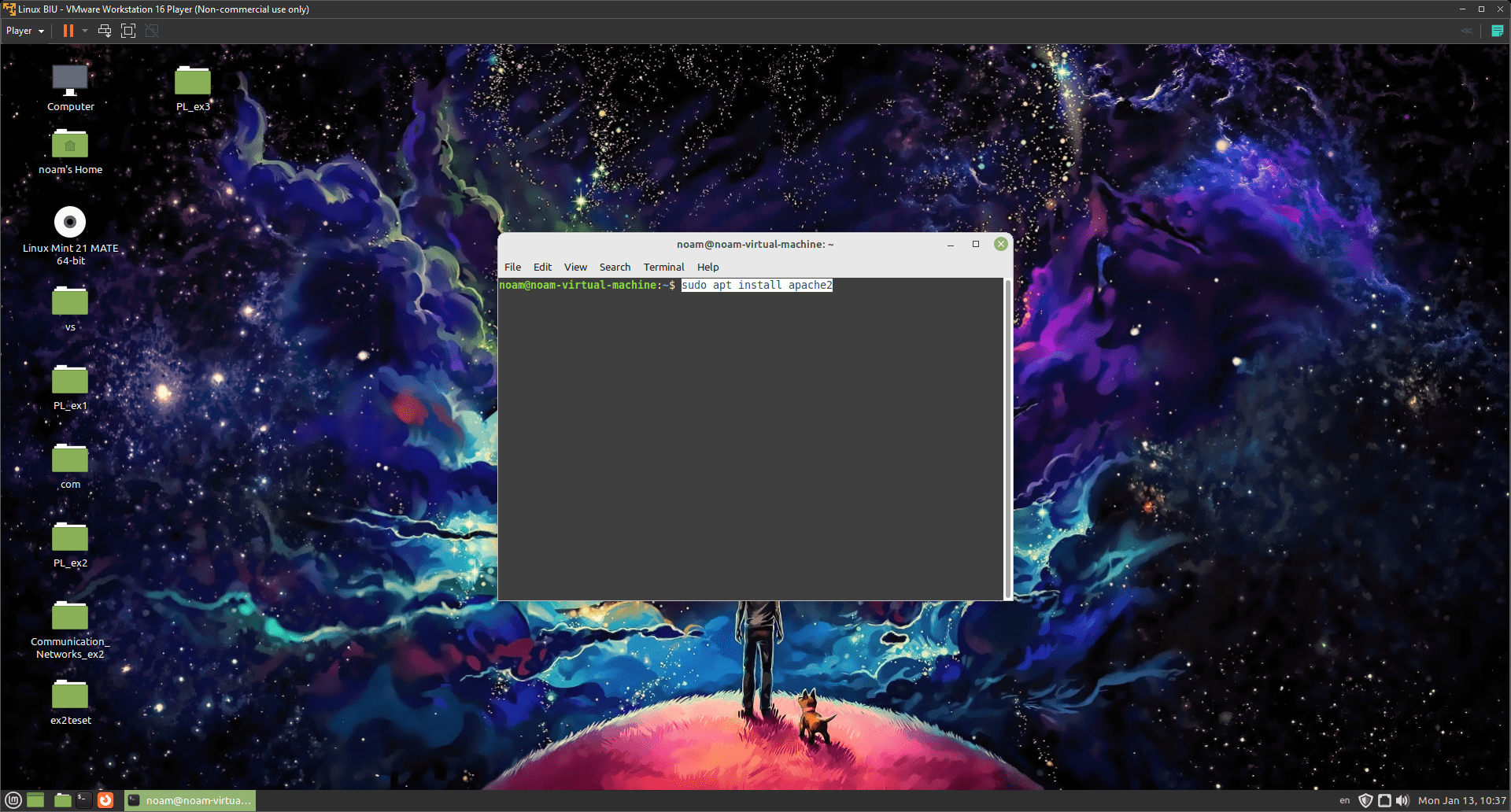
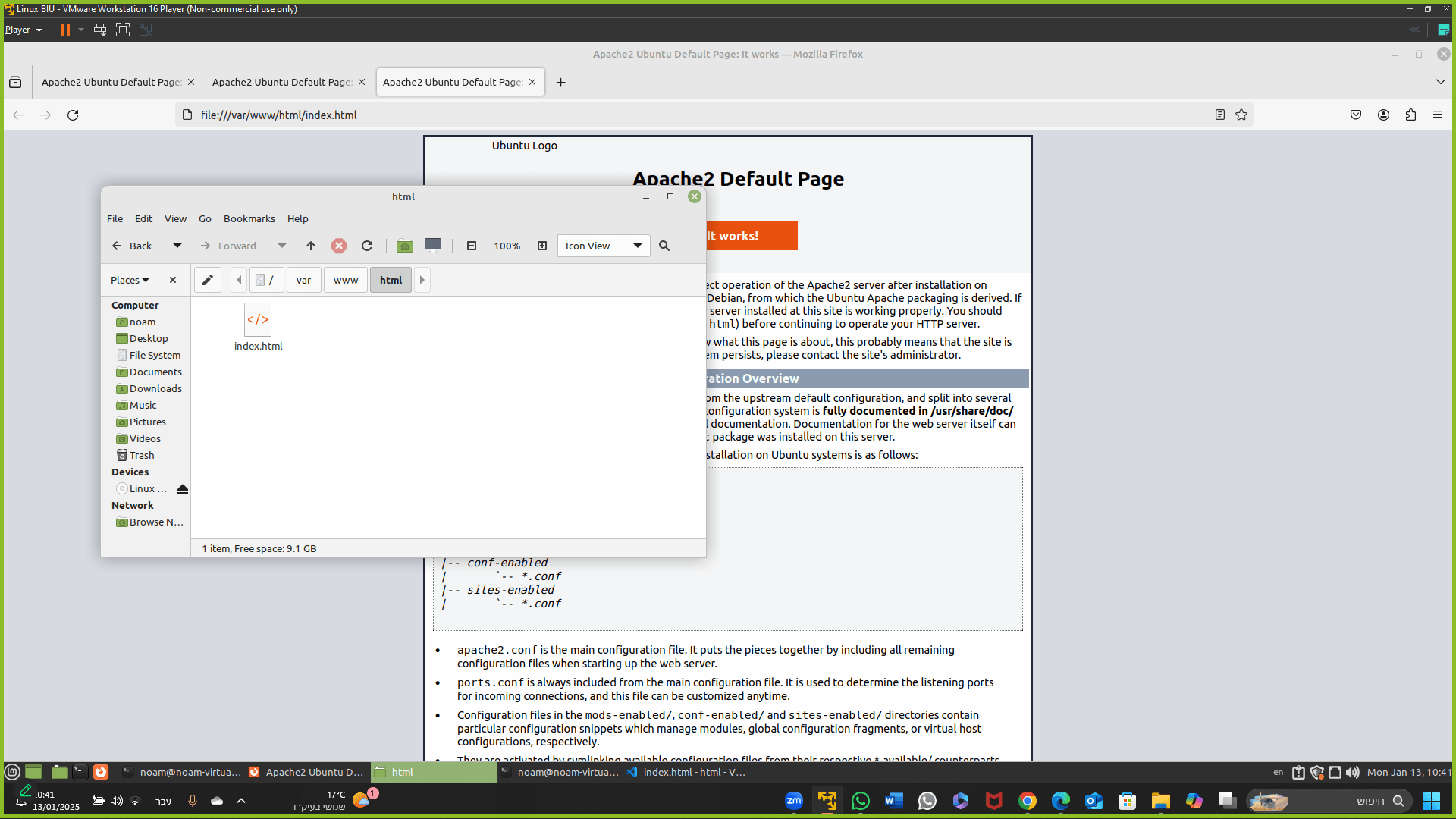
מטלה מס' 3 – אילאי בן יהושע ונועם ליבוביץ

*חלק ראשון*

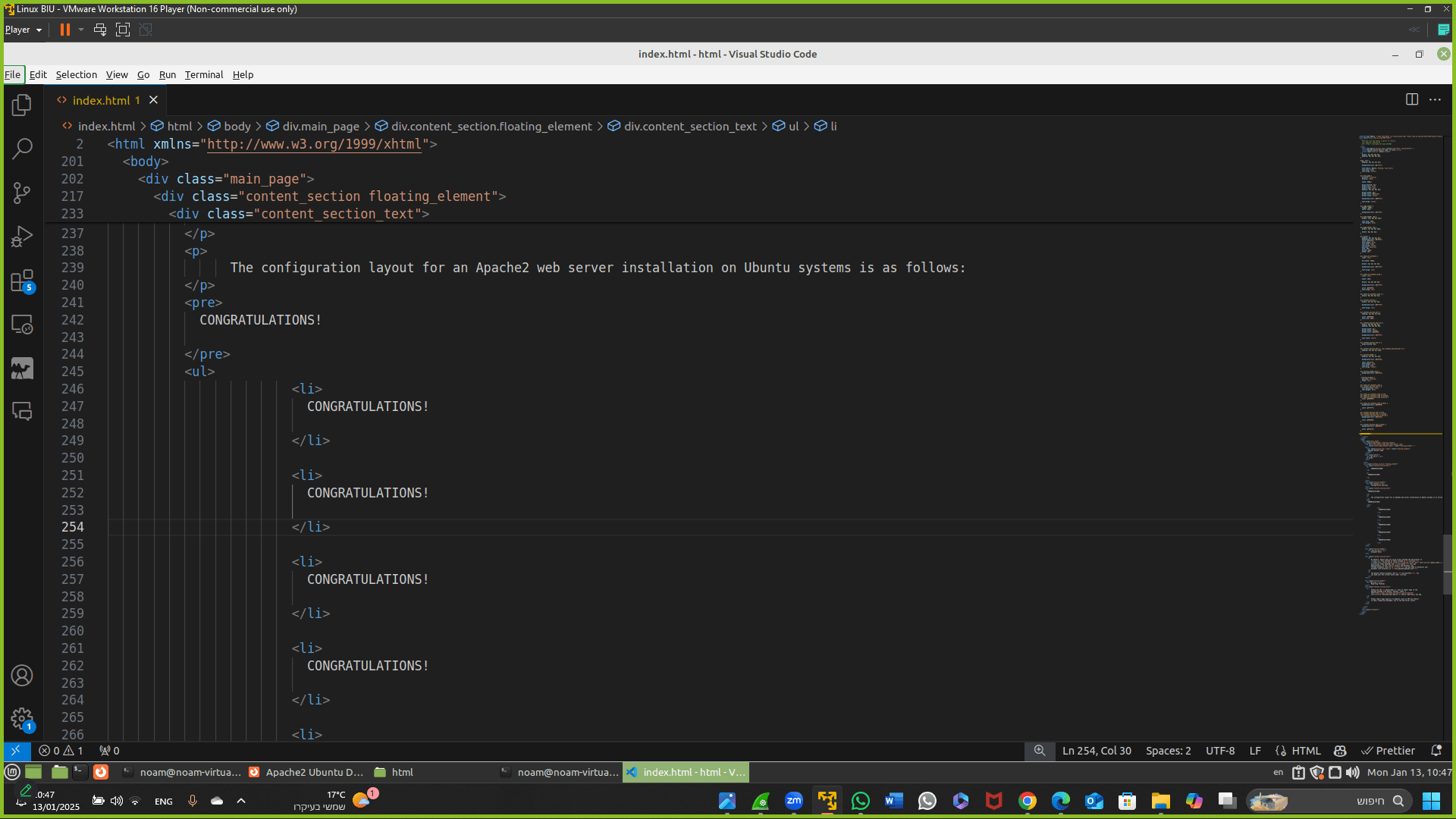
נתקין את שרת האינטרנט – apache.



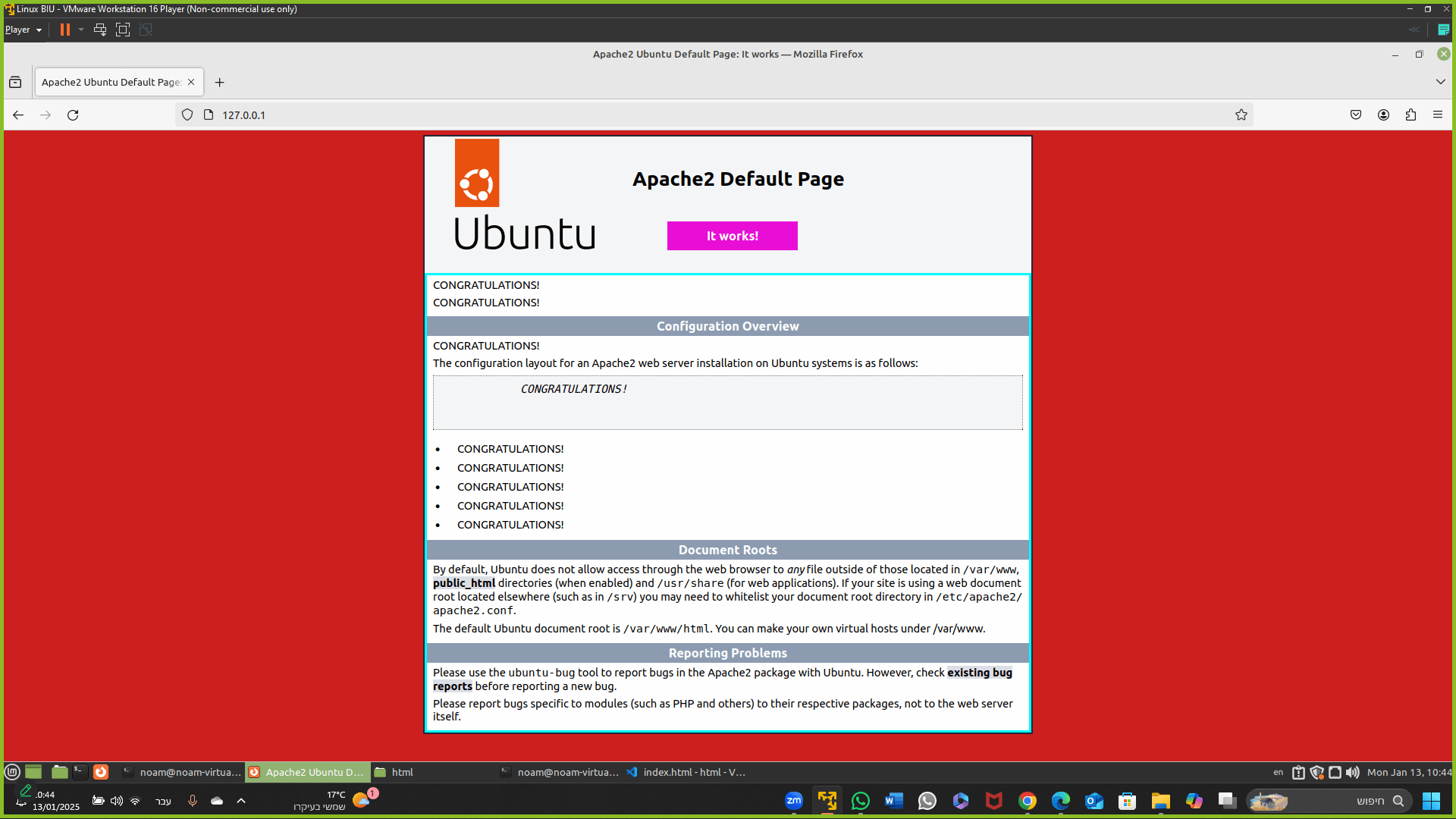
נראה את קובץ הhtml הדיפולטיבי בנתיב – var/www/html.



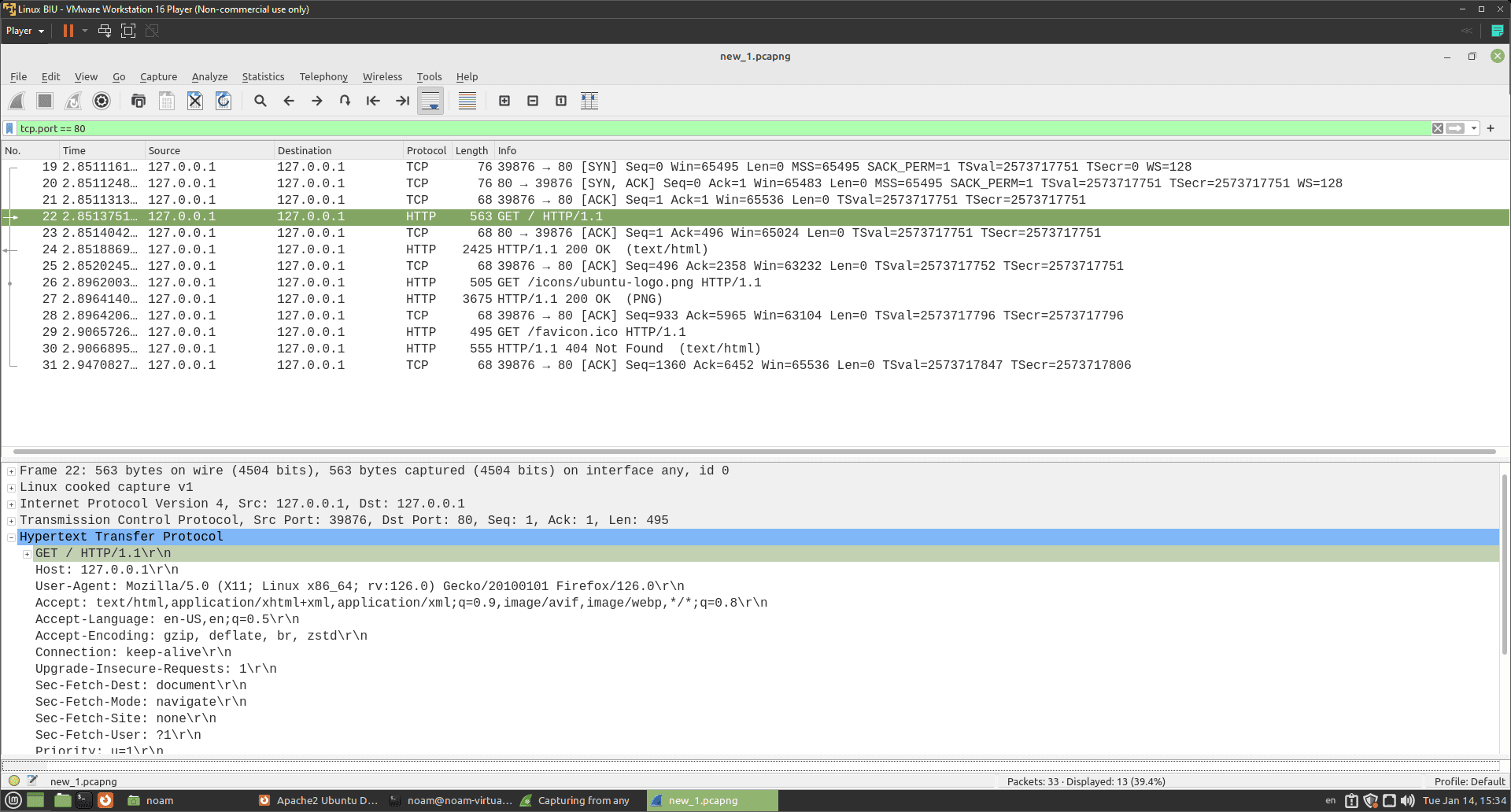
נפתח אותו בערוך הטקסטים vscode ונבצע בו מס' שינויים גרפיים וטקסטואליים.

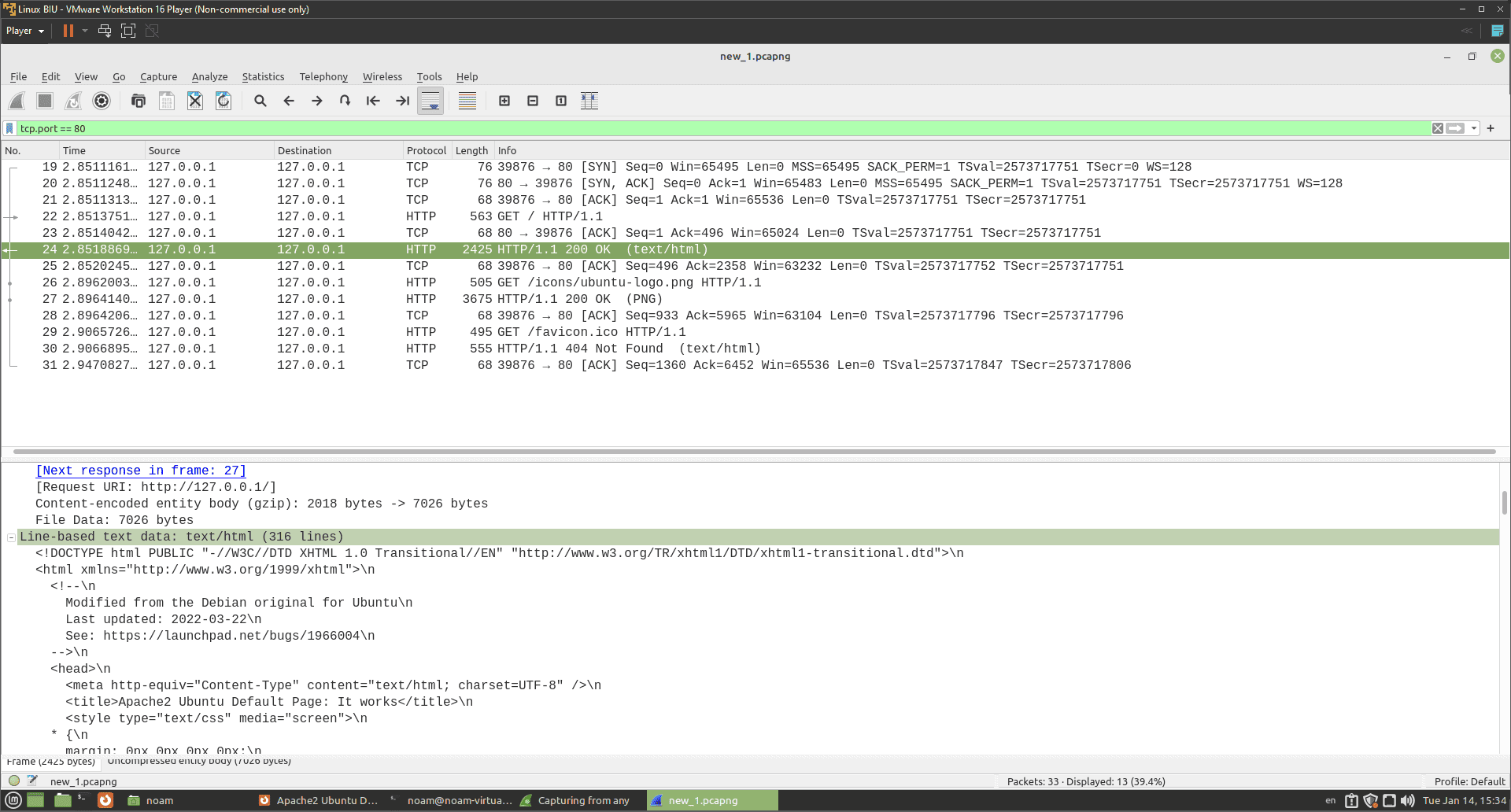


נשים לי כי השינויים מוצגים כעת בדף הדיפולטיבי אותו מחזיר השרת לאחר רענון של העמוד.

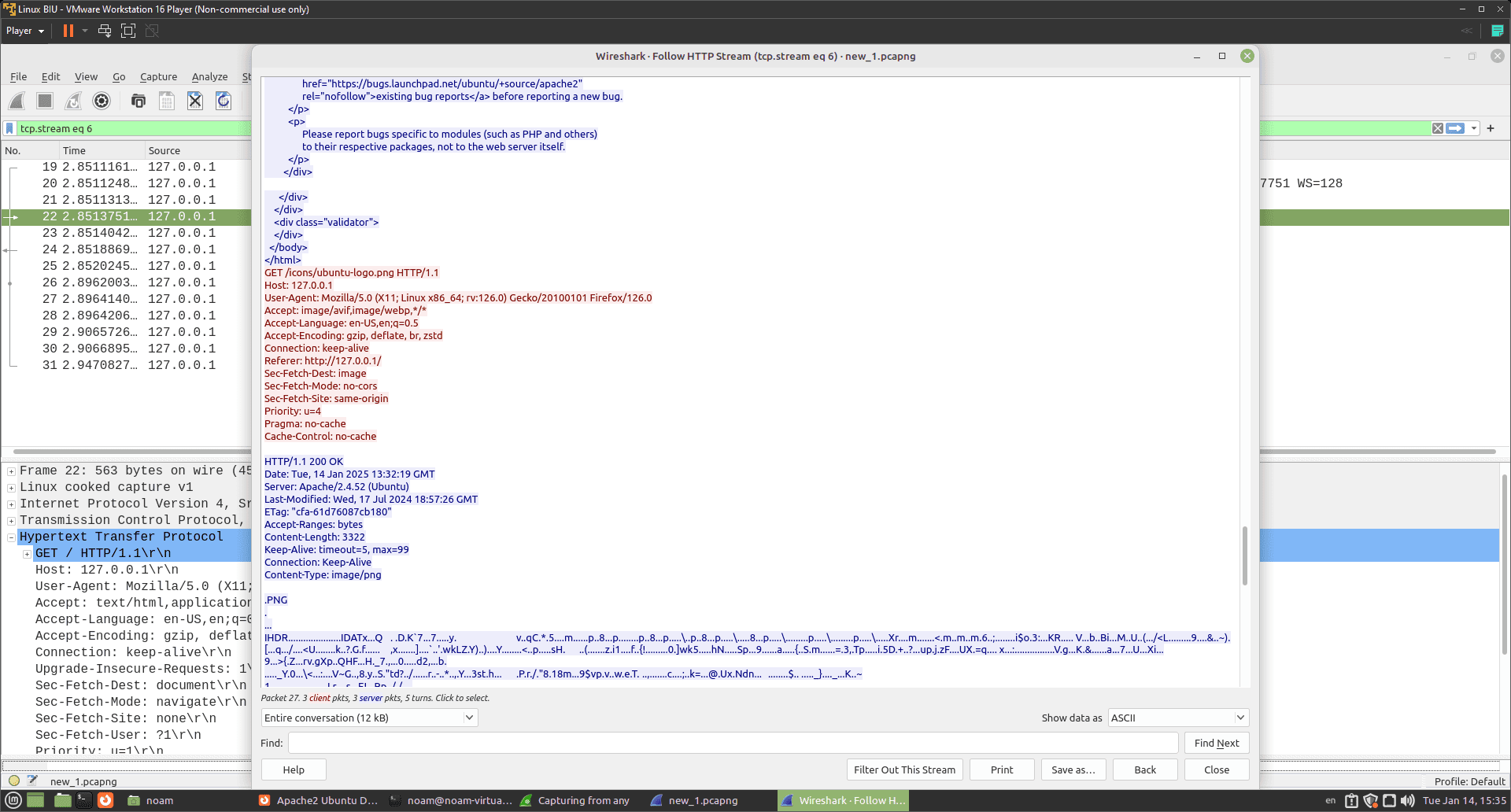


נפתח את תוכנת הwireshark (מעכשיו תקרא בהמשך הטופס 'כריש'), ונסניף בעזרתה את התעבורה. נסנן כך שיוצגו חבילות הרלוונטיות לפרוטוקול TCP ולפורט 80 עליו רץ שרת האינטרנט שלנו.





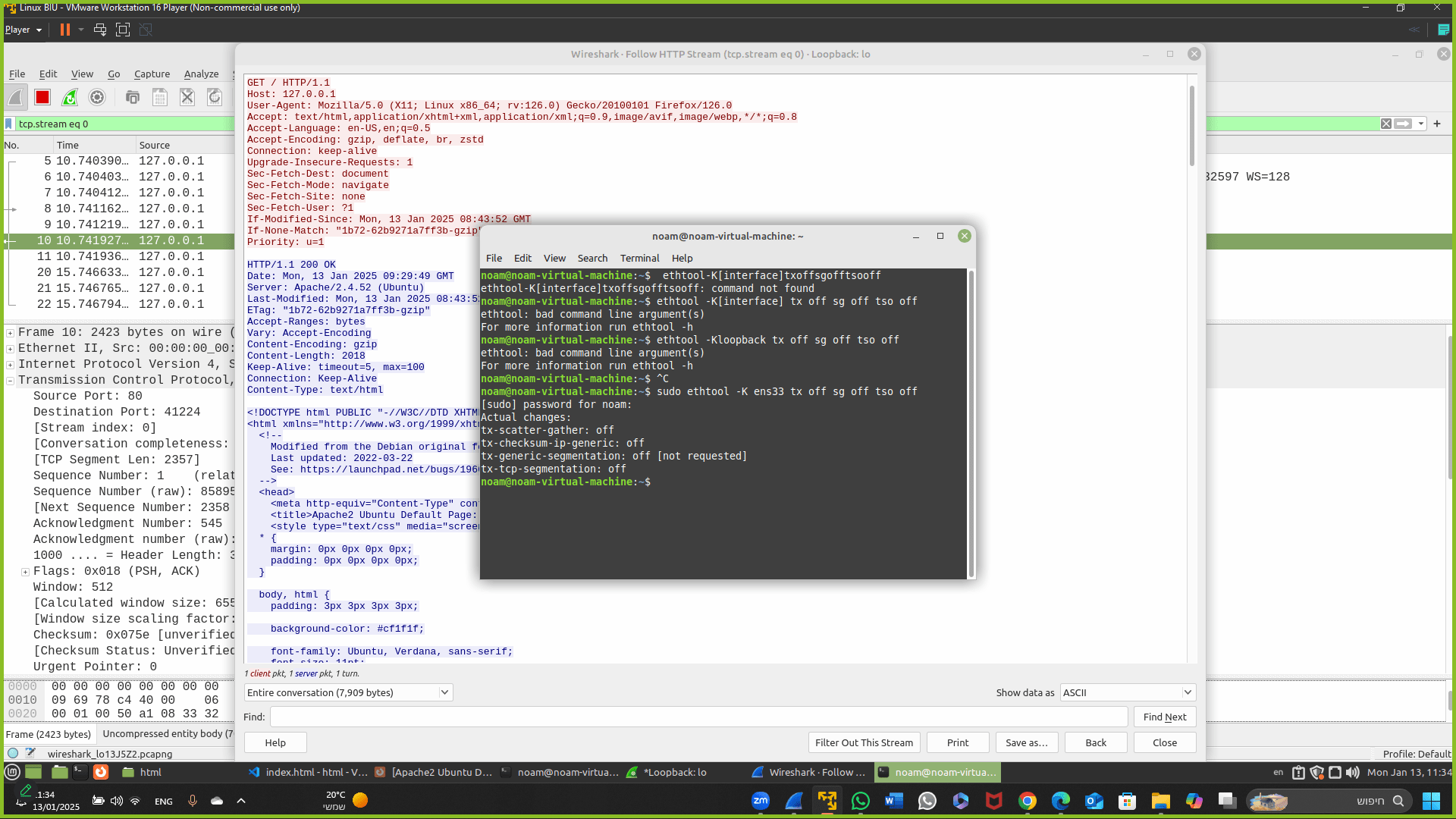
נסביר את הפלואו של הדברים בתמונות לעיל. תחילה אנו רואים לחיצת ידיים של   
syn-syn&ack-ack בין הלקוח (הדפדפן) לשרת. לאחר מכן הלקוח שולח בקשת http מסוג get לשרת על מנת לקבל את "דף הבית" (הלו הוא '/'). השרת מחזיר ללקוח ack כדי לתת חיווי שאכן בקשה זו נקלטה אצלו, ובהודעת הtcp העוקבת אנו הוא מחזיר לו את העמוד 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.



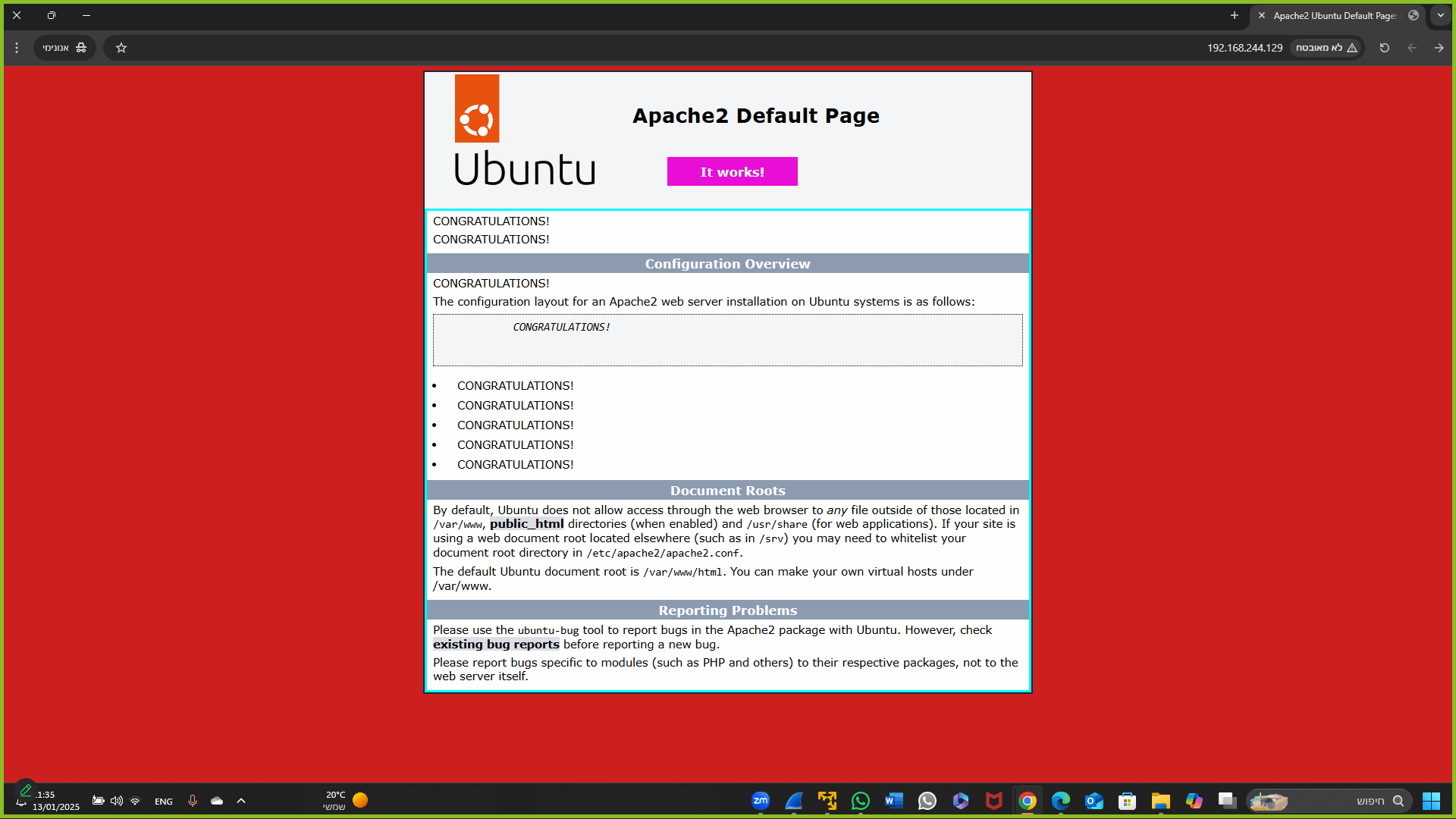
התמונה לעיל אנו רואים את הפלואו של תקשורת הhttp שקרתה, כאשר באדום אנו רואים את סוף תוכן העמוד שהשרת החזיר בבקשה הראשונה ללקוח, בכחול אנו רואים את הבקשה הנוספת של הלקוח עבור התמונה, ובירוק את התשובה של השרת עם הקידוד שלה.

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.

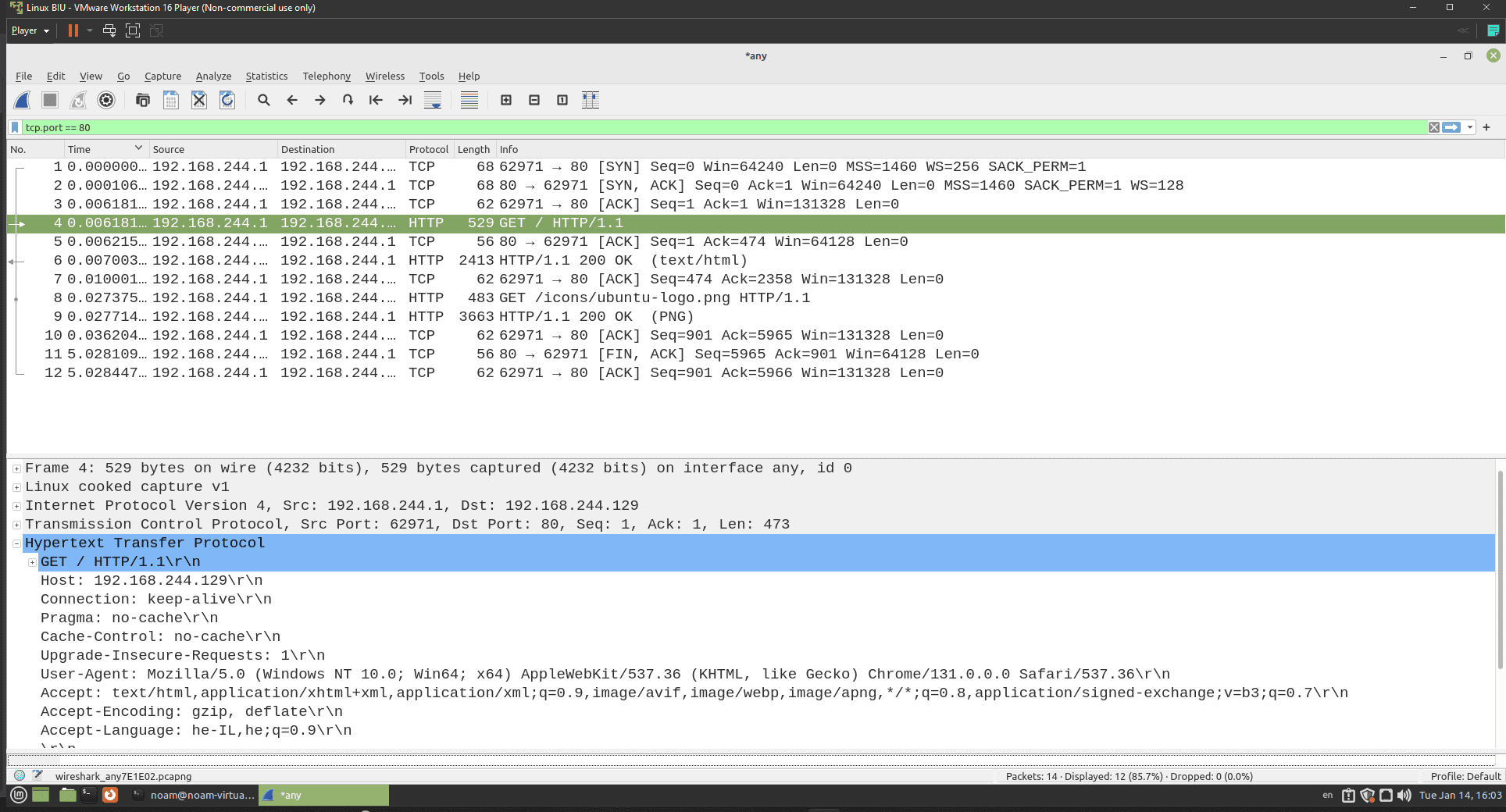


נגלוש לשרת מהמחשב **השני** דרך מצב incognito וכן דרך מצב דפדפן 'רגיל'. ראשית נראה שזה אכן עובד.

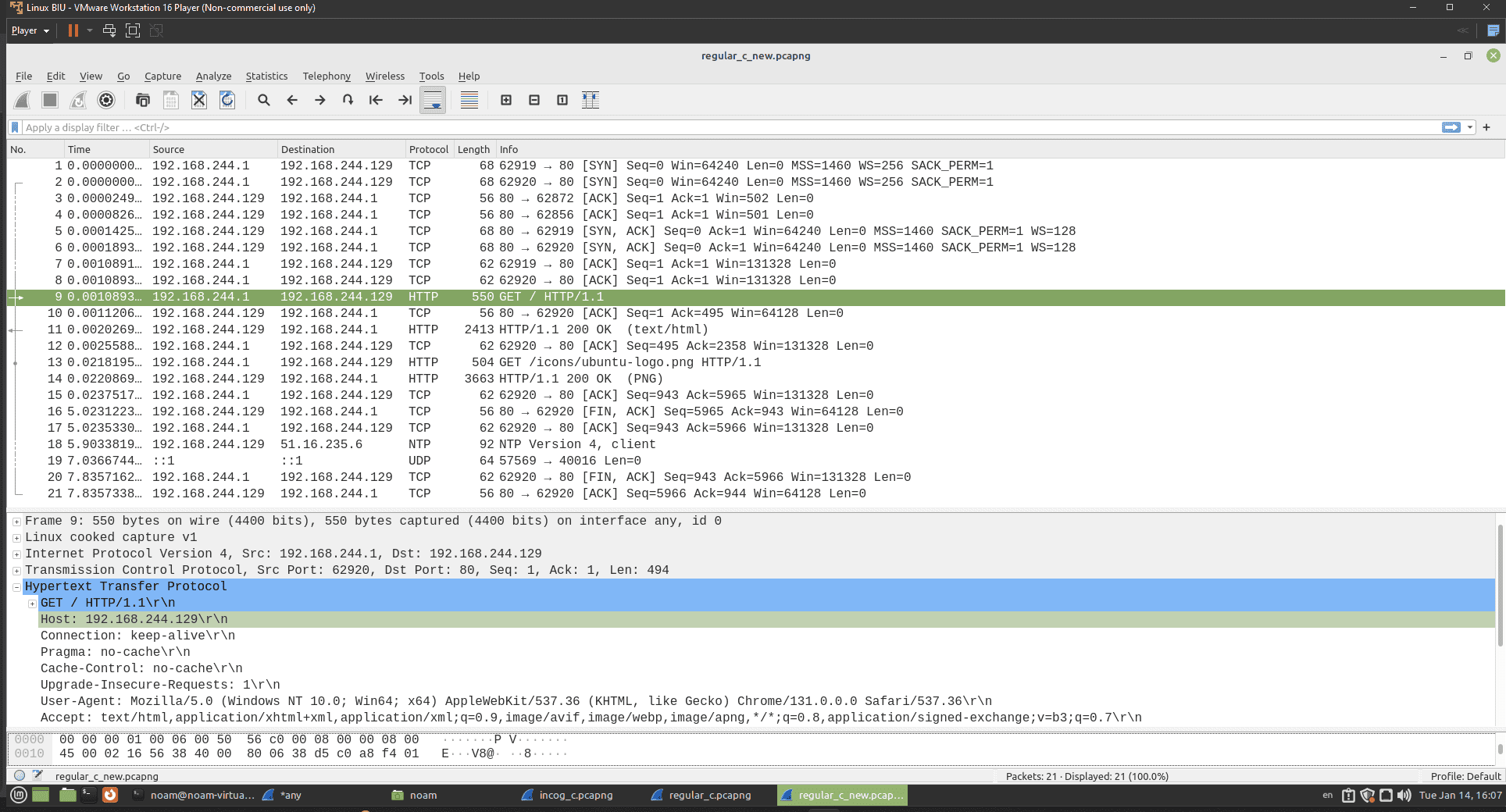


עובד.

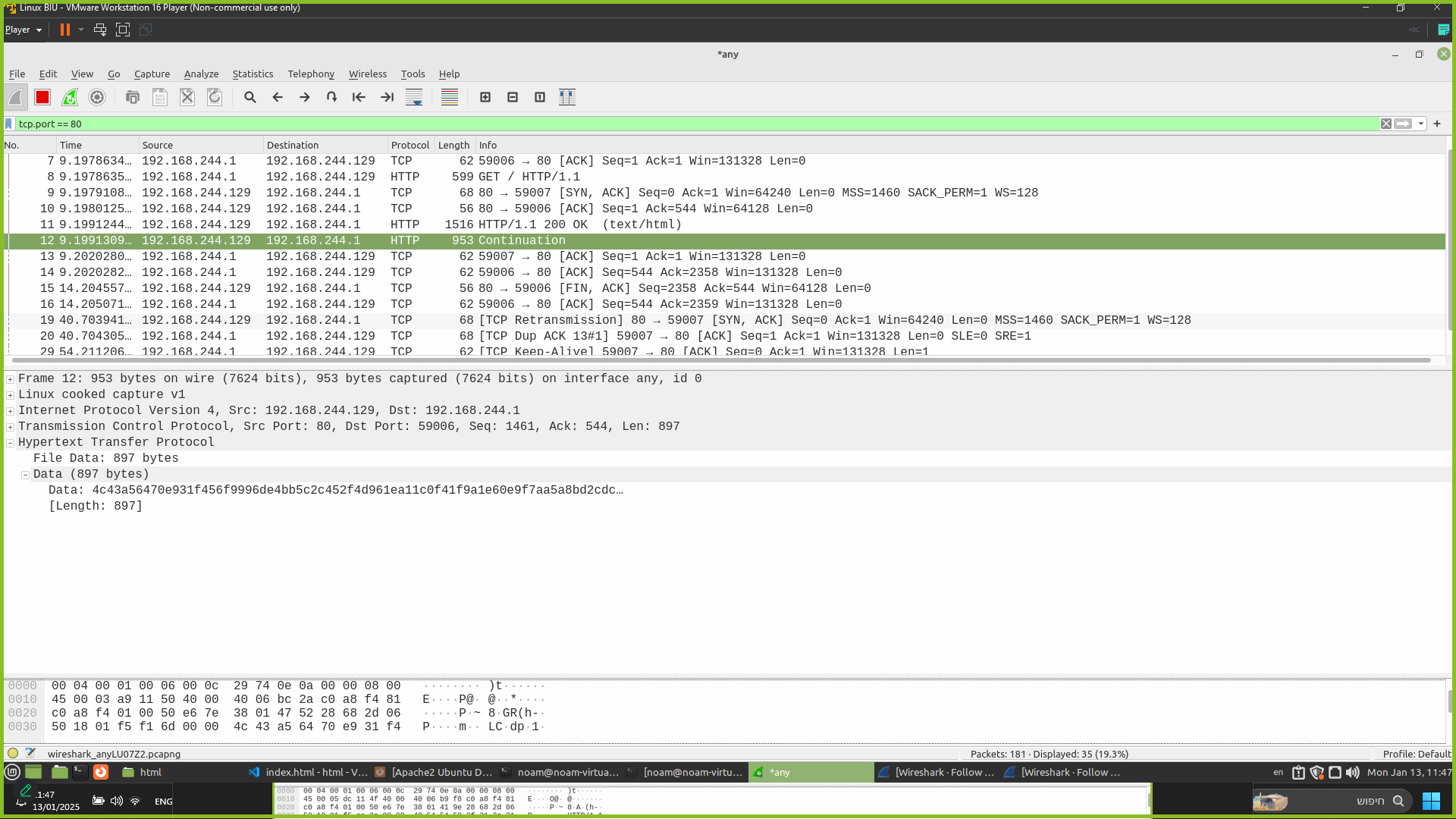
נבחין בתעבורה דרך הכריש. נתחיל דווקא בגלישה דרך מצב incognito. נשיב לב שקיבלנו משהו מאוד זהה לגלישה דרך המחשב עליו רץ השרת. גם פה ישנן 2 בקשות אל השרת, הראשונה עבור עמוד הhtml הדיפולטיבי של השרת, והשנייה עבור התמונה שהעמוד מציג (בדומה למה שראינו קודם לכן). אולם נשים לב שהפעם לא מתבצעת פנייה שלישית עבור 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. מעבר לכך לא הבחנו בהבדלים נוספים בין מצבי הדפדפן.



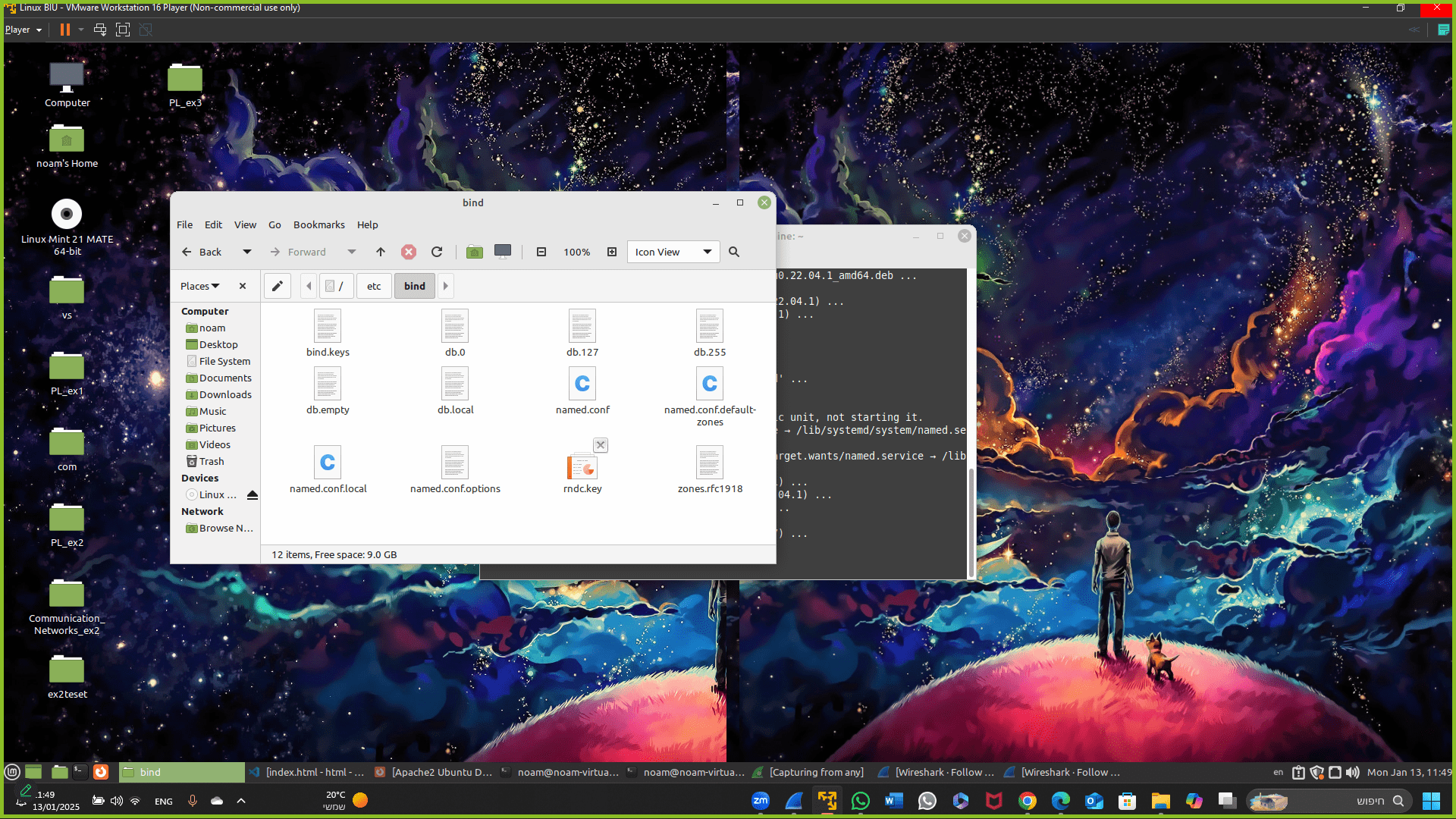
נבטל בהגדרות את הקונפיגורציה לגבי איחוד חבילות tcp תחת http ונראה שאכן עכשיו אנו רואים את שכבת האפליקציה פר-חבילה, שלא כמו שהיה קודם לכן.

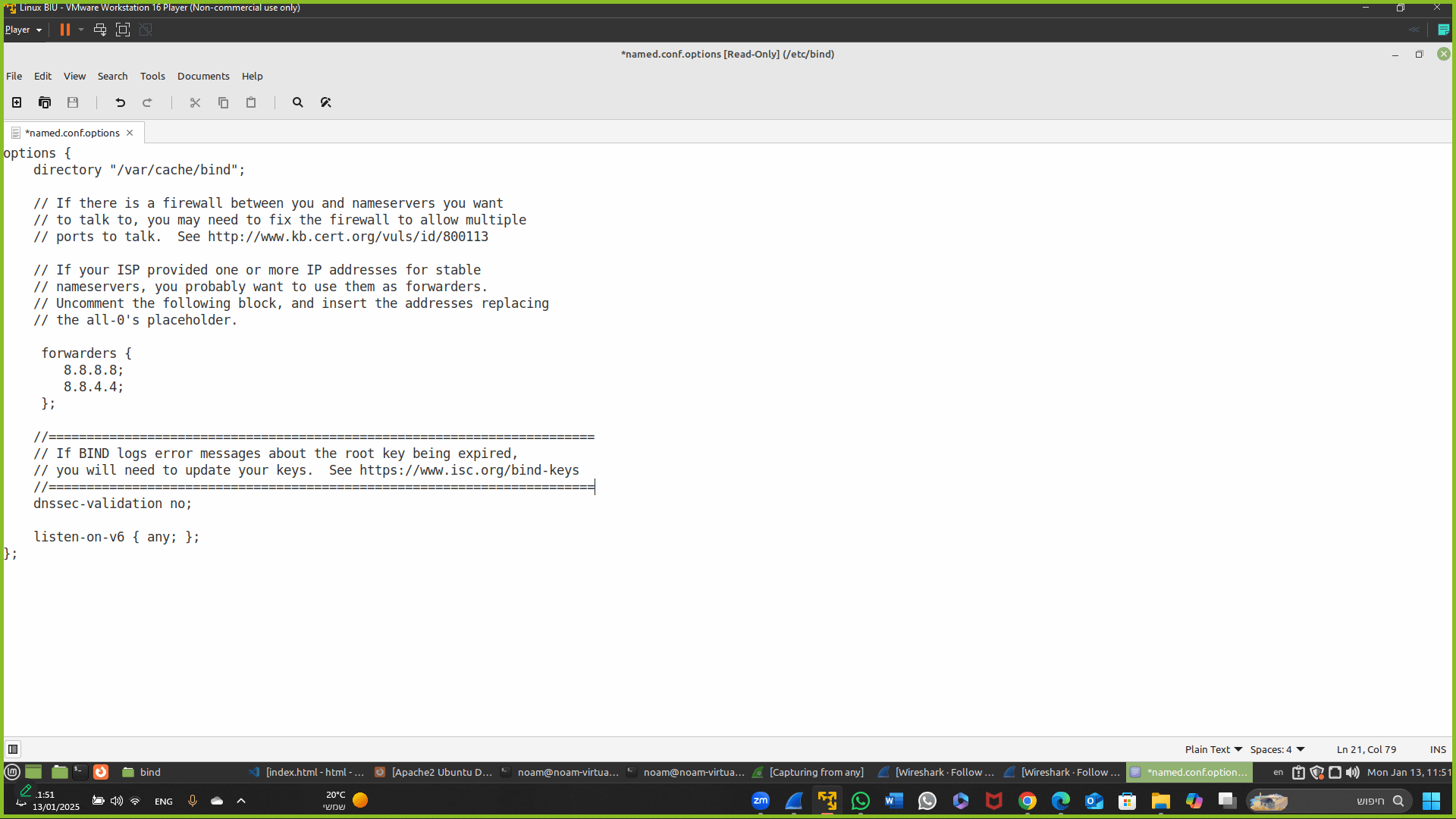


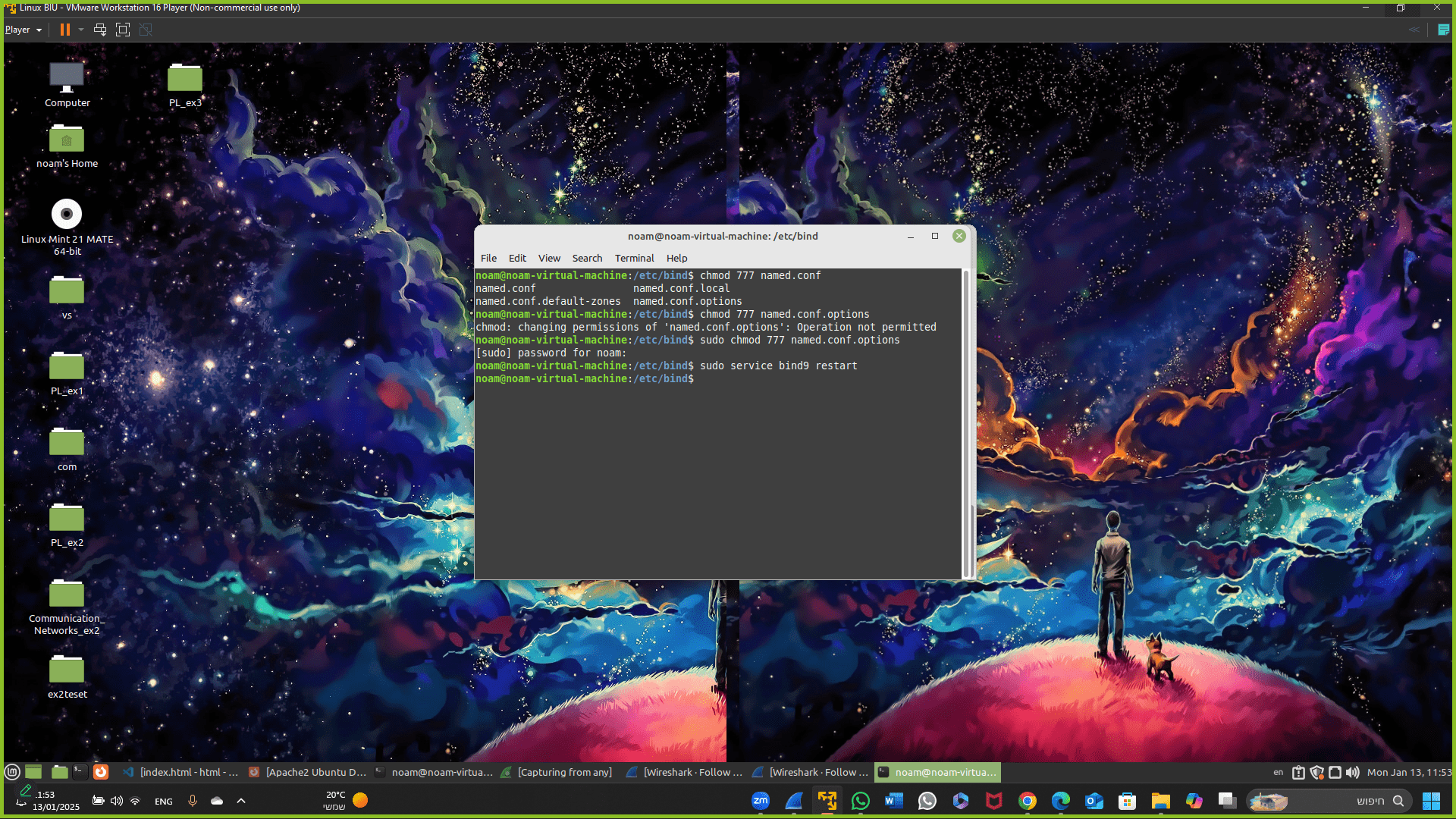
000

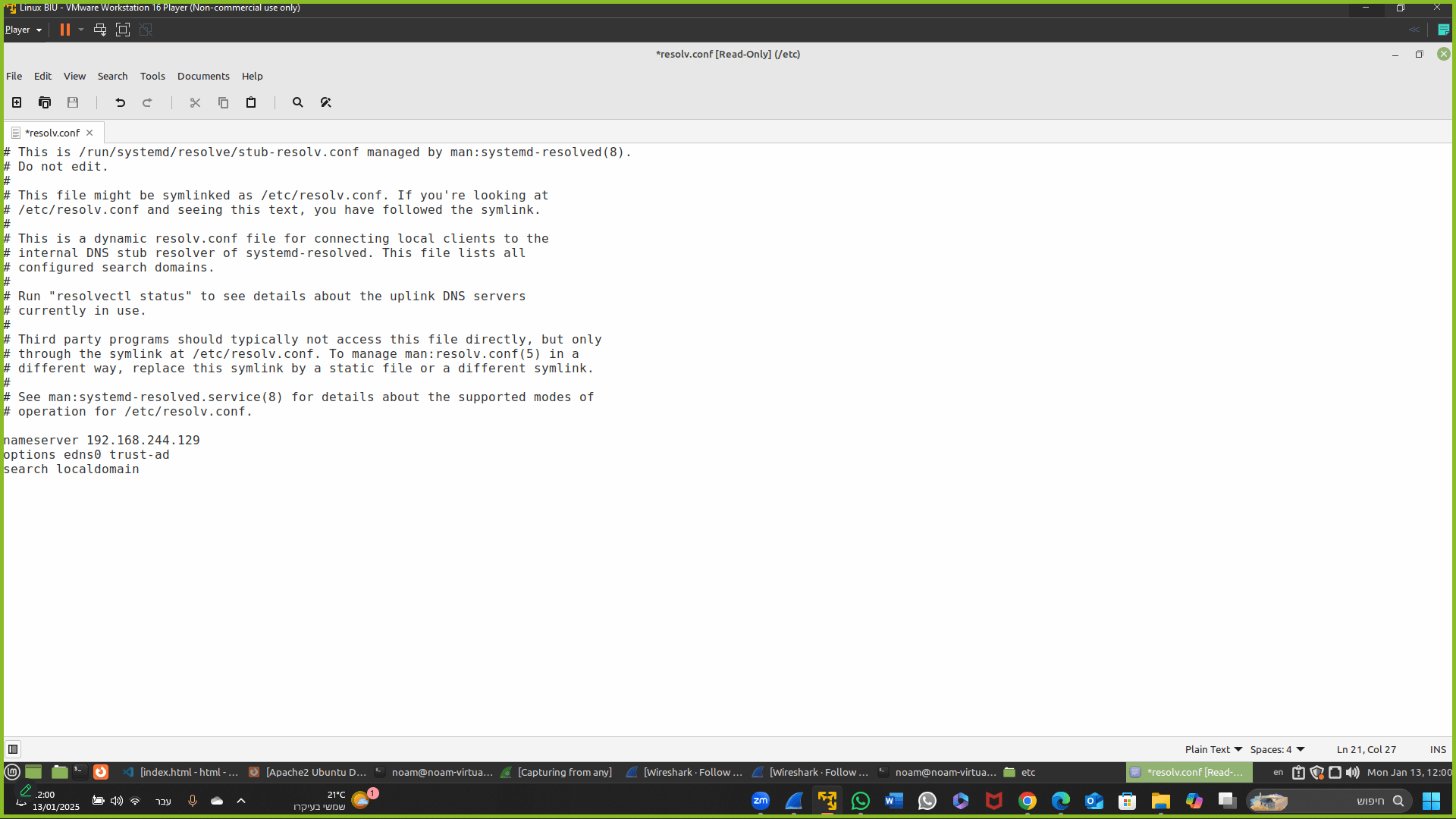
000

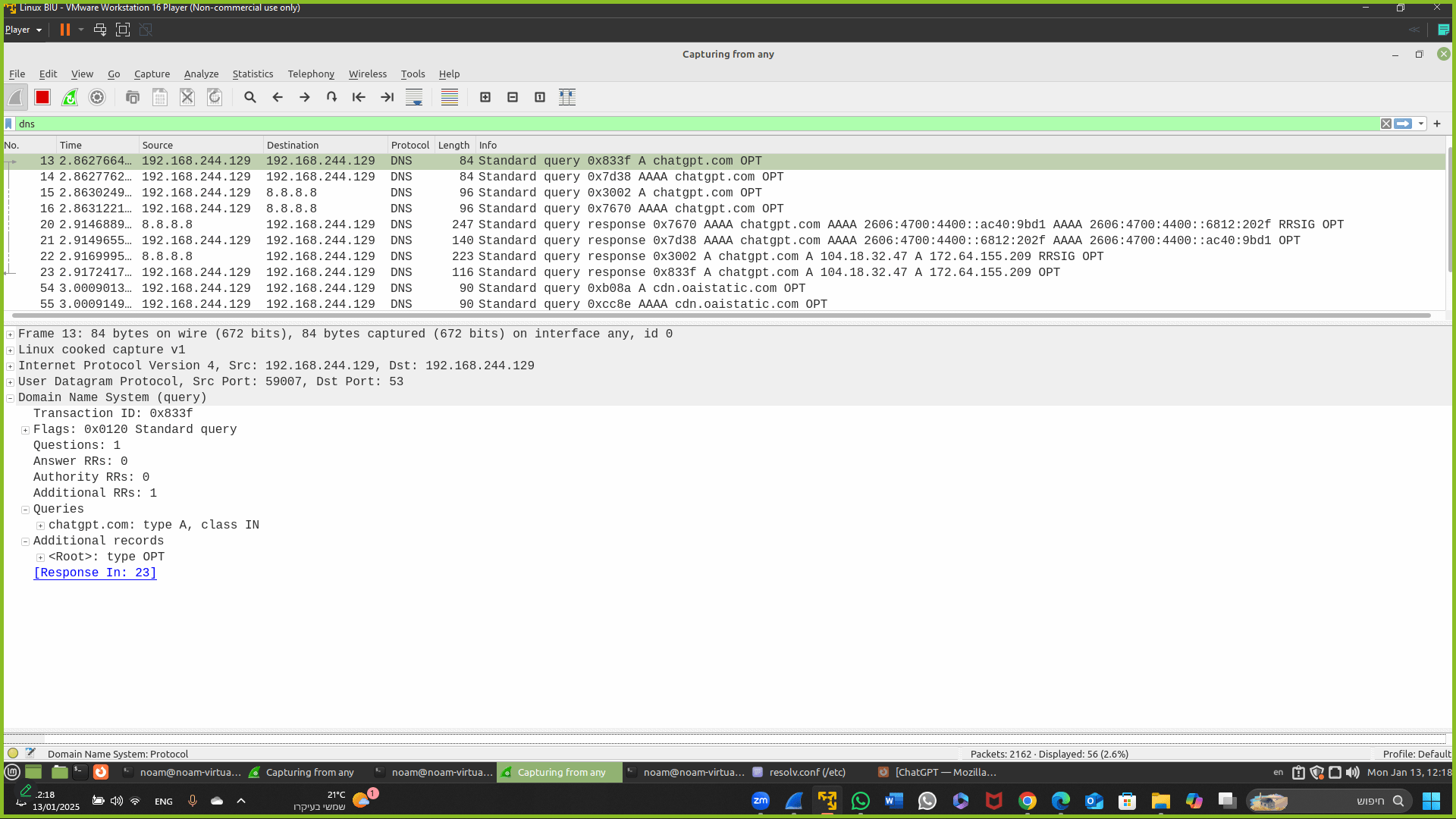
*עד כאן חלק ראשון*

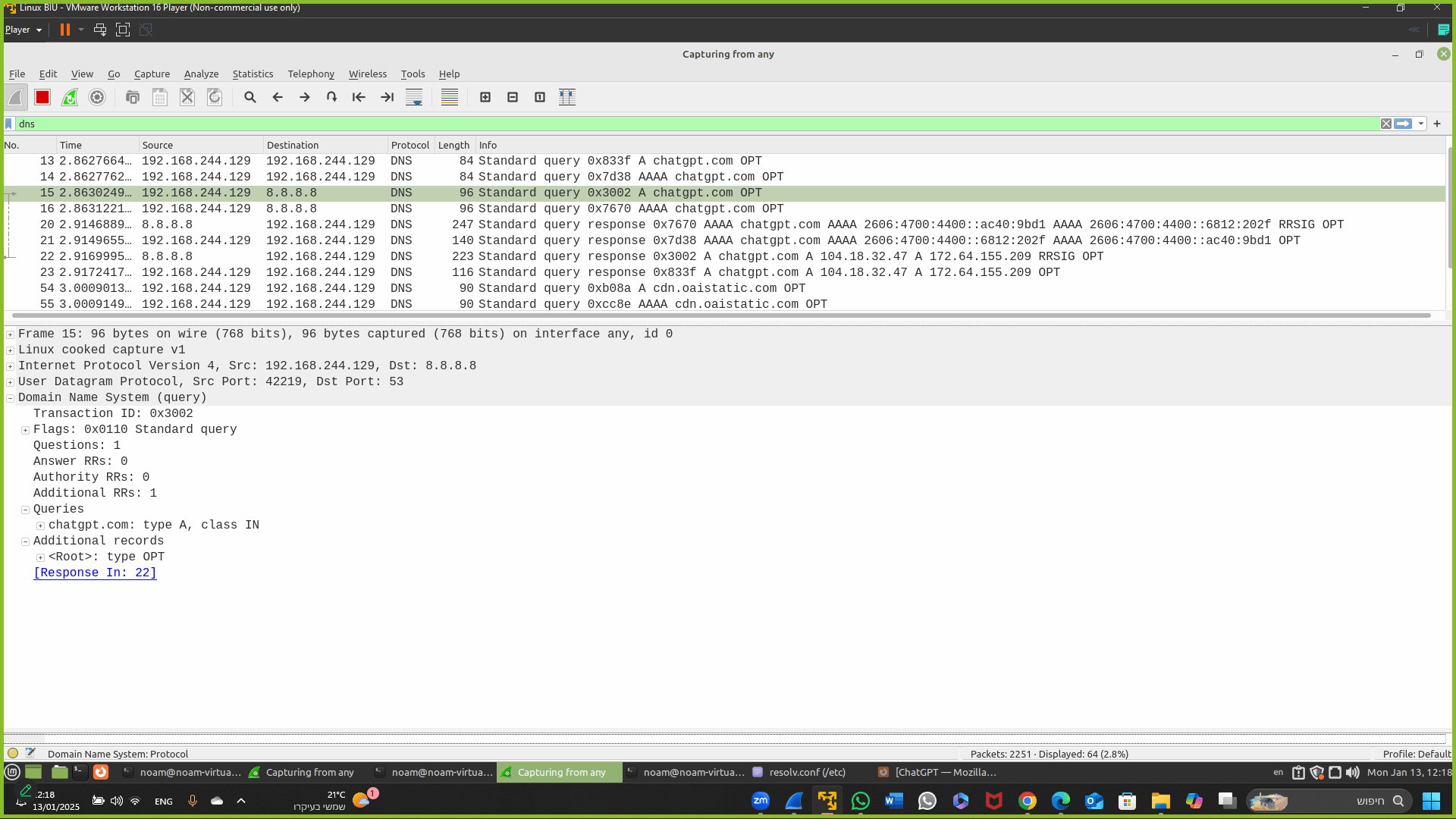


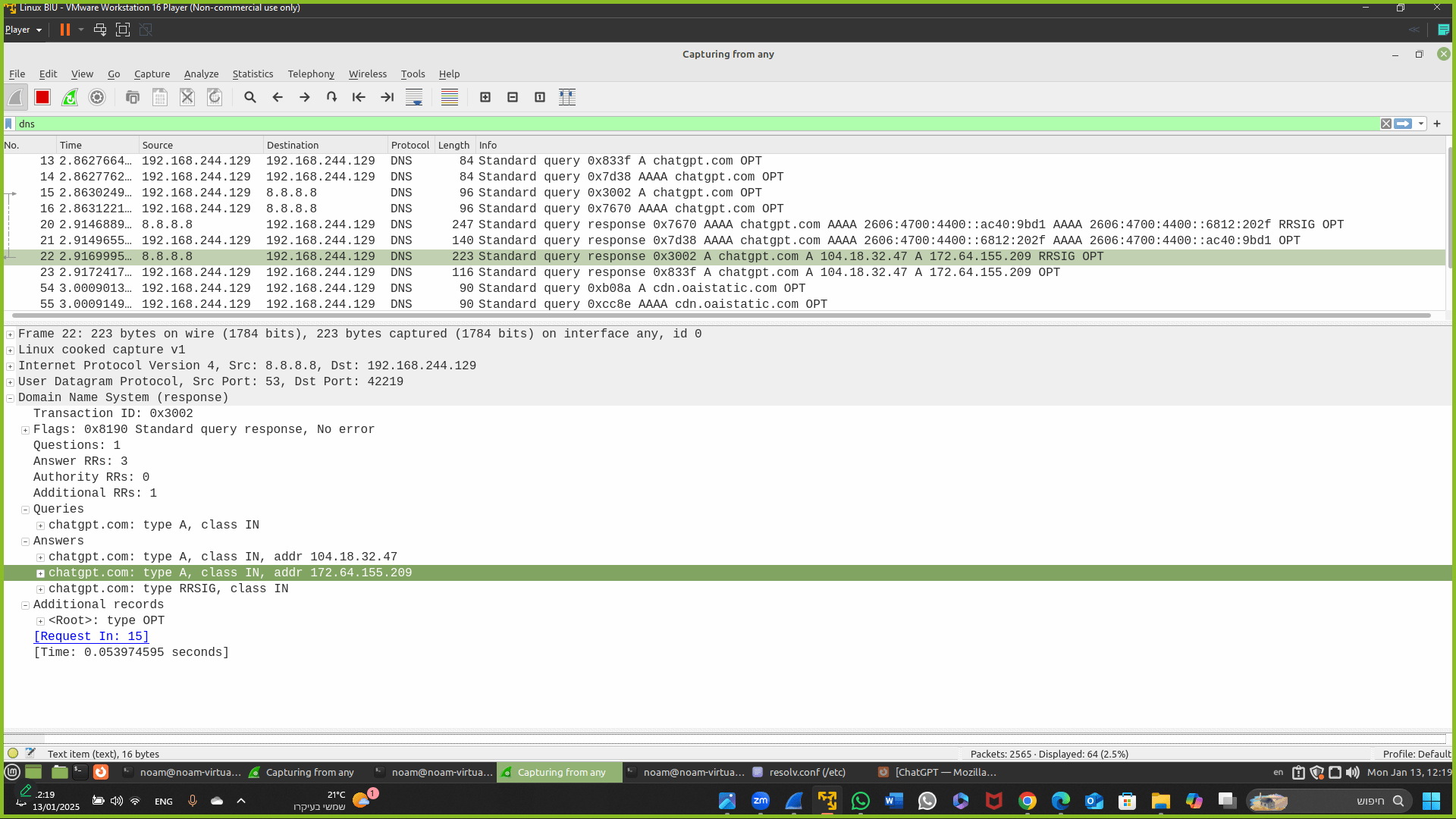


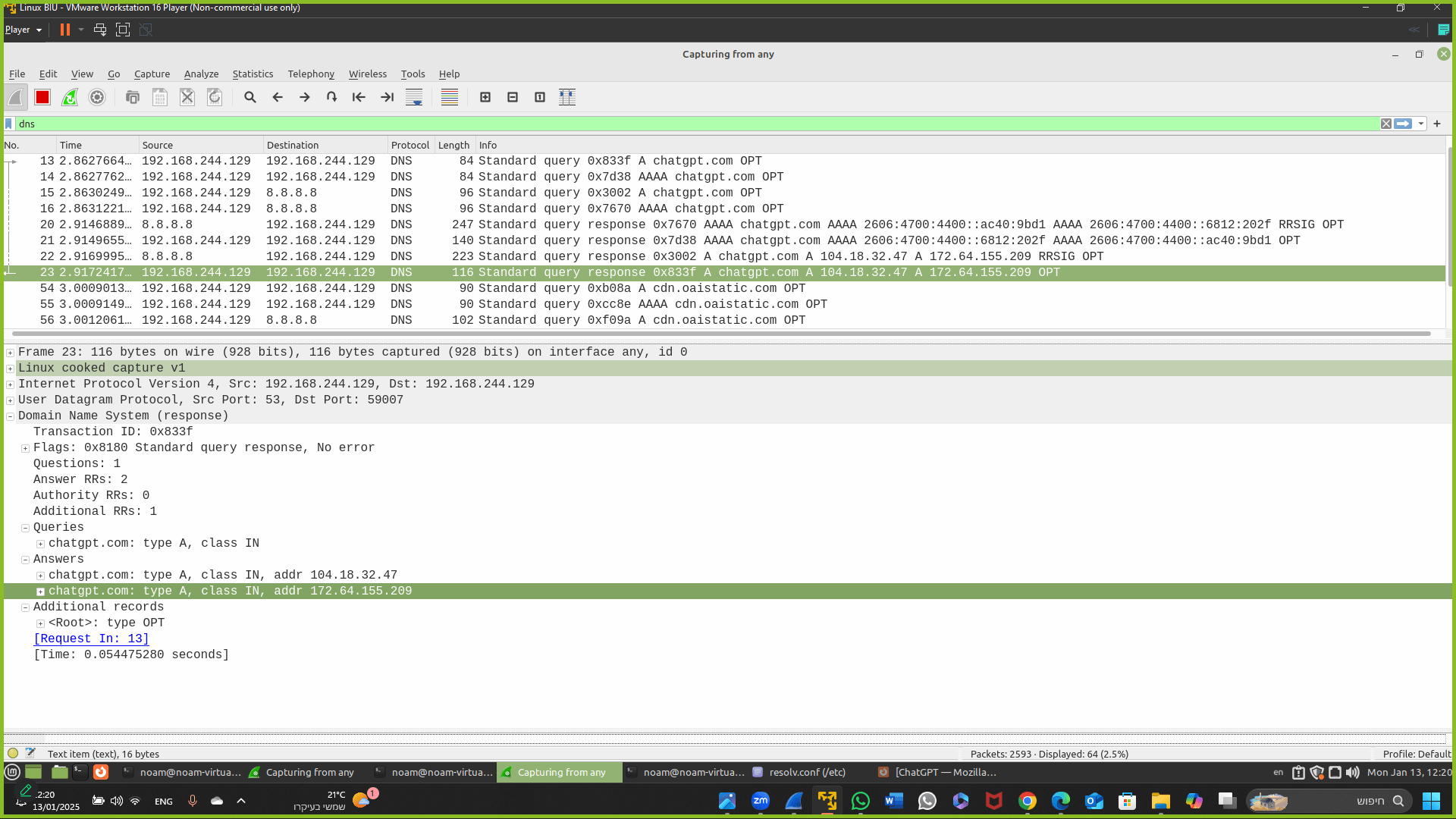


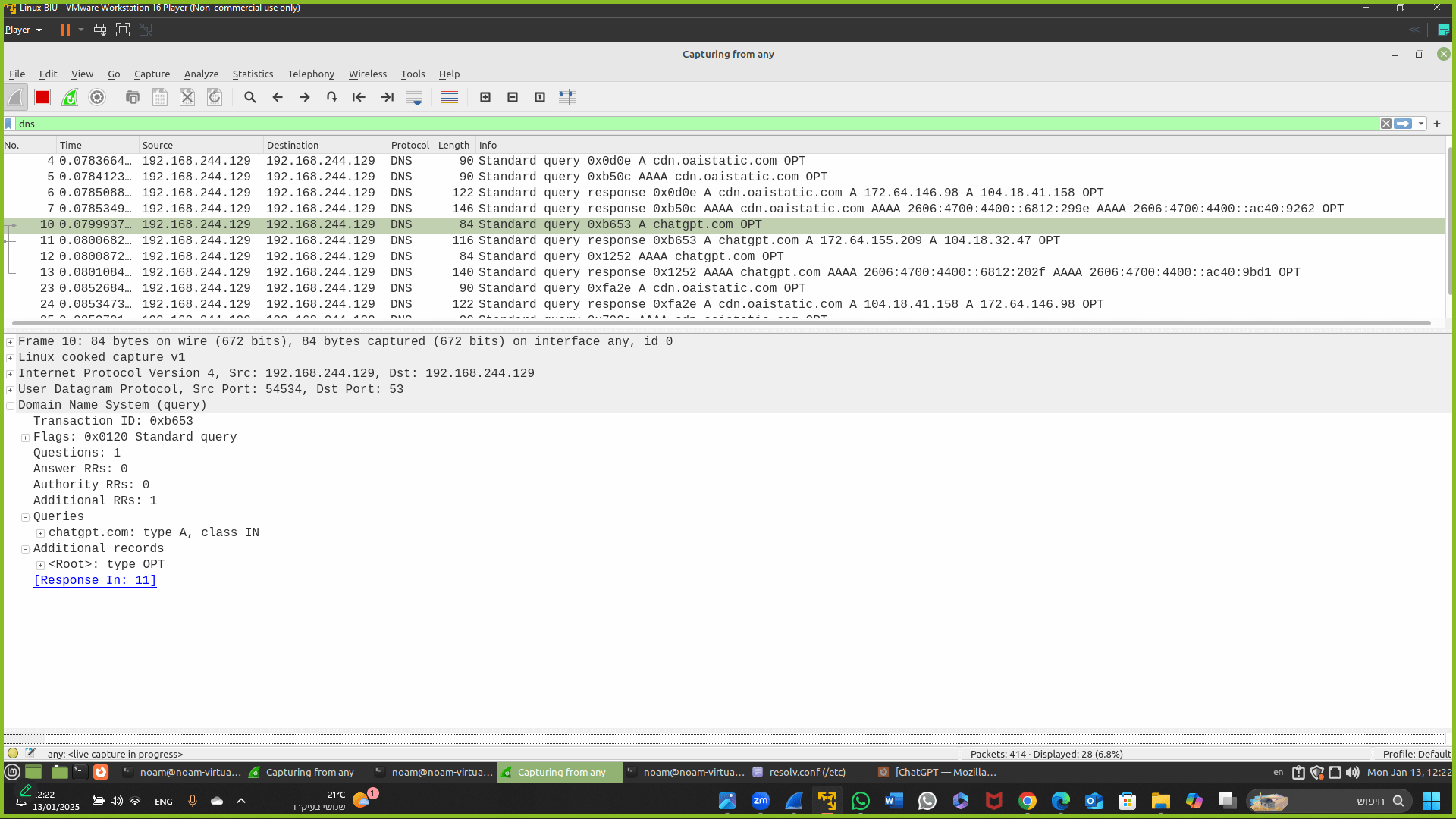


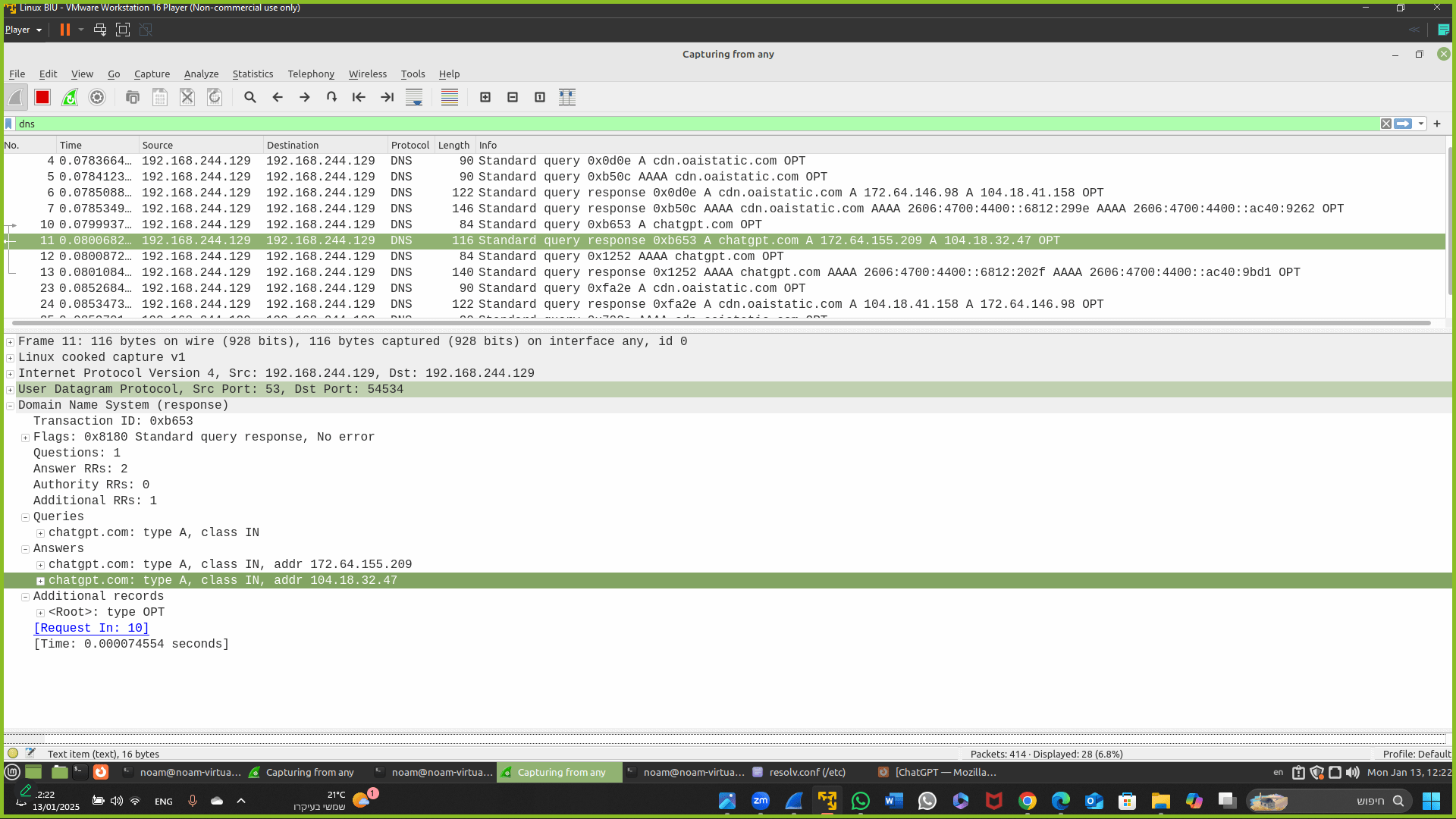


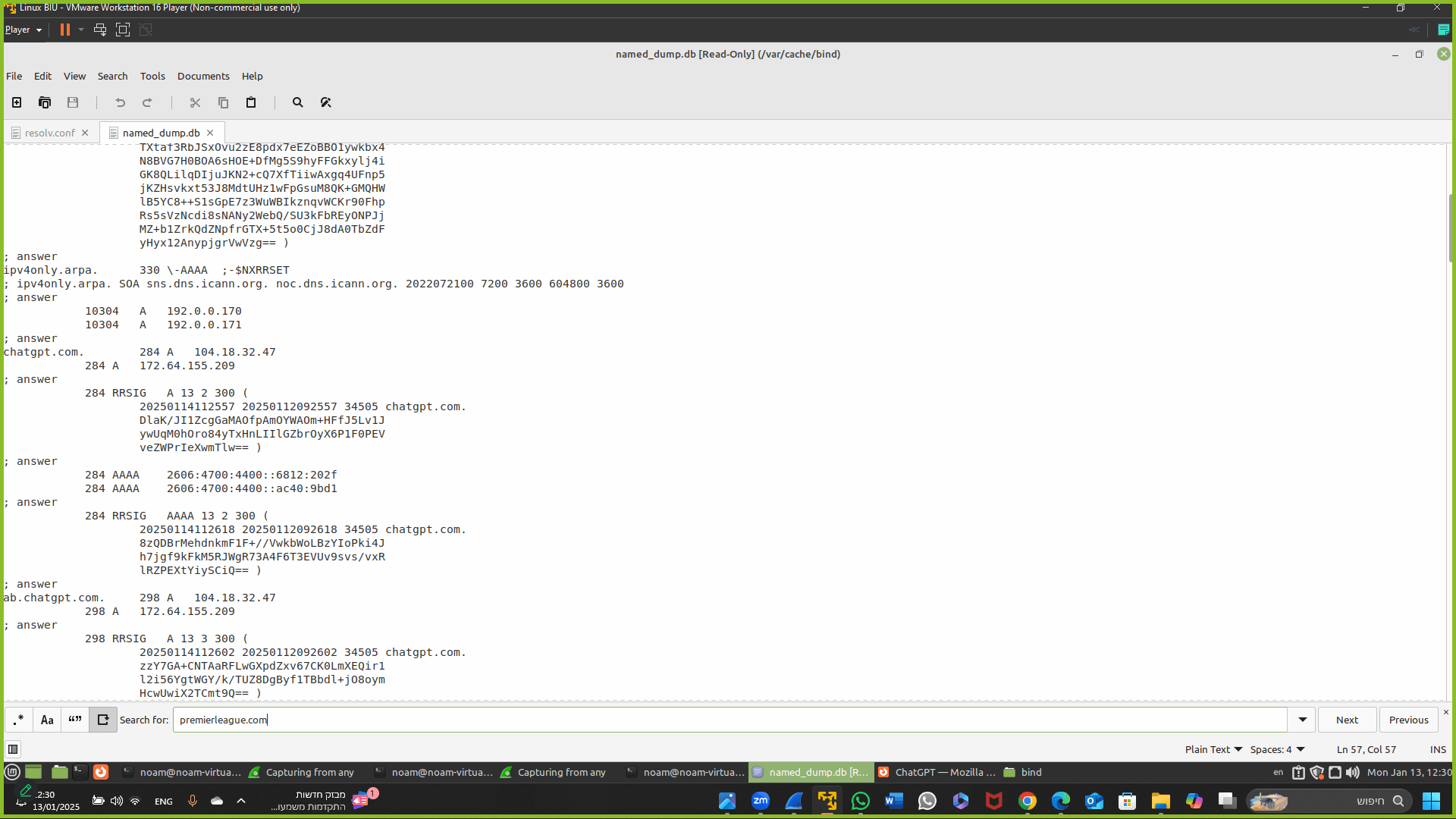


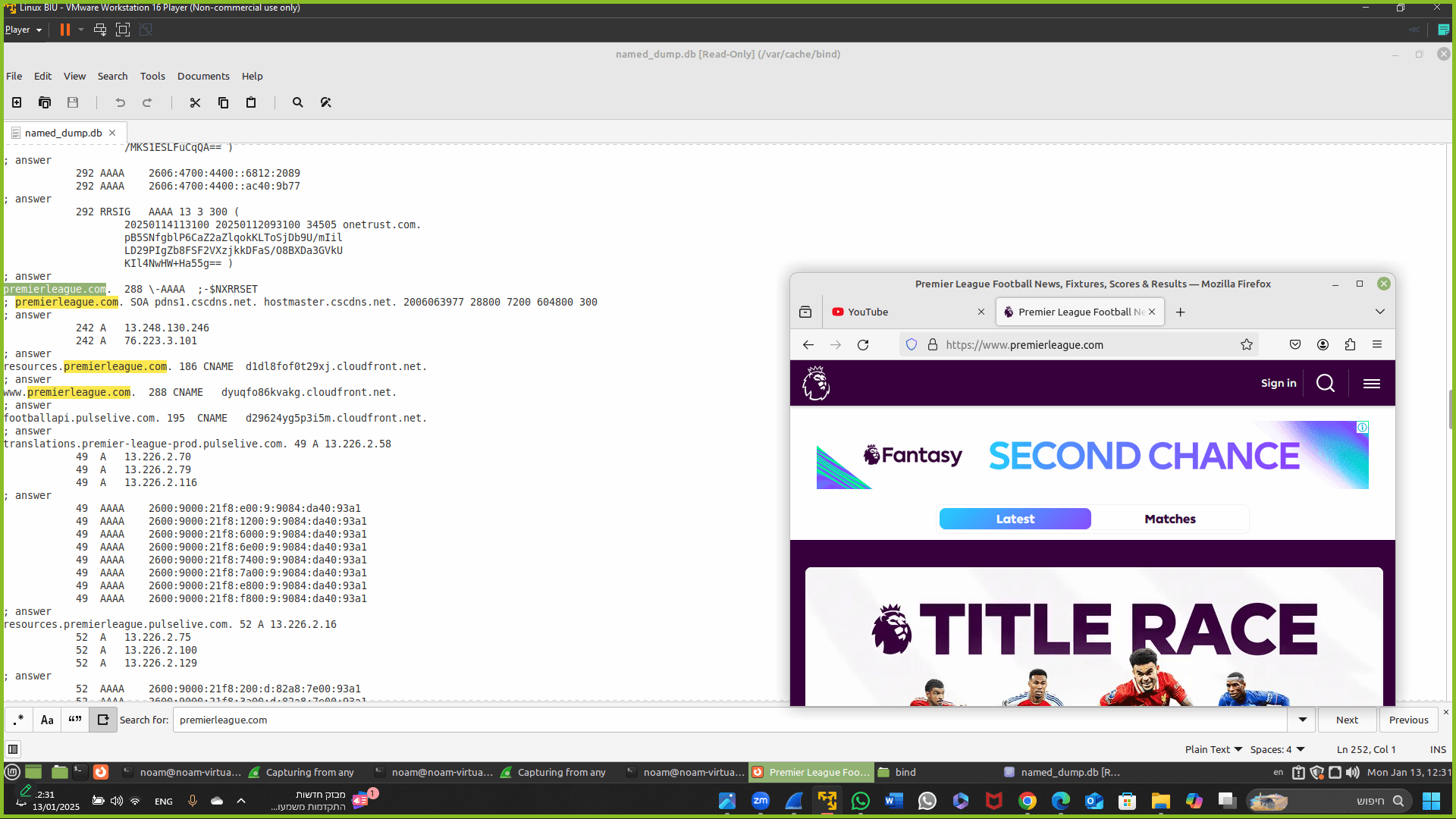


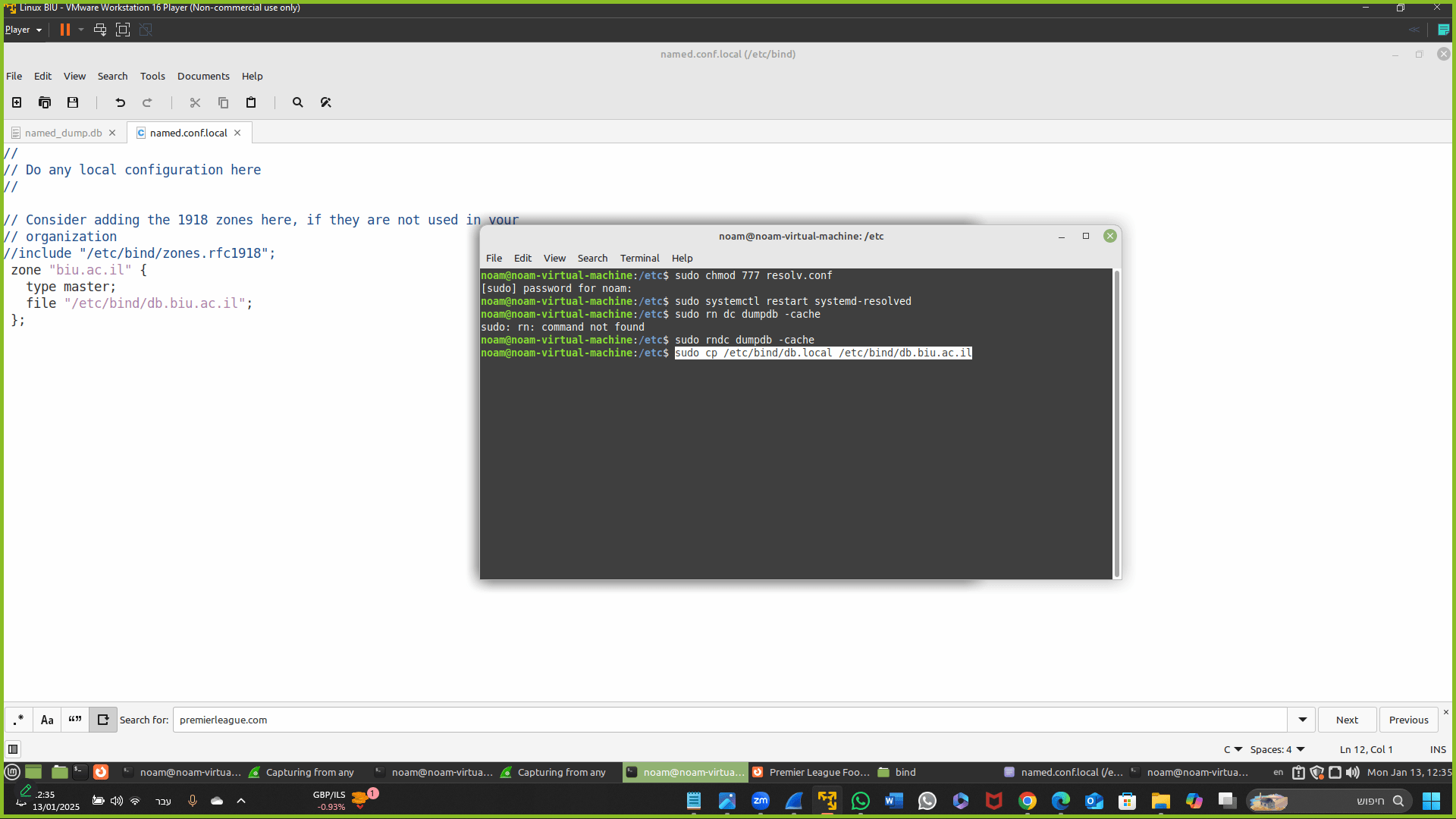


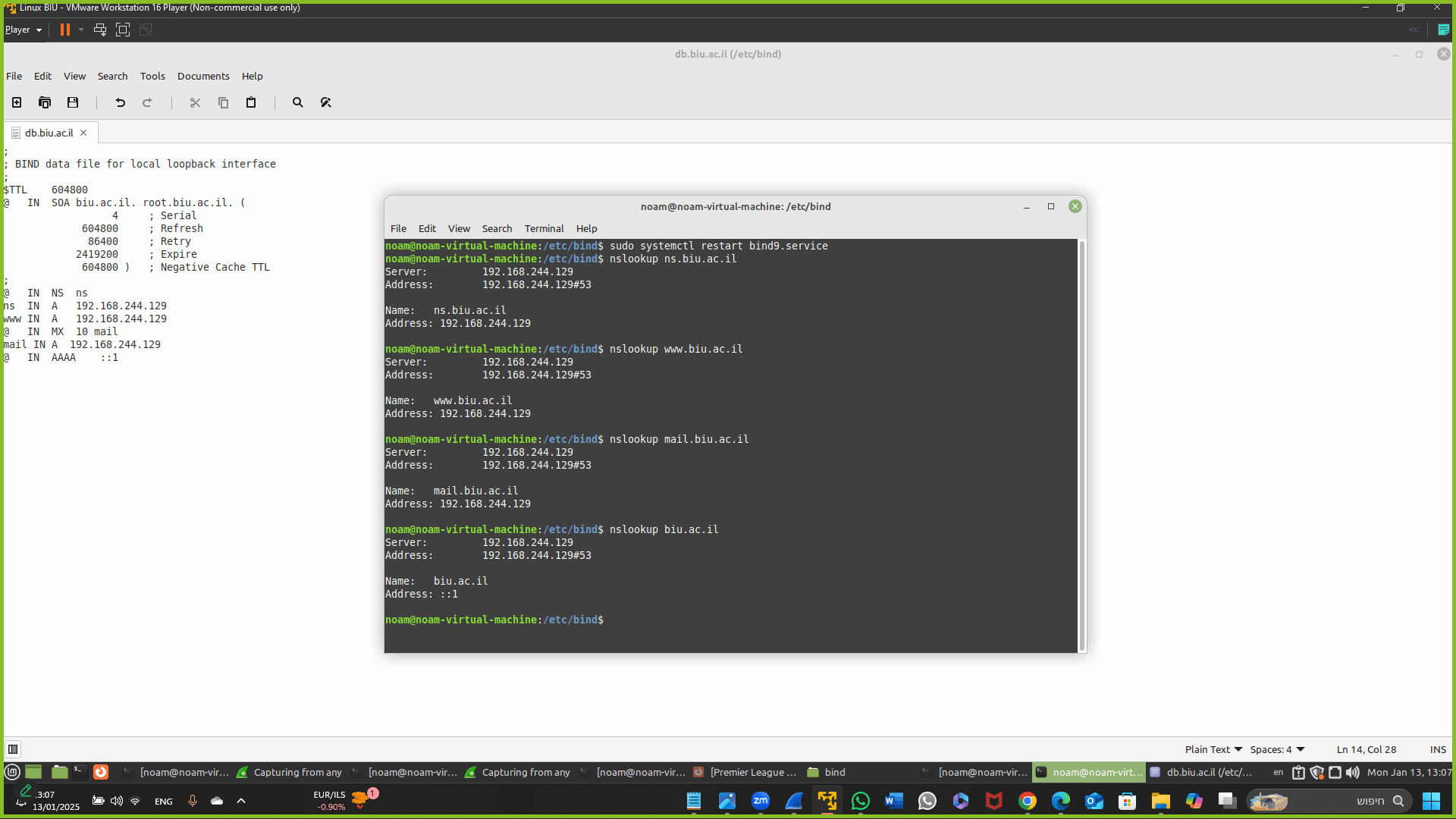


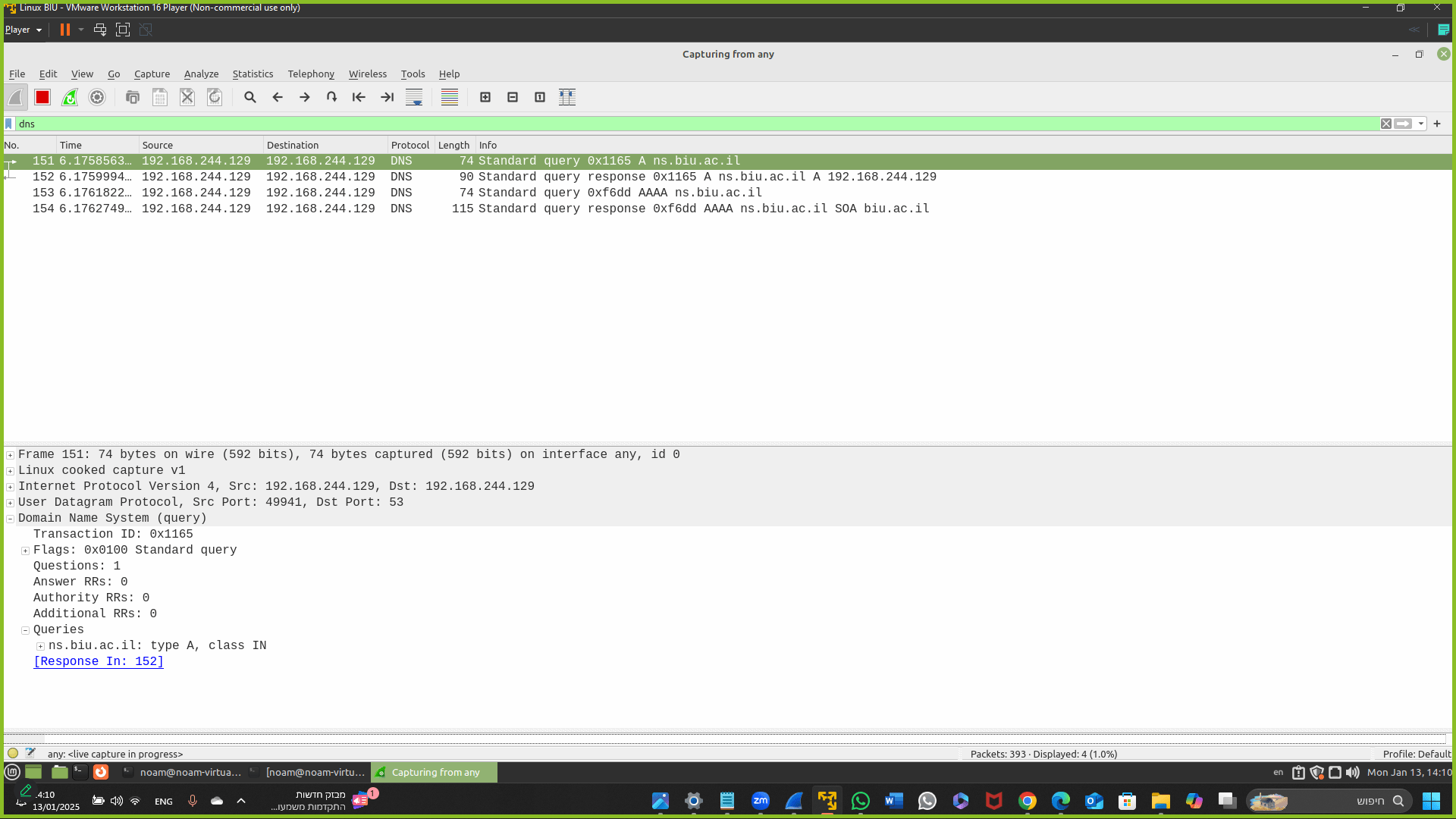


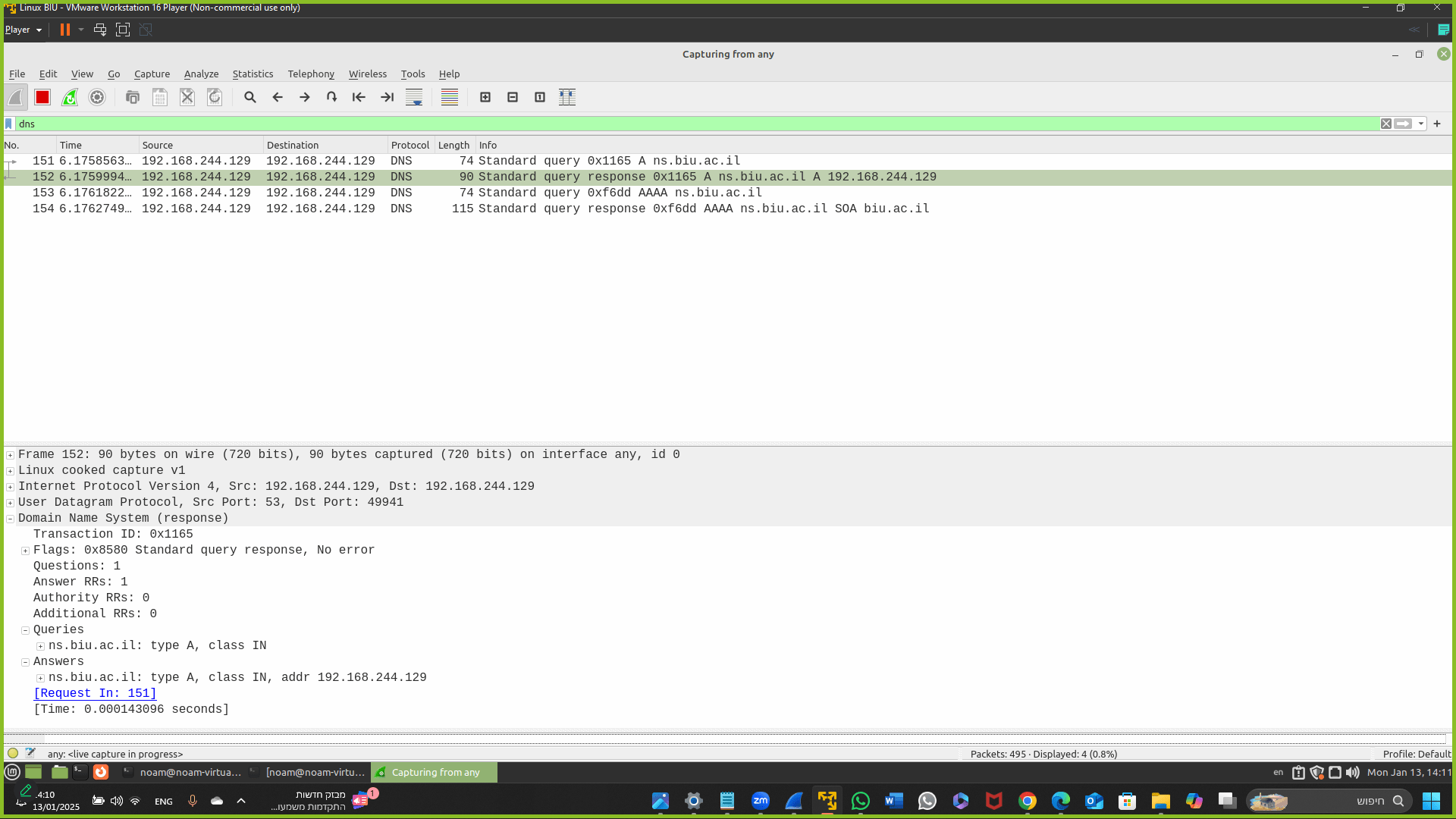


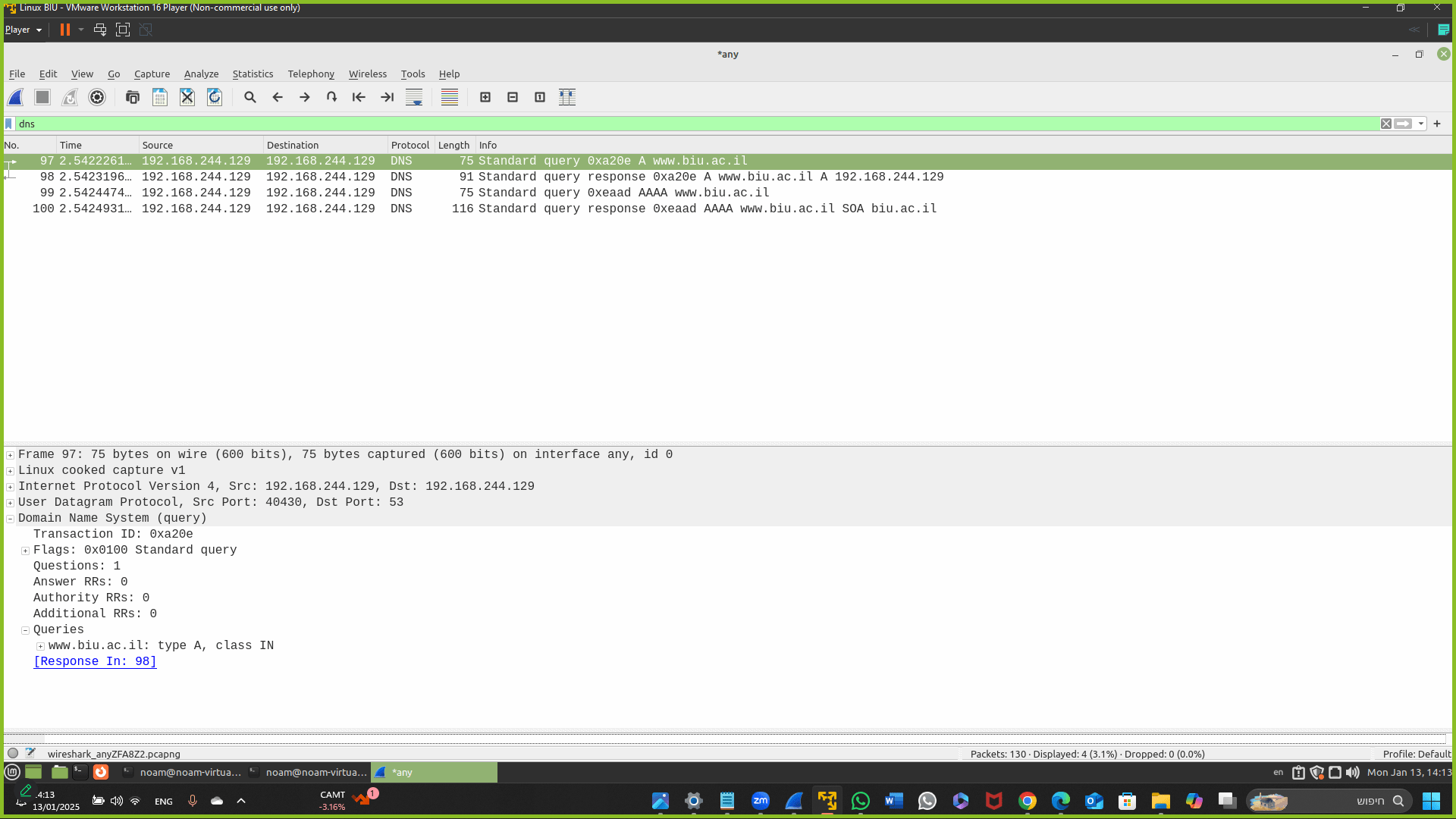


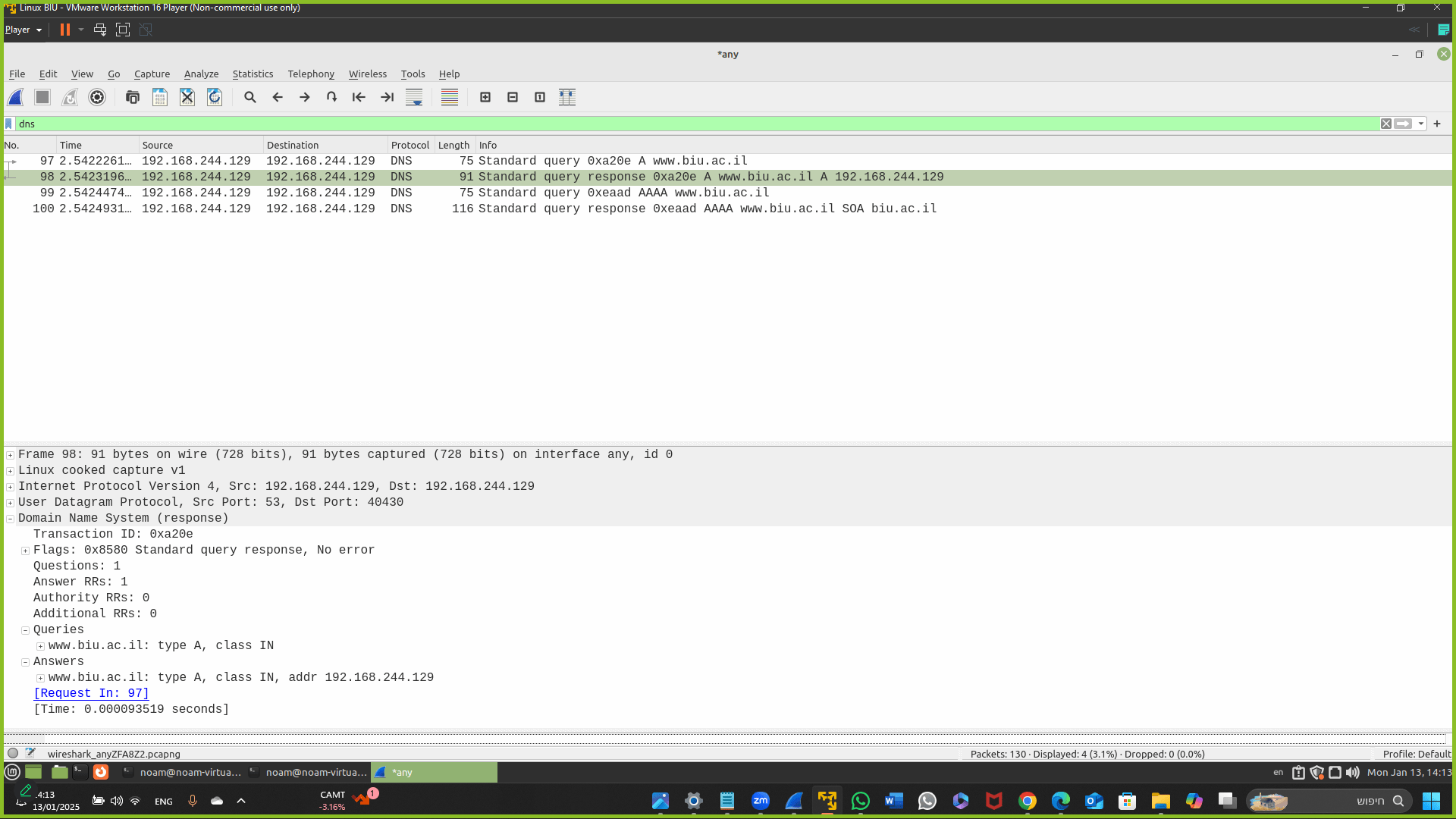


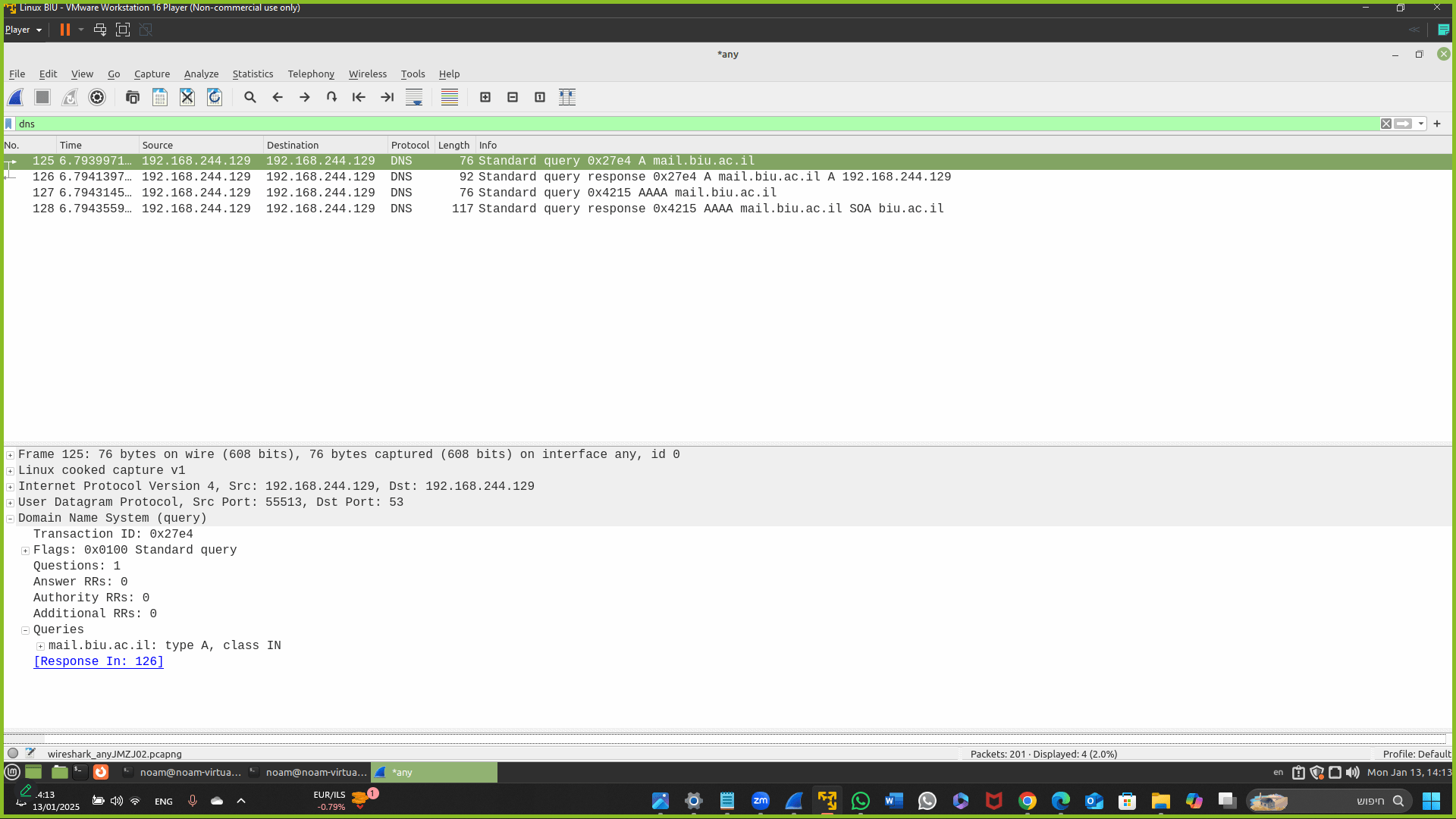


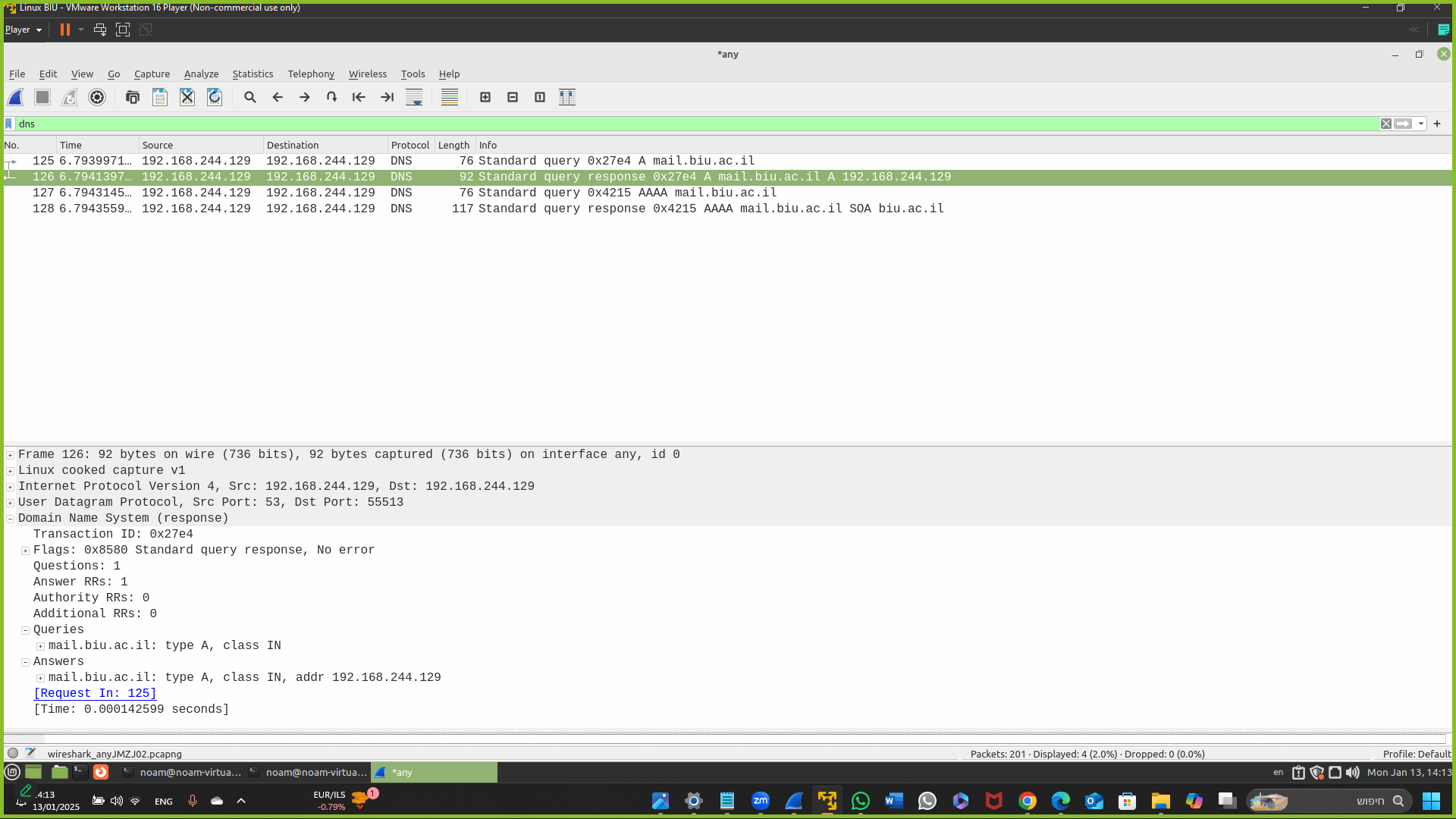




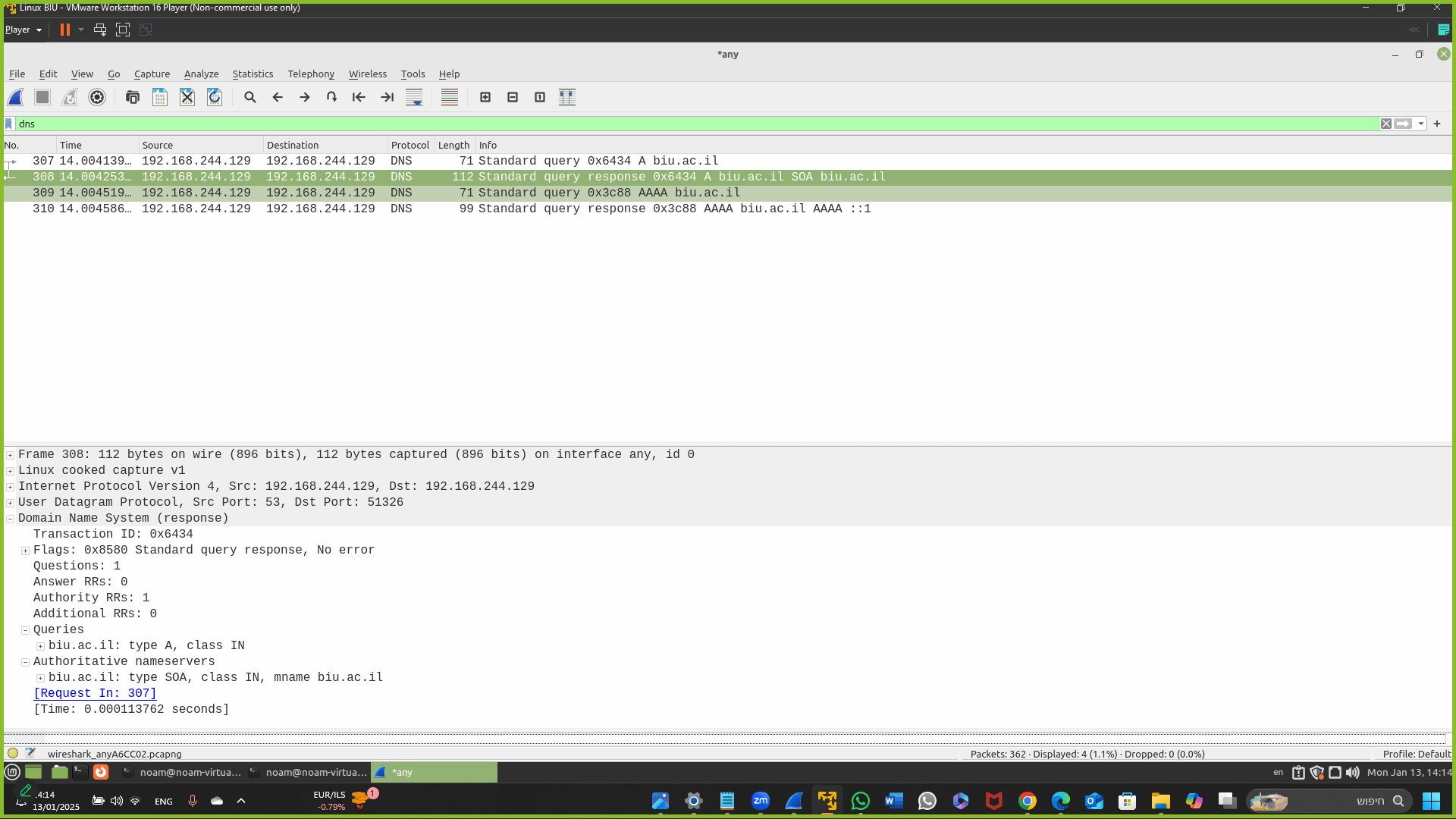




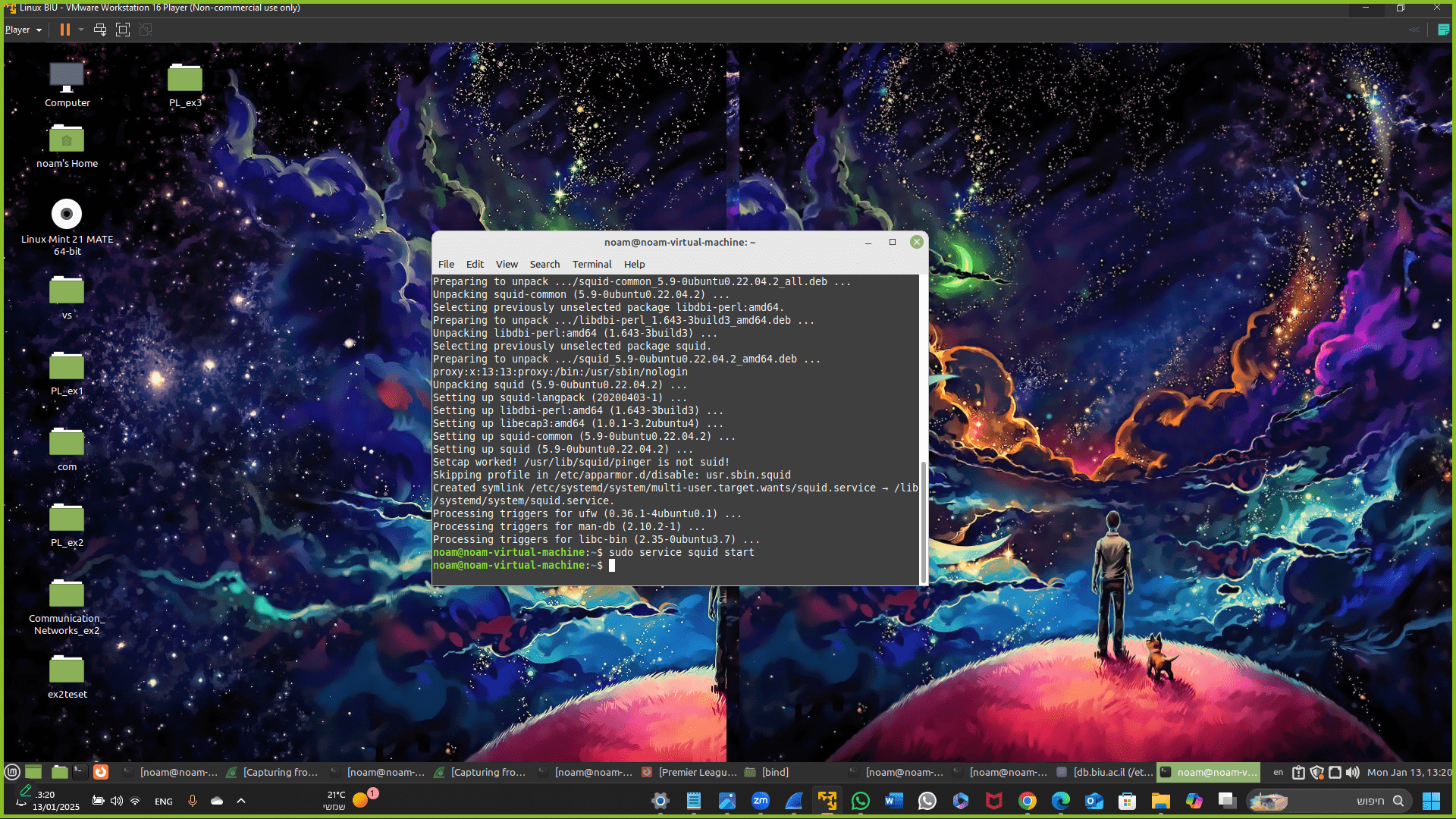


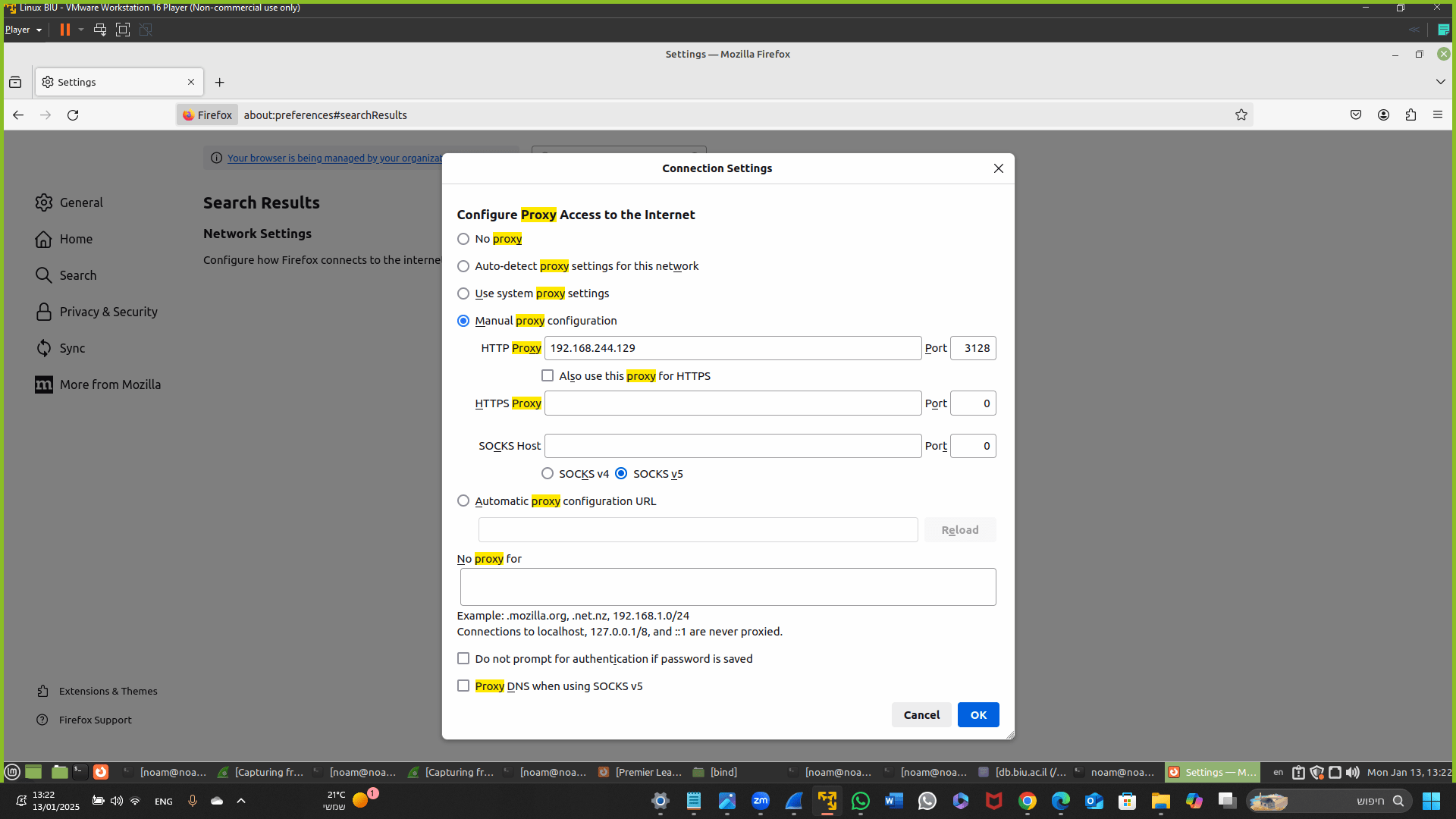


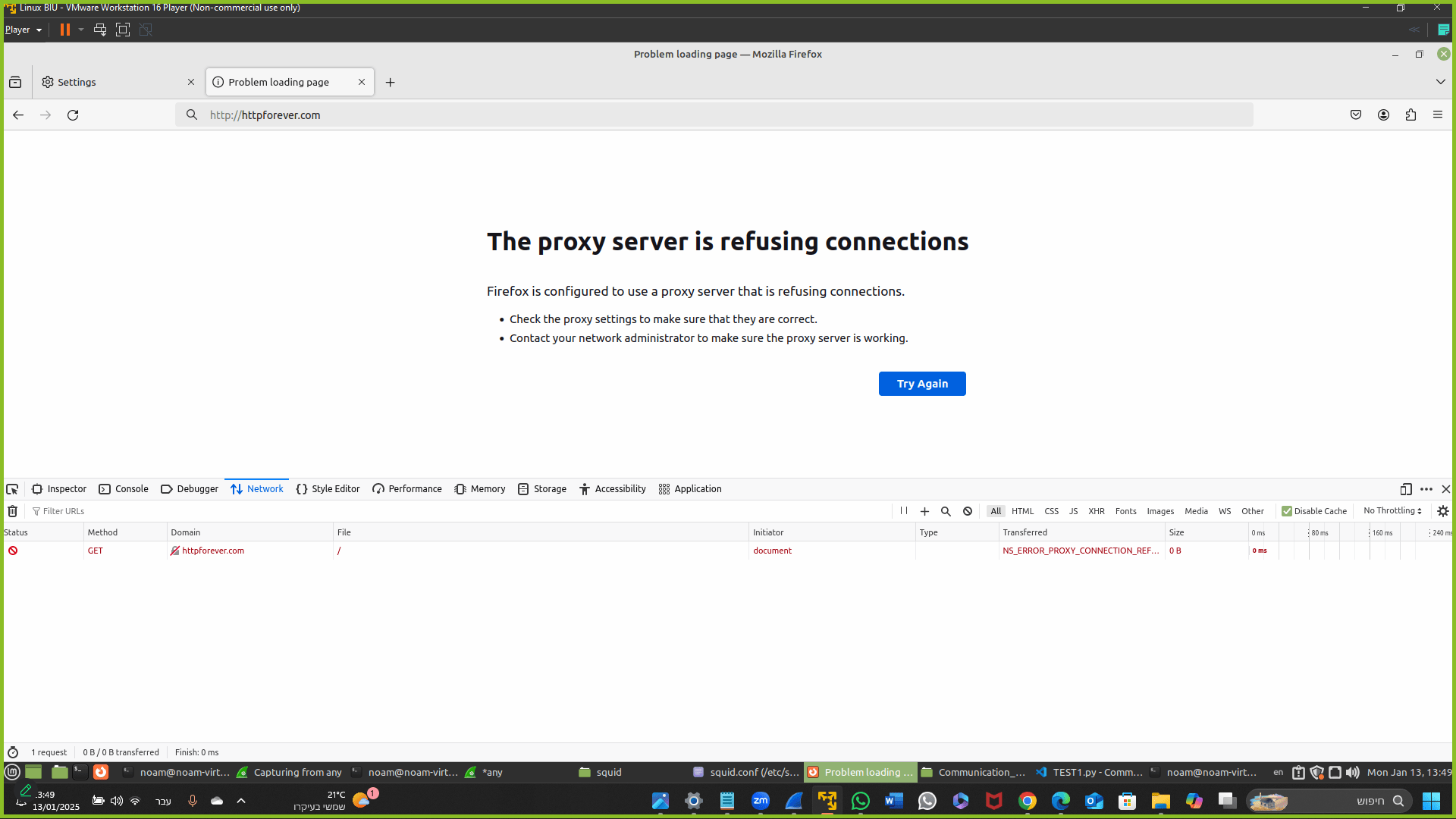


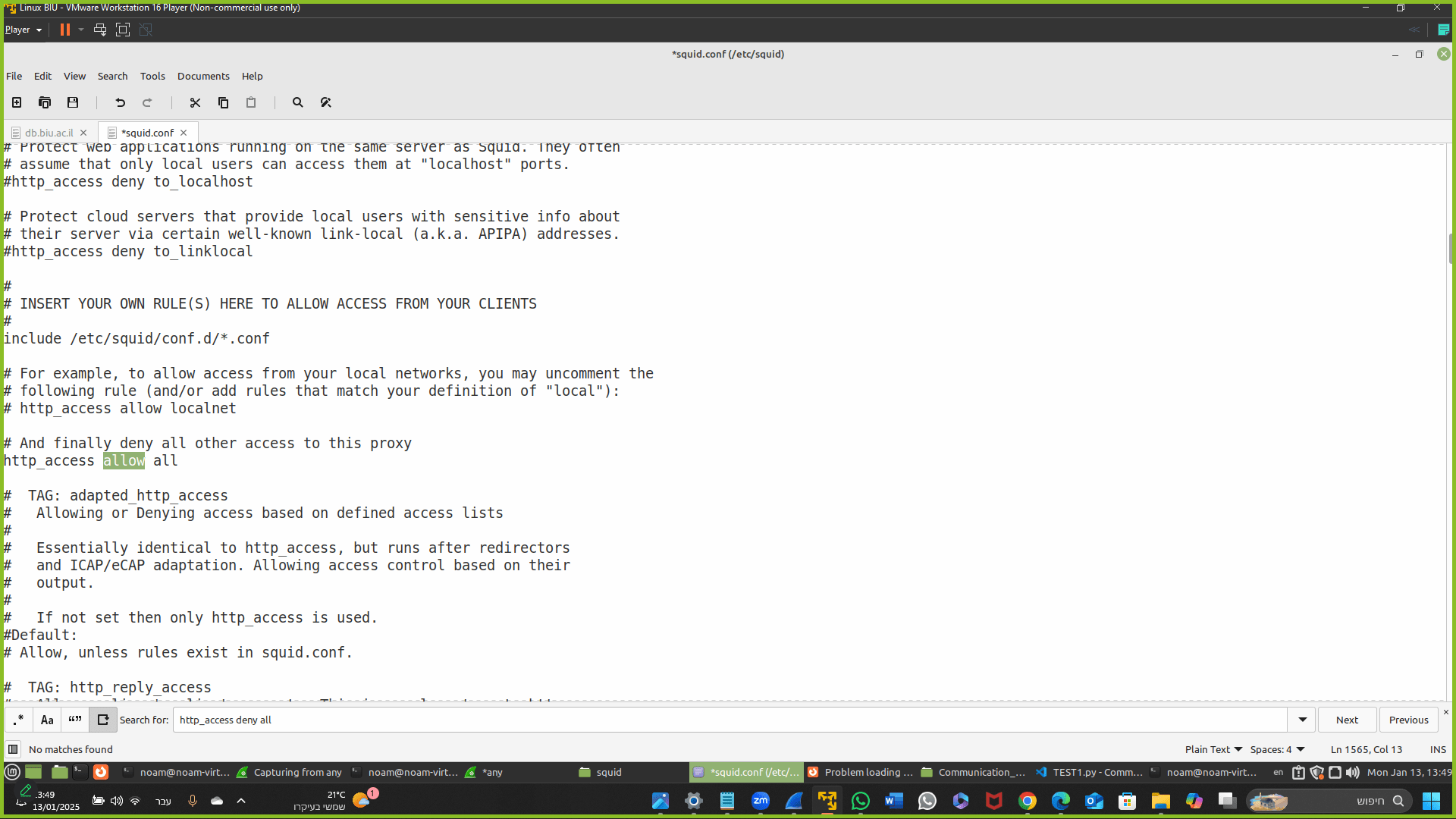


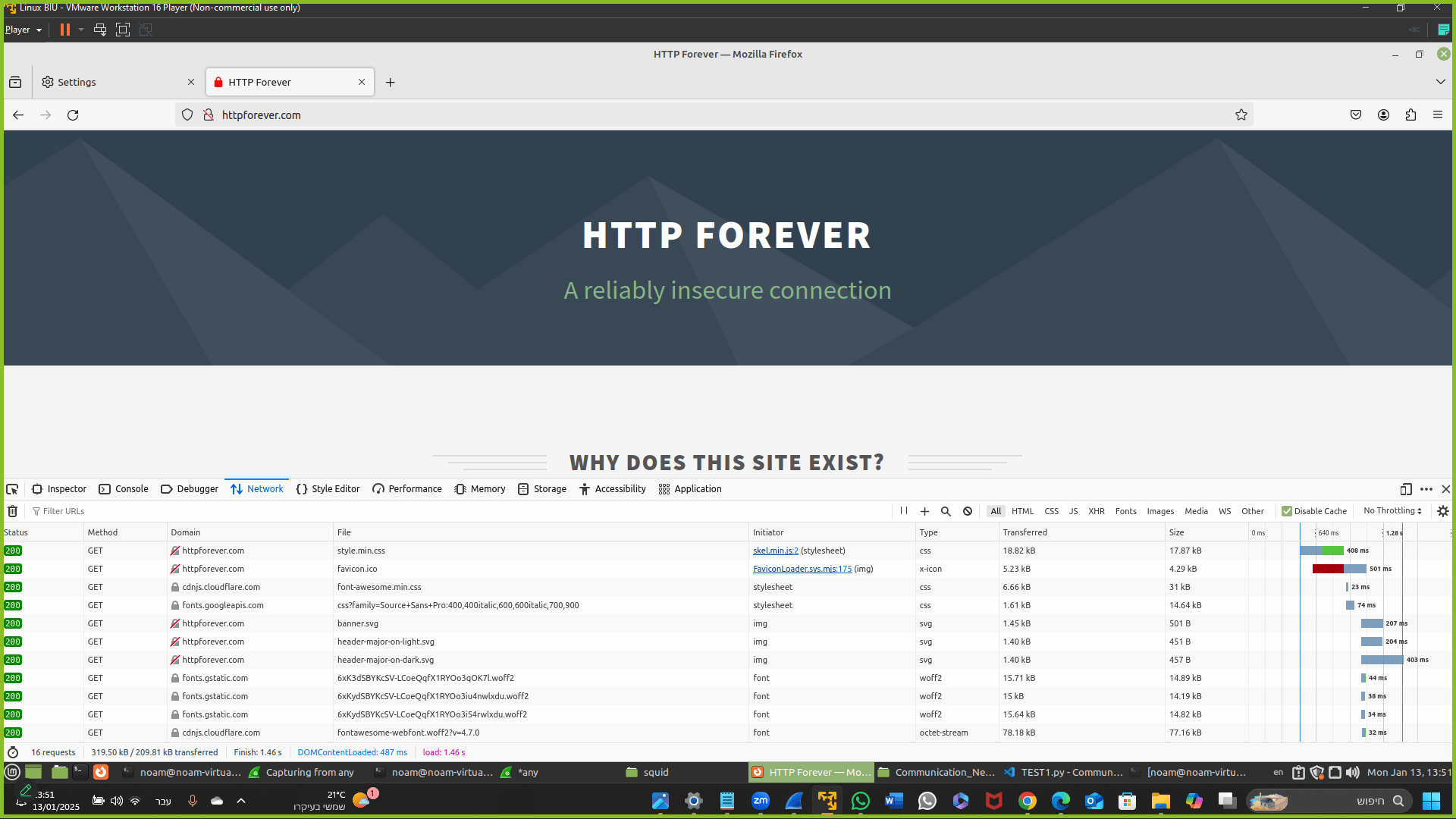
עד כאן חלק 2

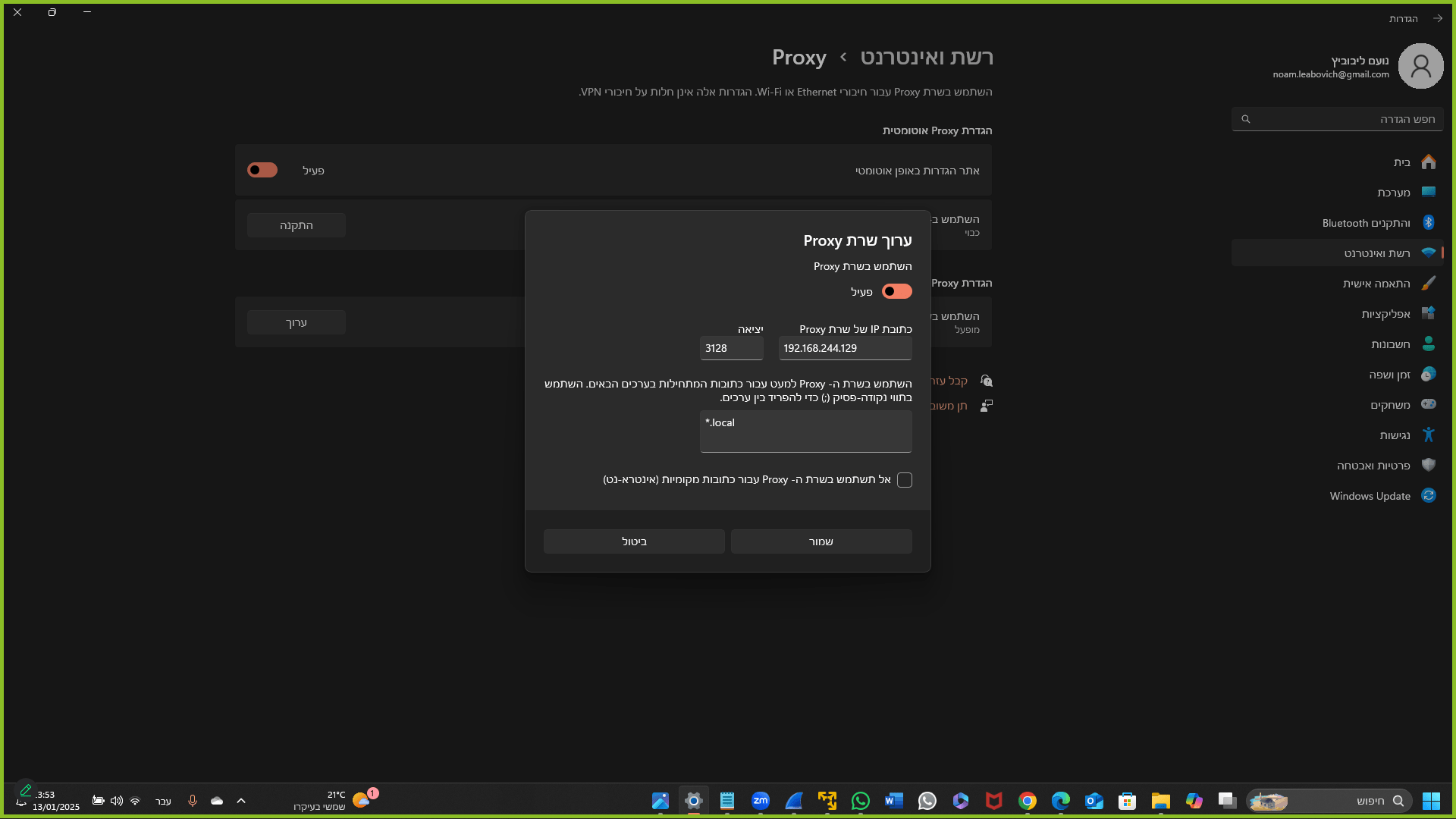


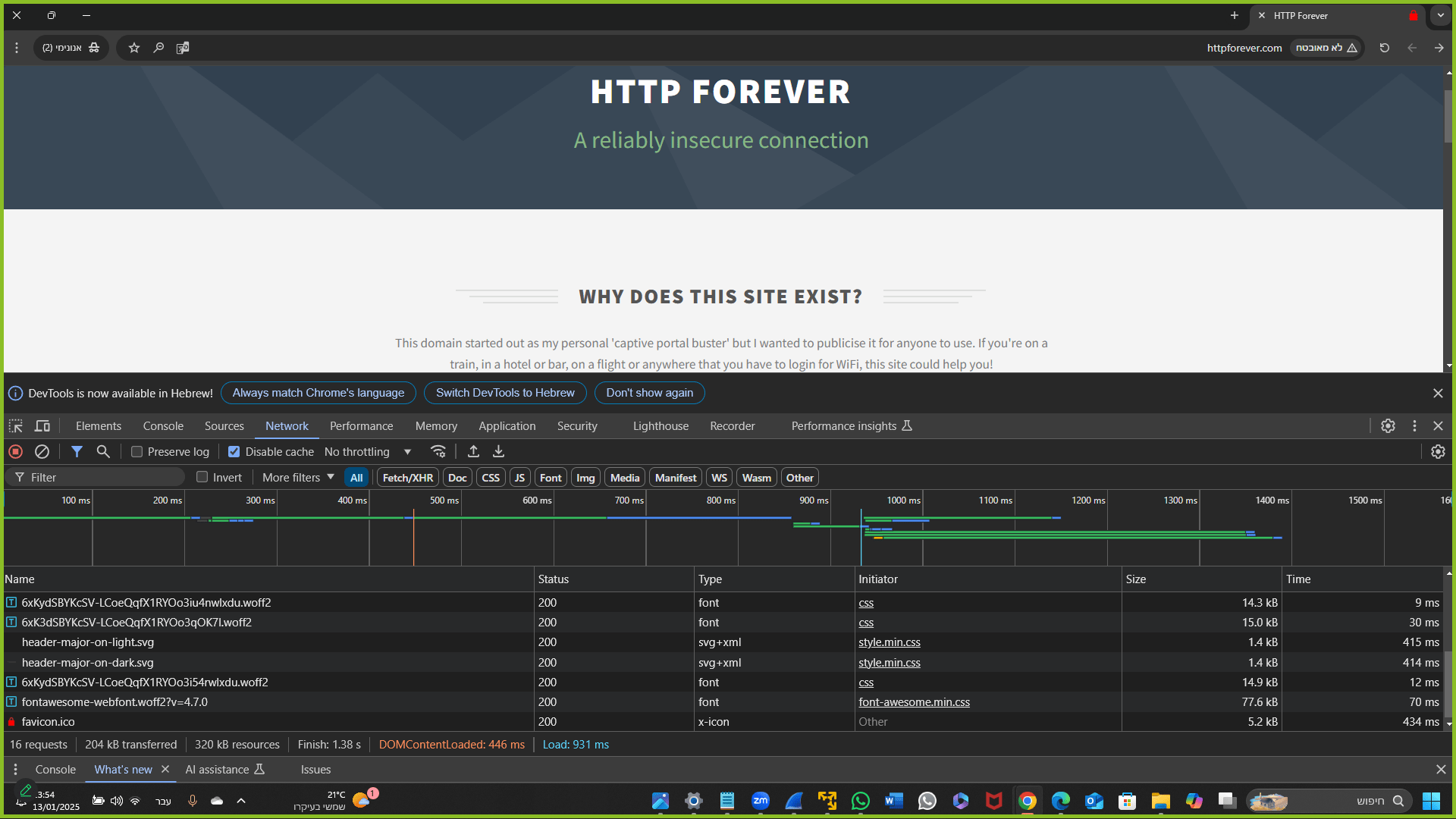


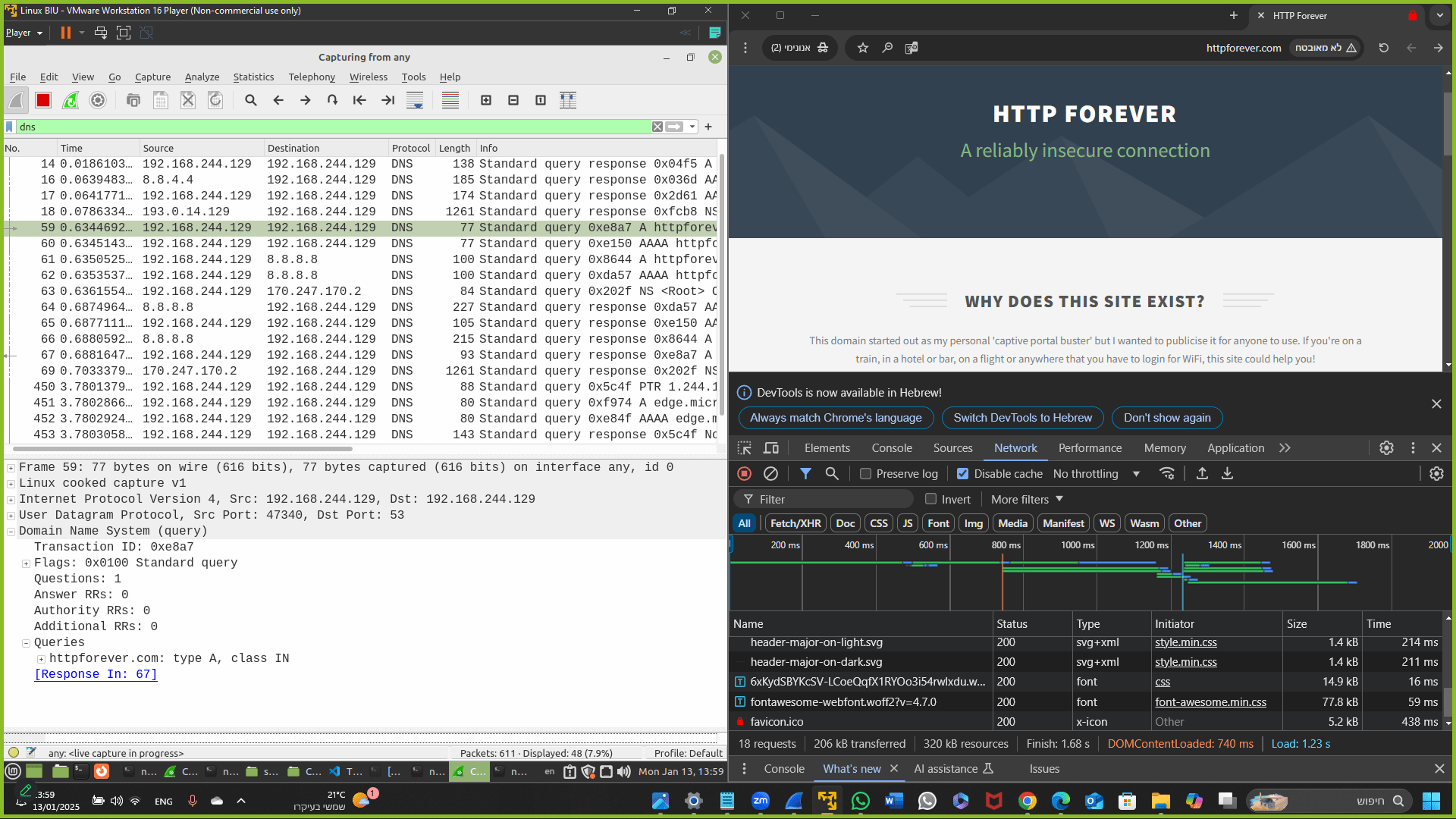


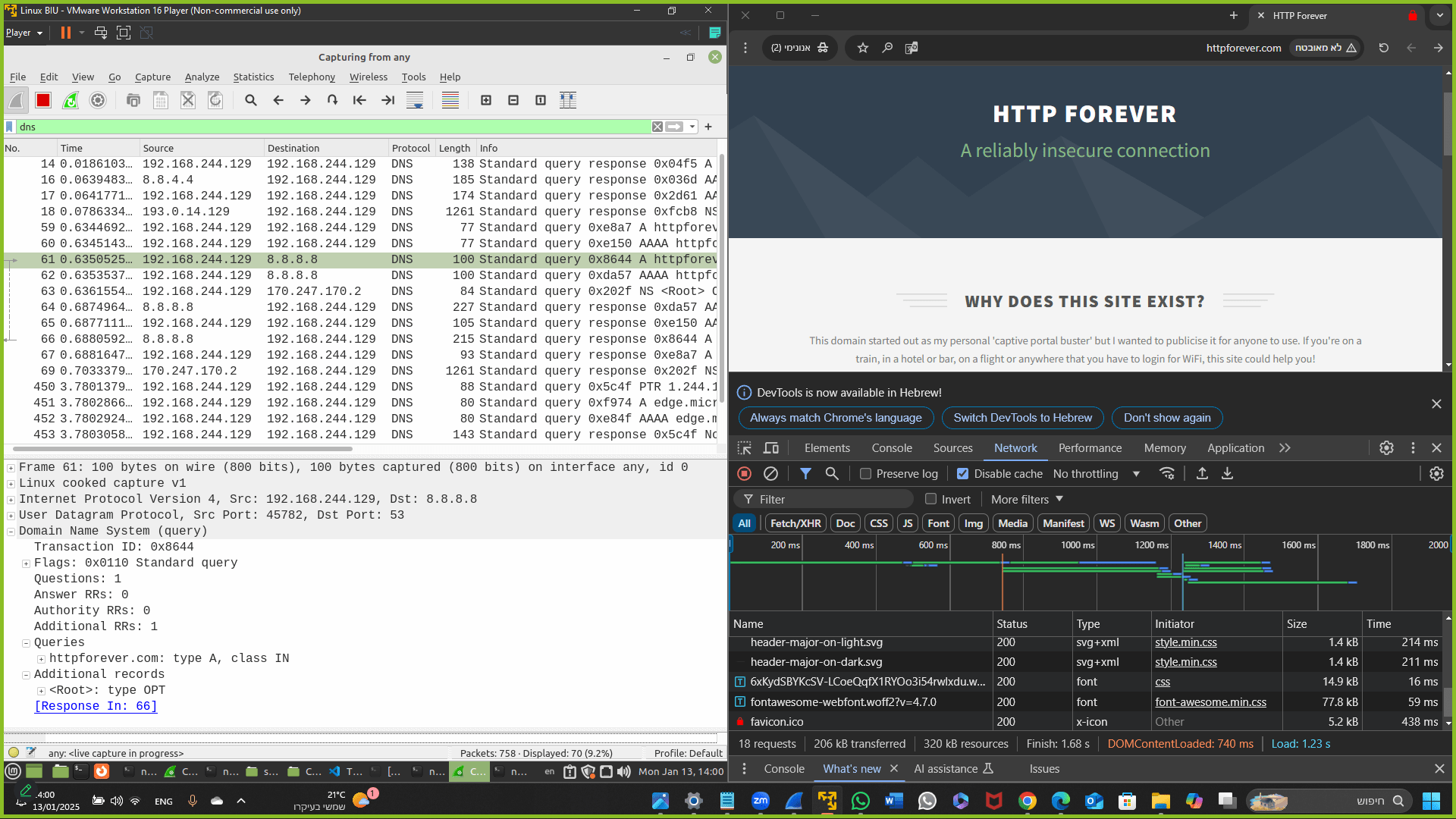


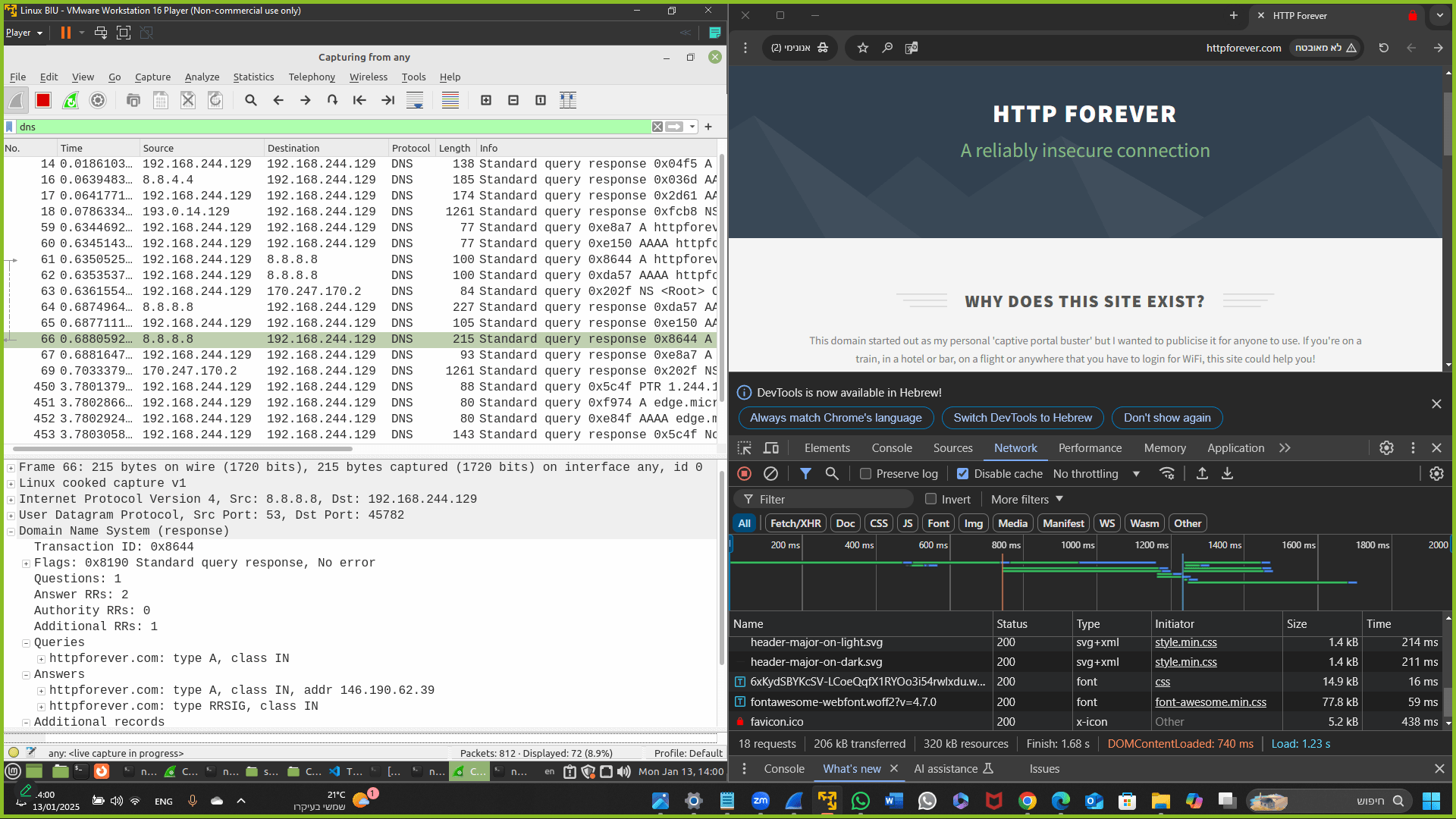


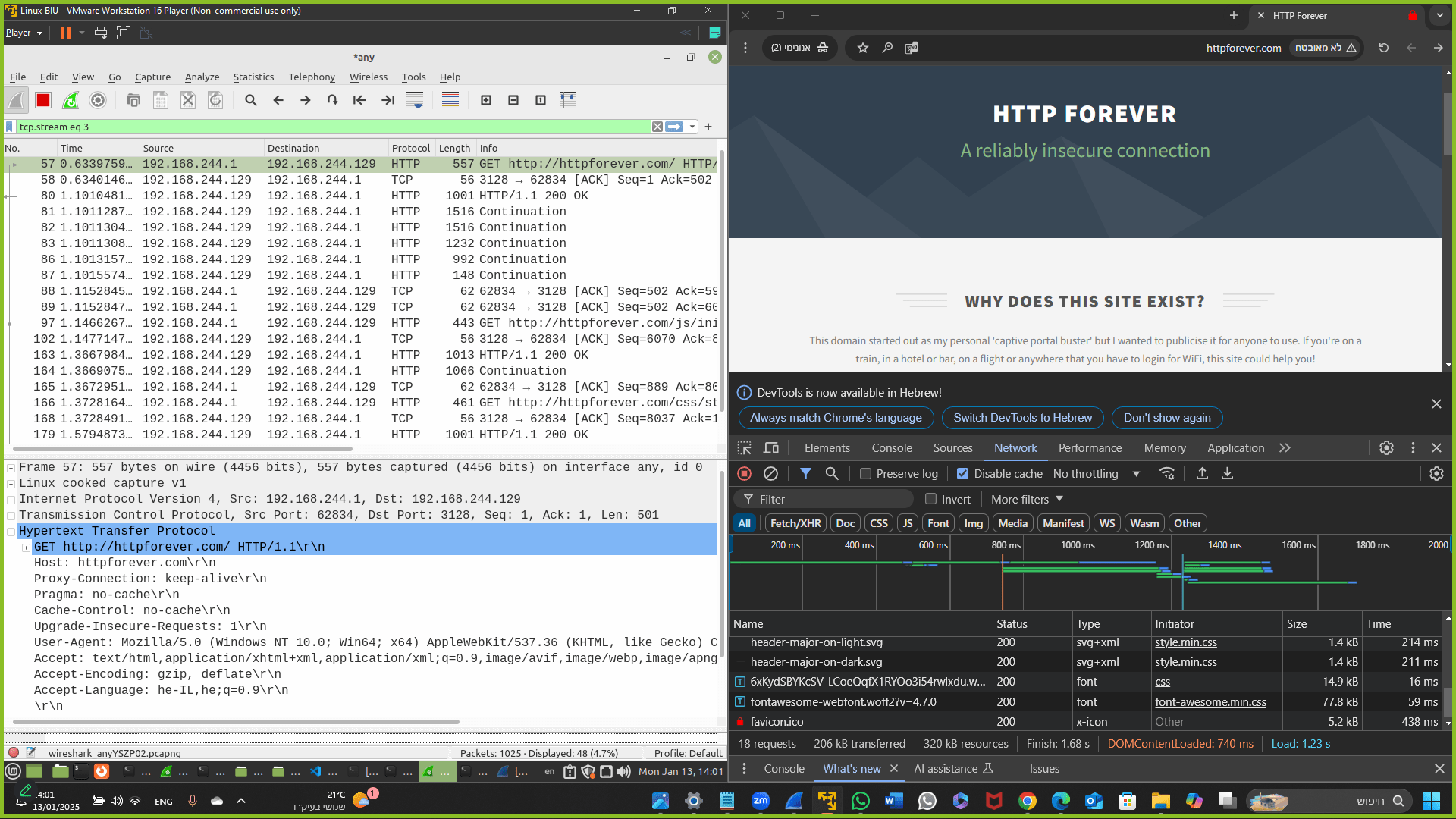


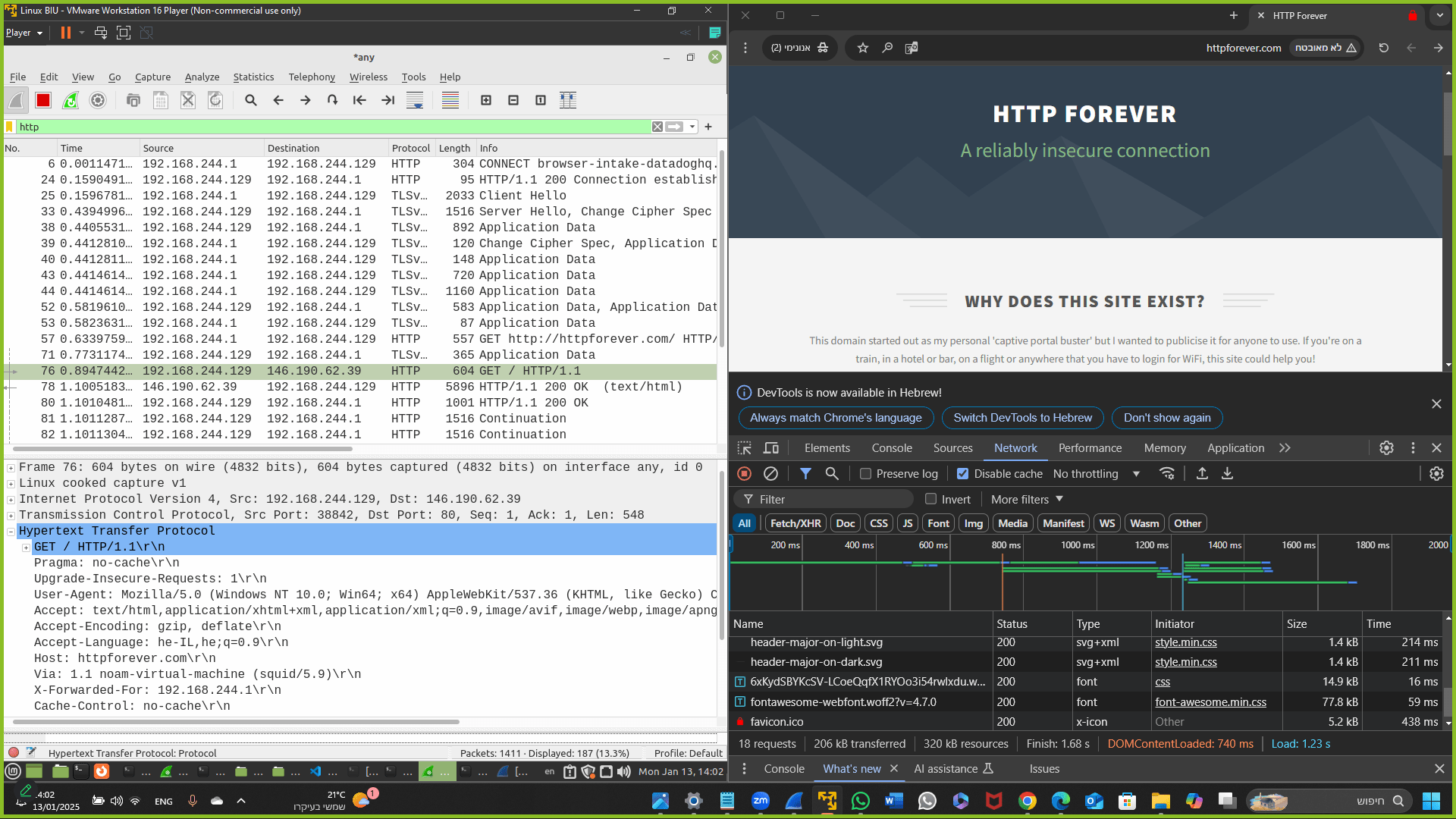


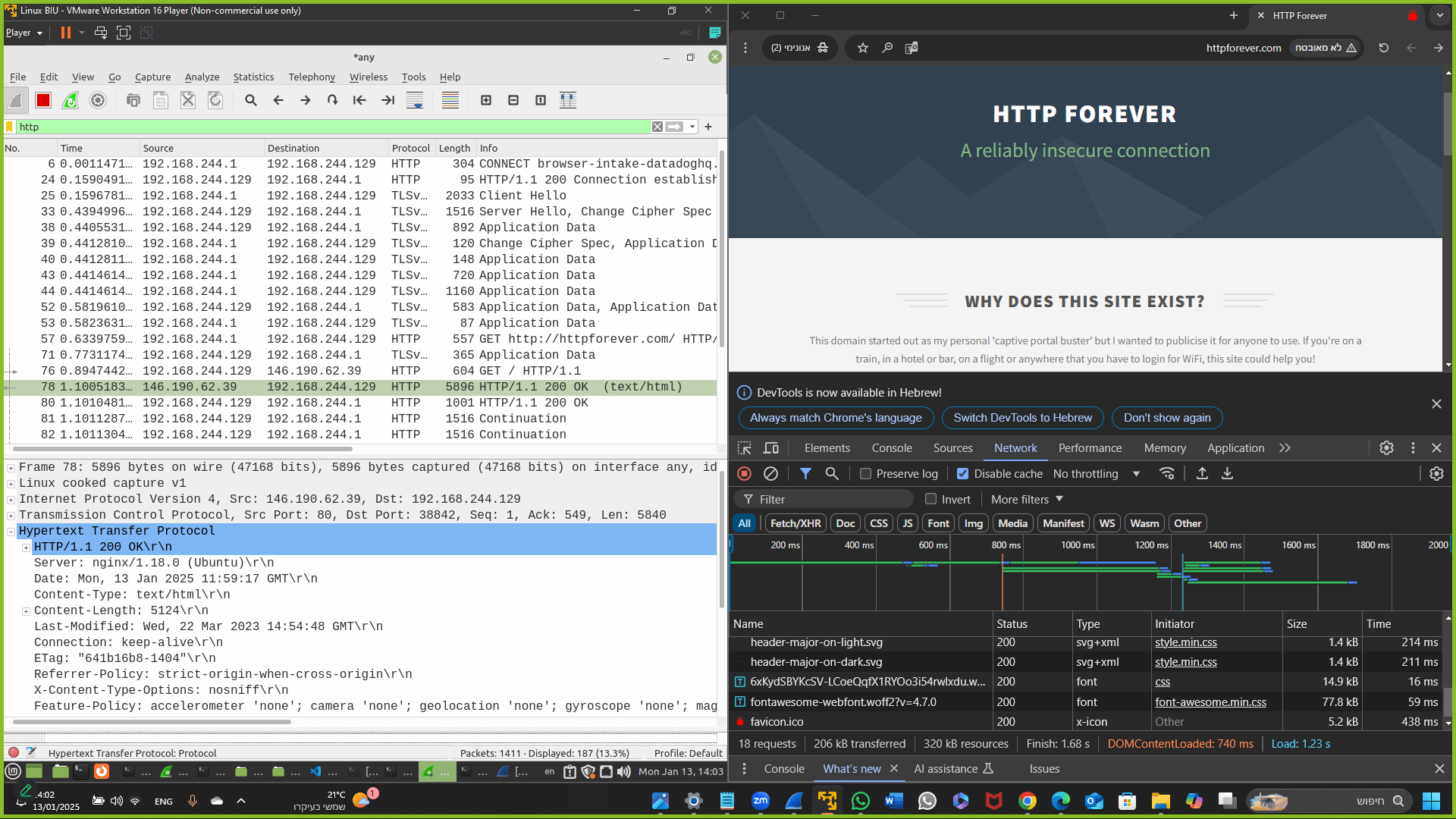


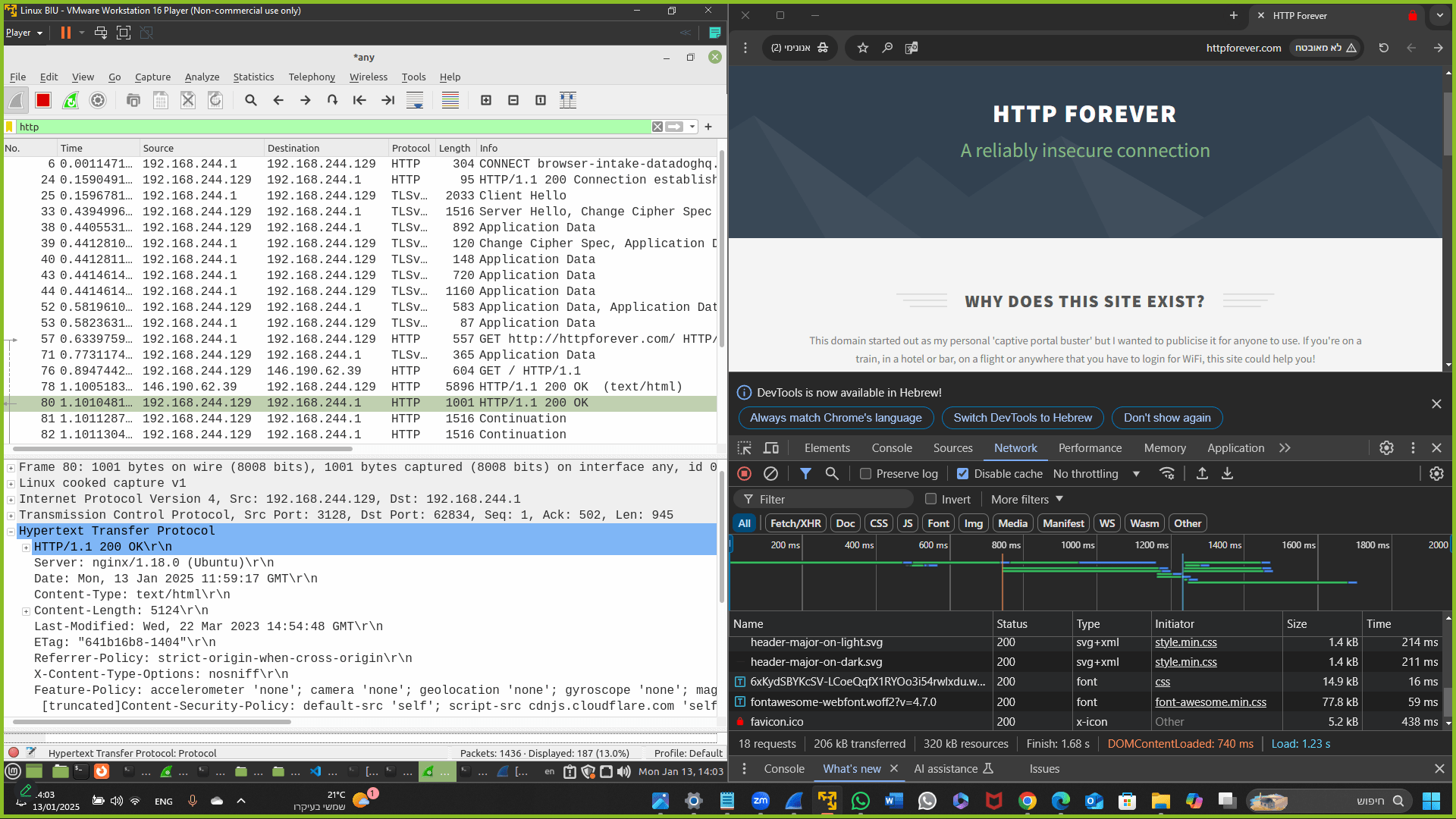












עד כאן חלק 3