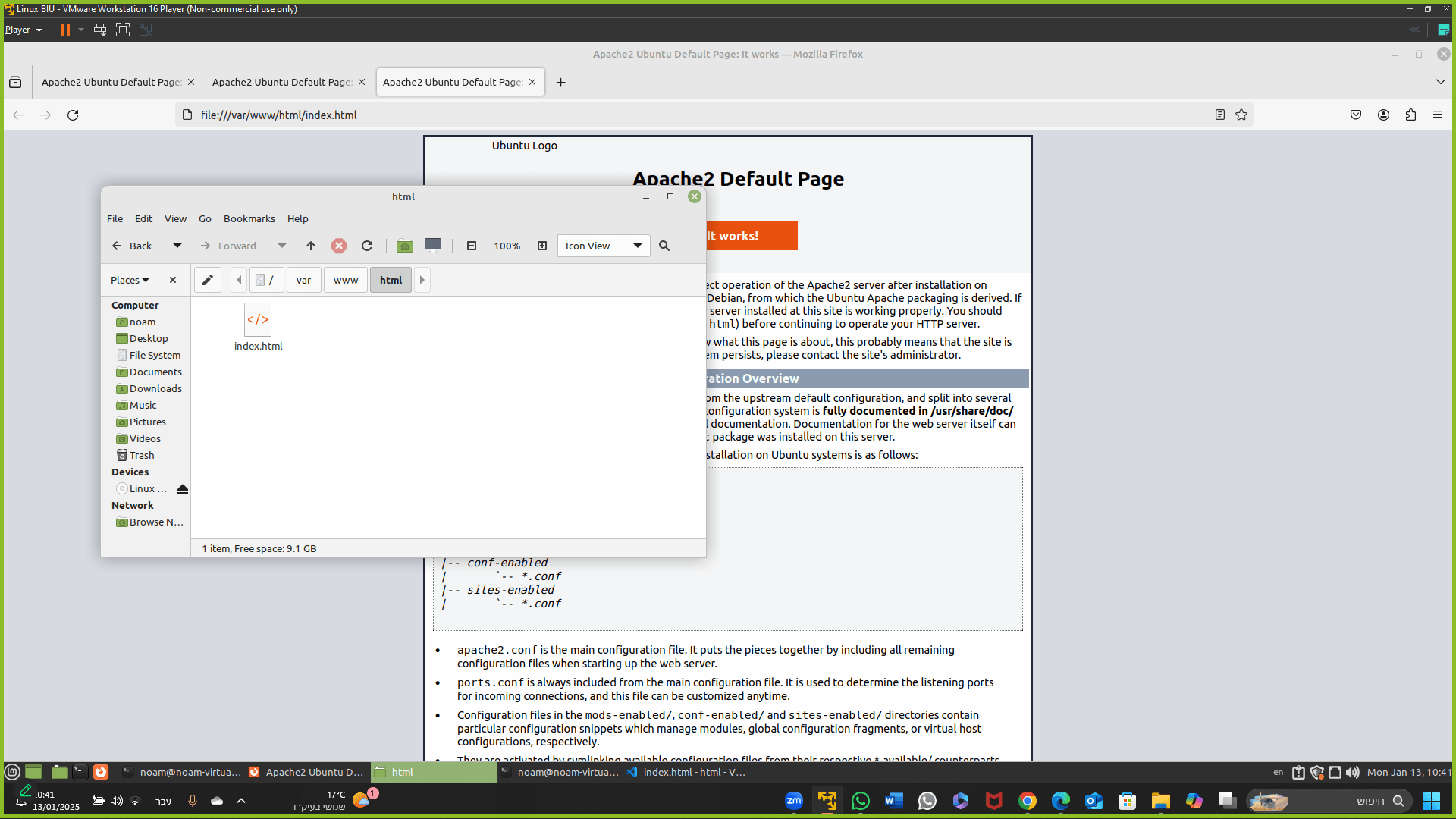
מטלה מס' 3 – אילאי בן יהושע ונועם ליבוביץ

*חלק ראשון*

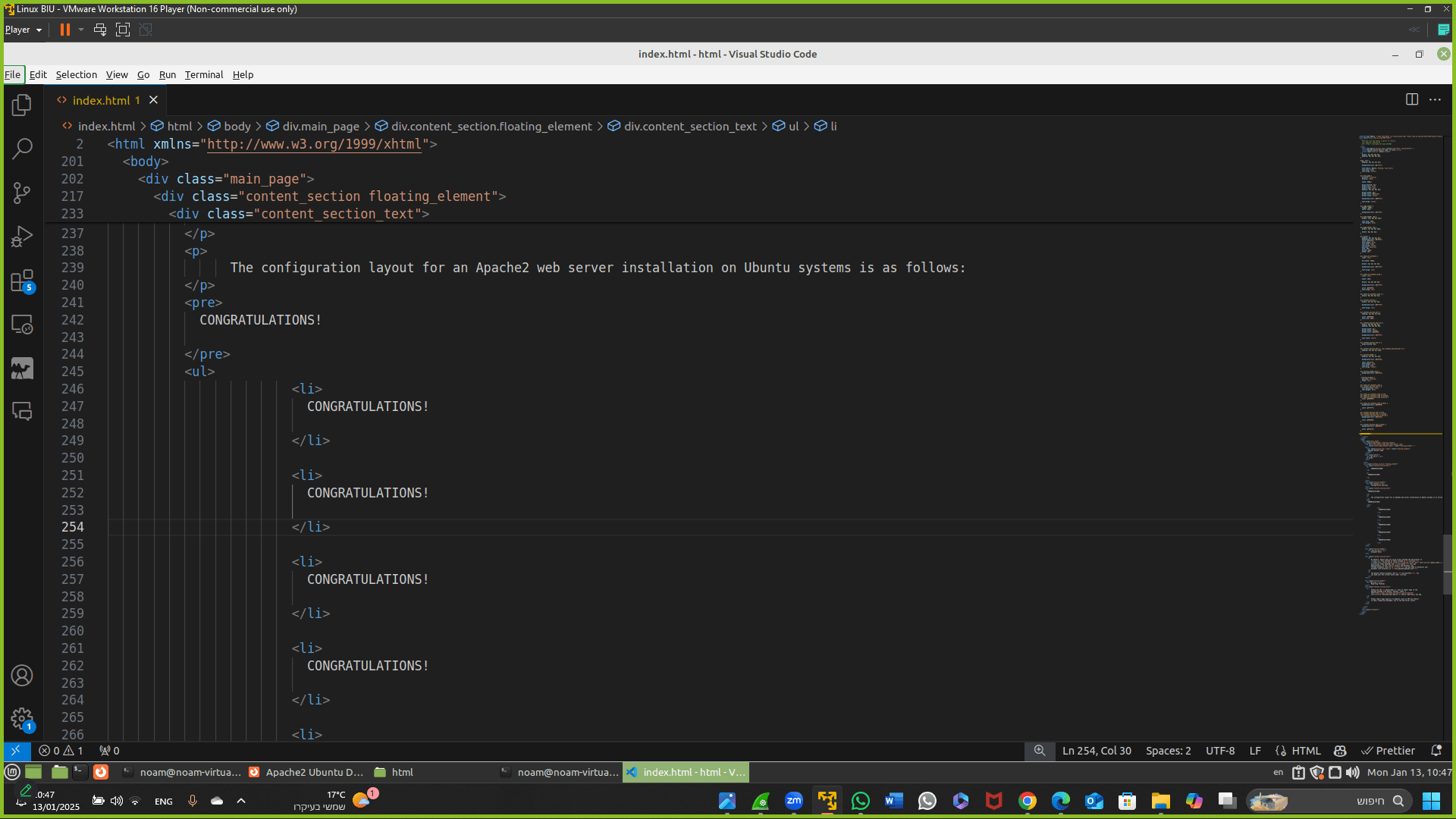
נתקין את שרת האינטרנט – apache.



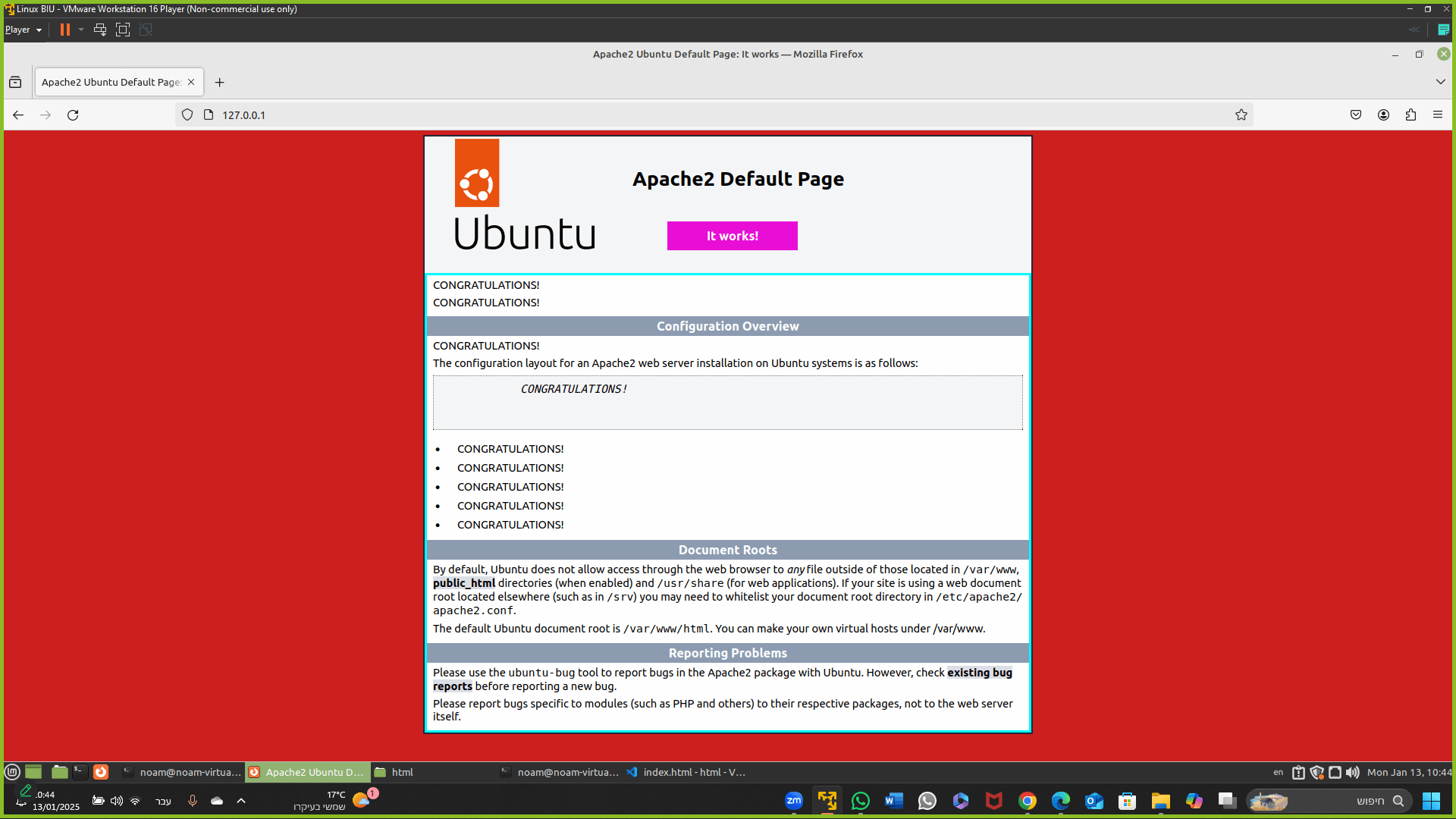
נראה את קובץ הhtml הדיפולטיבי בנתיב – var/www/html.



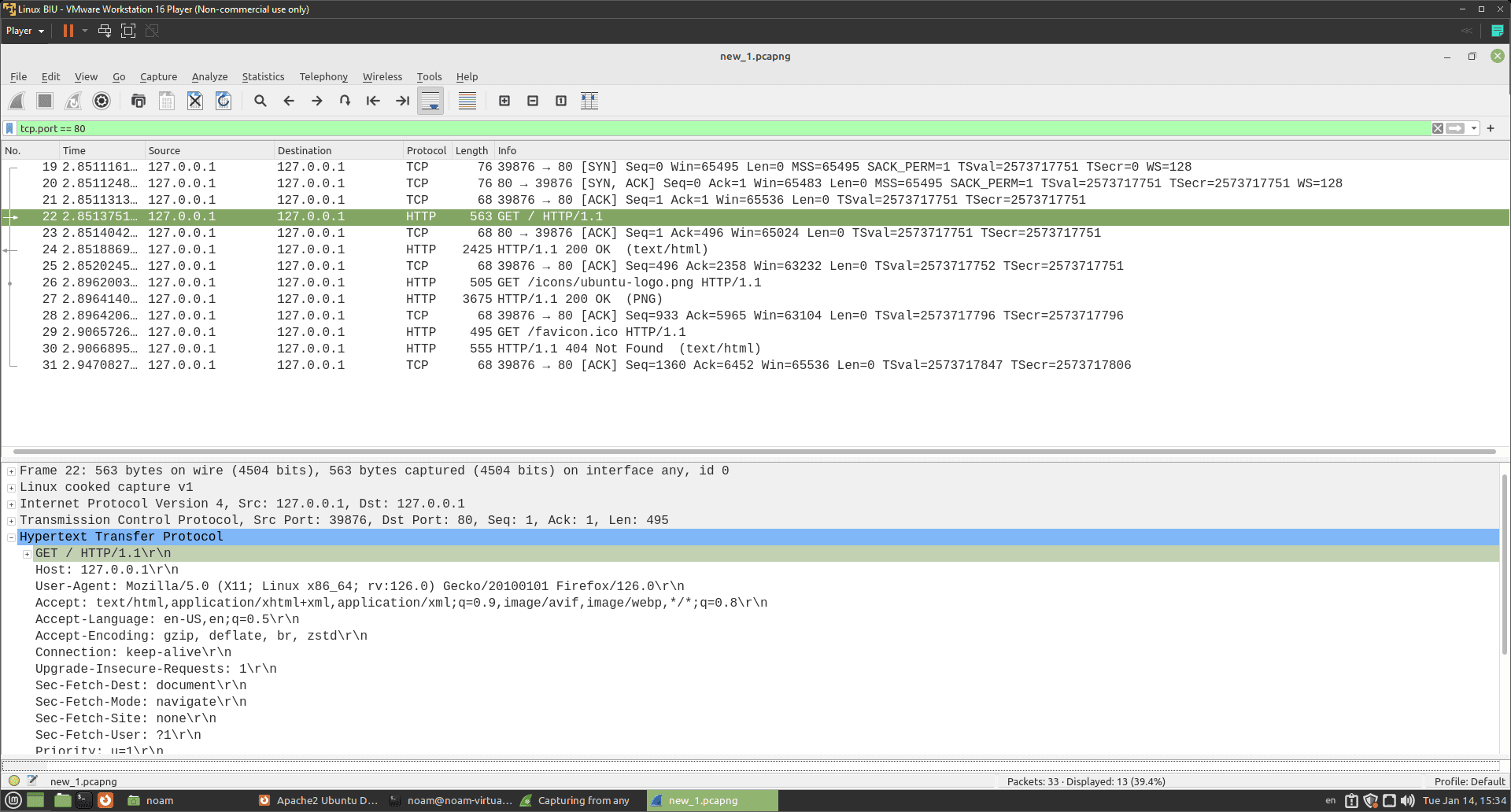
נפתח אותו בערוך הטקסטים vscode ונבצע בו מס' שינויים גרפיים וטקסטואליים.

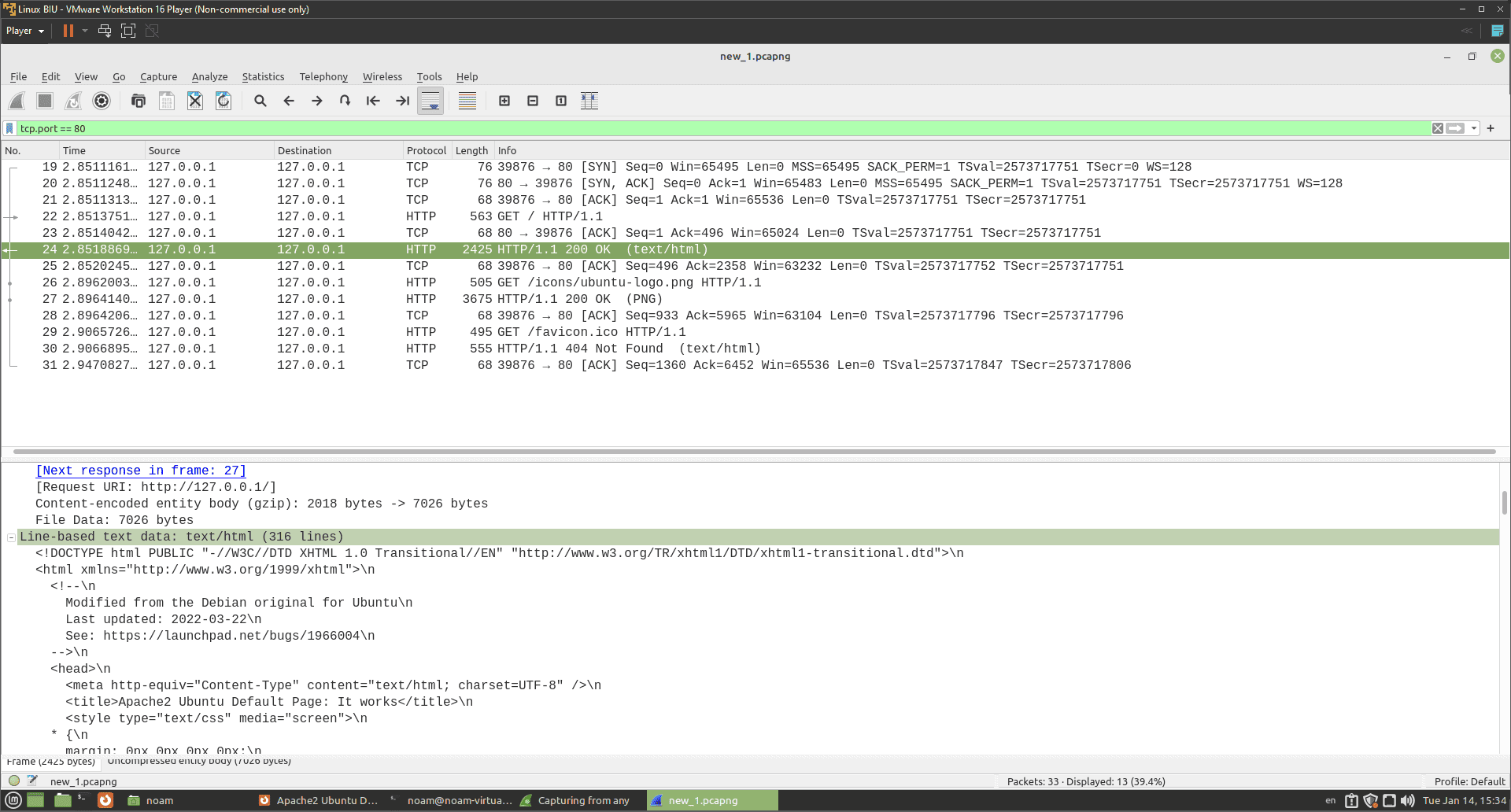


נשים לי כי השינויים מוצגים כעת בדף הדיפולטיבי אותו מחזיר השרת לאחר רענון של העמוד.

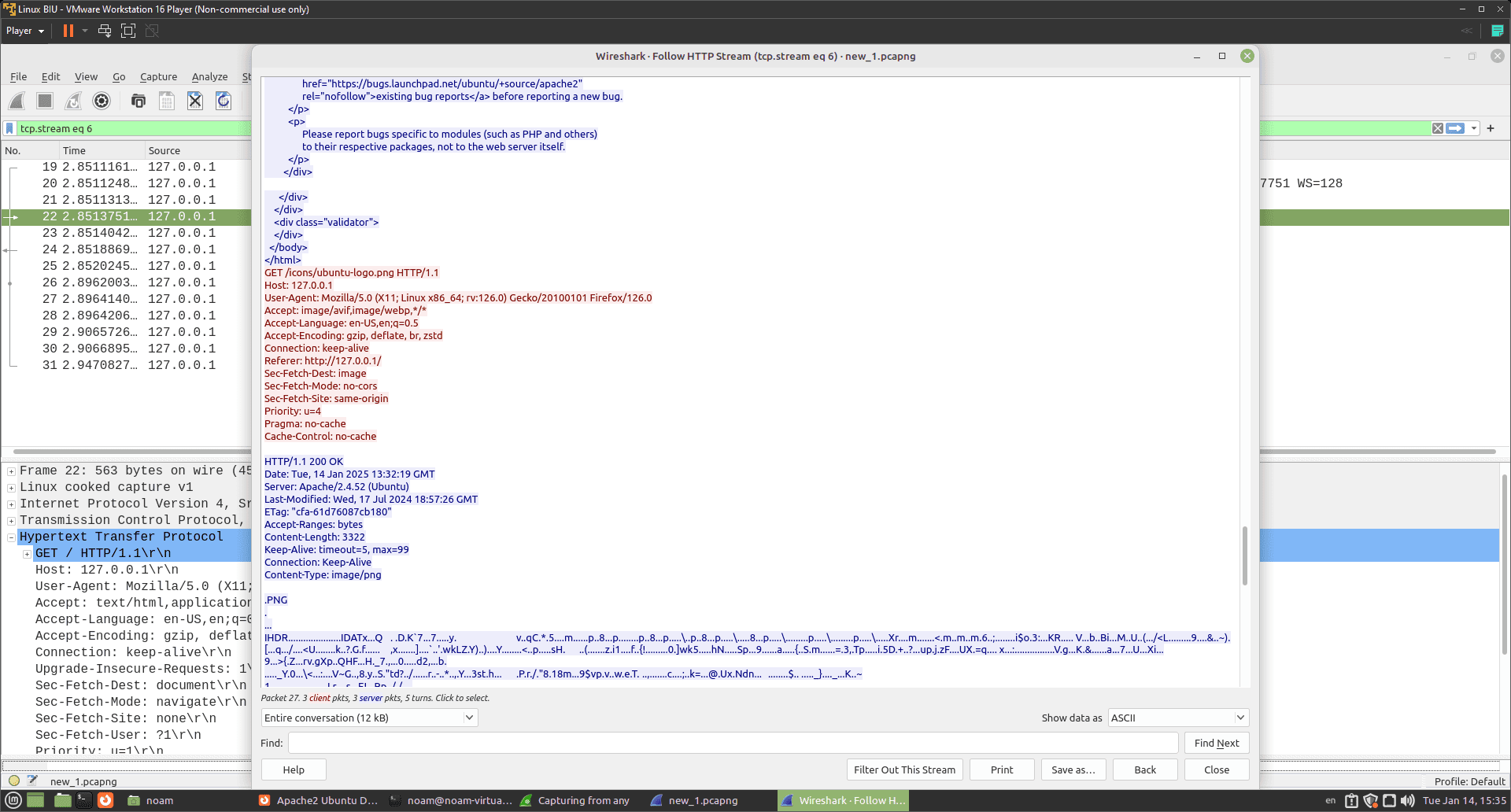


נפתח את תוכנת הwireshark (מעכשיו תקרא בהמשך הטופס 'כריש'), ונסניף בעזרתה את התעבורה. נסנן כך שיוצגו חבילות הרלוונטיות לפרוטוקול TCP ולפורט 80 עליו רץ שרת האינטרנט שלנו.





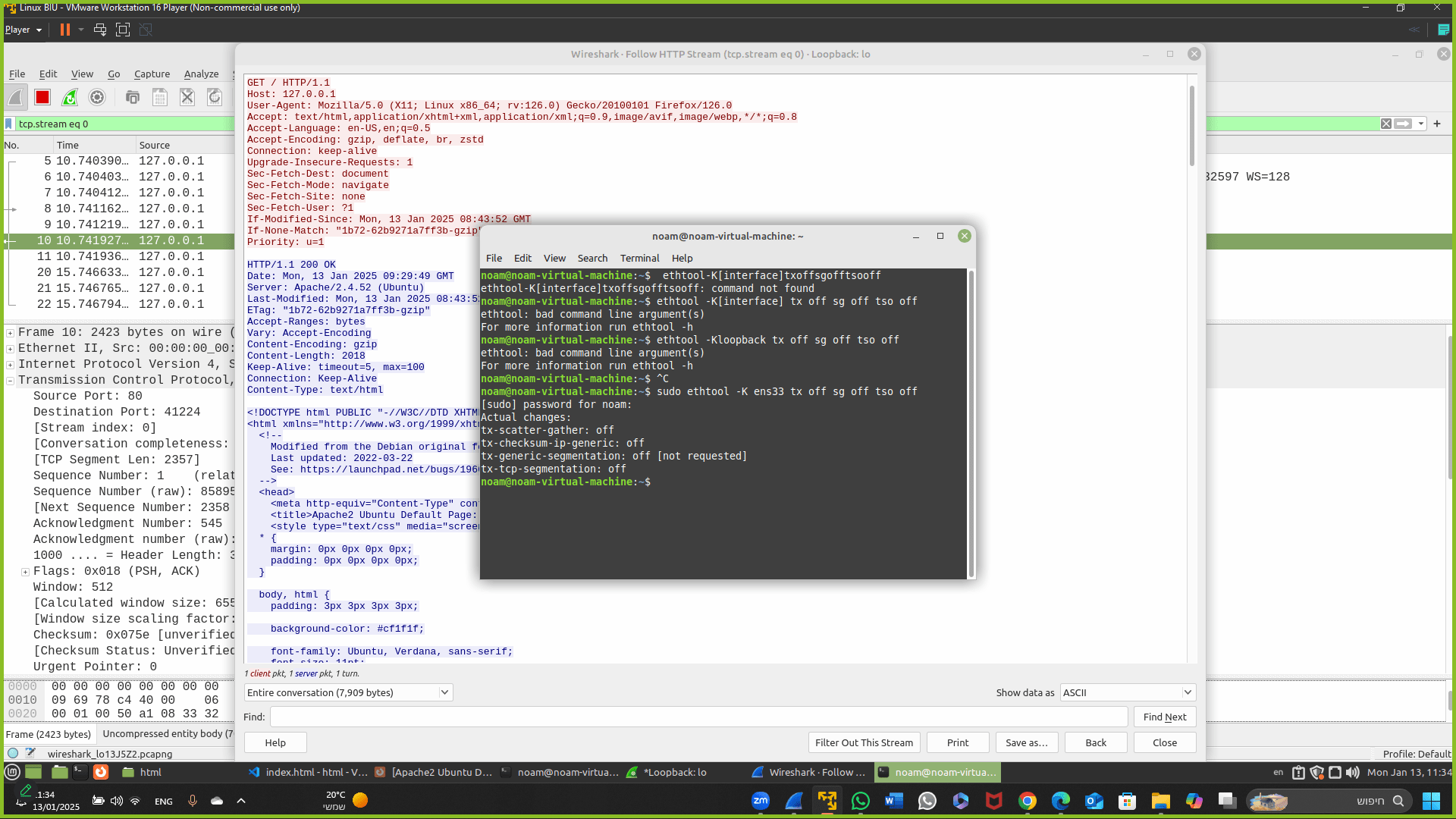
נסביר את הפלואו של הדברים בתמונות לעיל. תחילה אנו רואים לחיצת ידיים של   
syn-syn&ack-ack בין הלקוח (הדפדפן) לשרת. לאחר מכן הלקוח שולח בקשת http מסוג get לשרת על מנת לקבל את "דף הבית" (הלו הוא '/'). השרת מחזיר ללקוח ack כדי לתת חיווי שאכן בקשה זו נקלטה אצלו, ובהודעת הtcp העוקבת אנו הוא מחזיר לו את העמוד כאשר סטטוס ההודעה הינו 200 ok, כלומר הכל טוב. מה שקרה לאחר מכן הלקוח החל לקרוא את העמוד וראה שהוא זקוק למשאב נוסף – ubunto-logo.png, אז הוא ביצע על גבי אותו חיבור (שכן אנחנו בhttp בגרסה 1.1 והוא persistent) בקשה נוספת מהשרת ע"מ לקבל את התמונה הדרושה. השרת משיב ב200 ok עם תוכן התמונה, ואז הלקוח מבצע בקשה נוספת (שוב על גבי אותו חיבור) עבור משאב מסוג favicon, אך הפעם השרת לא מוצא אותו ועונה ללקוח ב404 not-found.



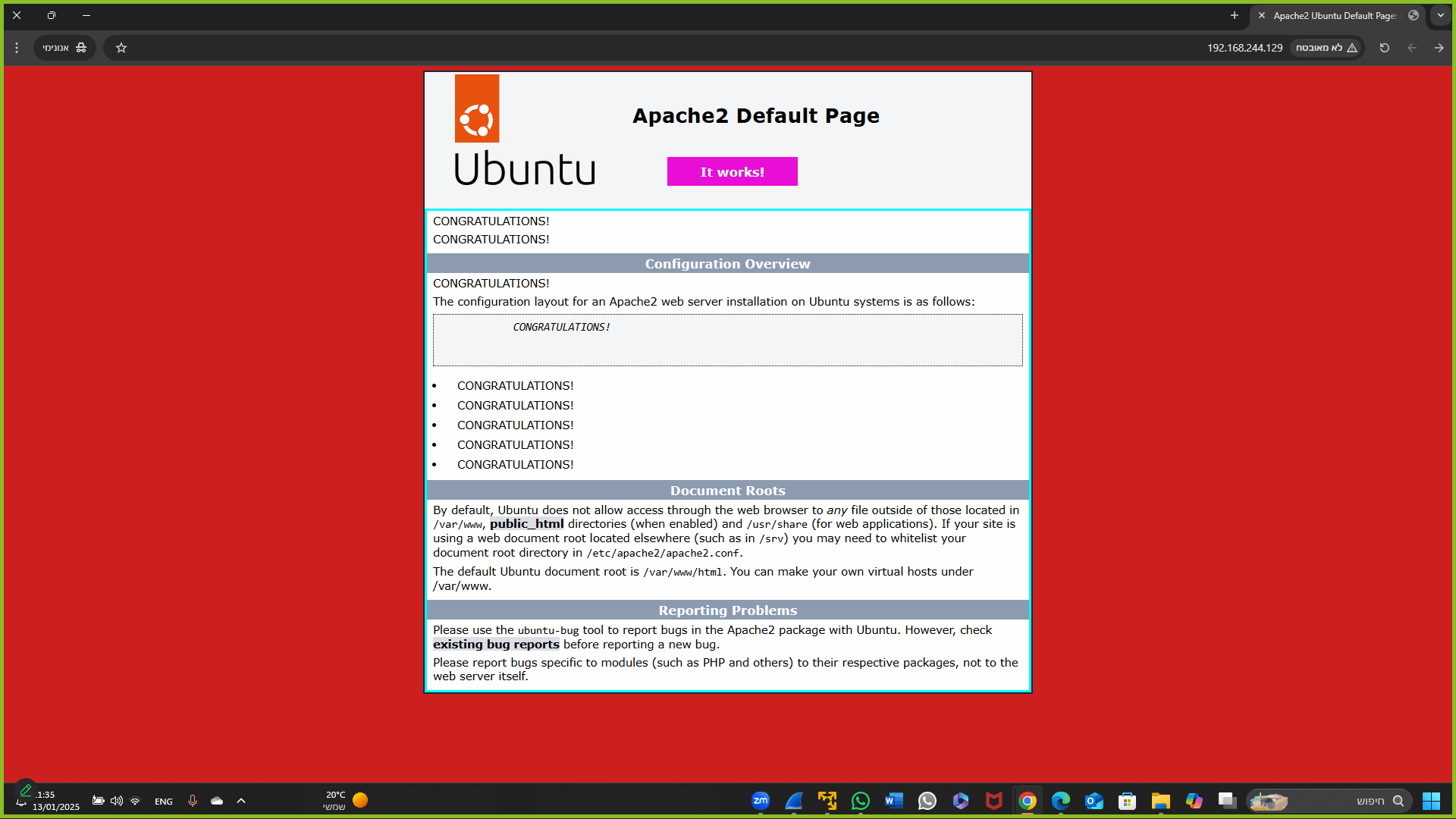
התמונה לעיל אנו רואים את הפלואו של תקשורת הhttp שקרתה, כאשר באדום אנו רואים את סוף תוכן העמוד שהשרת החזיר בבקשה הראשונה ללקוח, בכחול אנו רואים את הבקשה הנוספת של הלקוח עבור התמונה, ובירוק את התשובה של השרת עם הקידוד שלה.

נסכם – סה"כ התקשורת לכל אורכה התבצעה על גבי חיבור יחיד שהוקם בין השרת ללקוח, אשר דרכו עברו 3 בקשות. הראשונה החזירה קובץ html, השנייה החזירה תוכן **מקודד** שך תמונה בפורמט png, והשלישית החזירה header ללא תוכן, שכן המשאב לא נמצא ע"י השרת. נשים לב שאורכי התכנים שמועברים בהודעות מצוינים בהאדרים, כך שלדוג' בתמונה הראשונה אנו רואים כי אורך התוכן (לאחר דחיסתו) הינו 2018, וזה כמובן קיים בכל אחת מההודעות.

כעת נבטל את מנגנון הsegmentation offload.

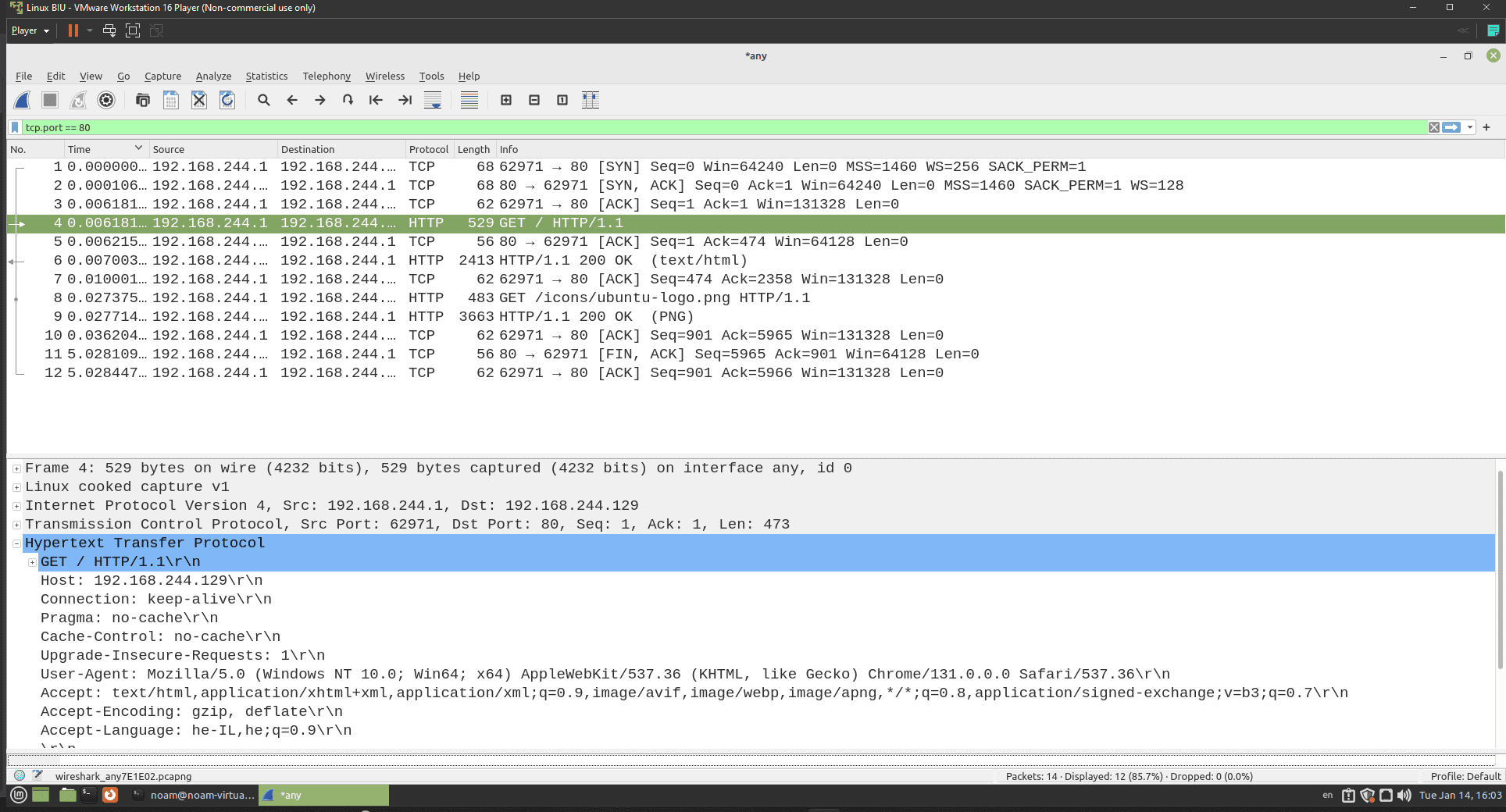


נגלוש לשרת מהמחשב **השני** דרך מצב incognito וכן דרך מצב דפדפן 'רגיל'. ראשית נראה שזה אכן עובד.

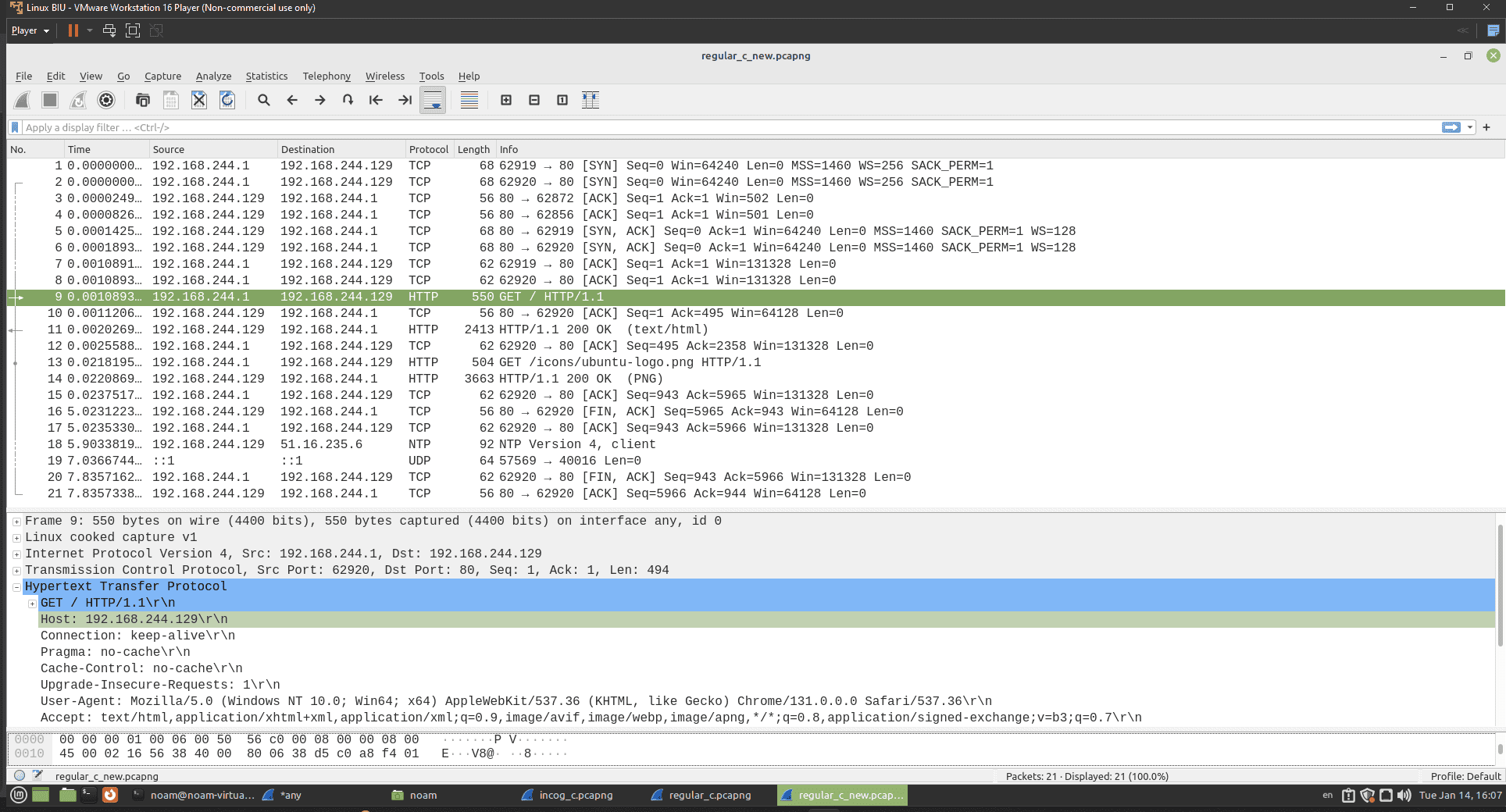


עובד.

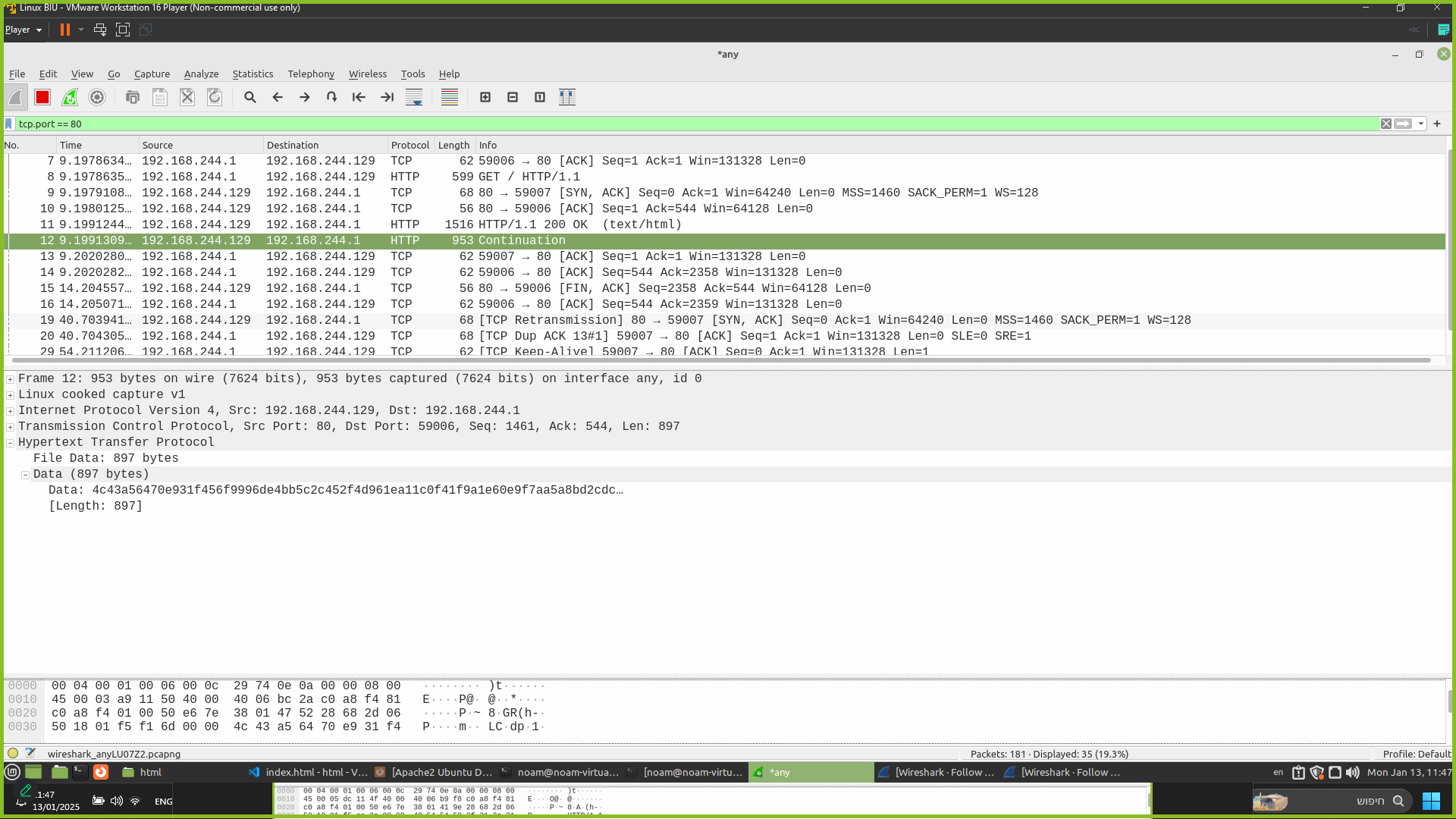
נבחין בתעבורה דרך הכריש. נתחיל דווקא בגלישה דרך מצב incognito. נשיב לב שקיבלנו משהו מאוד זהה לגלישה דרך המחשב עליו רץ השרת. גם פה ישנן 2 בקשות אל השרת, הראשונה עבור עמוד הhtml הדיפולטיבי של השרת, והשנייה עבור התמונה שהעמוד מציג (בדומה למה שראינו קודם לכן). אולם נשים לב שהפעם לא מתבצעת פנייה שלישית עבור משאב הfavicon, אלא ישר לאחר קבלת התמונה מהשרת הלקוח סוגר את החיבור מול השרת. נשים לב שהכל התבצע על גבי חיבור יחיד בין הלקוח לשרת. הבדל טריוויאלי הוא שהפעם התקשורת מתבצעת בין 2 כתובות ip שונות (וזה הגיוני שכן השרת והלקוח רצים על 2 מחשבים שונים), בשונה מהדוג' הקודמת. שם השתמשנו ב127.0.0.1.



כעת נבחין במס' שינויים כאשר ניגש לשרת דרך הדפדפן במצב הרגיל. תחילה נציין כי גם הוא מבקש **2** משאבים בלבד, אך הפעם הוא פותח חיבור נוסף, עם זאת הוא לא משתמש בו כלל לאורך הפלואו של התקשורת. ראינו כבר בתרגיל קודם כי זהו מנגון על מנת שלדפדפן יהיו מס' חיבורים זמינים בעת הצורך על מנת לבקש מס' משאבים במקביל. דבר מינורי שראינו הוא שבheader של השפה בincognito יש רק עברית, ואילו ברגיל מצויין שם גם כן אנגלית – en. מעבר לכך לא הבחנו בהבדלים נוספים בין מצבי הדפדפן.



נבטל בהגדרות את הקונפיגורציה לגבי איחוד חבילות tcp תחת http ונראה שאכן עכשיו אנו רואים את שכבת האפליקציה פר-חבילה, שלא כמו שהיה קודם לכן.



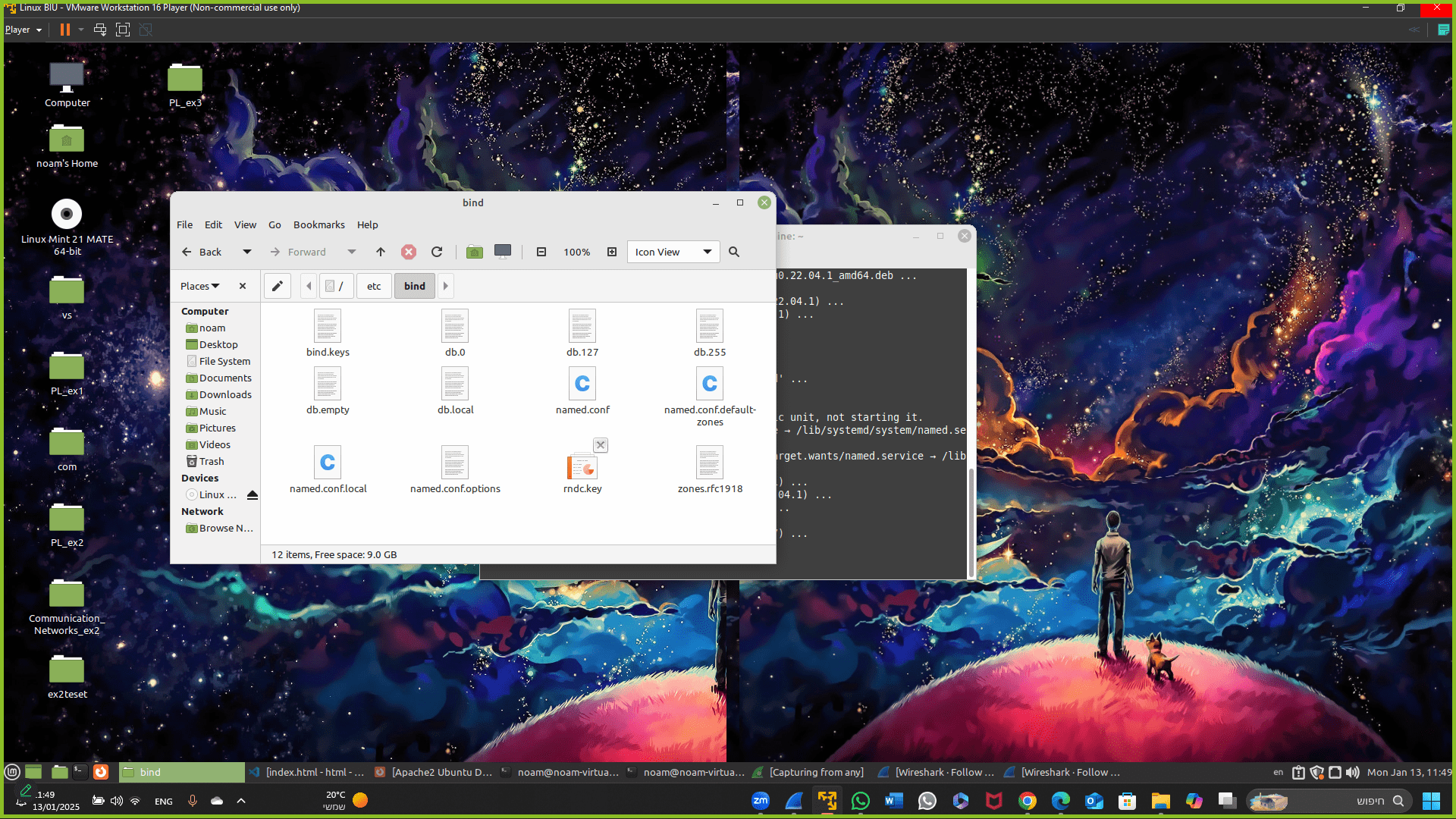
000

000

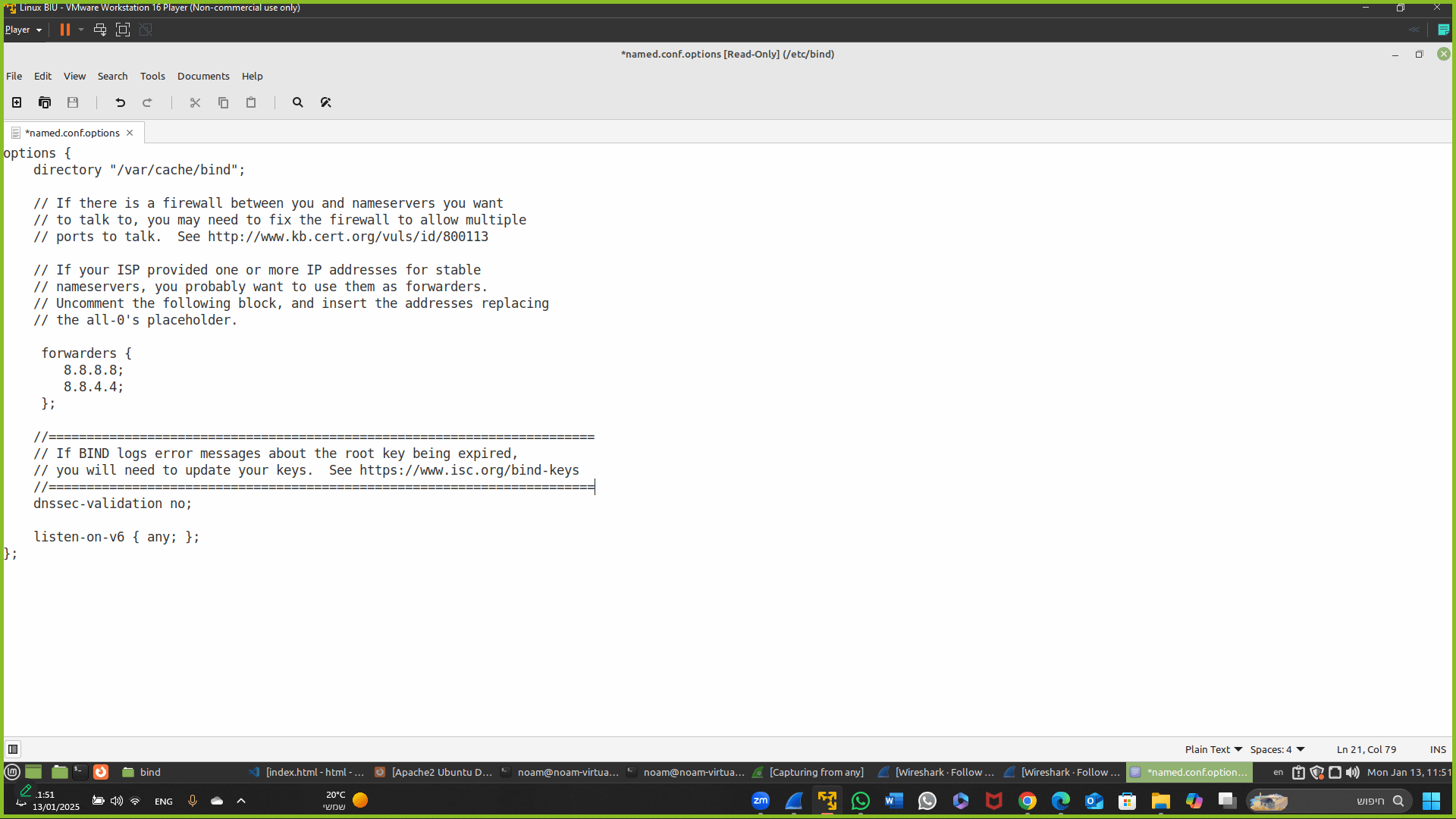
*עד כאן חלק ראשון*

*חלק שני*

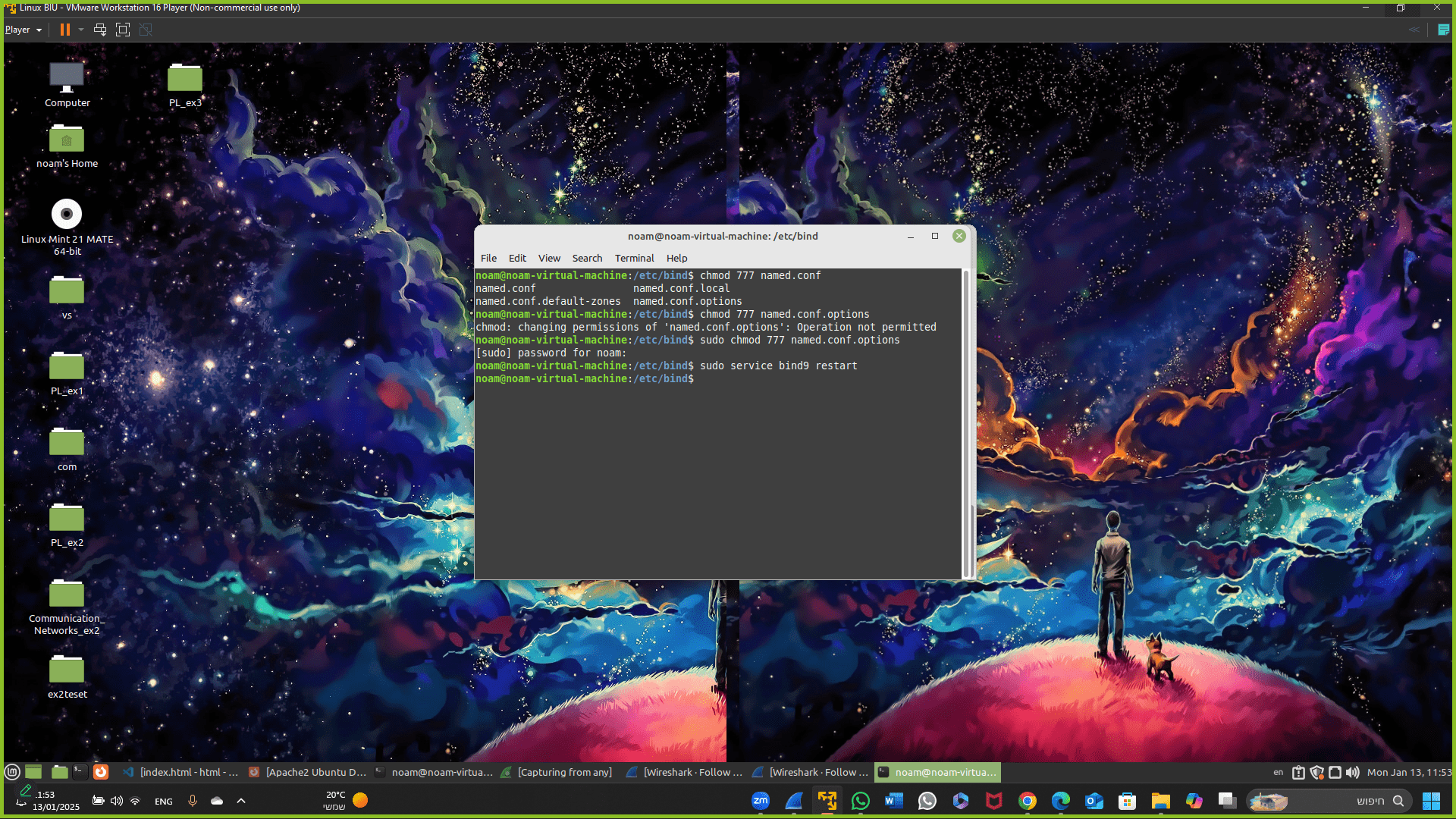
לאחר שהתקנו את שרת הDNS, נמצא את הקובץ *named.conf.options*.



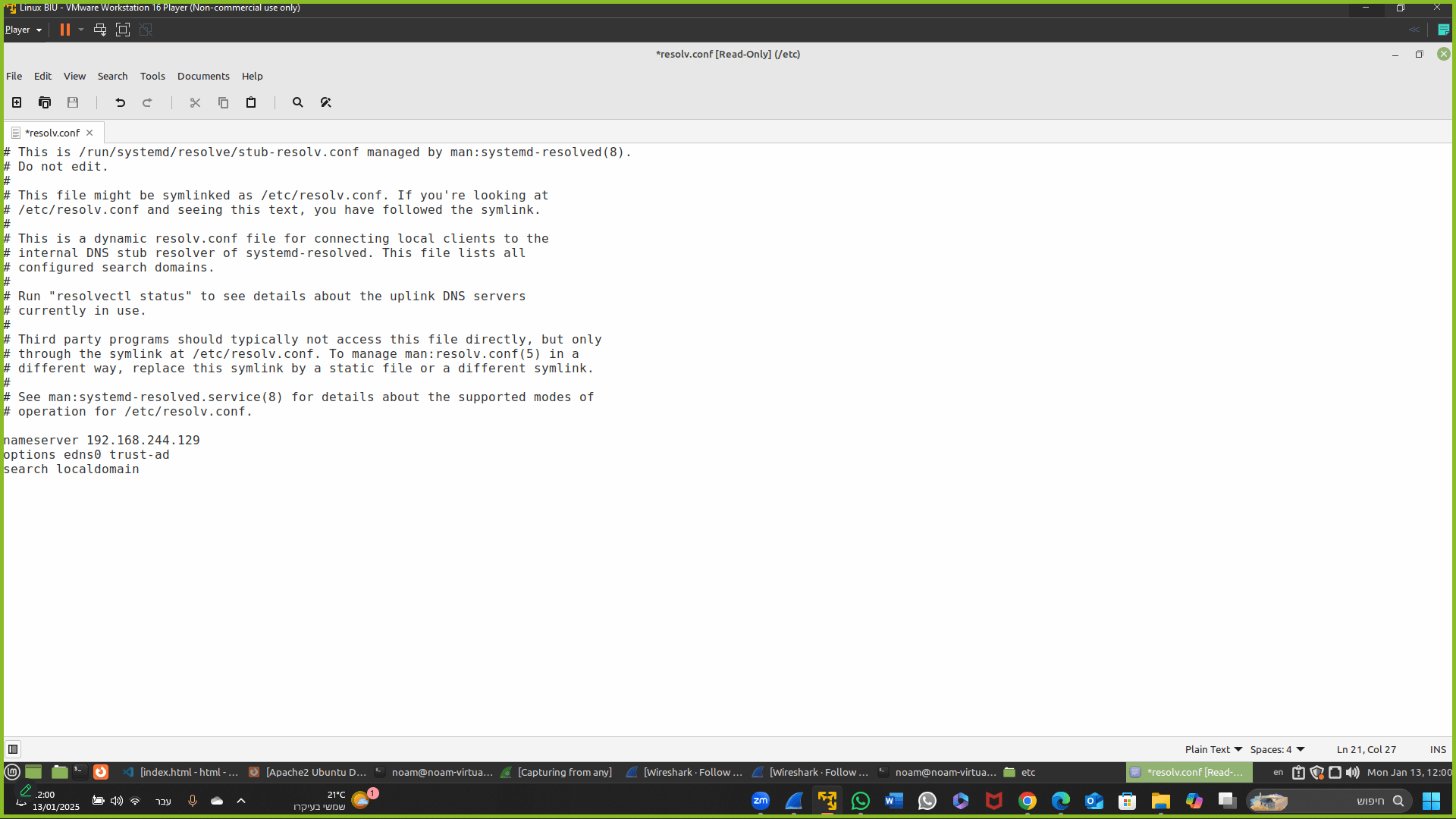
נגדיר את שרת הDNS שלנו כך שכל פעם שתגיע אליו שאילתת DNS, הוא יעביר אותה לשרת הDNS של גוגל (8.8.8.8 או 8.8.4.4). לאחר מכן, יקבל את התשובה חזרה מגוגל, ישמור אותה בקאש ואז יחזיר את התשובה למי ששאל אותו (עובד כמו ריזולבר).



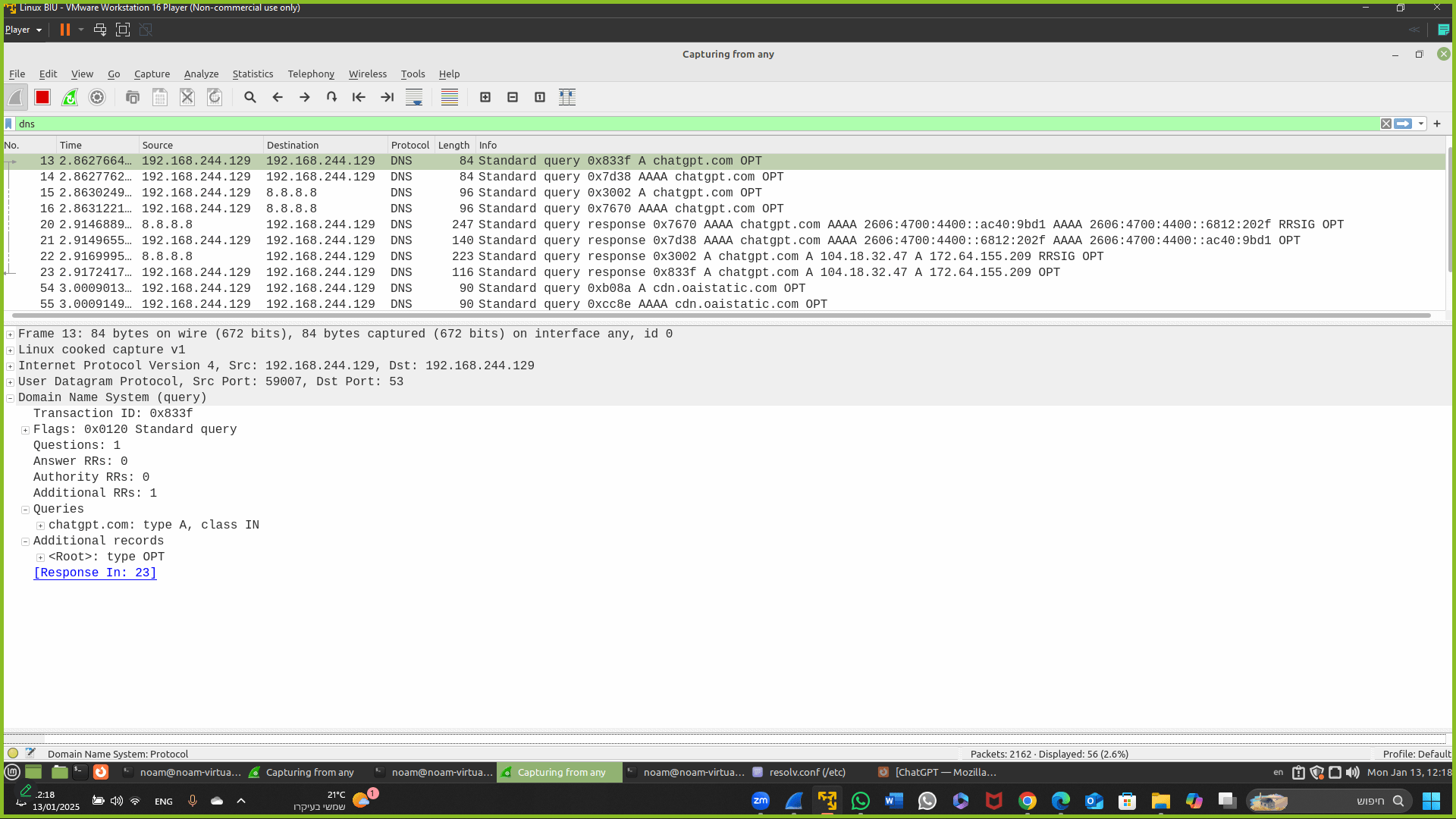
נשמור את השינויים ונאתחל את השרת כך שהשינויים יכנסו לתוקף.



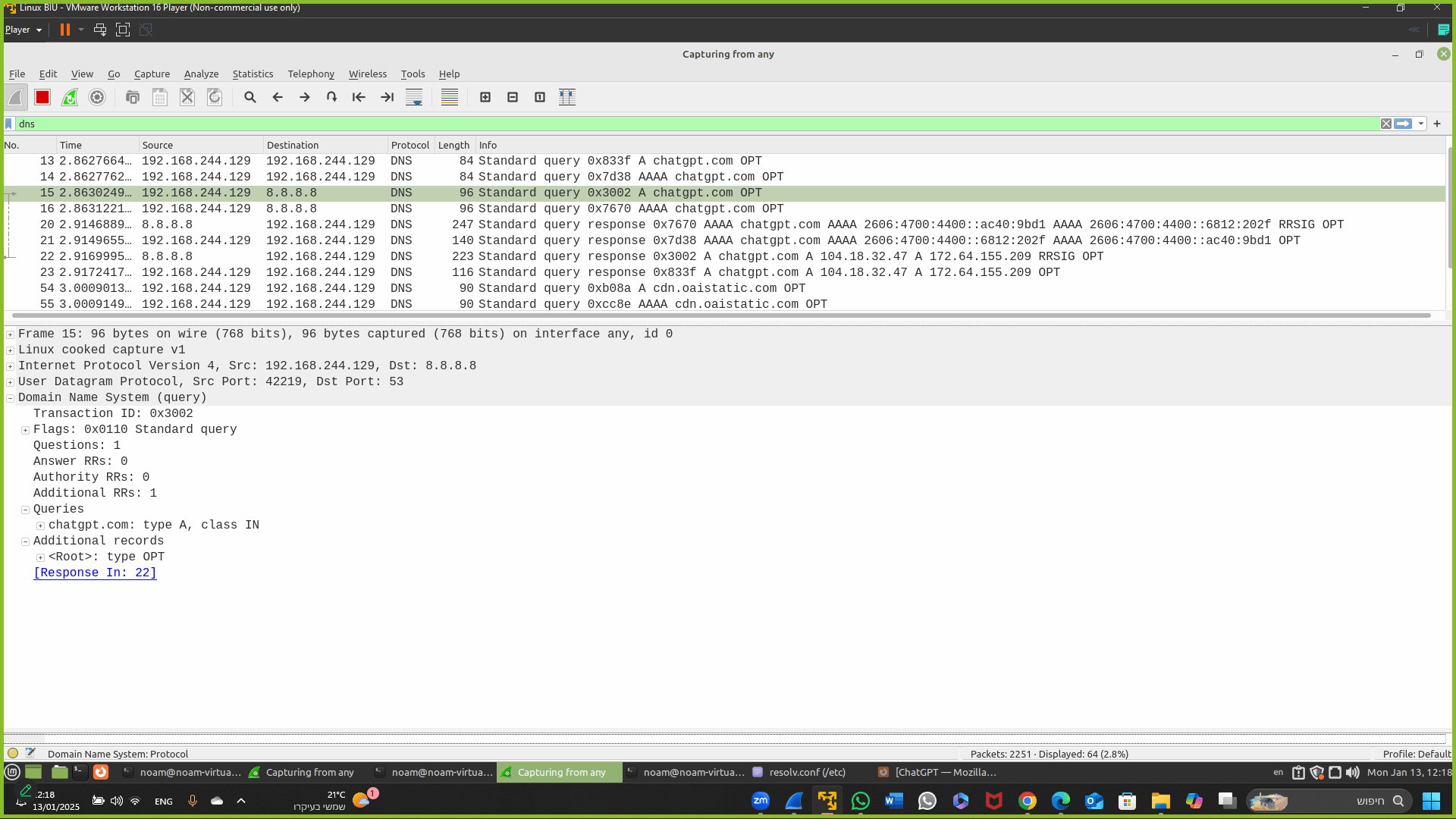
כעת, נגדיר את המחשב לעבור מול שרת הDNS שהגדרנו באופן דיפולטיבי.

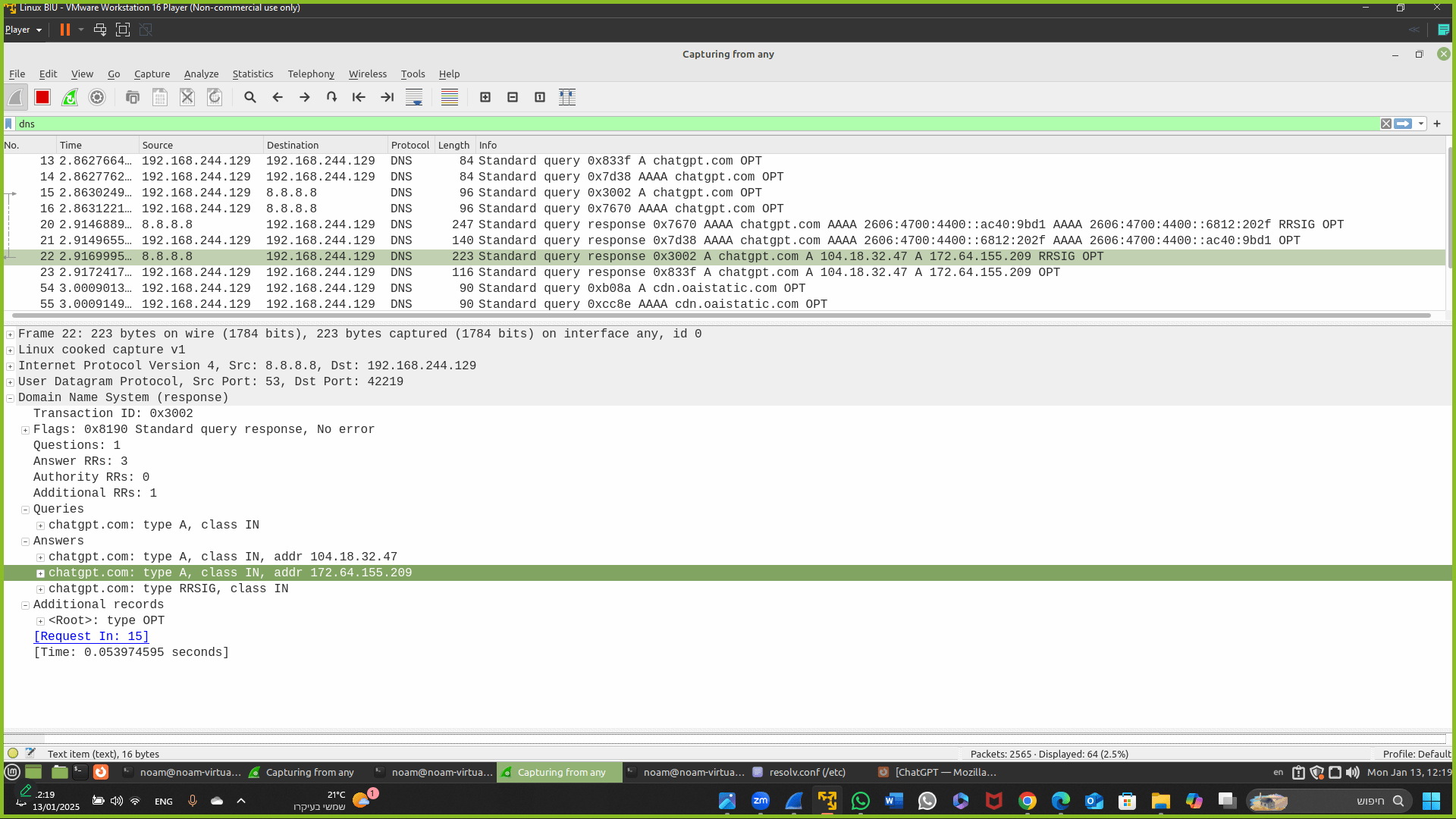


נבצע פעולה המצריכה ביצוע DNS, נגלוש לאתר של צאט GPT. ניתן לראות בכריש את בקשת הDNS של הלקוח (הלוא הוא אנחנו), כאשר הלקוח מבקש את כתובת הIP של האתר פעם אחת בפורמט של ipv4 באדום ומיד לאח"כ פעם נוספת בפורמט של ipv6 בכחול. נשים לב כמובן שפורט היעד הוא 53, הרי שהוא פורט המוקצה לשאילתות DNS. (הערה – נשים לב שבכל פעם מעכשיו בהקשרים של DNS תמיד תתבצענה 2 שאילתות – 1 עבור ipv4 ומיד לאח"כ תישלח שאילתה עבור ipv6. אנו נתייחס בהסברים לפלואו של הipv4, הipv6 שקול בהקשר של ההסברים שנתאר).

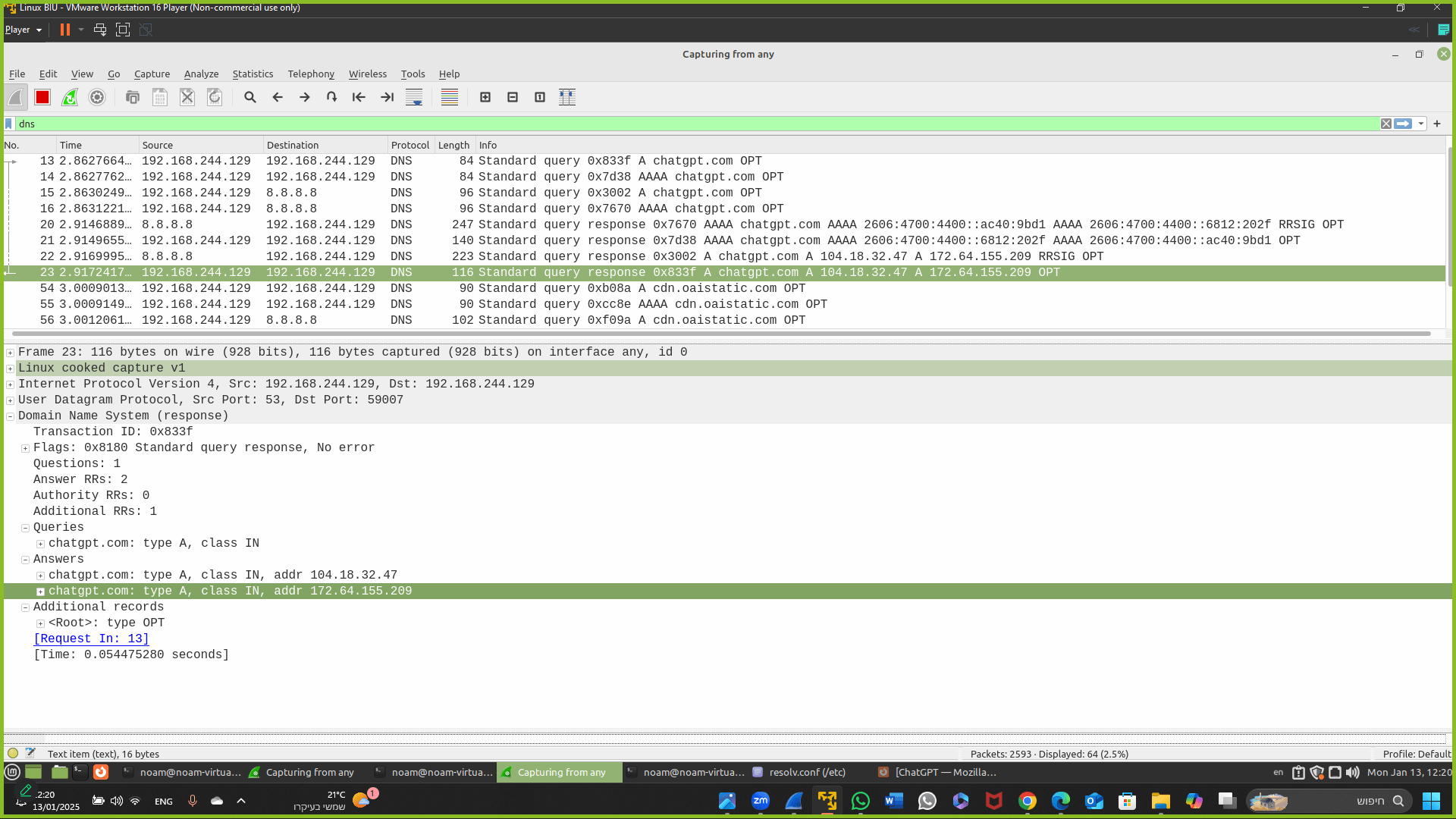


נשים לב שאכן לאחר ששרת הDNS שלנו מקבל את הבקשה הוא נעזר בשרת של גוגל 8.8.8.8 (בעזרת חיבור נפרד מזה שהיה לו עם הלקוח, שכן הוא פונה מפורט 42219 לפורט 53 בשרת של גוגל) בכדי להחזיר את התשובה (האתר שביקשנו). השרת שלנו כאמור מבקש זאת בעזרת IPV4 ולאחר מכן בעזרת IPV6.

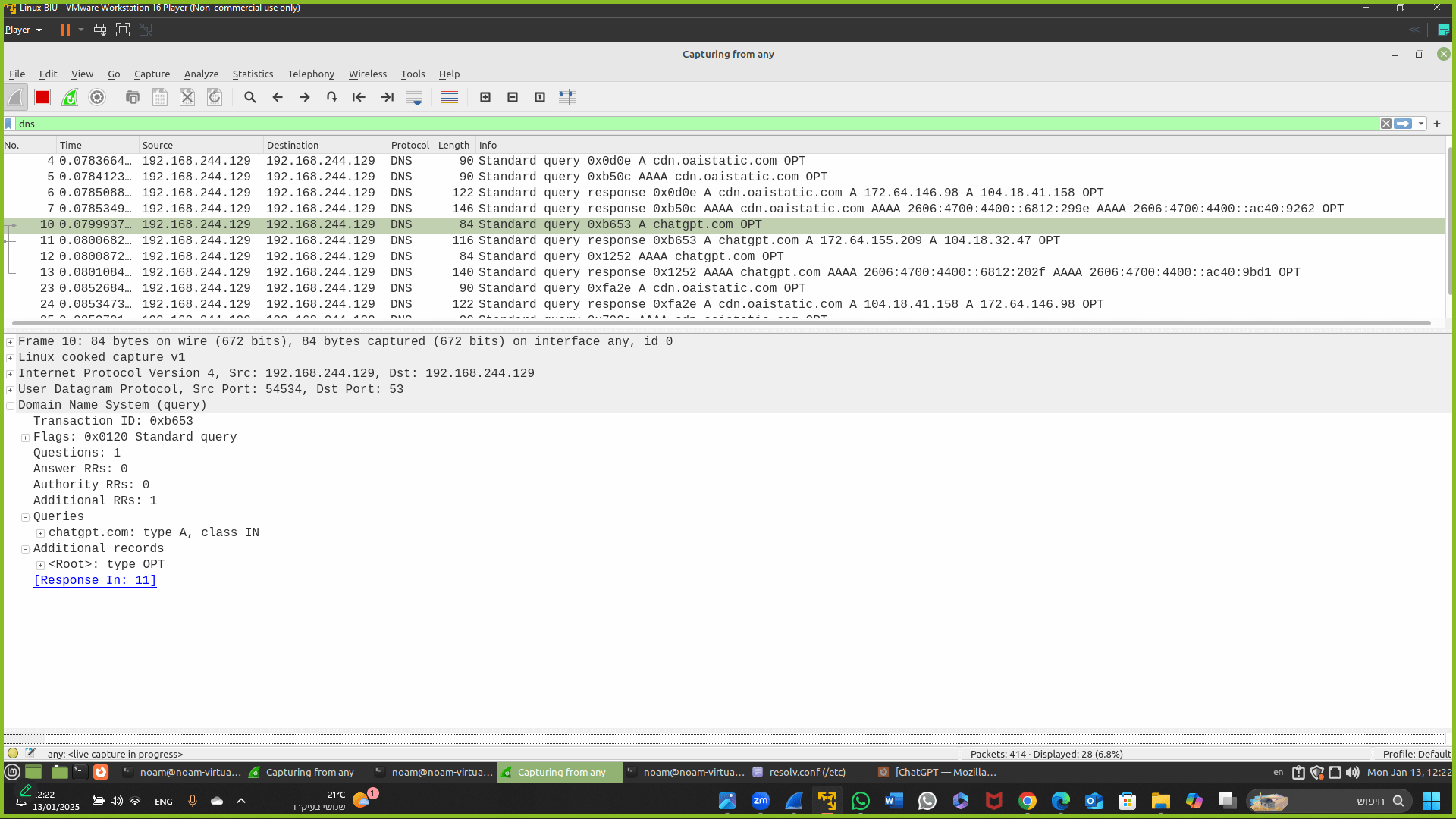


כאן, השרת של גוגל מחזיר אל השרת שלנו את התשובה עם כתובת הIP של האתר שביקשנו.  
זוהי הכתובת הסופית שכן השרת של גוגל שלח לנו רשומה מסוג A.

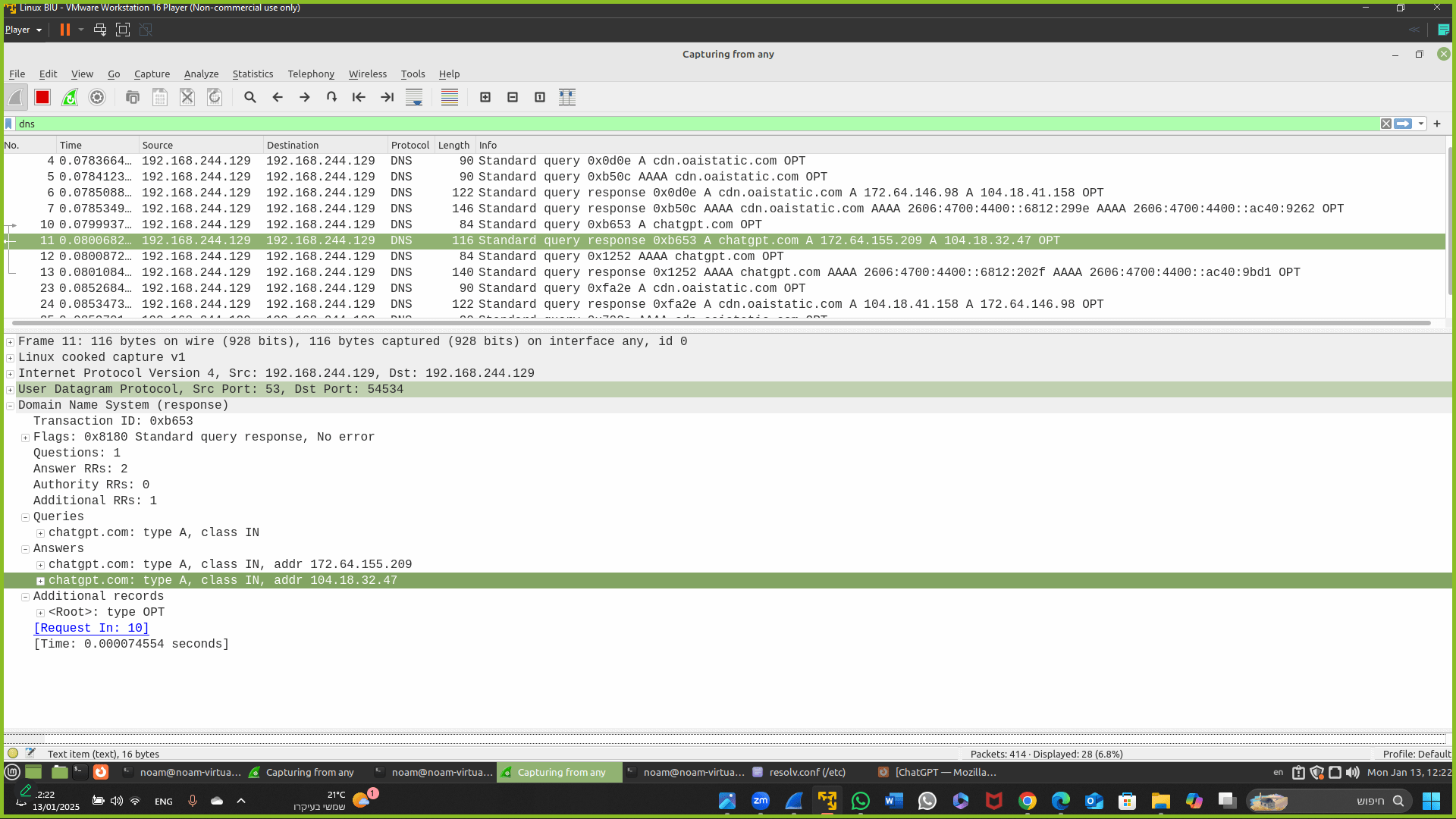
השרת שלנו, שולח אלינו (הלקוח) בחזרה את התשובה שקיבל מהשרת של גוגל.   
נראה כי בתשובה שקיבלנו, קיבלנו שתי אופציות לIP של האתר שביקשנו, יכול להיות מכמה סיבות שאת חלקן הזכרנו בהרצאה למשל חלוקת עומסים, או חברה חיצונית המספקת שירותי DNS.



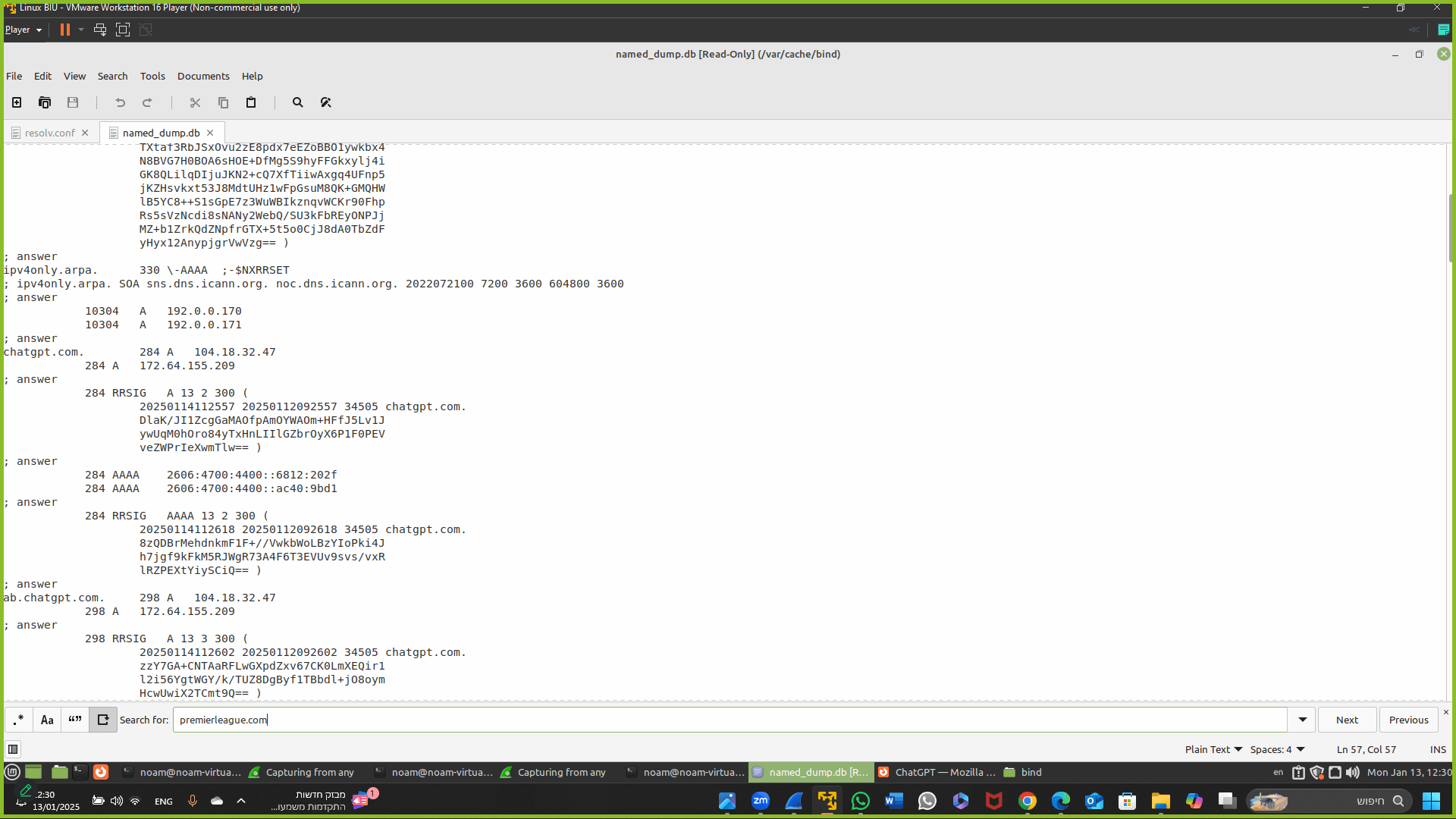
כעת, ננסה לגשת שוב פעם אל האתר (מתוך תקווה שכתובתו תהיה כבר בקאש).



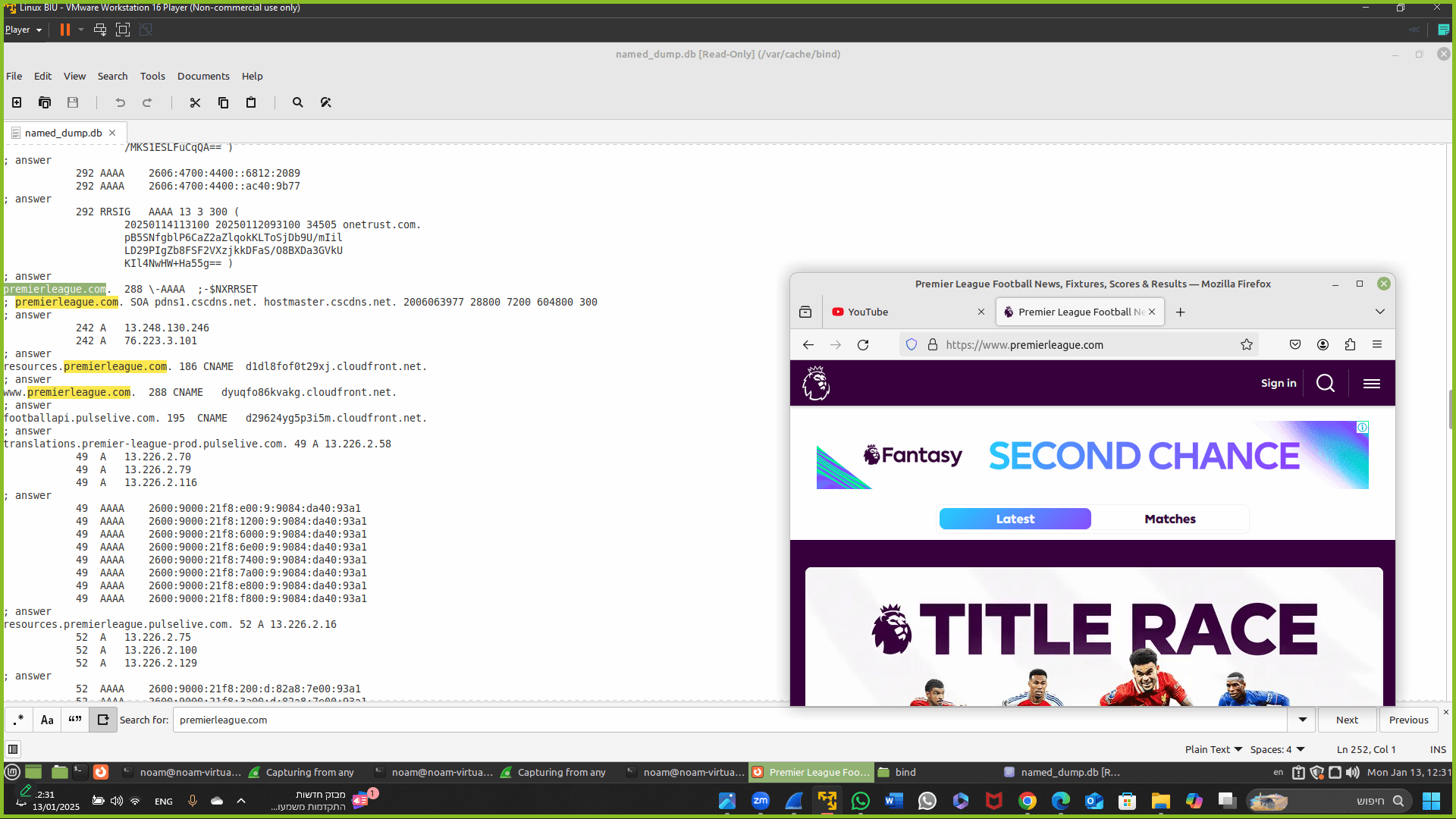
נשים לב שבהודעה העוקבת מיד השרת החזיר ללקוח את התשובה (ה- dst וה- src התהפכו). בשונה מפעם קודמת לא היה שלב ביניים של פנייה לשרתי הDNS של גוגל זאת כתוצאה מהעובדה שהשרת שלנו שמר את הכתובת כבר בקאש שלו עוד מהבקשה הקודמת, ולכן לא היה צריך לבקש מהשרתים החיצוניים שוב. סה"כ קיבלנו באמת את 2 הכתובות של האתר שביקשנו (בדיוק כמו הנסיון הקודם).



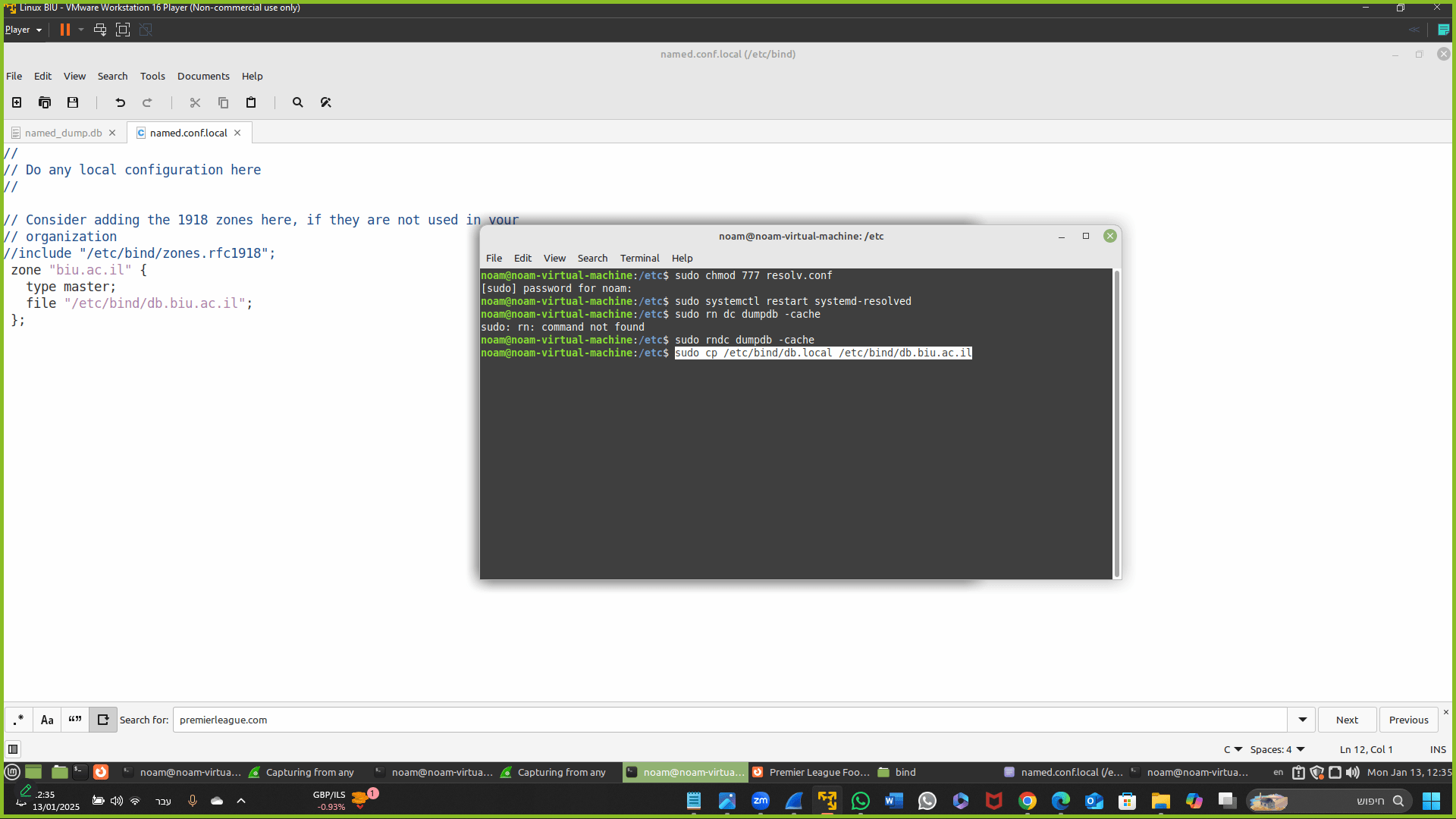
נייצא את הרשומות השמורות בקאש לקובץ בשם named\_dump.db (באמצעות ביצוע הפקודה - sudo rndc dumpdb -cache), ונכנס אל הקובץ שיצרנו. נבחין כי אכן הדומיין (האתר של צאט PGT) שחיפשנו נמצא בו.



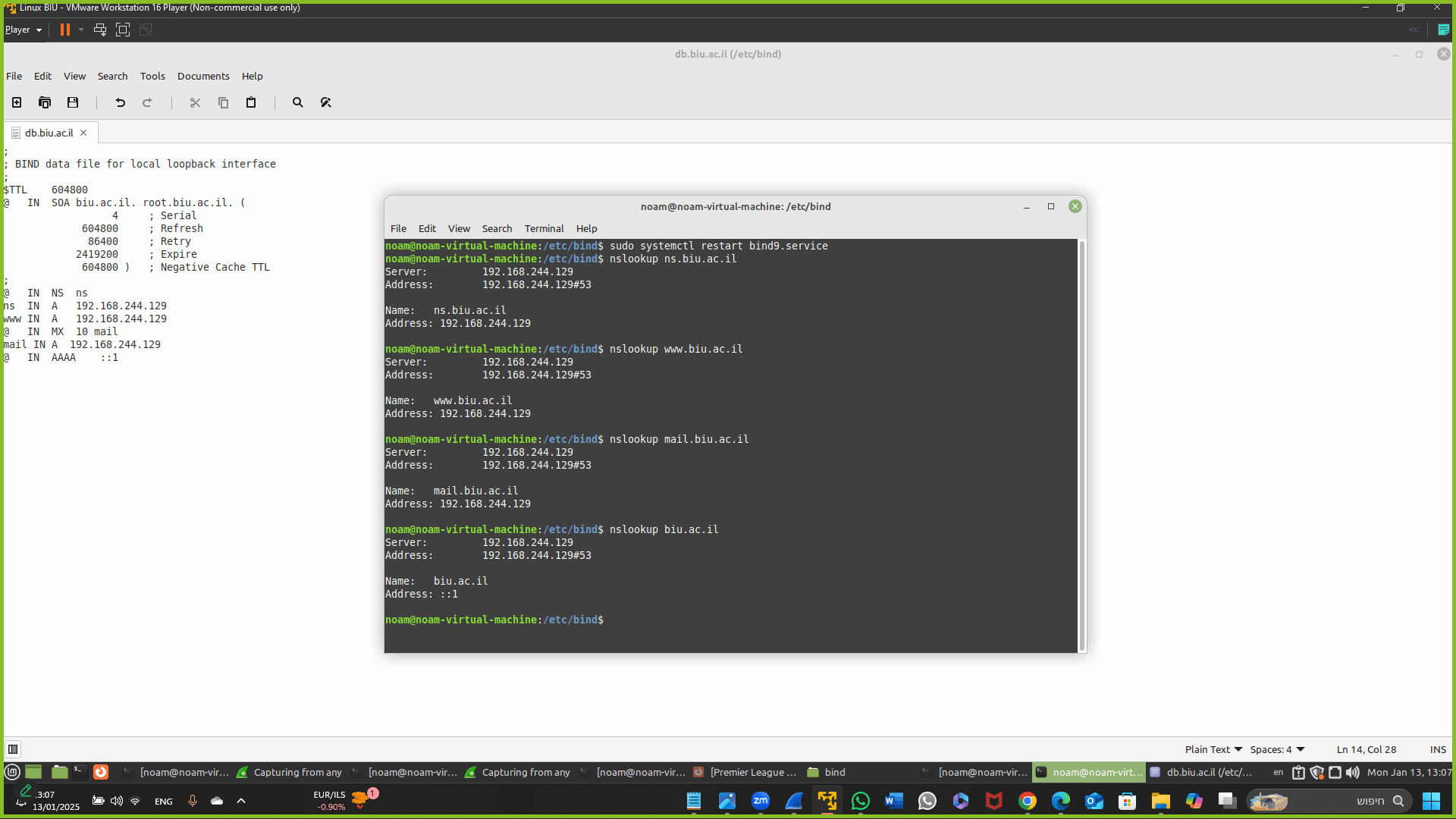
נוודא שאכן הדומיין שמחפשים נכנס אל הקאש. נשים לב כי בתחתית המסך חיפשנו את כתובת האתר של הליגה האנגלית ווידאנו שהוא לא קיים כרגע בקאש. כעת נגלוש אל אתר הליגה האנגלית ולאחר מכן נייצא שוב את הקובץ ע"פ הפקודה שציינו מקודם. ונגלה שאכן הדומיין נשמר בקאש של השרת שלנו.



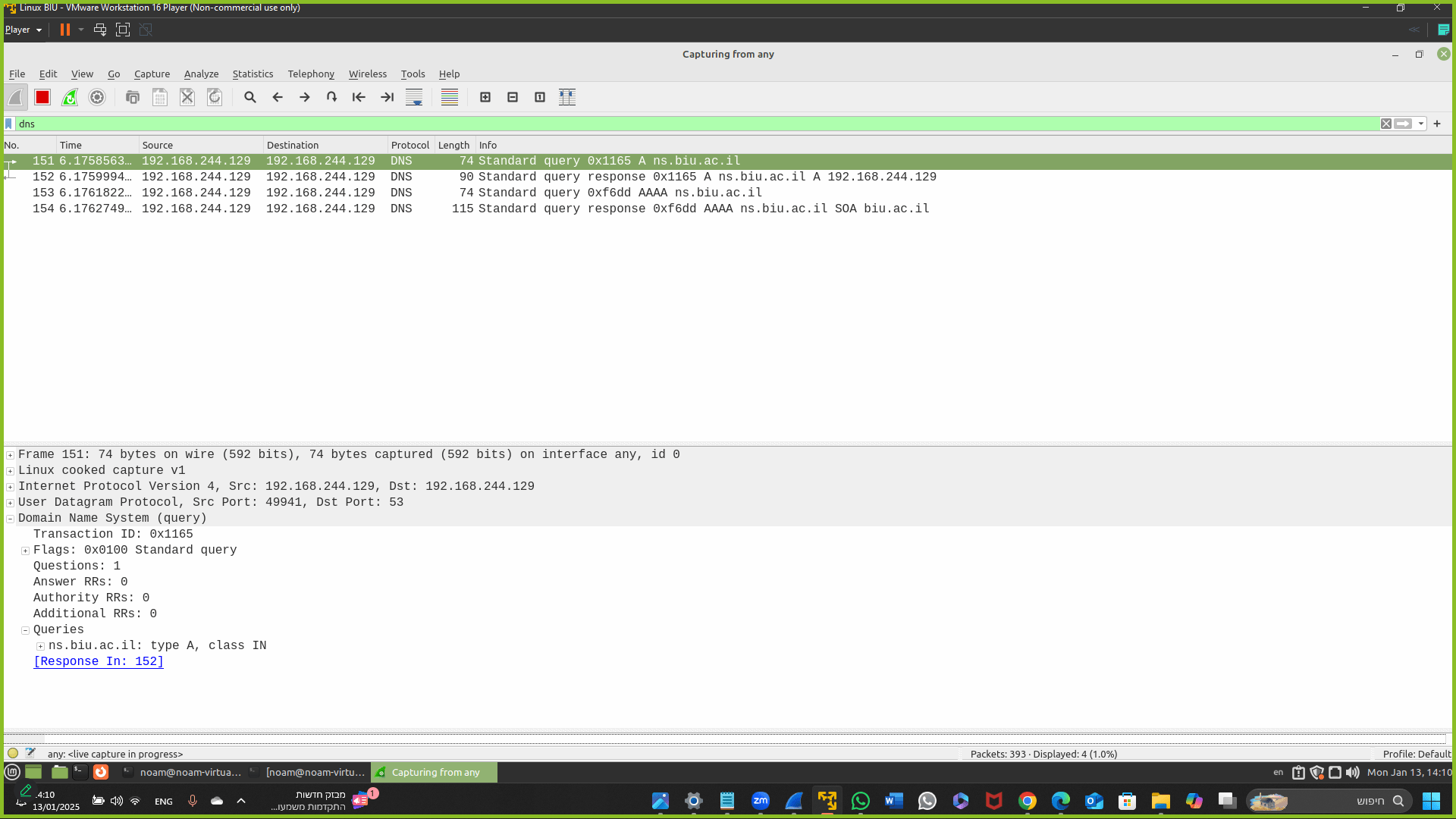
כעת, נגדיר את שרת הDNS שלנו להיות אוטורטיבי. נכנס אל הקובץ /etc/bind/named.conf.local ונגדיר לשרת קובץ שיכיל את הניהול של הדומיין biu.ac.il (במלבן האדום). ניצור את הקובץ בעזרת תבנית קיימת שנמצאת ב /etc/bind/db.local ולכן נריץ את הפקודה שבמלבן הכחול אשר תיצור לנו עותק של התבנית הדרושה.



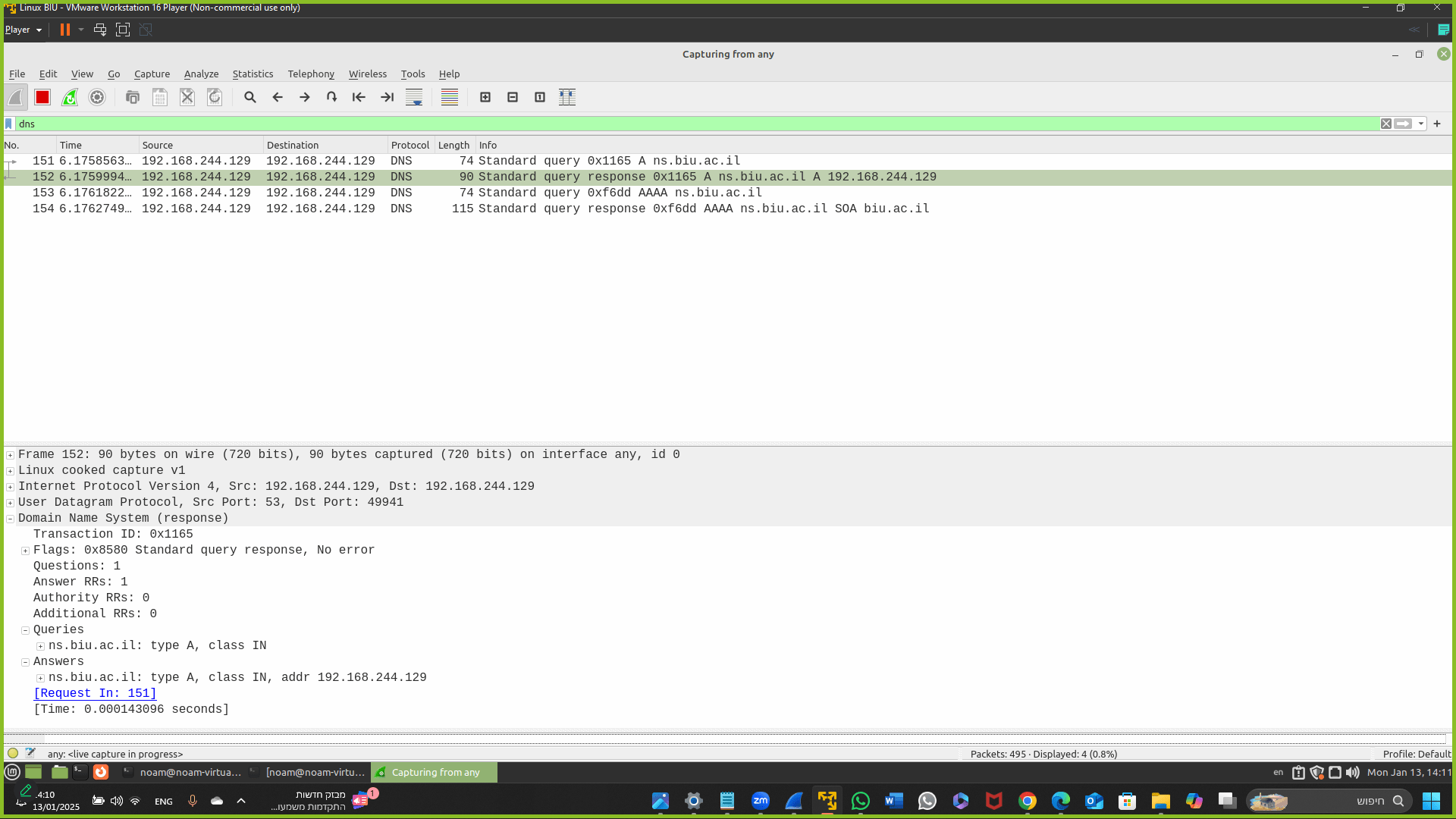
נכנס אל הקובץ שיצרנו ונגדיר את כל רשומות הDNS הבסיסיות ששרת שאחראי על דומיין צריך לדעת לענות עליהן. נבהיר שבכל מקום שבו רשום @ - הוא יוחלף אוטו' בדומיין biu.ac.il, מה שמאפשר כתיבה נקייה יותר ומעט מתחכמת.  
נשנה את רשומת הNS כך שתמפה בין הדומיין ns.biu.ac.il לכתובת הIP של המחשב שלנו (במלבן האדום).  
ניצור רשומת A שתמפה בין [www.biu.ac.il](http://www.biu.ac.il) לכתובת הIP של המחשב שלנו (במלבן הכחול).  
ניצור רשומת MX (מייל) ועל כך נמפה בין biu.ac.il ל - mail.biu.ac.il, בנוסף לרשומת A   
לmail.biu.ac.il - (במלבן הירוק) התשויך גם היא לIP של המחשב שלנו.  
נעלה את ערך הserial בכדי השינויים יכנסו לתוקף ונשמור אותם בעזרת הפקודה –sudo systemctl restart bind9.service. כעת נבצע שאילתות DNS מול השרת שלנו, ונוודא שאנחנו אכן מקבלים את הכתובות שהגדרנו.



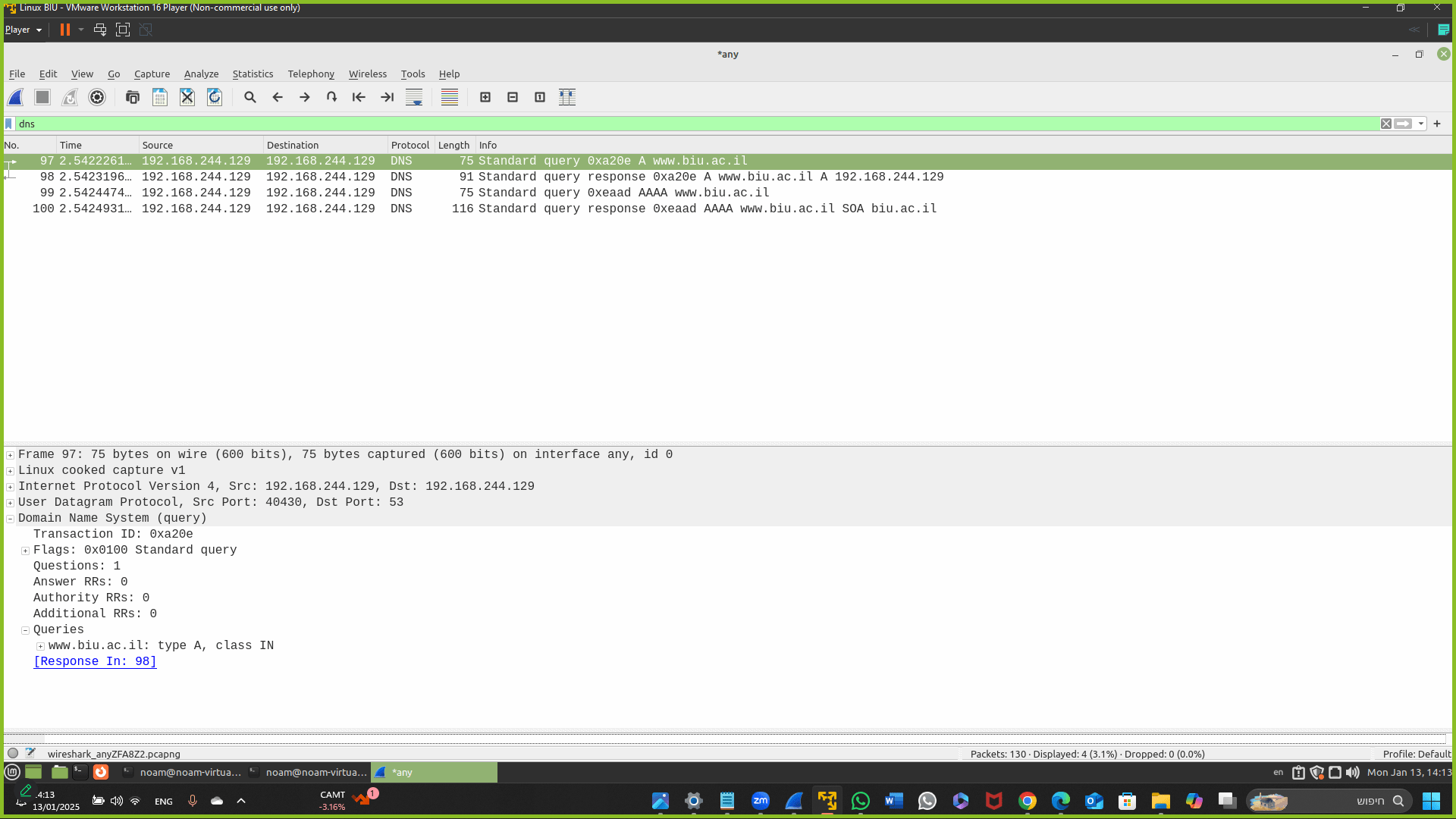
נתבונן בתיעוד השאליתות שביצענו דרך הכריש. ראשית, שאילת DNS בכדי עבור ns.biu.ac.il (ע"י הפקודהnslookup ns.biu.ac.il ). נראה בכריש שהלקוח שלח לשרת שלנו בקשה למצוא את כתובת הns.



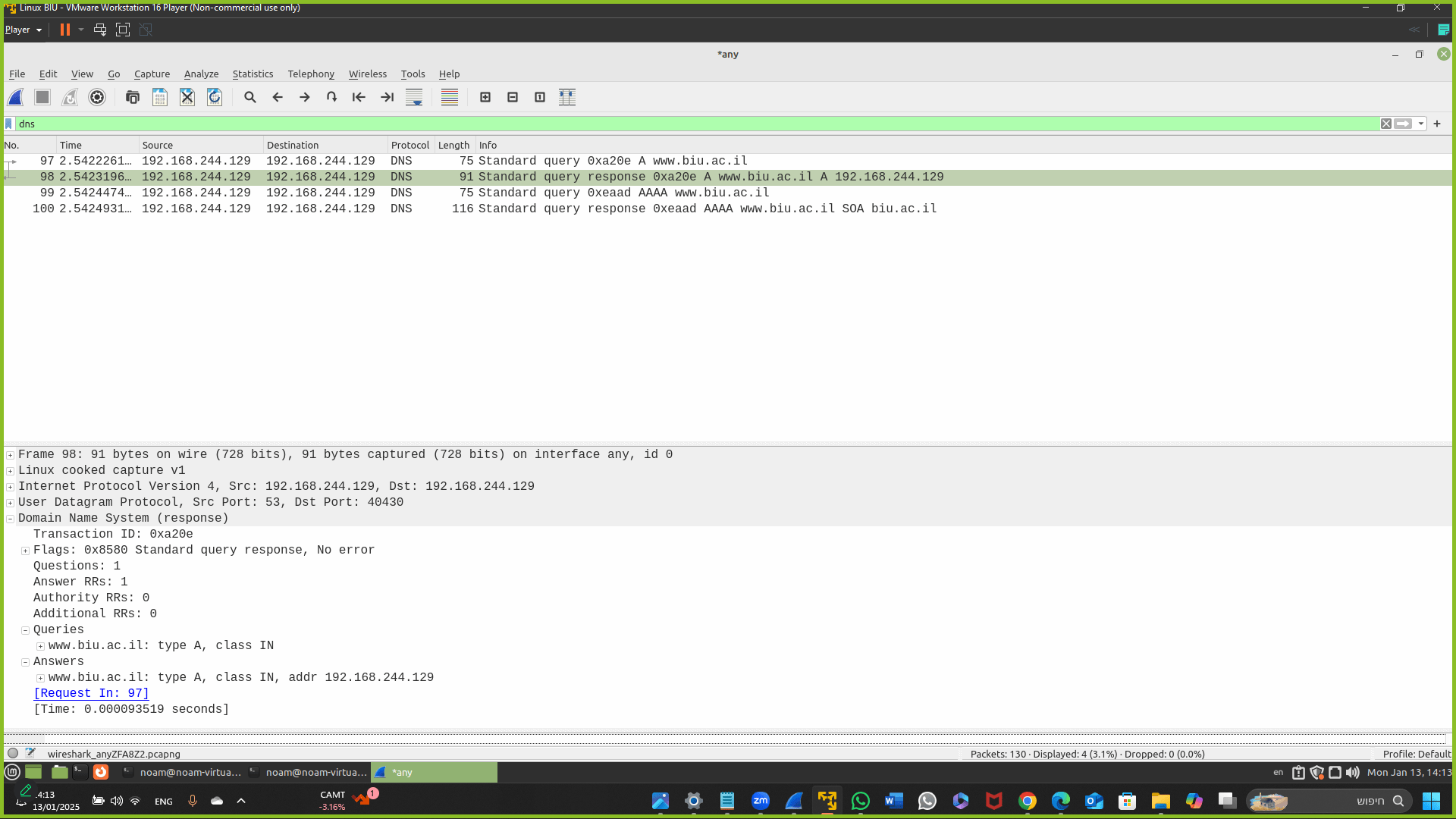
השרת מחזיר לו את התשובה ששמורה לו בקובץ שיצרנו מקודם, כלומר הוא מחזיר לו את IP של הדומיין ns.biu.ac.il כפי שהגדרנו. (כפי שאמרנו בהתחלה, מתבצעת מיד פניה גם כן עבור הipv6, וגם היא מקבלת מענה מהשרת שהגדרנו)



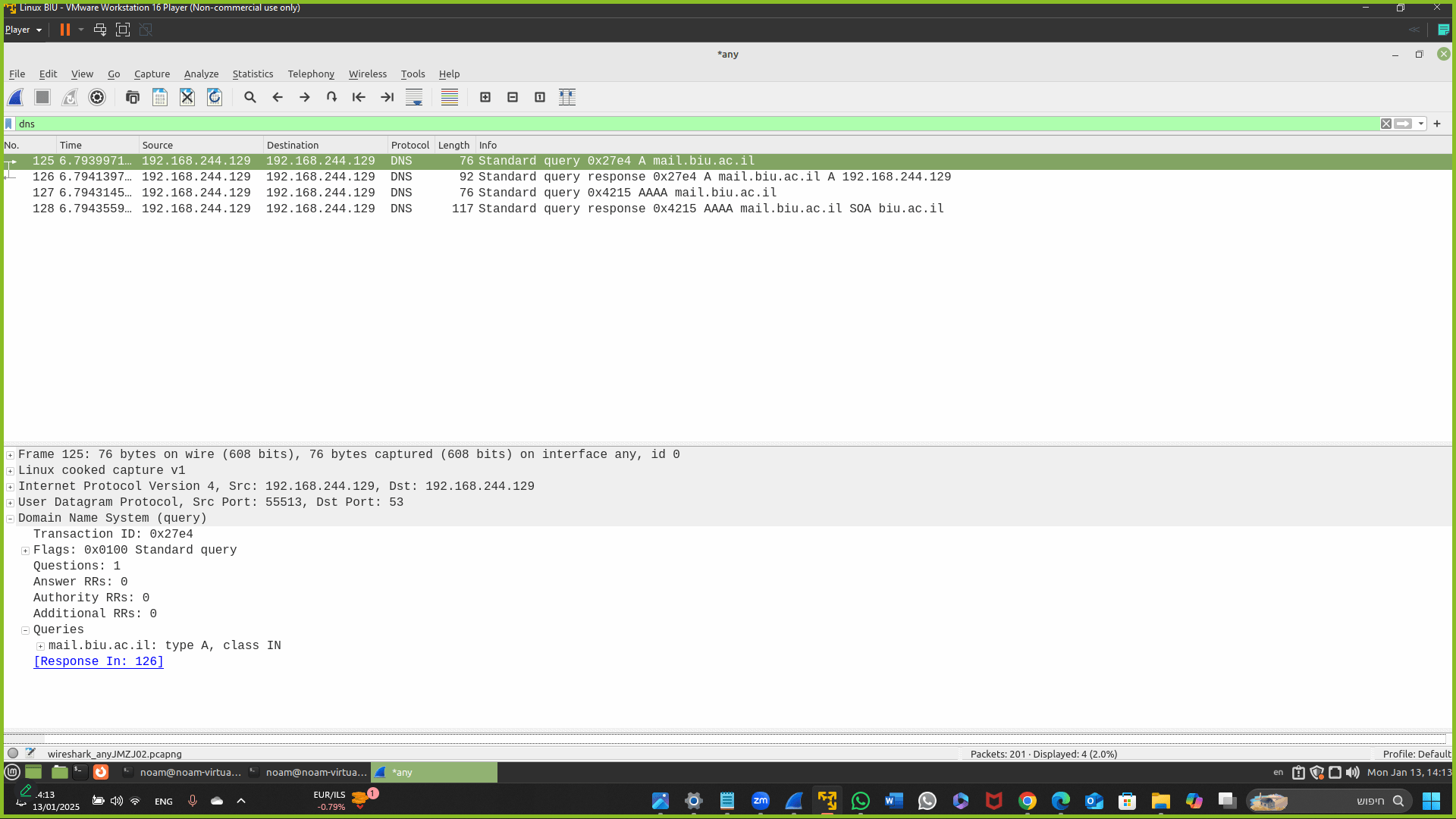
כעת נבצע שאילתת DNS אחרת בעזרת nslookup. נבקש את הIP של www.biu.ac.il מהשרת שלנו.



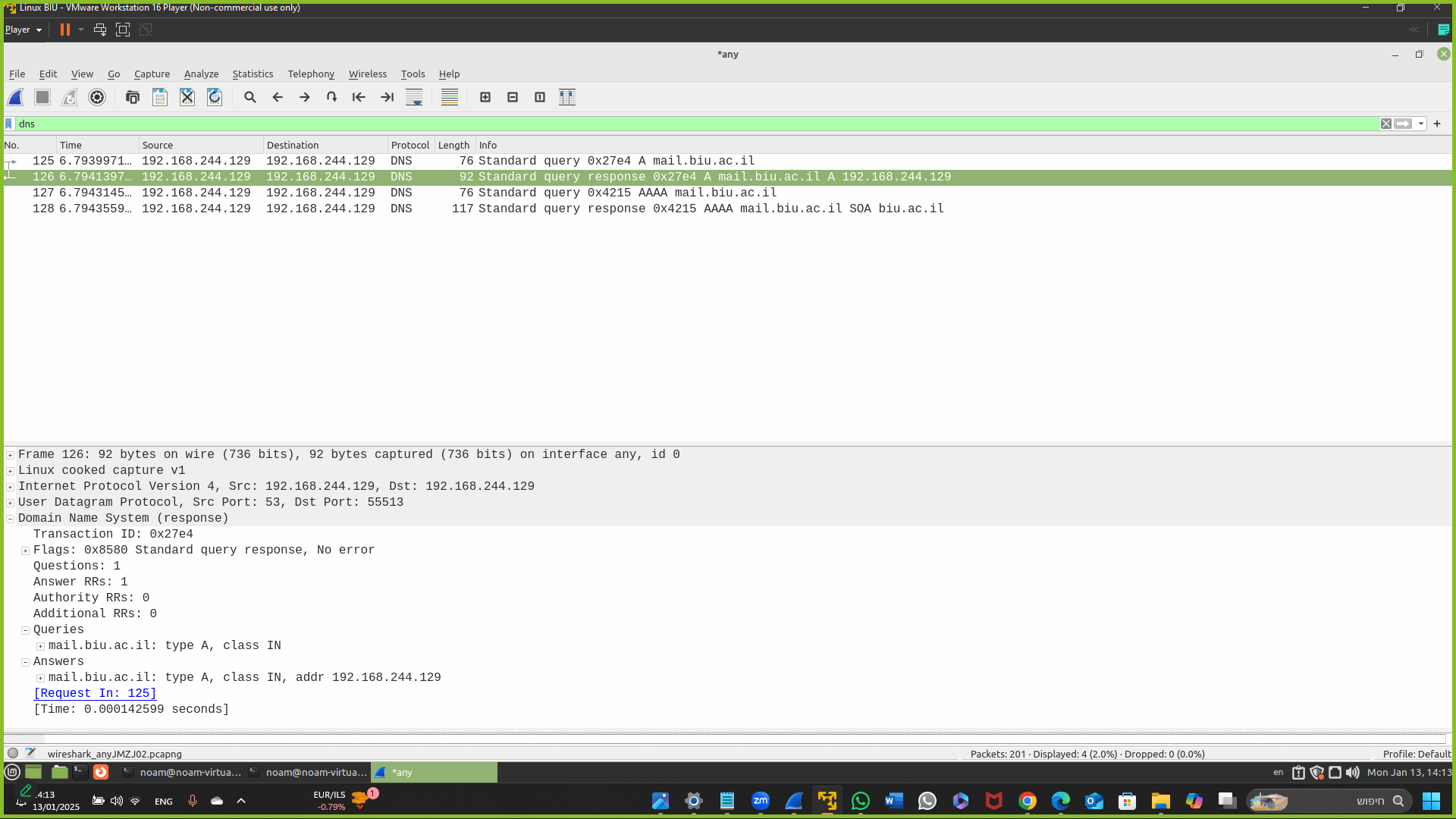
השרת מחזיר לו את IP של הדומיין www.biu.ac.il כפי שהגדרנו. (שוב פניה לipv6 לאחר מכן)



נבצע שאילתת DNS אחרת בעזרת nslookup. נבקש את הIP של שרת המייל [mail.biu.ac.il](http://www.biu.ac.il) מהשרת שלנו.



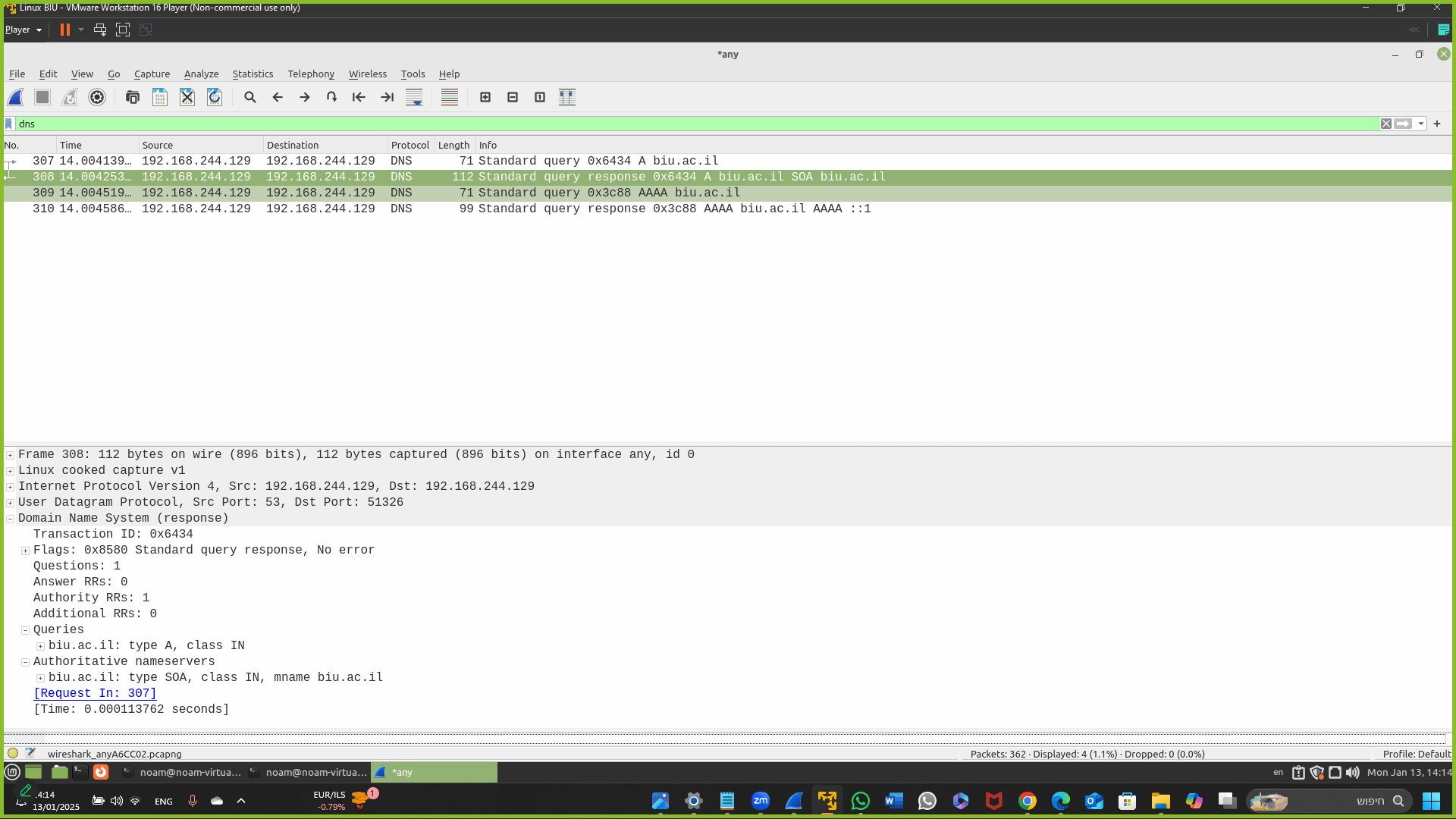
השרת מחזיר לו את IP של הדומיין mail.biu.ac.il כפי שהגדרנו. (שוב פניה לipv6 לאחר מכן)



נבצע שאילתת DNS אחרונה בעזרת nslookup. נבקש את IP של הדומיין biu.ac.il מהשרת שלנו.



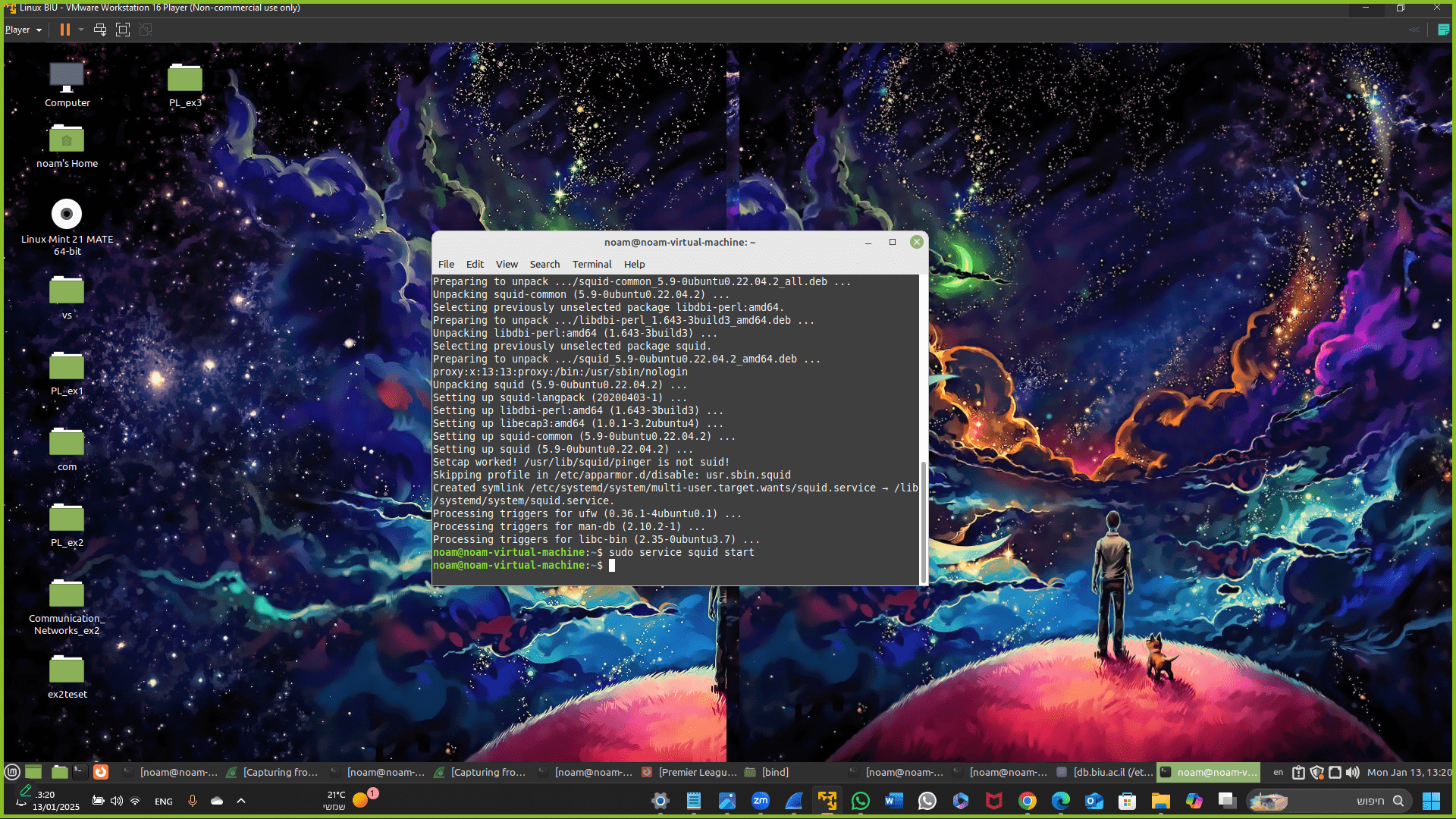
השרת מחזיר לו את שרת השמות של הדומיין biu.ac.il כפי שהגדרנו אותו. (שוב פניה לipv6 לאחר מכן)



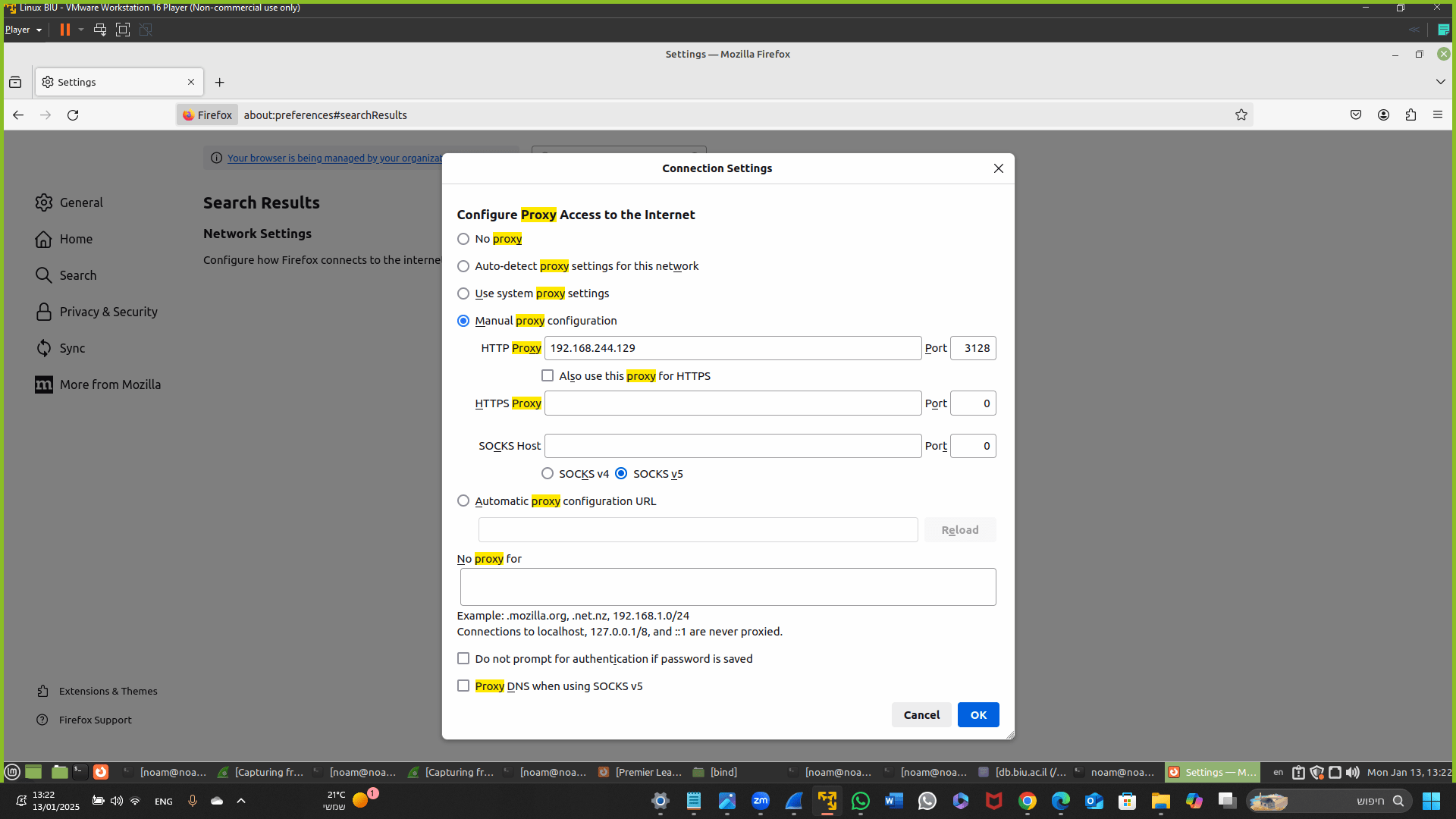
*עד כאן חלק שני*

*חלק שלישי*

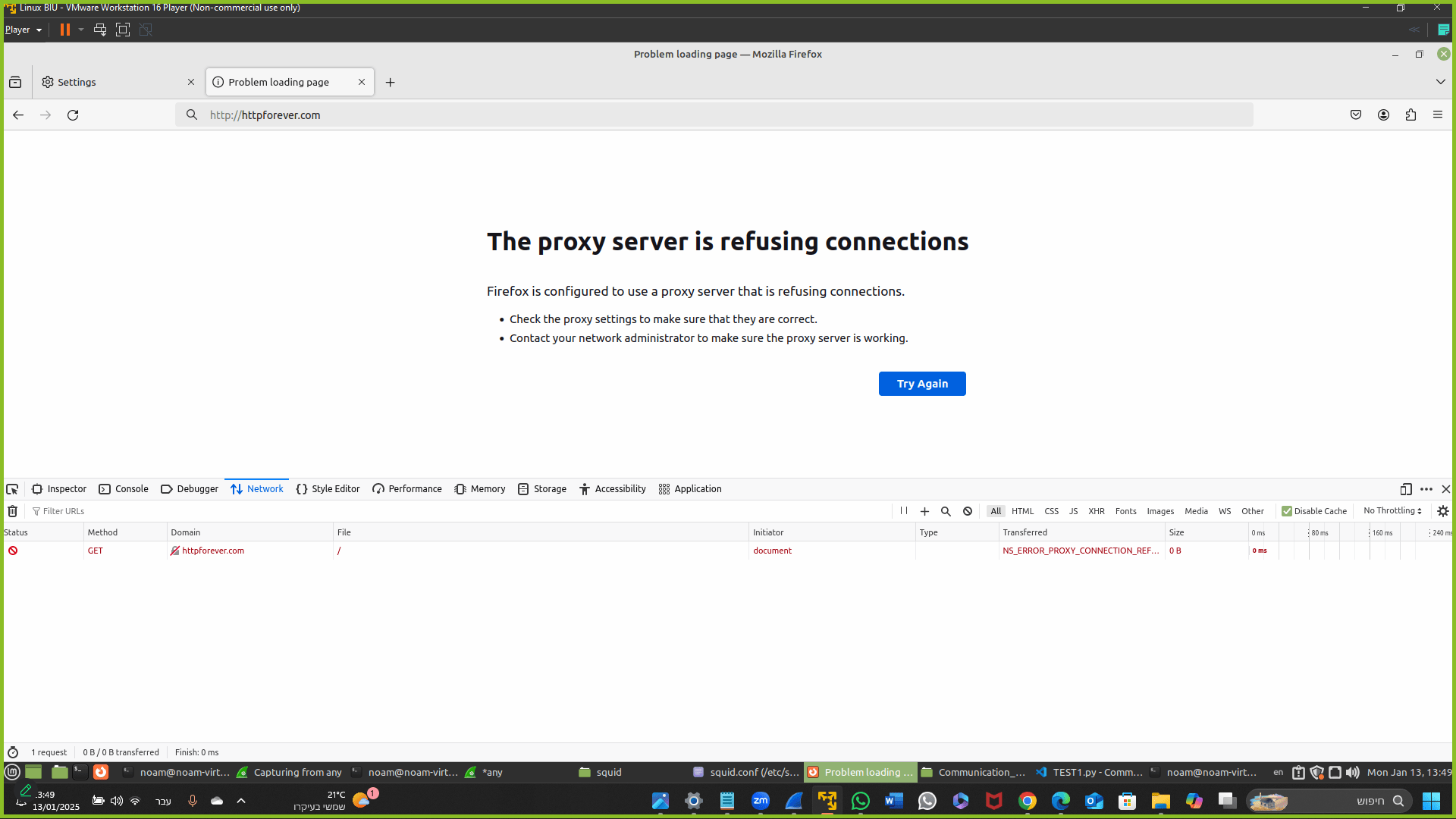
נתקין את שרת הפרוקסי – squid ונריץ אותו.



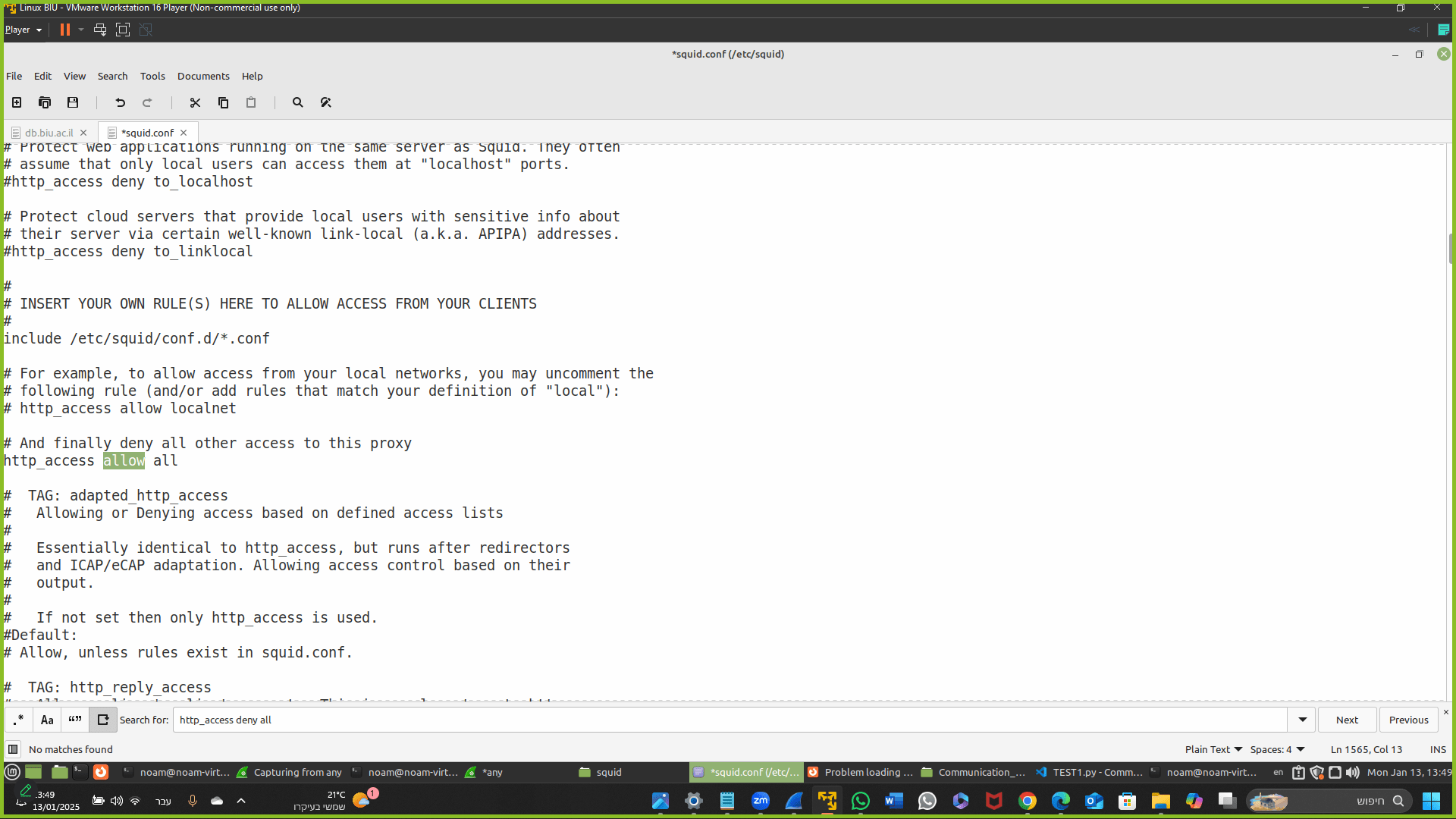
נפעיל את הדפדפן ונגדיר בהגדרות שלו כך ששרת הsquid שהתקנו יהווה שרת הפרוקסי שלו ע"י הזנת כתובת הIp של השרת את הפורט של הפרוקסי – 3128.



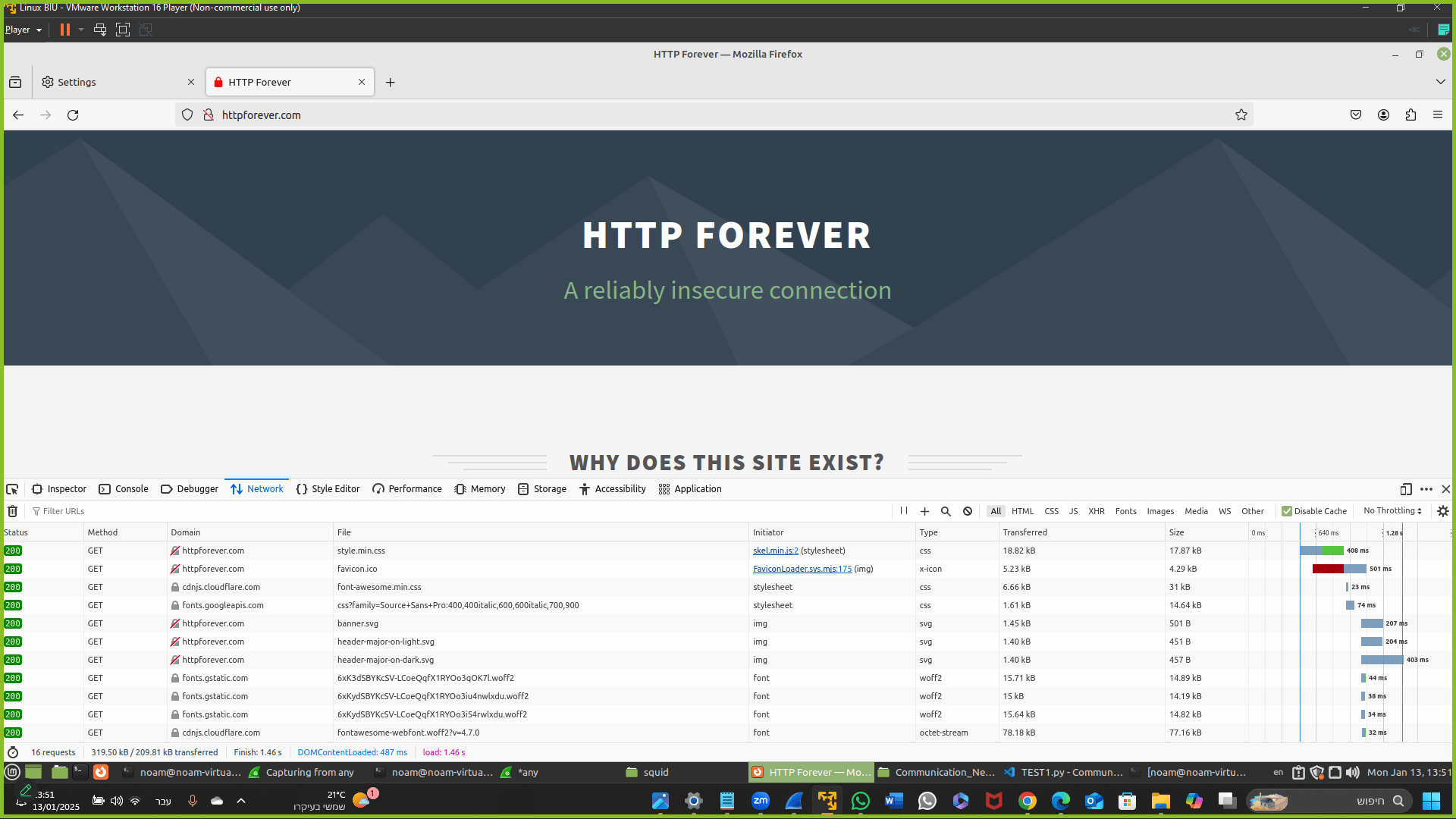
ננסה כעת לגלוש לאתר האינטרנט http:// httpforever.com, ונשים לב שאנו לא מצליחים ושרת הפרוקסי שהגדרנו מסרב לחבר אותנו לאתר.



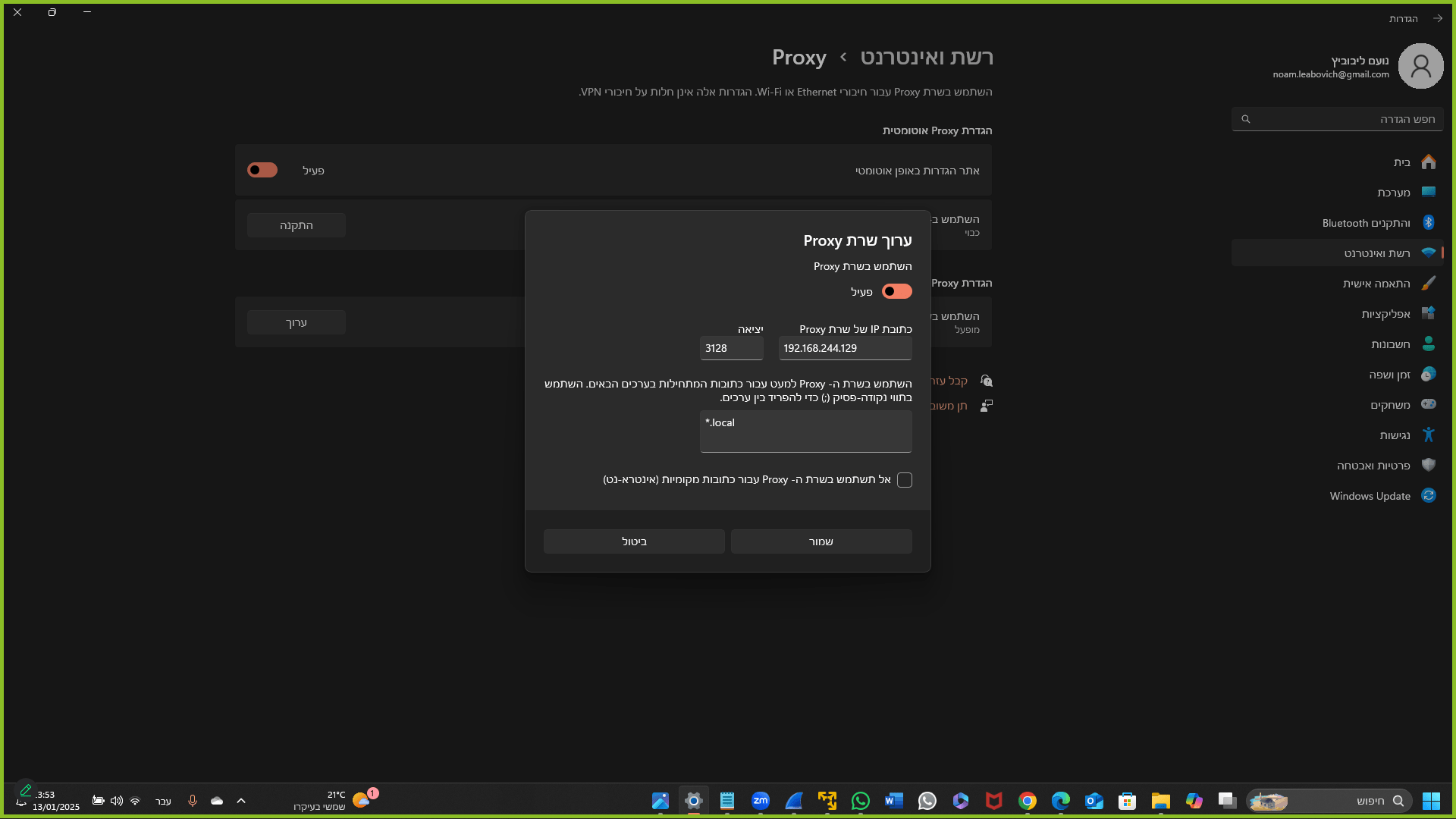
נשנה את ההגדרה בשרת הפרוקסי שגורם לזה. נשנה בקובץ squid.conf את הסטטוס deny להיות allow.



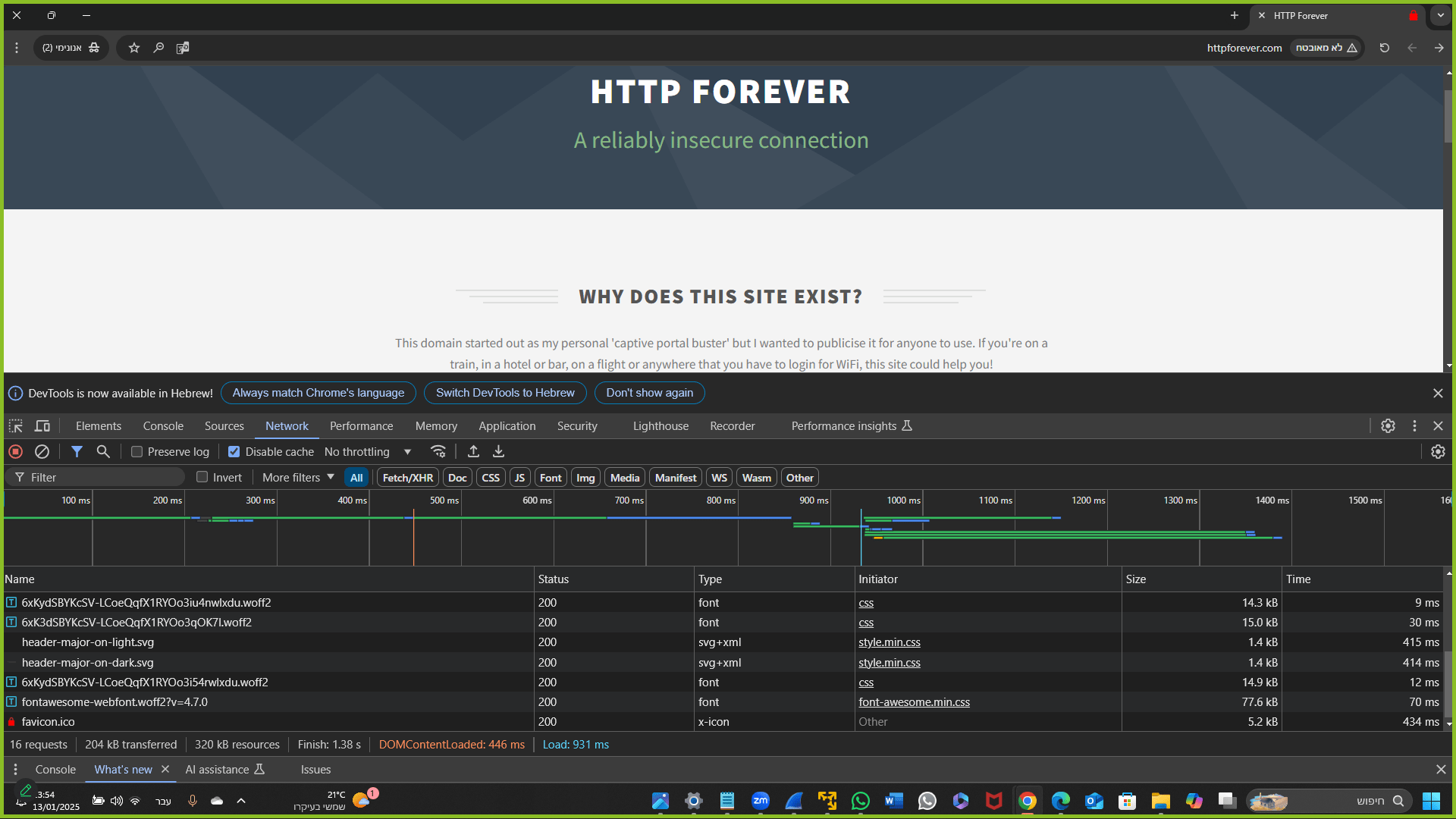
תאתחל את השרת וננסה לגשת שוב לאתר <http://httpforever>. נשים לב כי כעת העמוד נטען ומוצג במלואו, ואכן פניות האינטרנט עוברות בהצלחה את המיסוך של שרת הפרוקסי, אשר לאחר השינוי שעשינו אמור להתיר הכל.



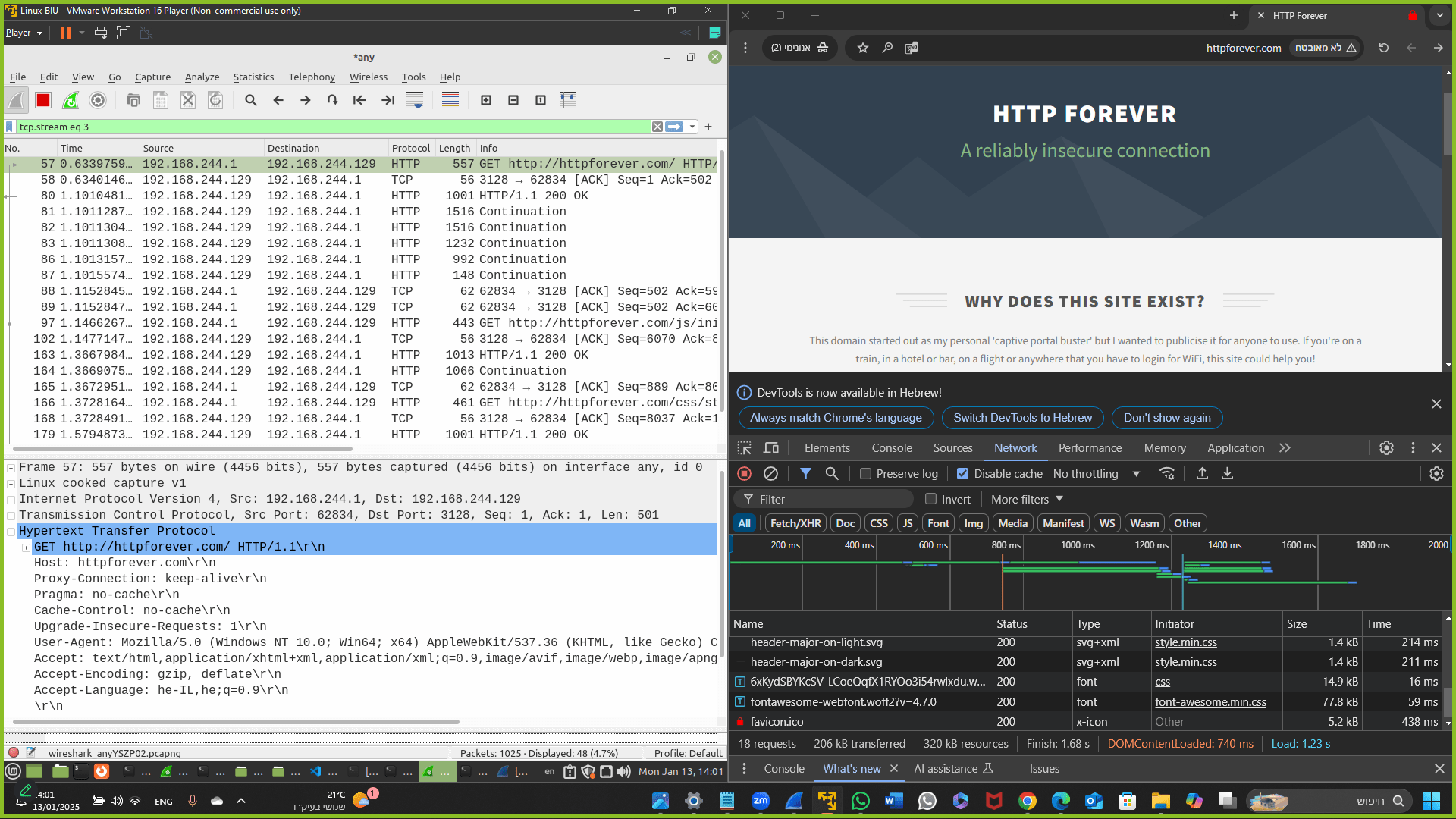
נגדיר גם כן את השרת השני שלא מריץ את הפרוקסי כך שיגלוש גם הוא דרך שרת הפרוקסי שלנו.



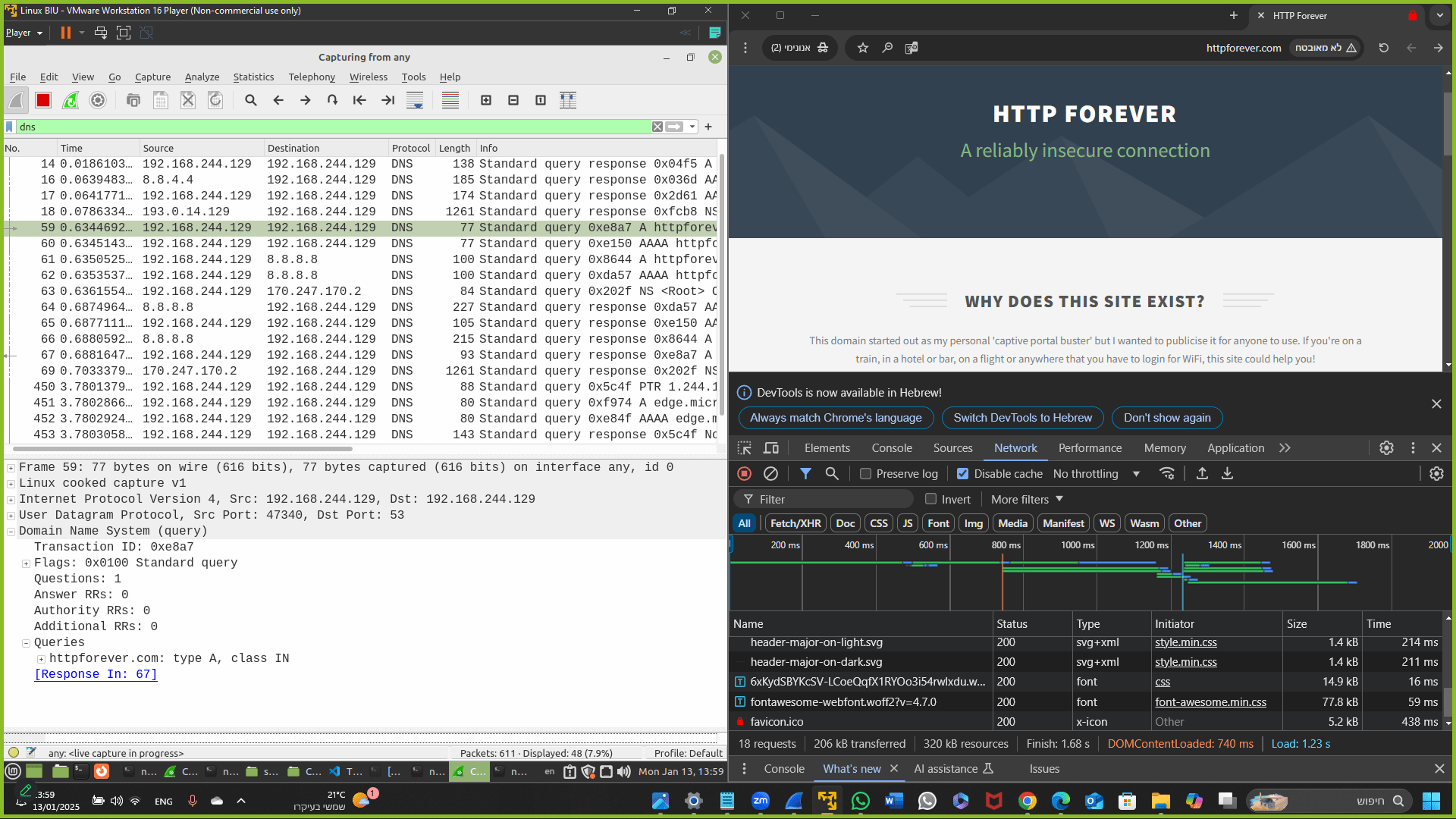
נוודא שאכן גם דרכו אנו מצליחים לגלוש לאתר. מצליחים.



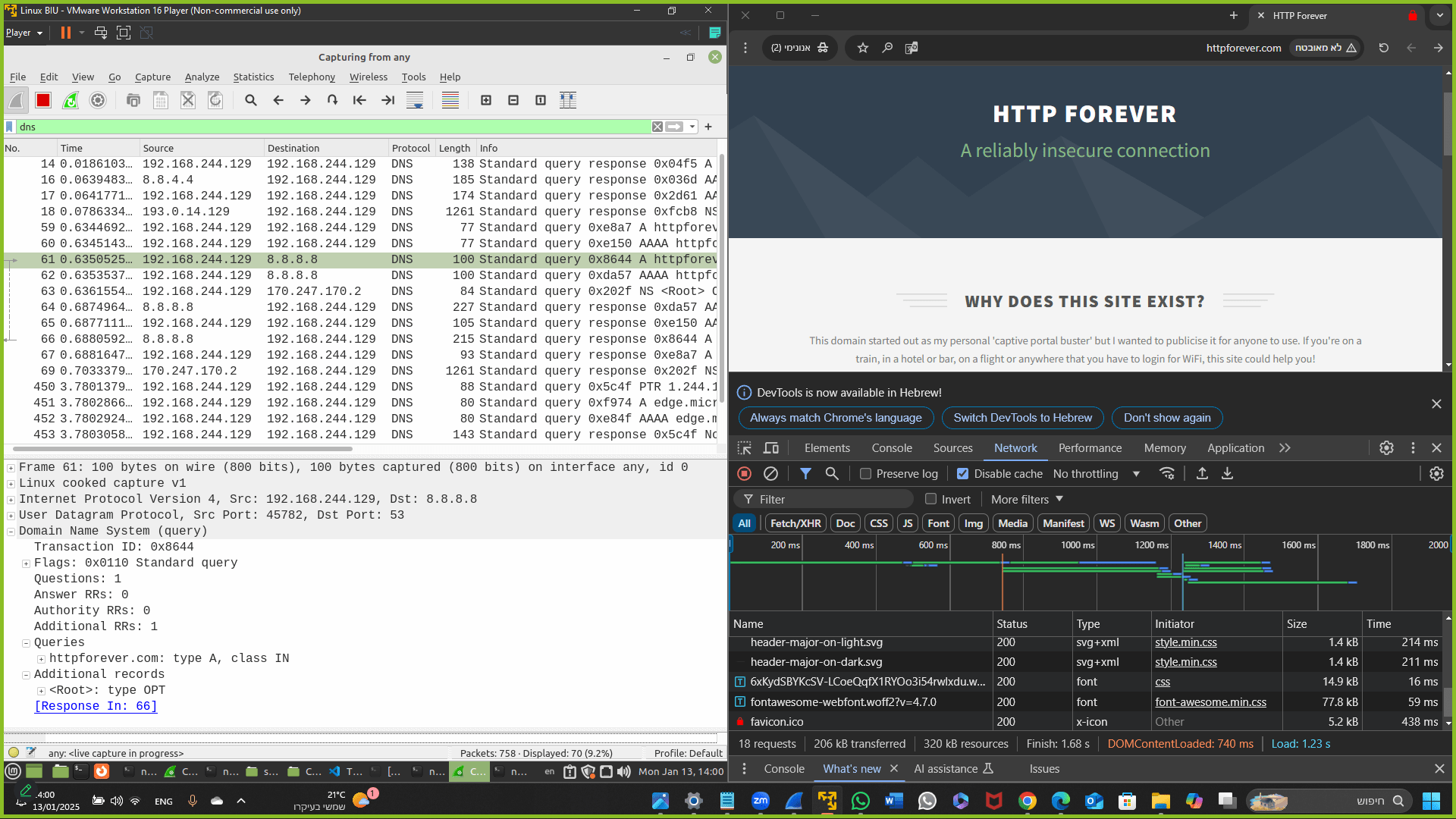
נתבונן כעת בתעבורה שקוראת בתקשורת הזו מאחורי הקלעים בעזרת הכריש. ראשית אנו רואים כי הדפדפן שלנו מבצע פניית get של פרוטוקול http **לשרת הפרוקסי** שלנו (הרץ בכתובת הIp – 192.168.244.129).



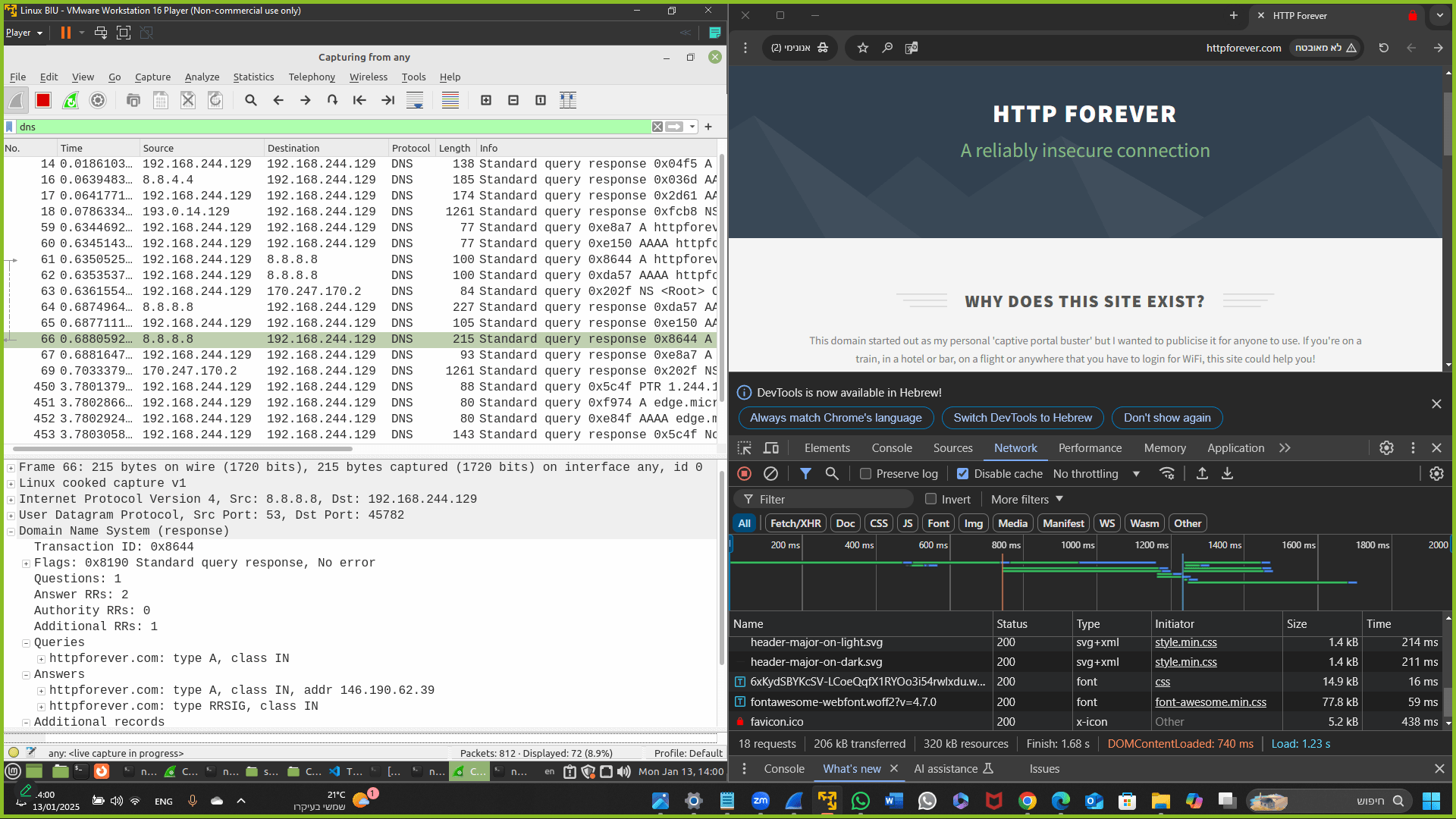
כעת לאחר שקיבל את הבקשה שרת הפרוקסי מריץ DNS על מנת למצוא את כתובת הIp של אותו שרת ממנו הוא צריך להחזיר את המשאב ללקוח (שהוא למעשה המחשב השני שלנו). נשים לב ששרת הDNS אליו הוא פונה זה השרת אותו הגדרנו בחלק הקודם (ניתן לראות זאת לפי הIp והפורט אליו הוא פונה שהם בדיוק הIp של אותו מחשב עצמו של שרת הפרוקסי), ובשל כך זה מתבצע בחיבור נפרד, מזה שדרכו שרת הפרוקסי קיבל את הבקשה של הלקוח.

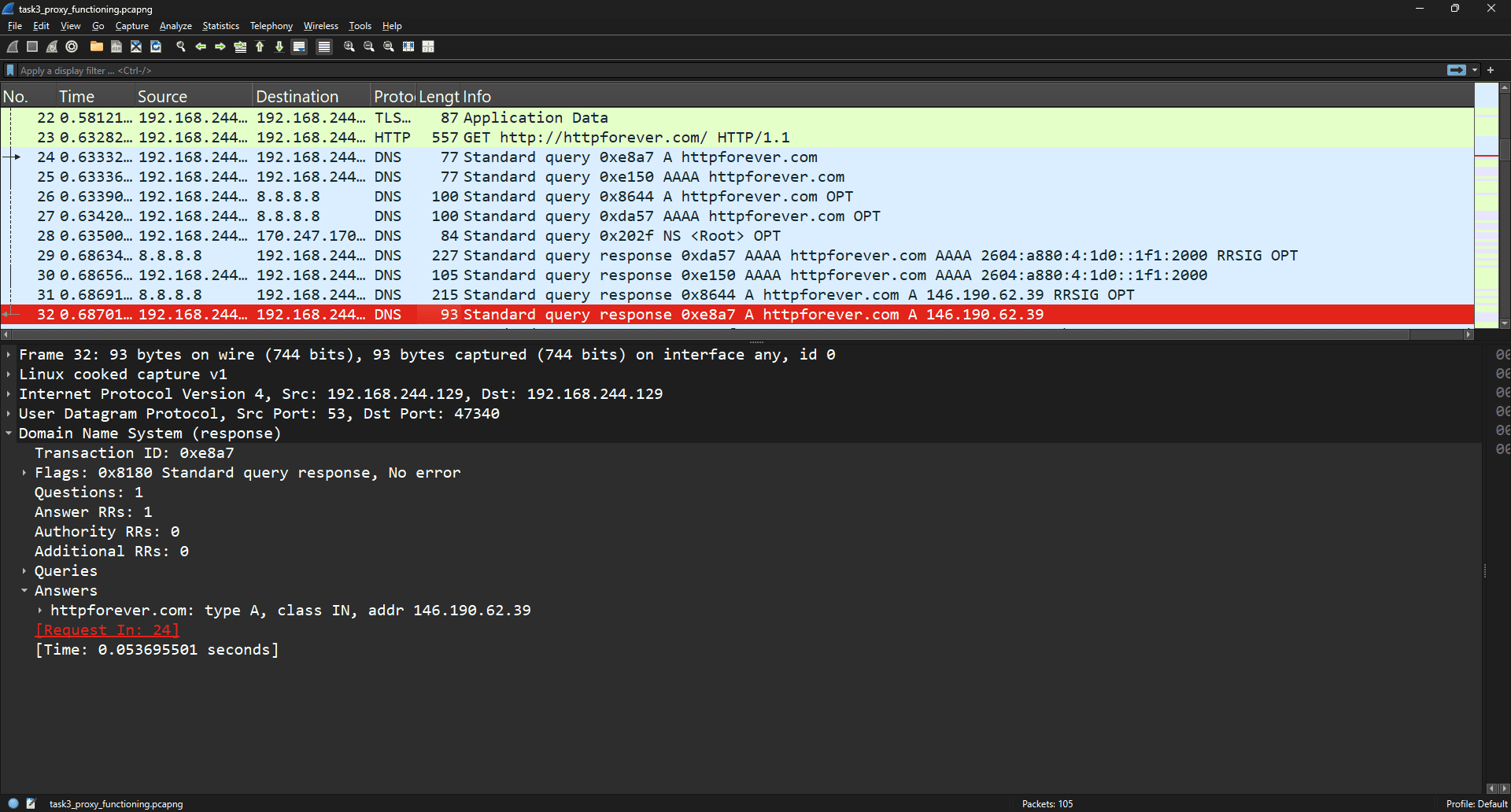


מכאן לאחר ששרת **הDNS** שלנו שרץ באותו מחשב של שרת הפרוקסי מקבל את בקשת הDNS, הוא מפנה את השאילתה לשרת של גוגל (בדומה למה שראינו בחלק השני), גם כן בחיבור נפרד.



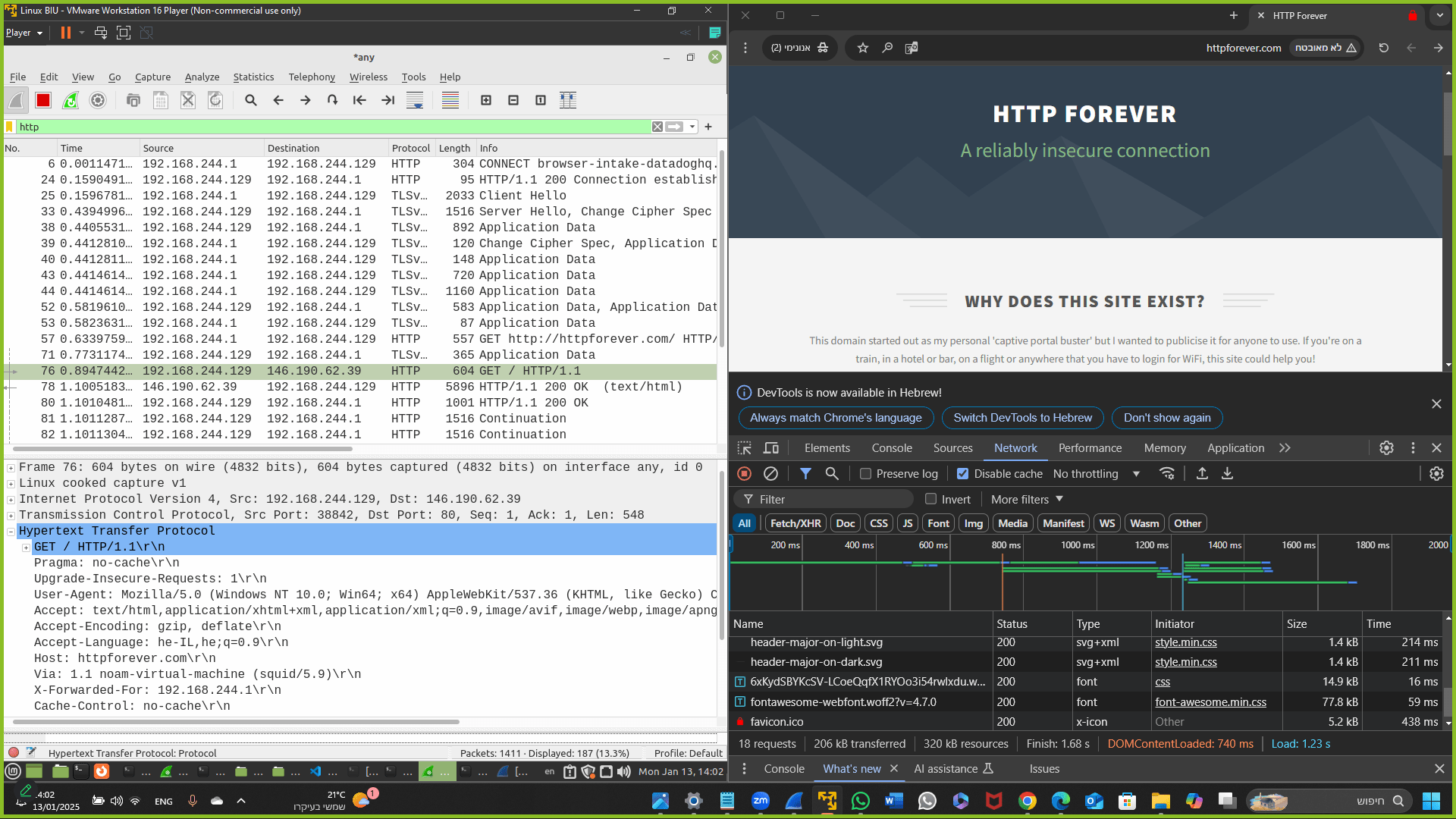
השרת של גוגל מחזיר כתשובה את כתובת הIp לשרת הDNS המקומי שלנו.

לאחר ששרת **הDNS** שלנו מקבל את הכתובת של האתר הוא מסוגל לענות בחזרה את התשובה (146.190.62.39) לשאילתת הDNS של כתובת האתר httpforever ששאל אותו שרת **הפרוקסי שלנו**.

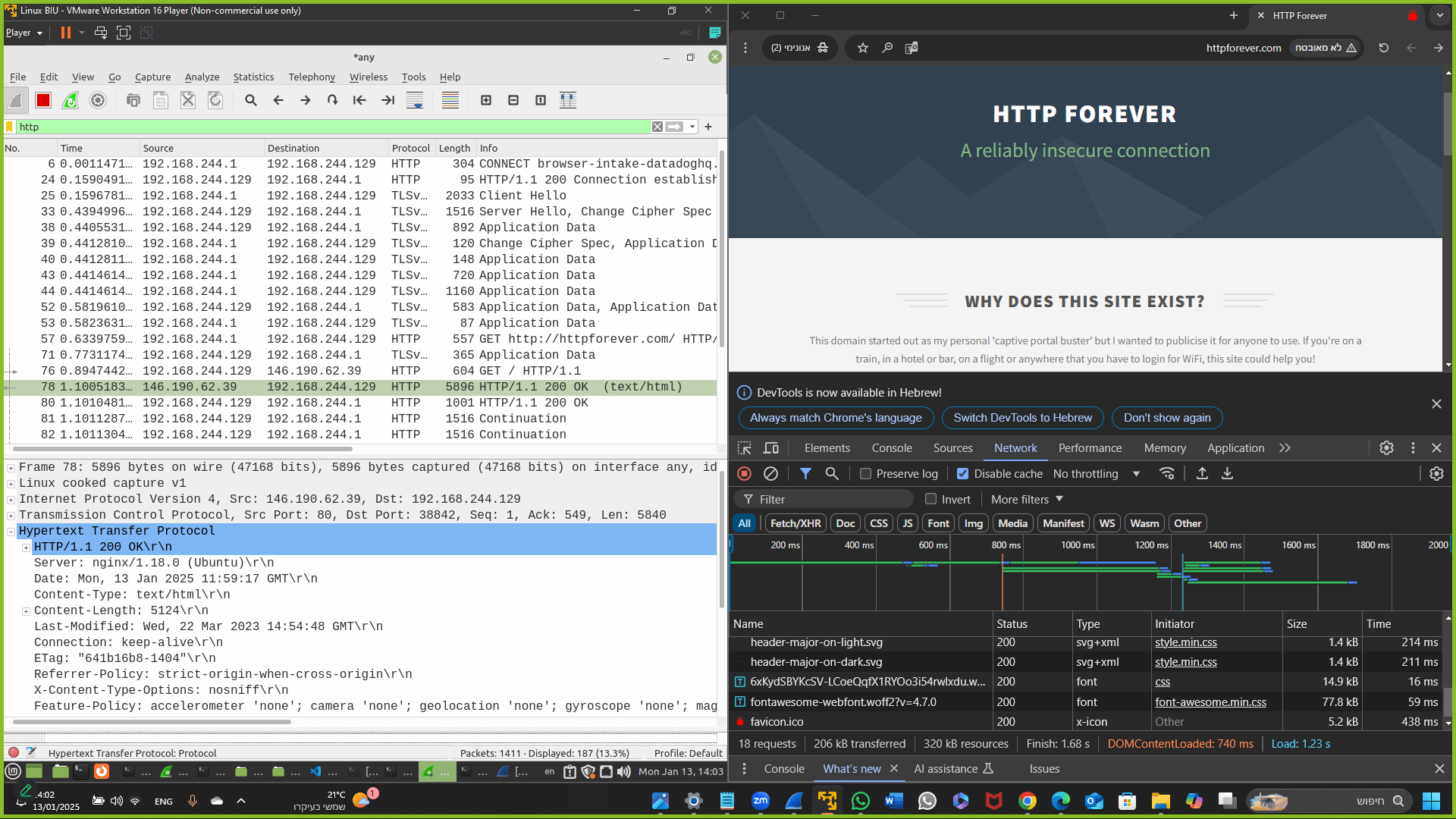


(כמו שאמרנו בצימוד לכל שאילתת ipv4 היה גם שאילתת ipv6 שנשלחו ביחד ונענו ביחד..)

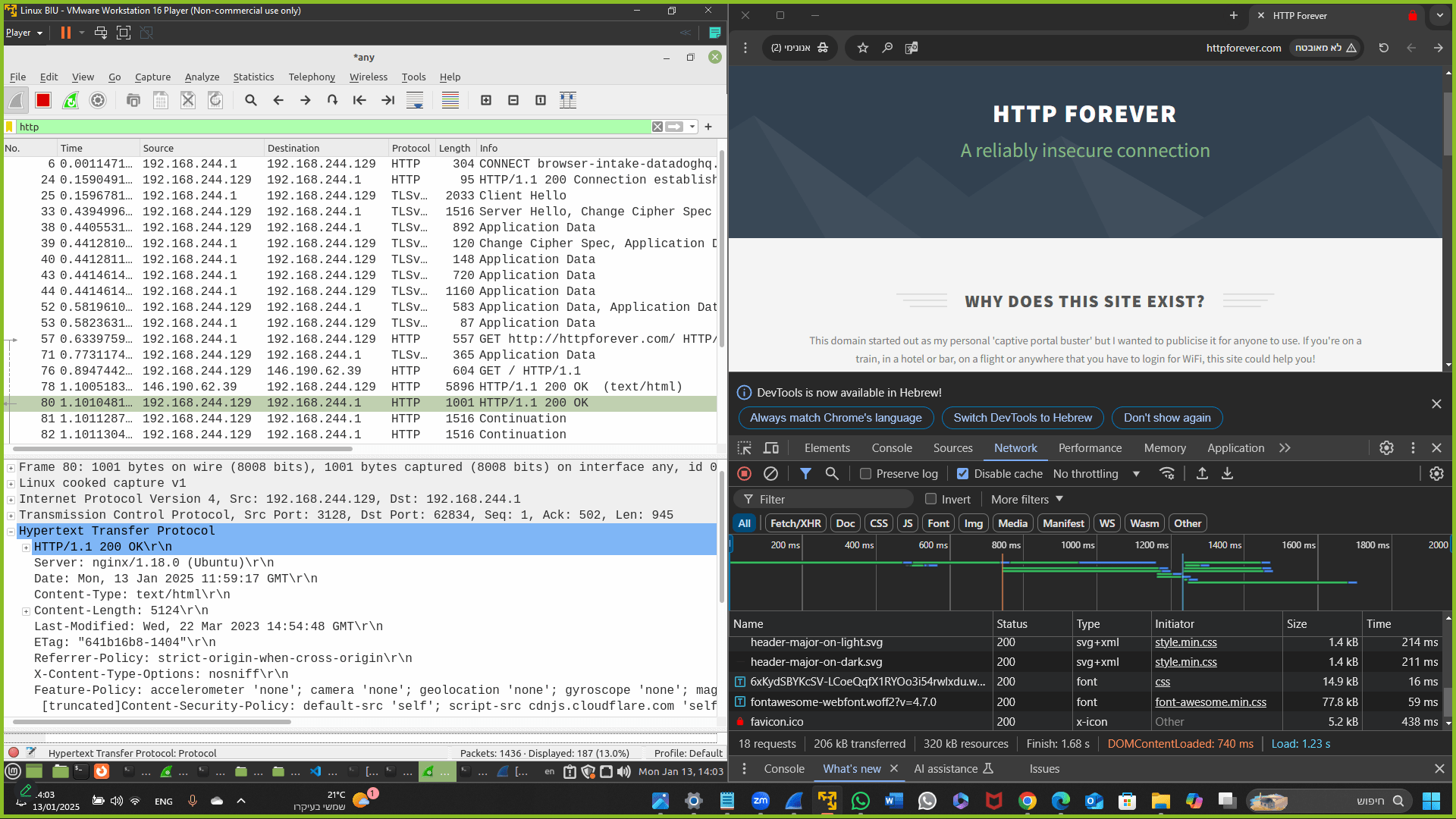
עכשיו כאשר שרת הפרוקסי שלנו יודע מהי כתובת הIP של עמוד האינטרנט ממנו הוא צריך להביא משאב חזרה ללקוח (הדפדפן), הוא מבצע פניית get בפרוטוקול http לכתובת הIp הזו - 146.190.62.39, כלומר אנו רואים ששרת הפרוקסי הוא הוא ממש זה שמבצע את הפניה לשרת האינטרנט ועושה את העבודה עבור הדפדפן.



שרת של httpforever (146.190.62.39) נותן את המשאב לשרת **הפרוקסי** ונותן חיווי של 200 ok.



ועכשיו סוף סוף שרת הפרוקסי אחרי כל התיווך הענק הזה יכול להחזיר את העמוד שהדפדפן ביקש ממנו בתחילה של דף הבית ('/') של אתר האינטרנט httpforever, כך שהדפדפן יוכל להציג לנו אותו באופן ויזואלי. החל מאתה והילך כאשר הדפדפן יזדקק לעוד משאבים (נניח תמונות המוצגות בדף שהוא זה עתה קיבל), התהליך שראינו עכשיו יקרה שוב. כלומר הדפדפן יפנה שוב לפרוקסי, אבל עכשיו הפרוקסי יוכל לפנות ישירות לשרת של httpforever מתוך הנחה שהוא לא יזדקק שוב לבצע שאילתת DNS כיוון שכתובת הIp תהיה בקאש (כמובן תלוי בttl), וכך התיווך יהיה יותר קצר מאשר הפעם הראשונה שהוא התבצע.



לסיכום ראינו כיצד שרת הפרוקסי מבצע עבור הדפדפן את פעולות הDNS כולן, וכן הוא "מוריד"/מבקש בעצמו את התוכן משרת האינטרנט ורק אז מעביר אותו לקליינט – הדפדפן בחזרה. כמו כן כל נדבך בתהליך התיווך שראינו התבצע על חיבור נפרד, ולמשל כאשר הפרוקסי דיבר בחיבור אחר עם שרת האינטרנט, החיבור שלו עם הדפדפן היה על סוג של "hold" בלי תקשורת זורמת ממש, עד השלב שהוא היה מסוגל להחזיר לו דרך החיבור, את התוכן שהוא קיבל משרת האינטרנט דרך החיבור הנפרד "המקביל".

*עד כאן חלק שלישי*