מטלה מס' 2 – אילאי בן יהושע ונועם ליבוביץ

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטיראשית נתאים ונשנה את הקוד כך שישלח את שמותינו לשרת, באופן הבא –

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

כעת נריץ את הקודים המעודכנים הנ"ל, נסניף את המידע בעזרת "כריש-הכבל" (בלעז – wireshark, מעתה והילך כאשר נכתוב 'כריש' בקיצור נתייחס למינוח הזה), וננתח את מה שקרה פה בהתאם לעקרונות TCP שראינו בהרצאה ובתרגול.

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטיזה מה שמוצג לנו בכריש לאחר הרצת הקודים -

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

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי חבילה מס' 1 –

אנו רואים פניה שמתבצעת לPORT - 12345 (אשר אותו נתנו ידנית לשרת), אשר מגיעה מפורט מקור 52970 אשר ניתן דינאמית ללקוח ע"י מ"ה (כיוון שלא צמדנו אותו לפורט ספציפי). הפנייה היא מסוג syn (ניתן לראות את הדגל הדלוק), כלומר הלקוח מבקש להסתנכרן עם השרת וזוהי תפקידה של החבילה הנ"ל עם דגל הsyn, אשר מהווה את השלב הראשון בטקס 'לחיצת הידיים המשולשת'. בנוסף לכך אנו רואים את ערך האופסט הגולמי (= raw) של הseq שבחר הלקוח, כאשר ניתן לראות שהוא החליט שתחילת התקשורת תתחיל מאופסט – '1305627461', ואילו ברור לנו שערך הack שלו כרגע יהיה '0', כיוון שהack שלו, זה הseq של השרת (אשר טרם נבחר..). כמובן שהlen הוא 0 שהרי אין דאטא בהודעת syn.

תמונה שמכילה טקסט, גופן, תוכנה, צילום מסך

התיאור נוצר באופן אוטומטידבר **מעניין** נוסף ניתן לראות כאשר נפתח את הoptions –

ובו, הלקוח מציין בפני השרת – "רק שתדע, הmss שלי הוא 1460b.".

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטיחבילה מס' 2 –

אנו רואים פניה שמתבצעת הפעם בכיוון ההפוך מPORT - 12345 (השרת), אשר מיודעת לפורט 52970 (הלקוח). הפנייה (או יותר נכון 'תגובה') מצד השרת הינה מסוג syn, ack (גם פה ניתן לראות את **שני** הדגלים דלוקים), כלומר השרת ראשית מכיר בסנכרון של הלקוח איתו, אך בנוסף הוא מבקש להסתנכרן גם הוא עם הלקוח. זוהי למעשה תפקידה של החבילה הנ"ל עם דגלי הsyn, ack, אשר מהווה את השלב השני בטקס 'לחיצת הידיים המשולשת'. ראוי לציין כי השלב הנ"ל עשוי להתבצע לעיתים ב2 חבילות שונות (1 לack ואחת לsyn) אך כמו שאנו רואים כאן, אלו יכולות להישלח גם כן יחדיו בחבילה אחת. בנוסף לכך אנו רואים את ערך האופסט הגולמי (= raw) של הseq שבחר השרת, כאשר ניתן לראות שהוא החליט שתחילת התקשורת תתחיל מאופסט – '3834986605', ואילו נשים לב שערך הack שלו יהיה '1305627462', שזה 1 יותר משהיה הseq של הלקוח בגלל הphantom-bit. גם כאן הlen הוא 0 שהרי אנחנו עדיין לא בשלב של העברת דאטא, אך מאוד מתקרבים לשם. אם נפתח את הoptions נשים לב שגם השרת מציין בפני הלקוח מהו הmss שלו – 1460b, כדי שזה ידע את הנתון כאשר הוא שולח אליו הודעות.

חבילה מס' 3 –

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

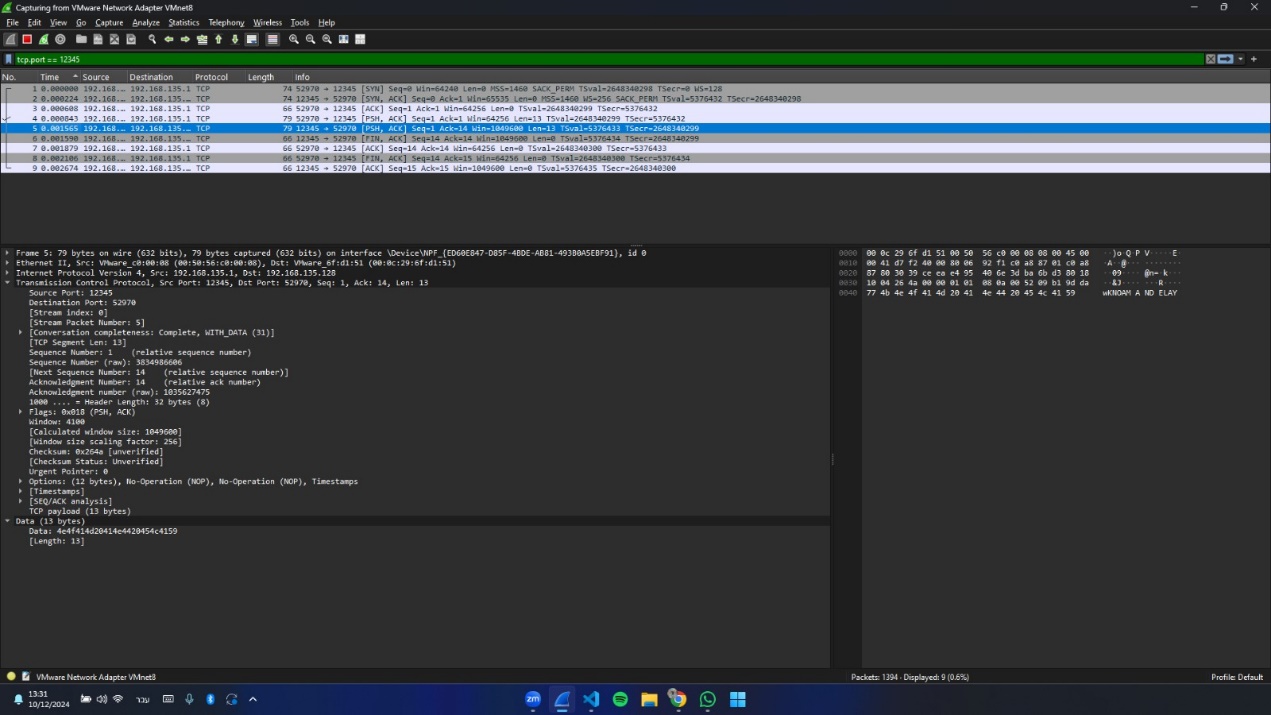
התיאור נוצר באופן אוטומטי

כעת ניתן לראות פניה נוספת מהלקוח 52970 (הלקוח), אל השרת. הפנייה מצד השרת הינה מסוג ack (שוב הדגל דלוק ולמעשה מעתה והילך הוא יהיה דלוק עד סוף התקשורת, אז נפסיק לציין זאת...), דהיינו הלקוח גם הוא מכיר בהכרת השרת בסנכרון שלו, ולמעשה תפקידה של החבילה הזו מהווה את השלב השלישי **והאחרון** בטקס 'לחיצת הידיים המשולשת', שכן שני הצדדים הסתנכרנו אחד עם השני, וכל אחד מהם הכיר בסנכרון המשותף, ומכאן והילך הקשר שריר וניתן להעביר דרכו הודעות. שוב אנו רואים את ערך האופסט הגולמי (= raw) של הack ששולח הלקוח, והוא – '3834986606', באופן לא מפתיע זה 1 יותר מהseq שבחר השרת (שוב אנחנו מגדילים בגלל הפאנטום ביט של הack). הseq לעומת זאת לא השתנה שכן לא נשלח עוד דאטא, וכתוצאה מכך שוב הlen הוא 0 (אך לא לעוד הרבה זמן). אם נפתח את הoptions נשים לב שהmss שלו כבר לא נשלח, שכן שני הצדדים כבר יודעים ומודעים למגבלות כל אחד של השני.

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

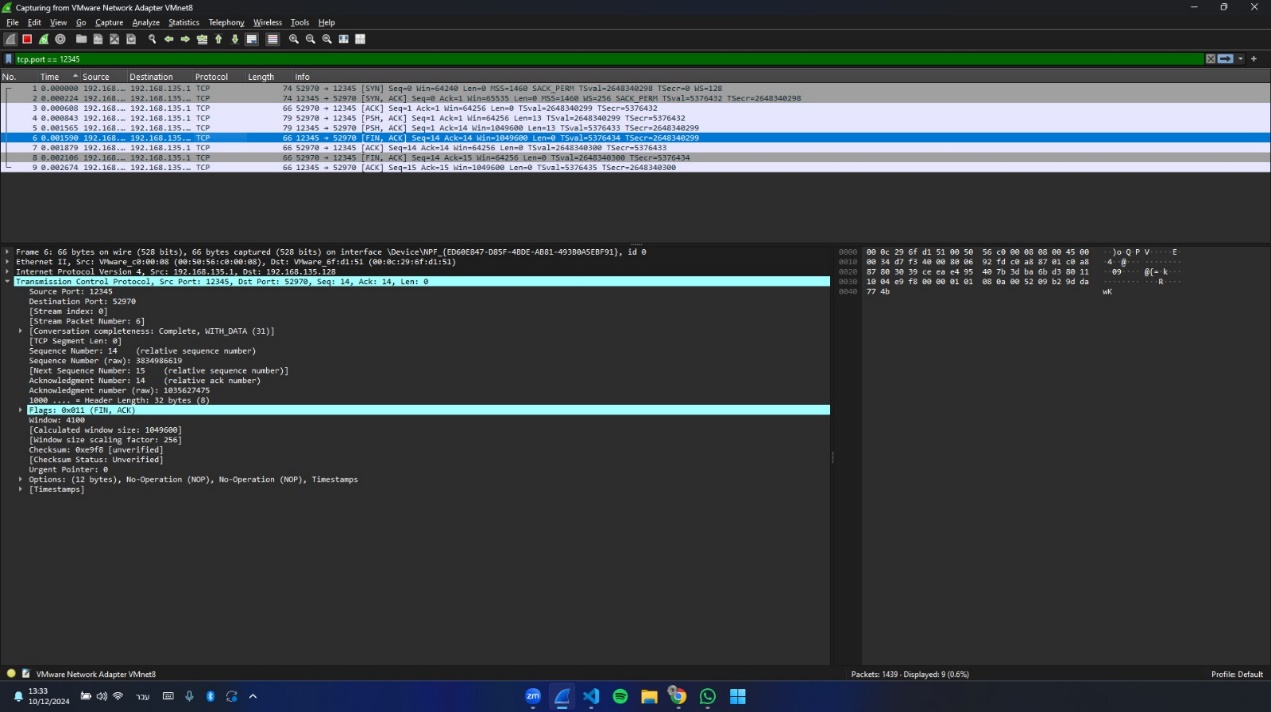
התיאור נוצר באופן אוטומטיחבילה מס' 4 –

סוף כל סוף הגענו לחלק המעניין ('פסגת הקורס' לפי חלק מהשיטות), אנו רואים הודעה מהלקוח (לפי הפורט 52970 שראינו קודם שהוא הפורט של הלקוח), השולח 13 בתים (לפי הlen), שאנחנו יודעים שהם הבתים של המחרוזת 'Noam and Elay' (כנדרש במטלה), ובנוסף לדגל הack (שאמרנו שמעתה והילך תמיד יהיה דלוק), גם כן דלוק דגל הpush שכאילו אומר להעביר בדחיפות לאפליקציה את המידע שהגיע. נשים לב שעכשיו יש לנו גם את שכבת הdata בהודעה (הבתים ששלחנו מאופן מקודד).

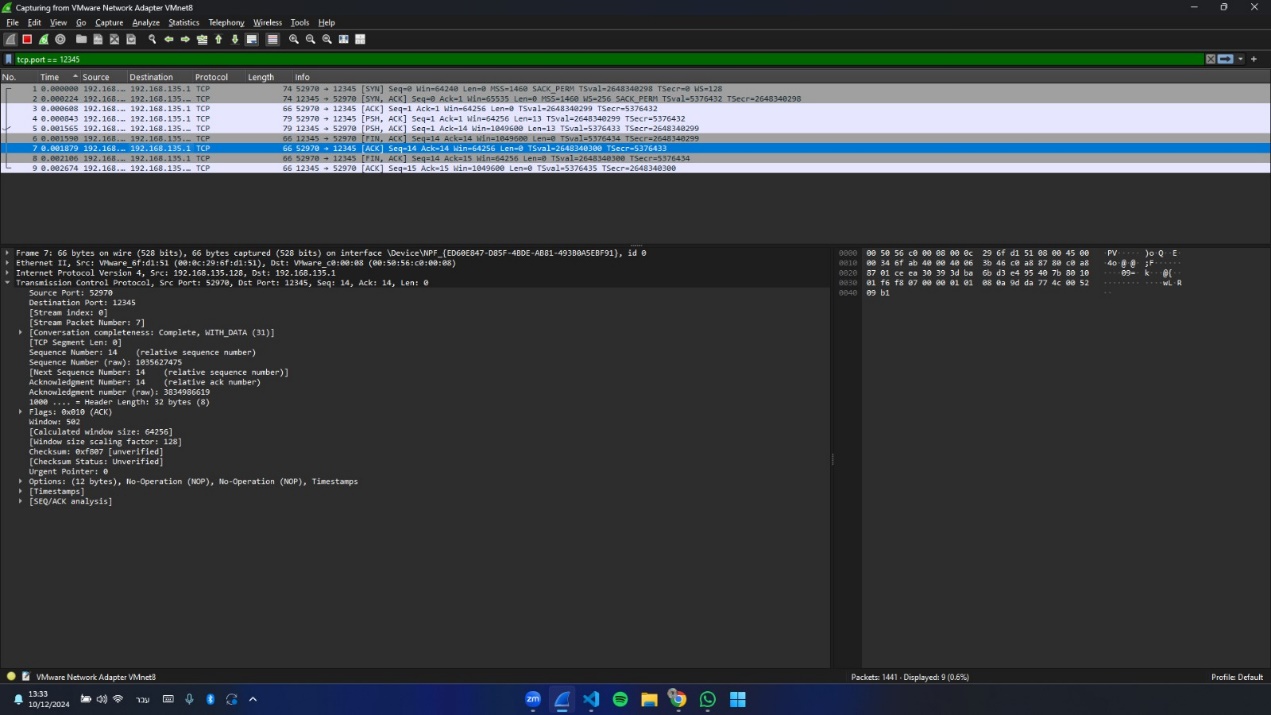
חבילה מס' 5 –

כעת נראה את תגובת השרת להודעה שקיבל מהלקוח בחבילה הקודמת שסקרנו. השרת ראשית **מעדכן** את המספר בשדה הack להיות 13 יותר משהיה – '1035627475' שכן עכשיו הוא אומר ללקוח – "שומע?! קיבלתי כבר **עד** בית מס' 1035627475 תן לי ממנו **והלאה**.". אמנם הseq לא משתנה כיוון שהוא טרם שלח דאטא משל עצמו ללקוח, אך כעת הוא מצרף להודעת הack את התוכן שקיבל בupper (ושוב אנו לא מופתעים שהlen הוא 13).

חבילה מס' 6 –



מיד לאחר שהשרת שולח את הודעת הack יחד עם התוכן בupper, הוא שולח הודעת fin (ניתן לראות דגל דולק), שכן הוא עשה את שלו, ותפקיד ההודעה הזו להודיע על כך שהוא מעוניין לסיים את מערכת היחסים עם הלקוח. נשים לב שעכשיו הseq של השרת גדל ב13 גם הוא לערך – '3834986619'.

חבילה מס' 7 –

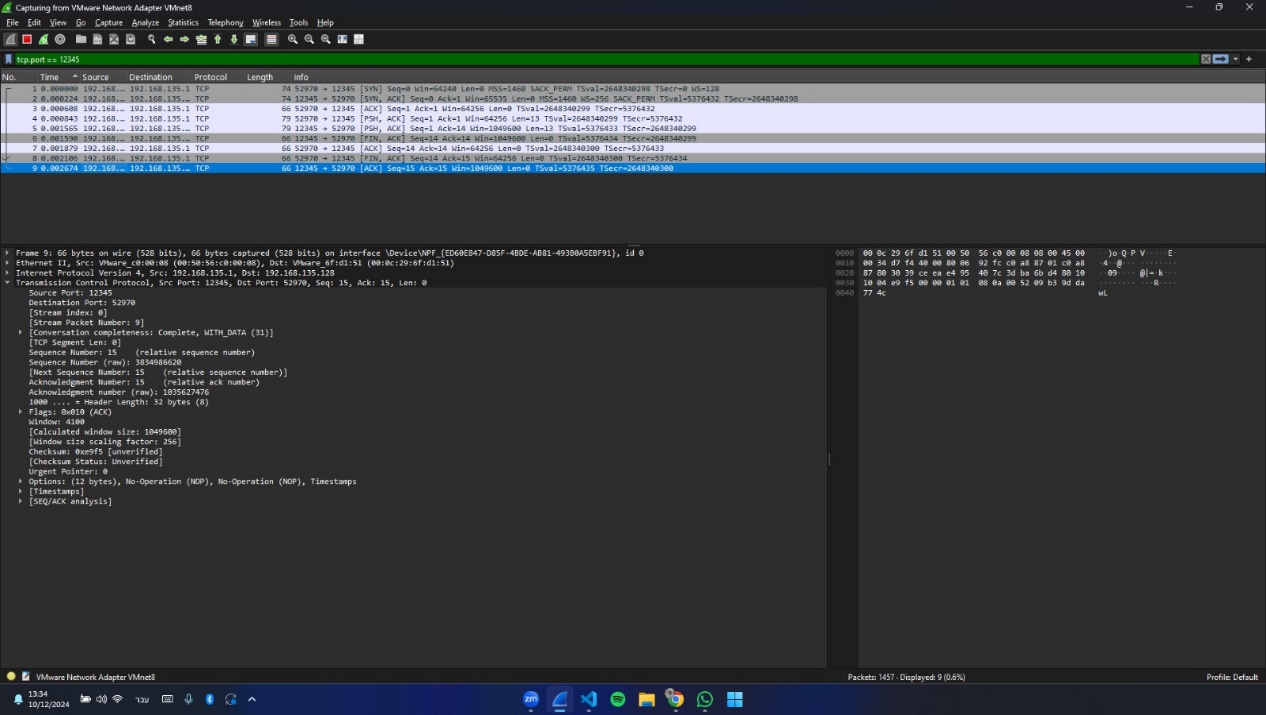
הלקוח מקבל את הדאטא חזרה מהשרת בupper, בהתאמה הוא מגדיל את ערך הacked ב13 בתים נוספים לערך – '3834986619'. כמו כן נשים לב שגם ערך הseq המצוין בהודעה גדל ב-13 מהפעם האחרונה בה ראינו חבילה נשלחת מהלקוח, וכעת הוא עומד על – '1035627475' (שוקינג זה הack של השרת..).

חבילה מס' 8 –

תמונה שמכילה טקסט, צילום מסך, תוכנה, תצוגה

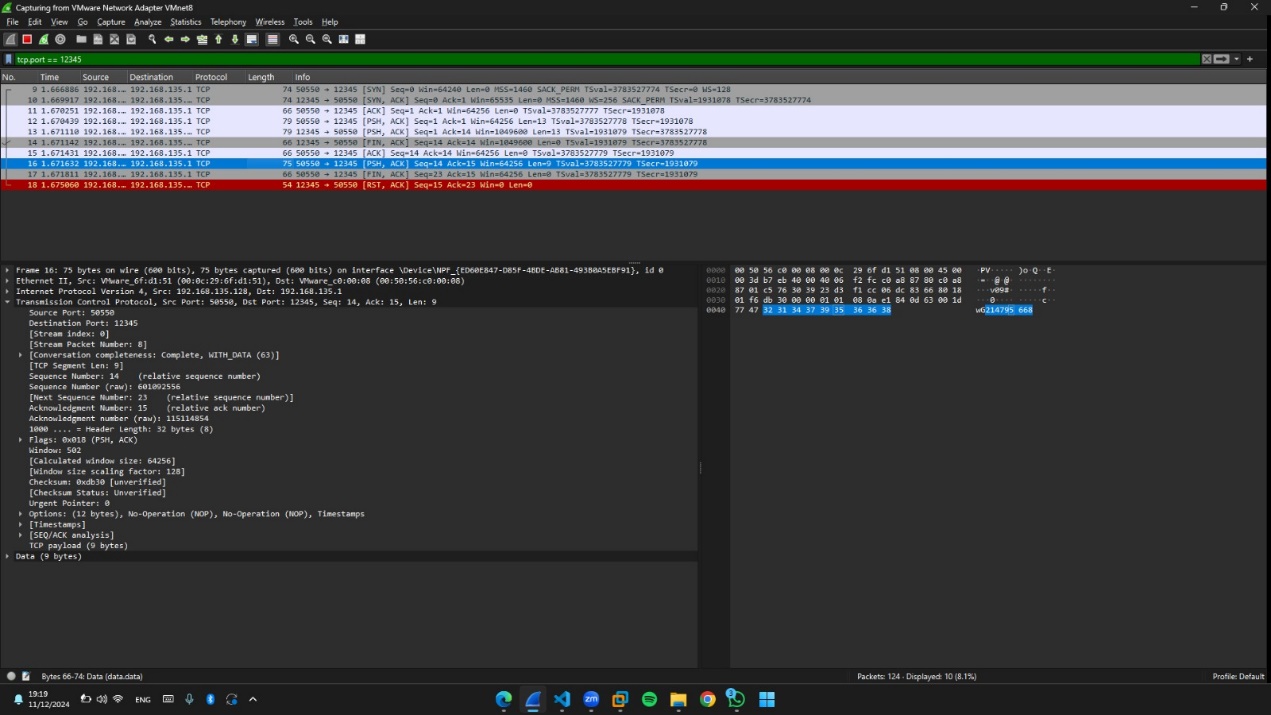
התיאור נוצר באופן אוטומטי

הלקוח כעת שולח הודעת ack לשרת, ולמעשה מעדכן שהוא קיבל את ההודעה שלו בנוגע לסיום ההתקשרות עימו. כיוון שהוא הבין שנגמר הקשר, הוא מסיים אותו גם מבחינתו ומצרף זאת להודעה (דגל הfin דלוק בה). דהיינו תפקיד ההודעה לומר לשרת שהסיום הינו הדדי. מעבר לזה אין מידע מעניין למעט העובדה שערך הack של הלקוח גדל ב-1 כתוצאה מהפאנטום ביט של הודעה הfin שמוכרת רק עכשיו.

חבילה מס' 9 –

בחבילה האחרונה (☹) שלנו אנו רואים את השרת מכיר בסיום ההתקשרות מצד הלקוח גם הוא בהודעת ack, ובאופן פרקטי תפקיד ההודעה הינו לסיים **לחלוטין בהחלט** את ערוץ התקשורת שהיה ביניהם. נשים לב שרגע לפני שהוא אומר "ביייי", הוא מגדיל גם הוא את ערך הack כתוצאה מהפאנטום ביט של הfin שהגיע מהלקוח, סה"כ ערך הack של השרת בסיום התקשורת עומד על – '1035627476'.

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

כל ההודעות בהתחלה תתנהגנה באופן זהה (לכן נחסוך ונתחיל רק מהמקום בו השינוי קורה), אבל עכשיו נשים לב –

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטילאחר שהלקוח החזיר ack על הfin של השרת (פאנטום ביט מגדיל ב1 את הacked), במקום לשלוח גם הוא fin (כמו שקרה קודם), הוא מנסה לשלוח הודעה נוספת ולעשות לה psh לאפליקציה (הלוא היא הת.ז.), ורק לאחר שההודעה הזו נשלחת, הוא שולח גם כן fin מצידו –

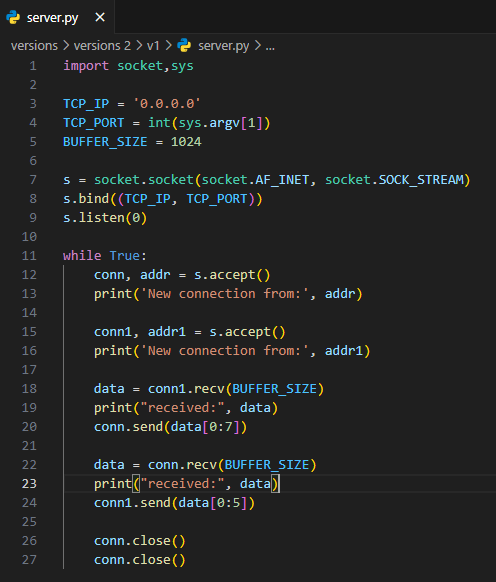
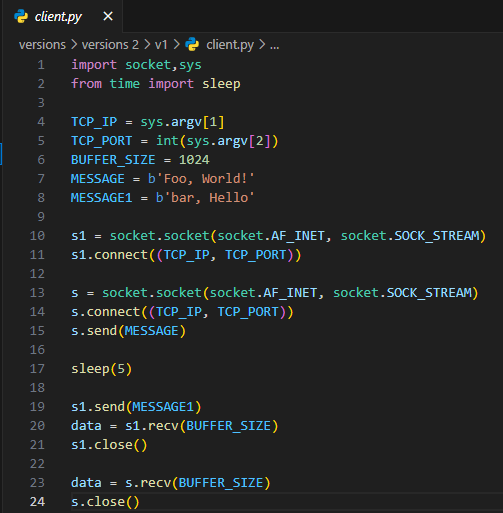
תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

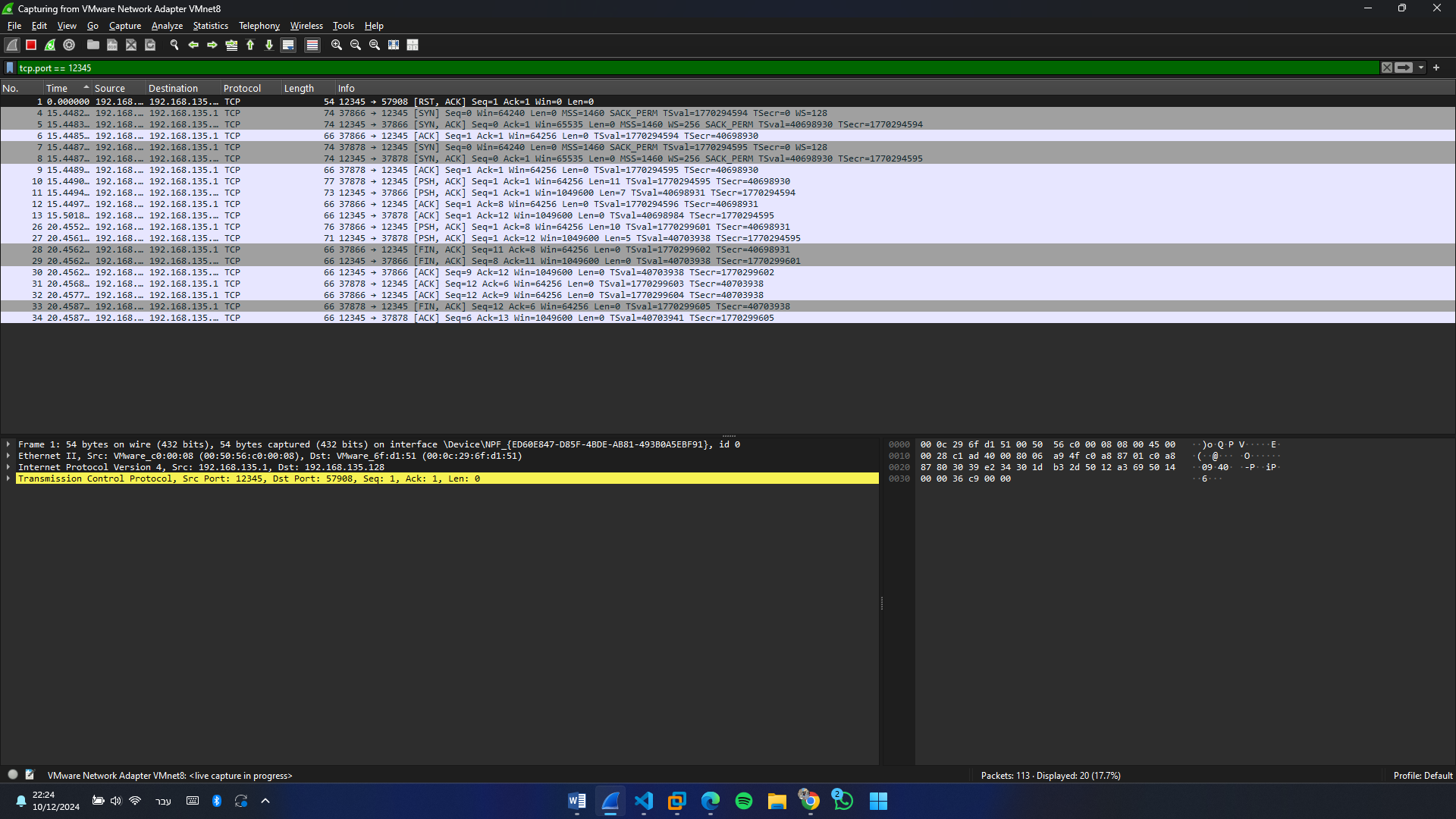
התיאור נוצר באופן אוטומטיהseq שלו גדל ב-9, והוא מסכים עכשיו לסיים את התקשורת. אלא שבגלל שהשרת לא ידע שעליו לצפות לחבילה נוספת הוא כבר סיים וסגר את העניין, ולכן נקבל ממנו הודעת rst –

כאילו רוצה לומר – "חדש אח שלי, חדש. מה שהיה אני כבר לא שם, ואין מי שישמע ויקבל את מה שיש לך לשלוח. אם אתה רוצה בוא נתחיל מההתחלה, ומשם נשתמע הלאה..".

ובכך סיימנו לנתח גם את המקרה בו הודעה נשלחת לאחר שהשרת עושה קלוז קודם לרגע בו היא מגיעה.

סעיף 2:

כעת, ננתח בקצרה את הקודים של הגרסאות השונות ואת התעבורה בהן:  
  
גרסה v1:



הלקוח מקבל את הIP והפורט מהארגומנטים לתוכנית, לאחר מכן הוא מתחבר בשני חיבורים נפרדים בעזרת שני סוקטים נפרדים לשרת (שורות 4 עד 9). הלקוח תחילה שולח הודעה דרך החיבור השני, מחכה 5 שניות ושלח הודעה אחרת דרך החיבור הראשון. לבסוף, הלקוח מקבל את שתי ההודעות מהסוקט בסדר הפוך לסדר בו התבצעה השליחה, ולאחר מכן סוגר את החיבורים. צד שני, השרת מאזין לכל החיבורים כיוון שהוא בחר בכתובת 0.0.0.0 ומקבל פורט כקלט. הוא מרים את השרת לאוויר כאשר הוא מוכן שיהיו ברגע נתון לכל היותר חיבור 1 בהמתנה (ליסטן שווה 0). הוא עושה accept לשני חיבורים (מדפיס מי הם), מקצר את ההודעה השנייה ל-7 תווים הראשונים שנקלטו ושולח אותה לחיבור הראשון, ואת ההודעה הראשונה ל-5 תווים הראשונים שנקלטו ושולח אותה לחיבור השני. לבסוף, יוזם ניתוק רק עם החיבור הראשון שהוא פתח.

נשים לב שבפועל בכריש קרה משהו מעניין. לאחר האתחול עד שורה 9, נשלחת ההודעה מהסוקט שנוצר שני בלקוח, והשרת לפני שהוא מחזיר עליה ack לאותו סוקט שני, קודם שולח את ההודעה המקוצרת (7 בתים), לחיבור הראשון (שורה 11). יתרה מזאת, הוא מקבל מהחיבור הראשון ack על ההודעה המקוצרת (שורה 12), ורק אז מחזיר ack על עצם זה שהוא קיבל את ההודעה המקורית מהחיבור השני. לאחר מכן השרת מקבל את ההודעה מהסוקט שנוצר ראשון בקליינט, ובאופן דומה לפני שהוא מחזיר עליה ack, הוא מעביר את 5 הבתים הראשונים מההודעה לחיבור סוקט השני. בשלב הזה מגיעה הודעת fin מהחיבור הראשון, אשר נענית בfin מצד השרת, ועל הדרך הוא גם נותן לו ack (רק בשלב הזה( על החבילה שנשלחה על ידו, ומיד הודעה נוספת של ack על הfin עם הפאנטום ביט. החיבור השני מחזיר ack רק עכשיו על ההודעה שהוא קיבל (5 הבתים), החיבור הראשון מחזיר ack על הודעות הfin שקיבל מהשרת (שוב פאנטום ביט), ולבסוף השרת מקבל fin מהחיבור השני ומשיב על כך בack (הוא לא שולח לו fin שכן בקוד הfin ששולח השרת הינו עבור החיבור הראשון, שכבר נסגר מיוזמתו קודם לכן..).

תמונה שמכילה טקסט, צילום מסך, תוכנה, גופן

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, צילום מסך, תוכנה, מערכת הפעלה

התיאור נוצר באופן אוטומטיגרסה v2:

תמונה שמכילה טקסט, חשמל, צילום מסך, תוכנה

התיאור נוצר באופן אוטומטי

כמו בתוכנית הקודמת, הלקוח מקבל את הIP והפורט מהארגומנטים לתוכנית, לאחר מכן הוא מתחבר לשרת ושולח לו את ההודעה המצוינת. הוא עושה recv מהשרת, ולאחר שקיבל את ההודעה שהגיעה חזרה מהשרת הוא מדפיס אותה. מצד שני, כמו בתוכנית הקודמת, השרת מאזין לכל החיבורים כיוון שהוא בחר בכתובת 0.0.0.0 ומקבל פורט כקלט. הוא פותח חיבור, מקבל את ההודעה בצאנקים של בתים 5 (כגודל הבאפר), עד שבתים מפסיקים להגיע, מדפיס כל צאנק, ושולח אותו חזרה ללקוח באופן של upper. לבסוף, הוא סוגר את החיבור.

כמו התפיסה האחרונה, ניתן לראות את לחיצת הידיים בין הלקוח לשרת (שורות 4 5 6).  
אך בשונה מהתפיסה האחרונה, השרת קולט את המידע שמגיע אליו מחיבור אחד בחתיכות של 5 תווים. לכן רואים בכריש את הצאנק הראשון מגיע ומוחזר מהשרת (בupper כמובן), ולפני שאנו רואים את המשך קבלת הצאנקים בשרת, אנו רואים את הack שהלקוח החזיר על הצאנק הראשון. משום מה השרת מאחד את 2 הצאנקים הבאים ושולח אותם יחדיו ללקוח, כאשר האחרון מחזיר עליהם ack. עם זאת, מכיוון שיש רק recv אחד בלקוח, הוא ממשיך בשלו ולאחר קבלת הצאנק הראשון הוא שולח fin כדי לסיים את ההתקשרות. השרת מקבל את fin ומחזיר עליה ack, אולם מעתה והילך שאר הצאנקים שהשרת שולח זוכים למענה של rst כיוון שאין אף אחד בצד השני ששומע וזמין לקבל. כנל לfin שהשרת שולח – הכל מקבל rst, שכן הלקוח ניתק בצד השני...

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטיגרסה v3:

תמונה שמכילה טקסט, צילום מסך, תוכנה, מערכת הפעלה

התיאור נוצר באופן אוטומטי

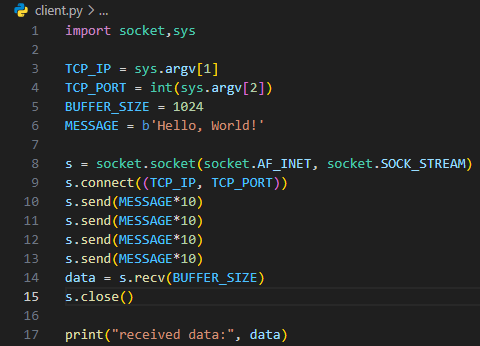
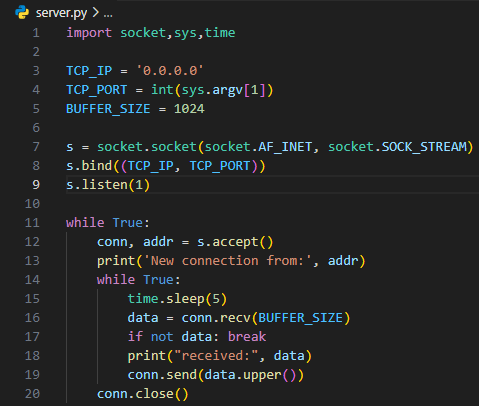
תמונה שמכילה טקסט, חשמל, צילום מסך, תצוגה

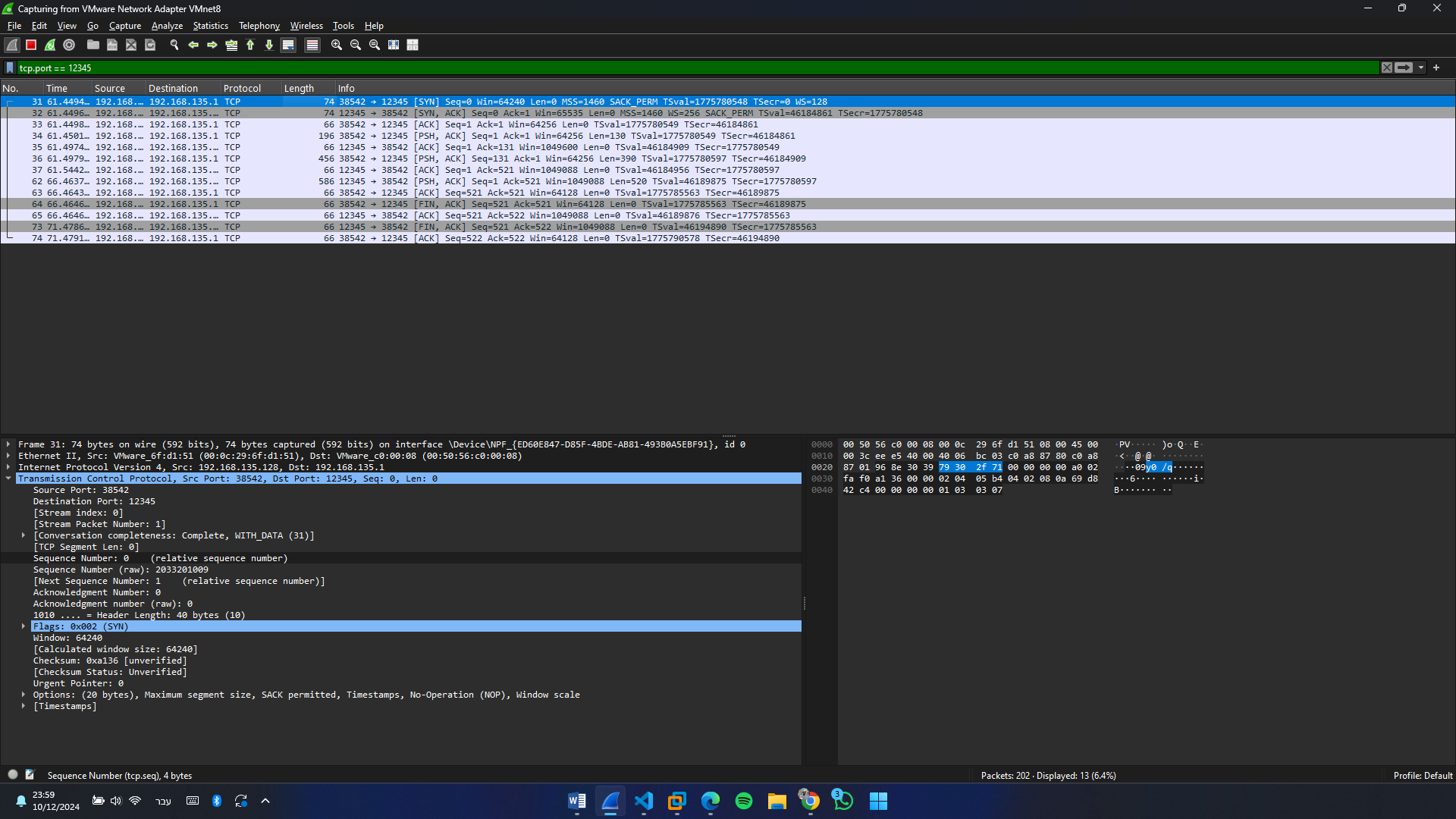
התיאור נוצר באופן אוטומטי

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 ההודעה חזרה מהשרת הוא מושך ומדפיס את ה1024 בתים הראשונים ולאחר מכן שוב הוא מנסה למשוך עוד 1024 בתים. כמו כן בצד שני, כמו בתוכנית הקודמת, השרת מאזין לכל החיבורים כיוון שהוא בחר בכתובת 0.0.0.0 ומקבל פורט כקלט. הוא פותח חיבור, מקבל את 1024 הבתים הראשונים ושולח אותם חזרה ללקוח באותיות גדולות בהכפלה של 1000 פעמים הקלט. לבסוף, סוגר את החיבור.

כמו בתפיסה האחרונה, ניתן לראות את לחיצת הידיים בין הלקוח לשרת. מיד לאחר מכן את 13 הבתים שהלקוח שולח, ואז השרת קולט את כל המידע שמגיע אליו, ושולח אותו באותיות גדולות כפול 1000 פעמים, לכן 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 הבתים הראשונים) וכן את השנייה (אלו שבאים אחרי), מחזיר ack באופן מאוחד על 13000 הבתים שהגיעו, וסוגר את החיבור. אבל יש עוד חבילות בבאפר שלא נקראו ע"י האפליקציה כאשר היא נסגרת, לכן נשלחת הודעת rst לשרת שידע שהלקוח ניתק מבלי לקרוא עד הסוף.

גרסה v4:



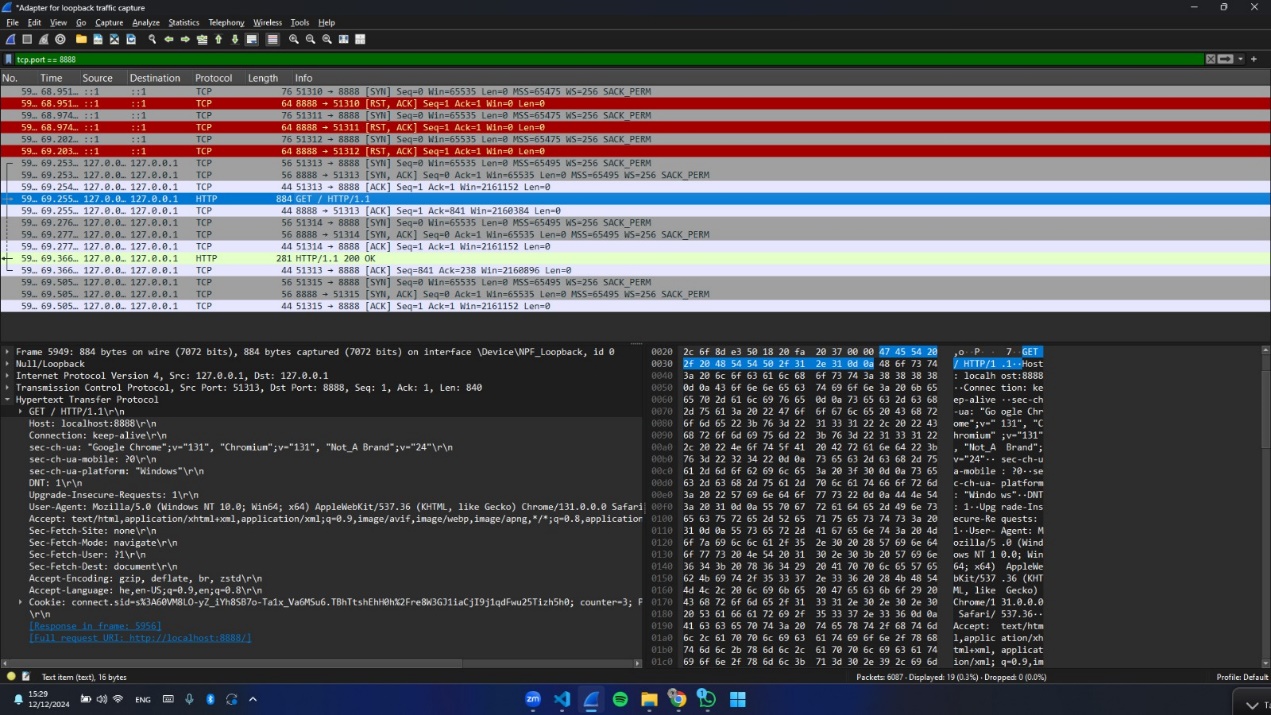


כרגיל הלקוח מקבל את הIP והפורט מהארגומנטים לתוכנית, לאחר מכן הוא מתחבר לשרת ושולח לו 4 הודעות שכל הודעה היא 10 פעמים "Hello, World", השוני הפעם מהתוכנית הקודמת, הוא שלאחר שקיבל את ההודעה חזרה מהשרת הוא מדפיס את ה1024 בתים הראשונים. בצד שני, כרגיל השרת מאזין לכל החיבורים כיוון שהוא בחר בכתובת 0.0.0.0 ומקבל פורט כקלט. הוא פותח חיבור מחכה 5 שניות, מקבל את 1024 הבתים הראשונים ושולח אותם חזרה ללקוח באותיות גדולות. הוא חוזר על התהליך עד שהוא מפסיק לקבל מידע מהלקוח ולבסוף סוגר את החיבור.

בכריש אנו רואים תחילה את לחיצת הידיים בין הלקוח לשרת. אך בשונה מהתפיסה האחרונה, השרת מחכה 5 שניות ולכן התגובה שלו מתעכבת. ניתן לראות כי השליחה (של הלקוח) והתגובה (של השרת) קורית פה 2 פעמים (כנראה 3 השליחות האחרונות אוחדו בעת שליחתן). בכל פעם השרת (שנמצא בtimeout באפליקציה ולכן לא קורא מהבאפר), מחזיר ack שהוא קיבל את ההודעה. השרת שולח את כל ה520 בתים בחזרה בפעם אחת (מאוחדת) באותיות גדולות, ומקבל ack עליו מהשרת. מכאן והילך טקס הפרידה הסטנדרטי של fin-ack-fin-ack בין השרת ללקוח, כאשר כל אחד מעלה את הפאנטום ביט, ונפרדים לשלום. כאן לא הייתה בעיה של rst בשל העובדה שכל המידע נקרא מהבאפר בטרם האפליקציה סיימה את הריצה.

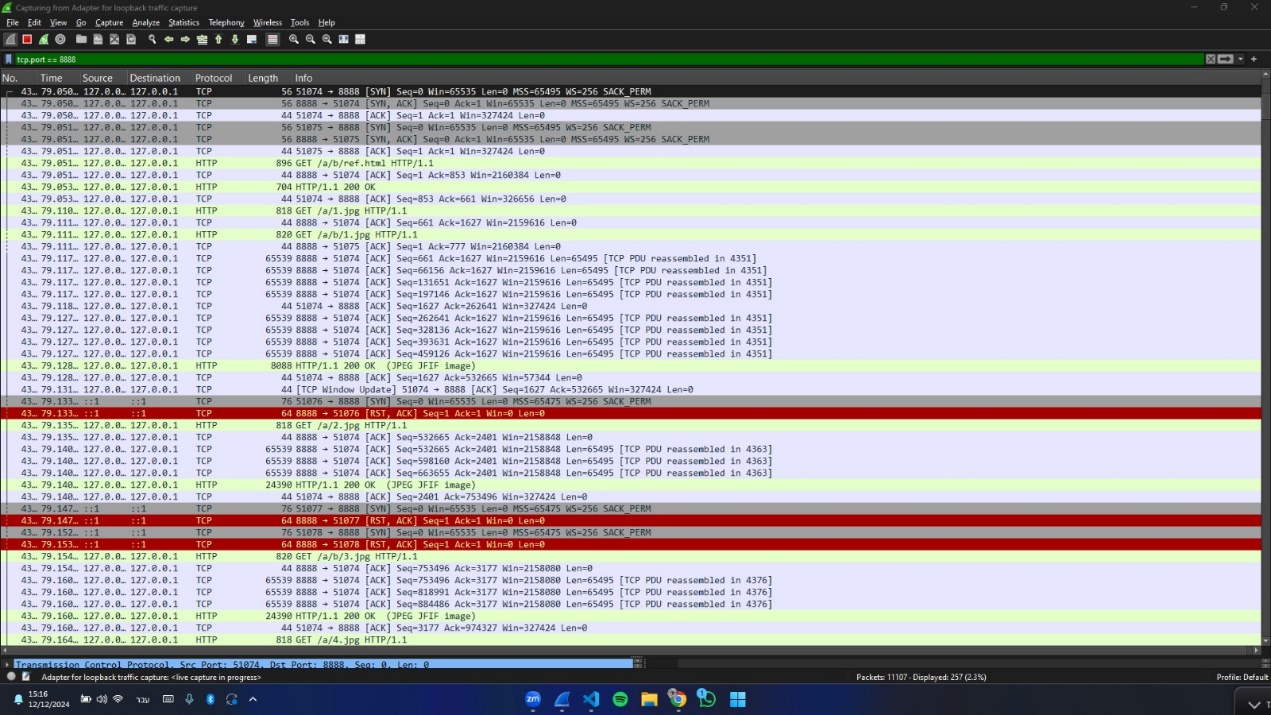
נסיים חלק זה של התרגיל בהצגת הכריש בזמן ריצת קוד השרת שלנו, כאשר האחרון מקבל בקשות מהדפדפן.

http://127.0.0.1:8888/



הכתובת פונה ומבקשת מהשרת את דף הבית, והוא מחזיר בתמורה את – index.html. ניתן לראות שדפדפן עושה מס' פניות מכמה (שלושה) פורטים שונים (לא ברור לנו ככ למה, אך טרם למדנו זאת לעומק בהרצאה :/), אולם רק מהראשון הוא שולח את הבקשה של הקובץ, ומקבל אליו חזרה את ה 200 ok. ניתן לראות שנשלח המון מידע עם הודעת הget מדפדפן, כמו סוג הדפדפן, הגרסה שלו, משהו עם cookie ועוד. מעבר לget וההחזרה של המידע מהשרת – 200 ok, אין משהו מיוחד, ואין בקשה נוספת מאותו פורט ראשי. לא ראינו שהתבצעה באמת תקשורת בפורטים האחרים שנפתחו, כנראה כיוון שהיו תקועים בליסטן (שכן אנחנו לא עובדים באופן מקבילי).

http://127.0.0.1:8888/a/b/ref.html



כעת מבקש הדפדפן קובץ שמכיל בתוכו מס' קבצים (תמונות) נוספים. ניתן לראות ששוב הוא פותח מס' חיבורים (כ-שלושה) דרך כמה פורטים, אבל בפועל רק דרך החיבור הראשון עוברת כל התקשורת של השרת. לאחר מכן הוא מבקש את הדף הראשון (זה שביקשנו בכתובת), ולאחר שהוא מקבל אותו בget אנו מתחילים לראות רצף של בקשות נוספות של כל הקבצים הדרושים כגון /a/2.jpg ושות' על מנת להציג את הדף (שאותו ביקשנו בהתחלה) בשלמותו. התמונות שהן די גדולות מועברות בכמה צאנקים ואחת לכמה זמן אנו רואים ack שחוזר עליהן, כאשר בסיום העברה של כל קובץ תמונה נראה שמגיע get "מאסף", כאשר שוב בקשות הget מלאות באותו מידע יבש על הדפדפן ושאר התשתיות.

http://127.0.0.1:8888/redirect

תמונה שמכילה טקסט, צילום מסך, תצוגה, תוכנה

התיאור נוצר באופן אוטומטי

http://127.0.0.1:8888/error

תמונה שמכילה טקסט, צילום מסך, תוכנה, תצוגה

התיאור נוצר באופן אוטומטי