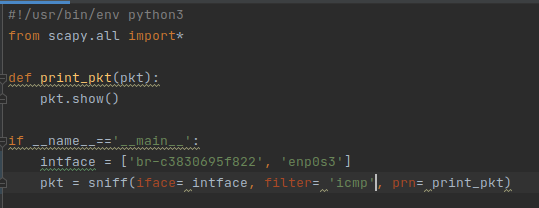
פרויקט גמר בתקשורת

**חלק 1:**

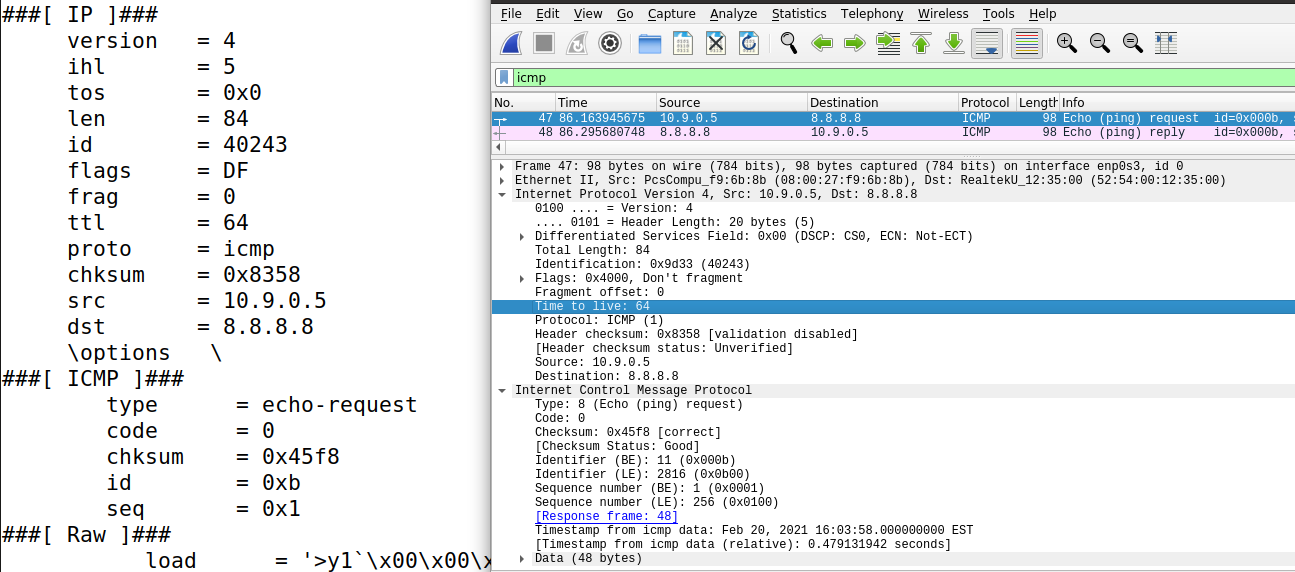
משימה 1.1:

**חלק A**:

כאן התבקשנו לבנות סניפר בפייתון, ולפלטר לפי icmp. להלן הקוד:

ניתן לראות את תוצאת ההרצה של הקובץ עם הרשאת גישה (sudo), וניתן לראות שהיא זהה לפאקטות שאנו רואים בwireshark:

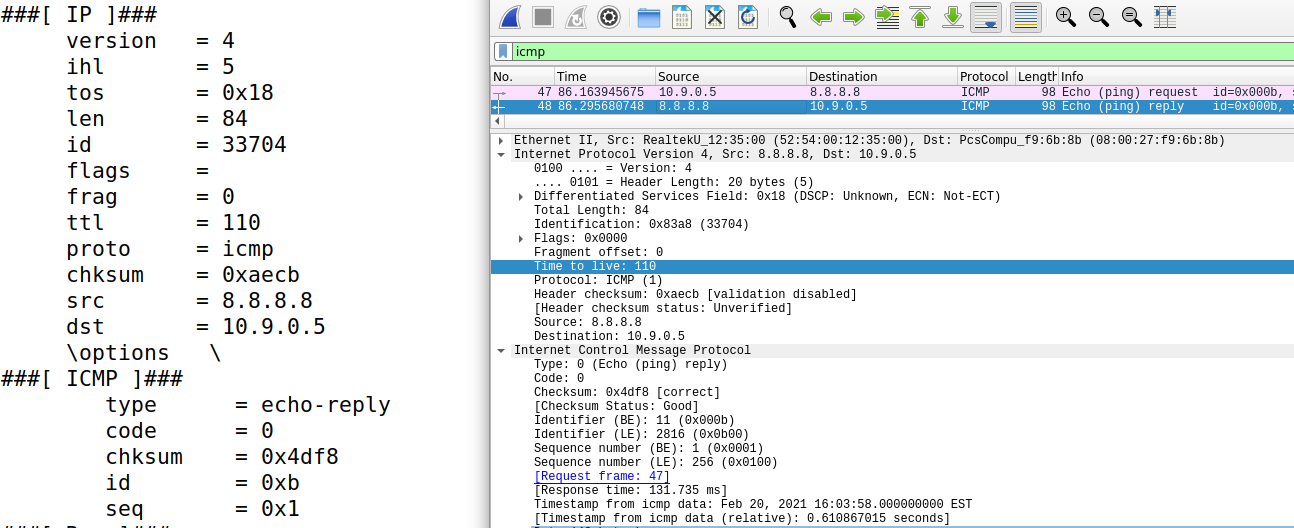
request:



terminal

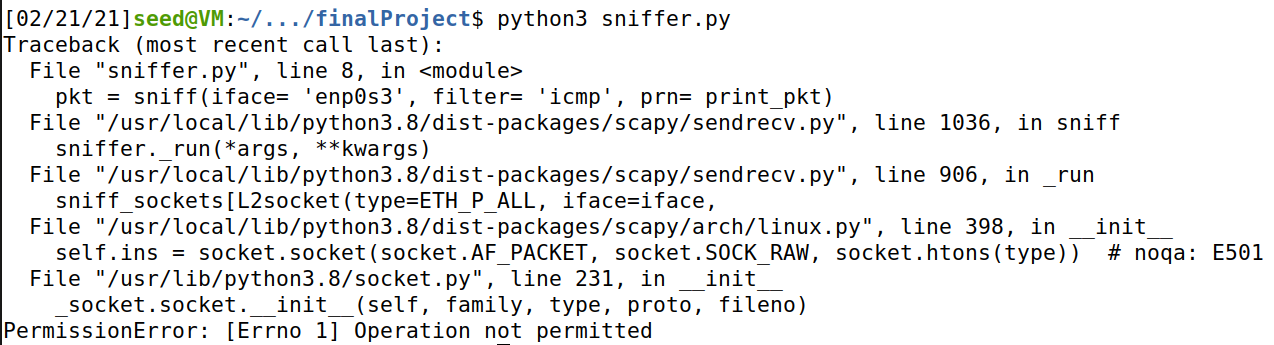
wireshark

reply:



terminal

wireshark

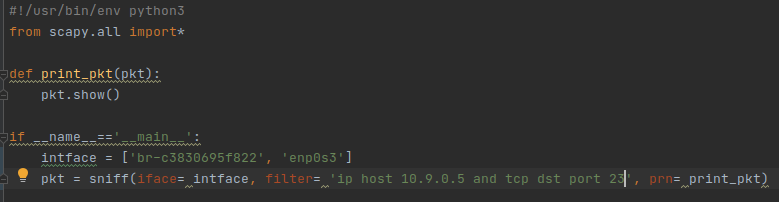
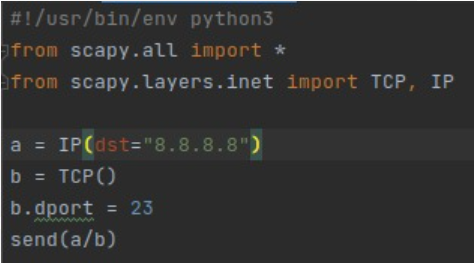
כאן ניתן לראות את תוצאת ההרצה של הקובץ בלי הרשאת גישה (sudo). הסיבה שמתקבלת שגיאה, כי האזנה לפקטות זה דבר שדורש הרשאת מנהל. בלי sudo אין הרשאת מנהל, ולכן לא ניתן להפעיל ,raw socket ולכן מתקבלת שגיאה.

**חלק B:**

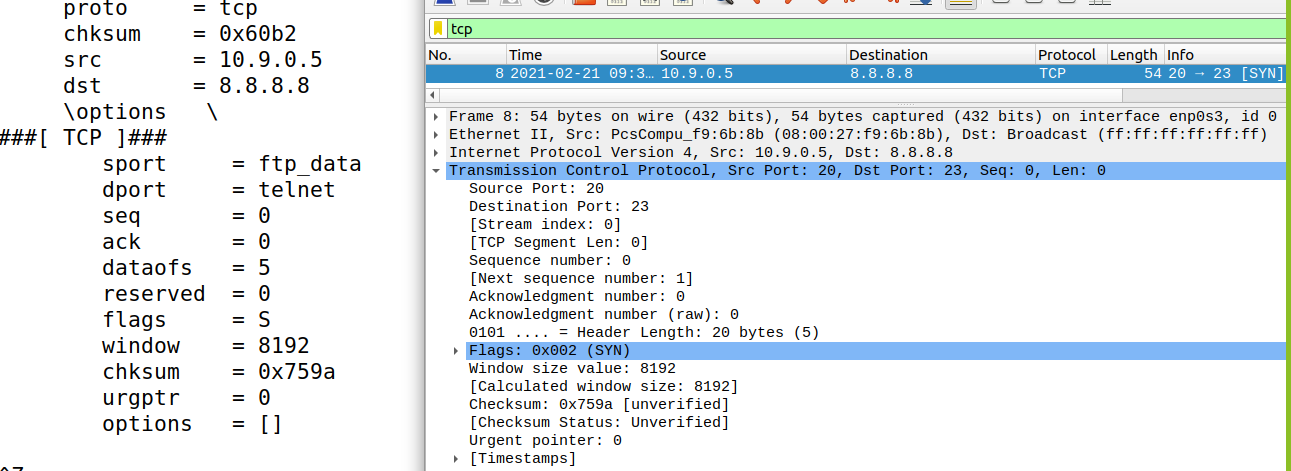
בחלק הזה נתבקשנו לפלטר את הסניפר שבנינו לפי סוגי פקטות:

-לפי icmp: ניתן לראות זאת בתמונה בחלק A, שם הפילטר כבר מסונן לפי icmp

-לפי TCP: כאן אנו מפלטרים פקטות TCP עם פורט יעד 23 עבור IP מסוים (8.8.8.8):



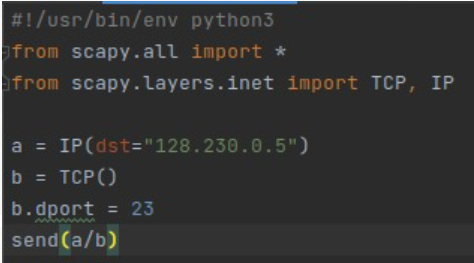
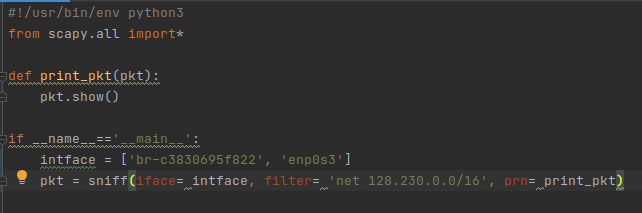
ניתן לראות שקיבלנו אותן פקטת TCP בטרמינל ובווירשארק:



terminal

wireshark

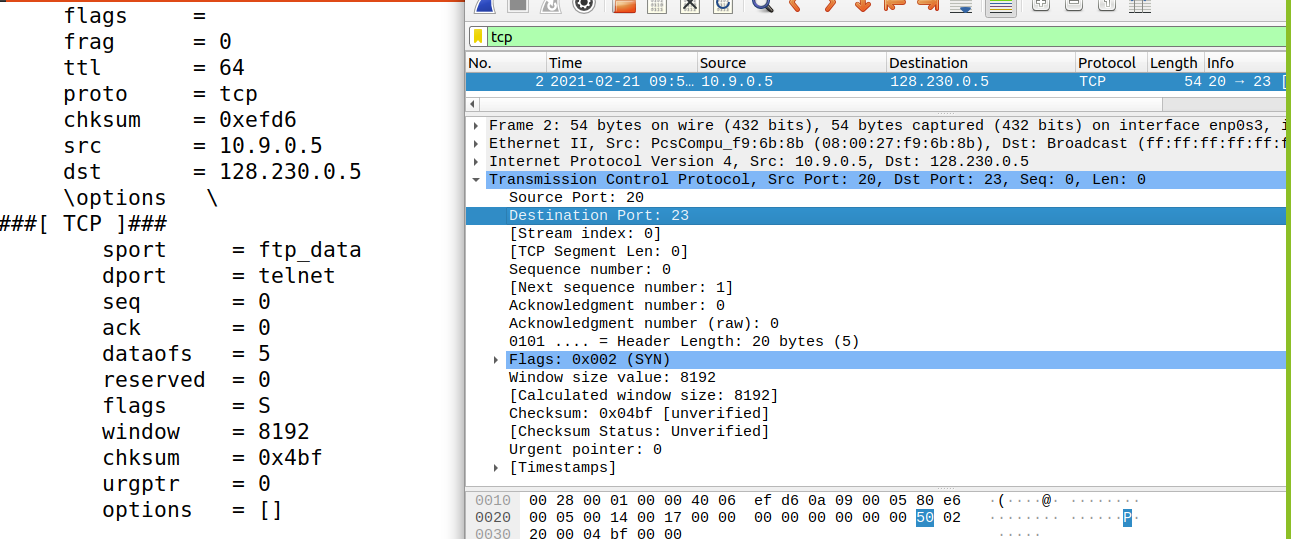
-לפי רשת פנימית: כאן נתבקשנו ללכוד פקטות שרצות ברשת פנימית מסוימת (128.230.0.0/16):



ניתן לראות שקיבלנו תוצאה זהה בטרמינל ובwireshark:

terminal

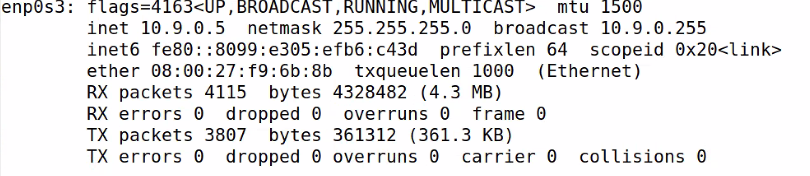
wireshark



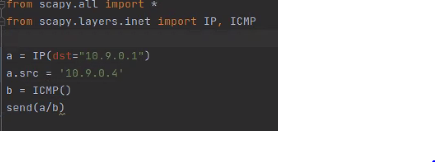
משימה 1.2:

בשאלה זו אנו מתבקשים לבצע spoofing ברשת פנימית באמצעות icmp request:

ראשית, נשים לב מה הip שלנו: 10.9.0.5:



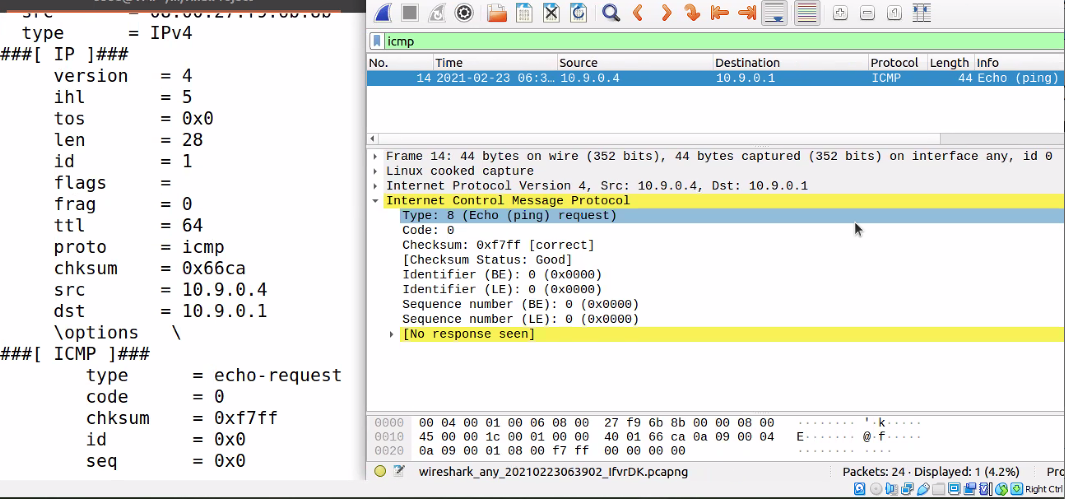
כעת, נבצע spoofing: נתחזה לip 10.9.0.4 ונשלח פאקטת icmp לip 10.9.0.1:



נריץ את הסניפר בטרמינל שמפלטר לפי icmp ונשלח את הפאקטה הנ"ל בטרמינל. ניתן לראות שבפרטי הrequest הsrc הוא 10.9.0.4 (כשבפועל, הip שלנו הוא 10.9.0.5):

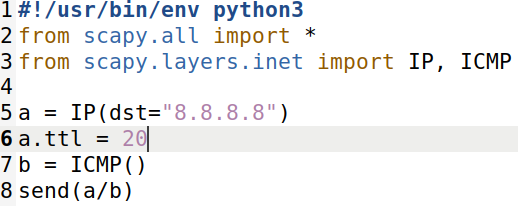
terminal

wireshark



משימה 1.3:

נשלח בקשת icmp לip 8.8.8.8 ונראה כמה נתבים הפקטה צריכה לעבור בדרך על מנת להגיע ליעדה. רק בttl = 20 הפקטה תגיע ליעדה ונקבל reply :



כאן יש תצלום של wireshark: הip שלנו הוא 10.9.0.4 , ולפי הפקטות השחורות ניתן לעקוב אחר המסלול של הקריאה עד 8.8.8.8. הפקטות השחורות מראות איזה נתב החזיר שהפקטה נפלה אצלו, כי לא הגיעה ליעד. במקים בהם פקטה המתינה זמן רב לתגובה מהנתב אליו היא צריכה לעבור, אנו רואים request ללא אף תגובה:



**חלק 2:**

משימה 2.1:

**חלק A**:

1. נסביר את הקריאות בסניפר שכתבנו:

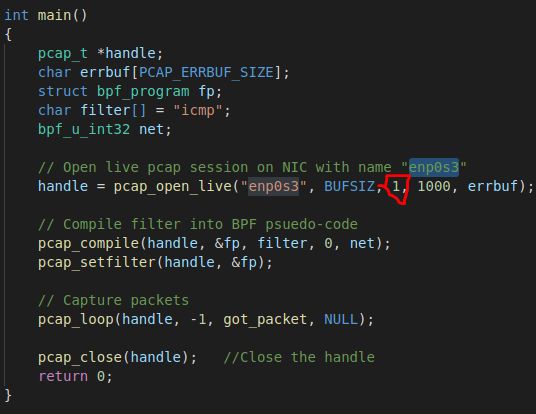
**pcap\_open\_live** – פתיחת raw socket והשמת כרטיס הרשת במצב האזנה.

**pcap\_compile&pcap\_setfilter** – פלטור פאקטות: כלומר, אנו מגדירים לסוקט אילו פקטות להרים (פקטות מסוג icmp, או מסוג tcp, או מסוג udp וכיו"ב), ע"פ סינטקס BPF.

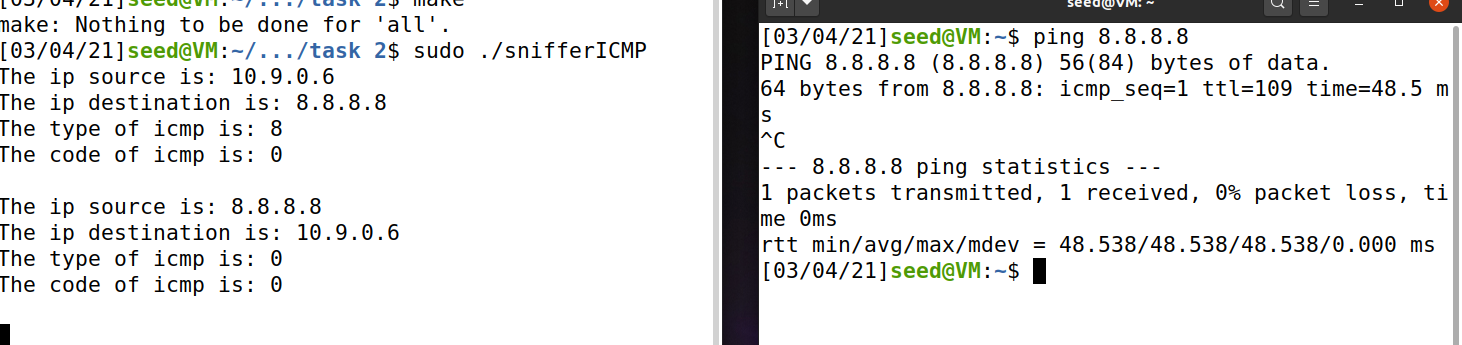
**pcap\_loop** – פקודה זו אחראית על הרמת הפאקטות ע"פ הפילטר שהגדרנו לעיל. היא רצה בלופ מספר פעמים על פי איך שמגדירים לה (1-/0 היא תריץ בלופ איסוף פעמים). כאשר מורמת פאקטה, הפקודה תפנה את הפאקטה לפונקציה שהגדרנו לה. (בקוד שלנו זו המתודה got\_packet שמדפיסה את פרטי הפאקטה).

**pcap\_close** – סגירת הraw socket ושאר המשאבים.

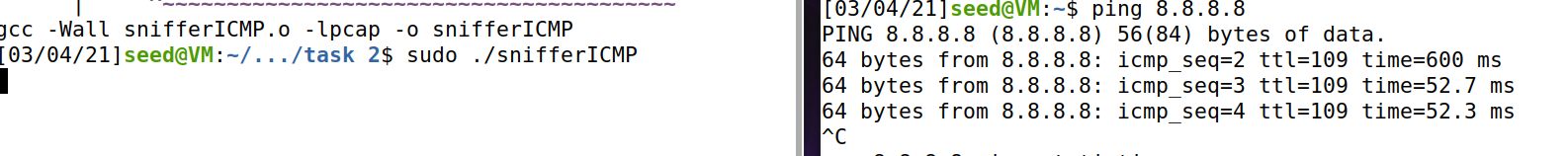
1. הסיבה שדרושה הרשאת מנהל היא שפתיחת raw socket דורשת הרשאות מנהל וכאשר אנו מריצים את הsiniffer אנו מבקשים לפתוח raw socket. אילו לא הינו מקשים הרשאת מנהל הריצה היתה נופלת בשורת הפקודה של pcap\_open\_live שם אנו פותחים את הsocket.
2. נפעיל את הסניפר עבור פקטות icpm, עבור התעבורה בכרטיס הרשת enp0s3. הip שלנו הוא 10.9.0.4. נשים את מצב האזנה על 1, כלומר: האזנה לכל פקטות הicmp

בכרטיס הרשת, גם אלו שלא קשורות ל 10.9.0.4:

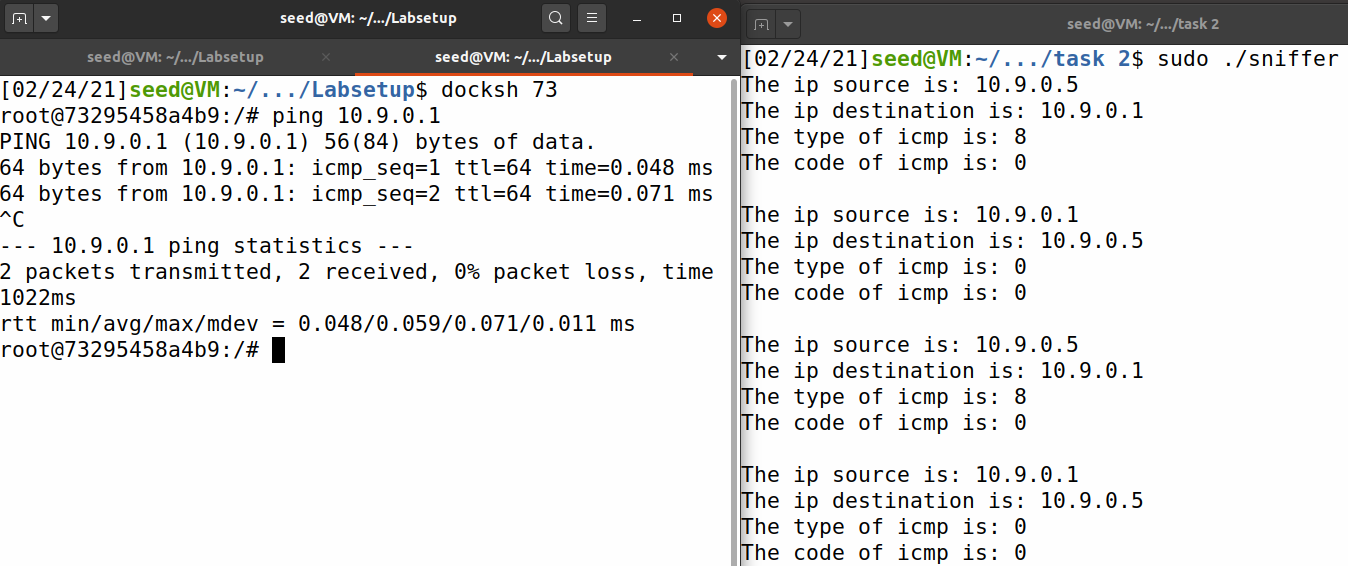
כעת, נפעיל את הסניפר שלנו. ממכונה אחרת שכתובתה 10.9.0.6 ונמצאת על אותו כרטיס רשת, נבצע ping 8.8.8.8: ניתן לראות בתצלום שהטרמינל משמאל (של 10.9.0.4) מדפיס את פרטי הפקטה, על אף שהודעת הping לא שורה אליו, כי הוא על pm = 1:

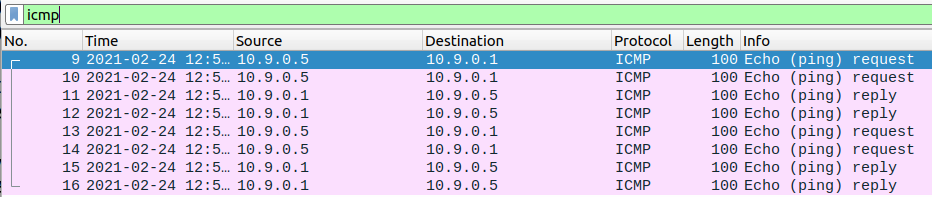


כעת, נכבה את pm (pm = 0), כלומר: האזנה רק לפקטות icmp בכרטיס הרשת דלעיל ורק כאלה שקשורות לip שלנו (10.9.0.4). כאשר נפעיל את הסניפר, ונבצע ping 8.8.8.8 מ10.9.0.6, הסניפר לא ידפיס דבר, כי הפקטה לא קשורה ל10.9.0.4:



**חלק B**:

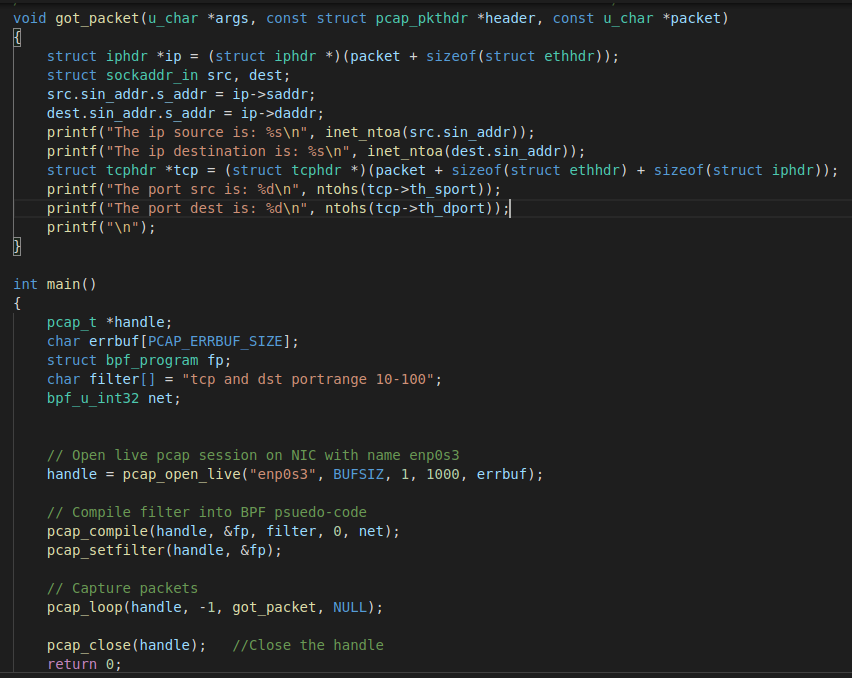
* נסניף תעבורת icmp בין 2 הקונטיינרים שלנו בdocker: host, שכתובתו 10.9.0.5 ישלח ping ל10.9.0.1 (הכתובת של seed-attacker). הקוד של הסניפר מצורף בתיקייה שנקראת task2):

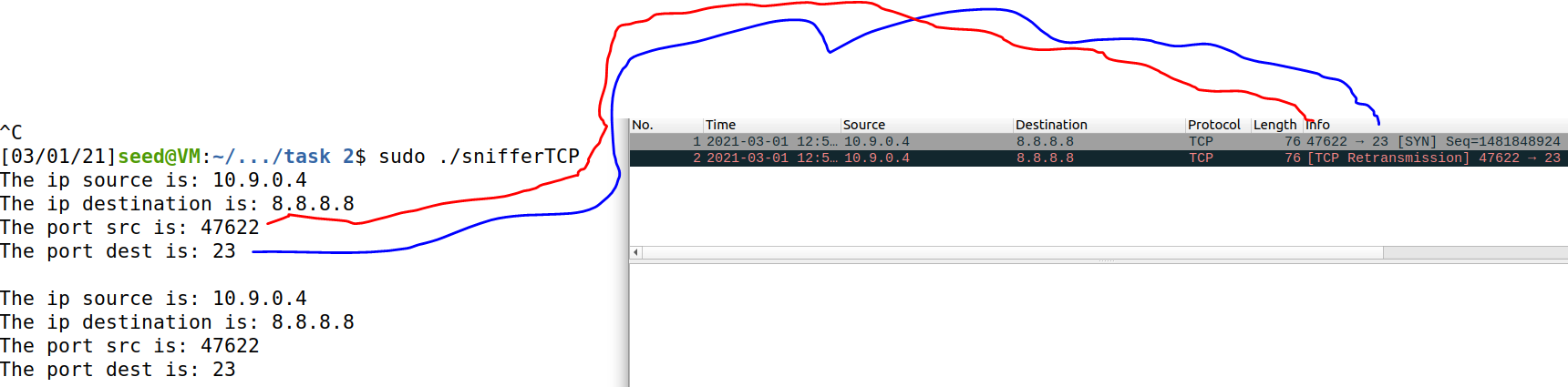
בחלון השמאלי שלחנו ping ל10.9.0.1, ובחלון הימני ניתן לראות שהסניפר מדפיס תעבורת icmp: על כל request (type=8), נקבל reply ((type=0. את הסוקט שכתבנו בקוד, הגדרנו עבור interface של אחד הקונטיינרים, ולכן רואים reply אחרי request יחיד, ולהפך.

ניתן לראות שהתוצאות שקיבלנו בsniffer מוצגות גם בwireshark.

הסיבה לכך שכל request וכל reply מופיע פעמיים היא ששמנו את הwireshark בהאזנה לכל הinterfaces ברשת ולכן אנו רואים פעמיים: פעם אחת כאשר המקור שולח ופעם שניה כאשר הפאקטה מגיע ליעד.

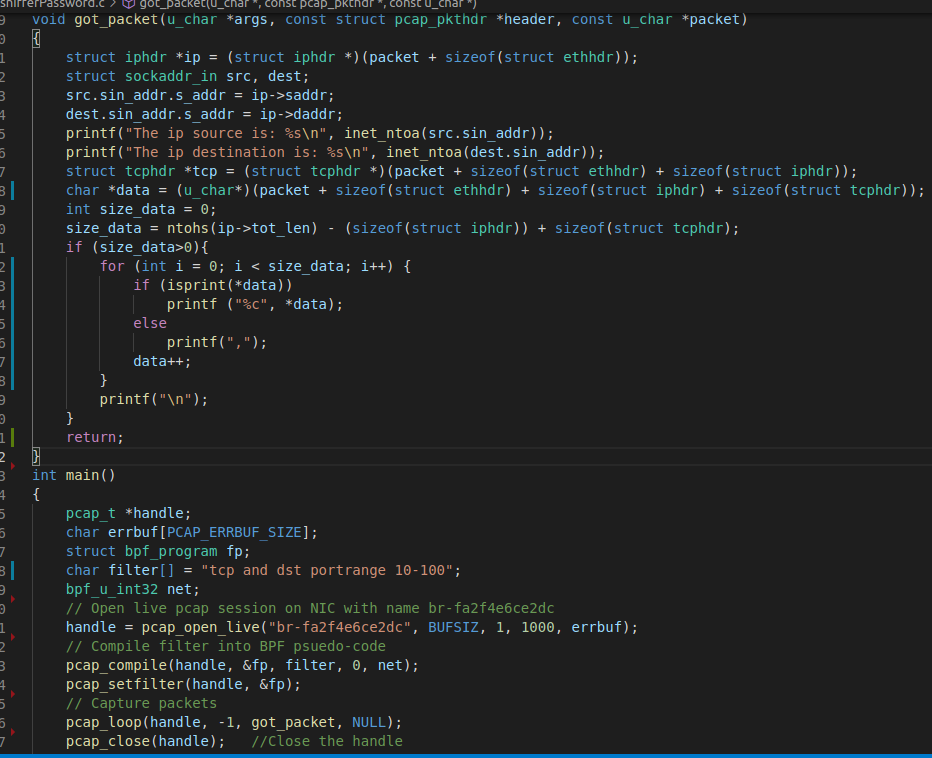
-נפלטר כעת פקטות tcp לפי פורט יעד בין 10-100:

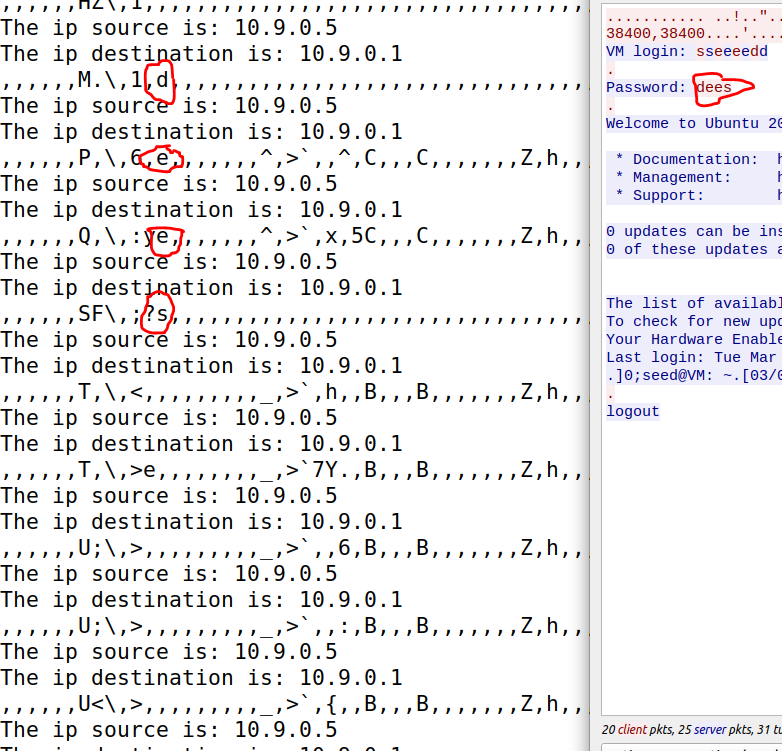
להלן הקוד:

בטרמינל אחד הרצנו את הפקודה telnet 8.8.8.8 שתשלח פקטת tcp לכתובת זו, ובאחר הרצנו את הסניפר. פורט היעד הוא 23 (ולכן הסניפר שלנו ידפיס את הפרטים, כי 23 בין 10 ל100). ניתן לראות, שהip של המקור והיעד, ופורט המקור והיעד – תואמים בטרמינל ובווירשארק:

חלק C:

נפעיל את הדוקר. אחנו נשלח פקטת tcp מhost שכתובתו 10.9.0.5 ל10.9.0.1, ע"י telnet. להלן הקוד של הסניפר עבור הסיסמה:

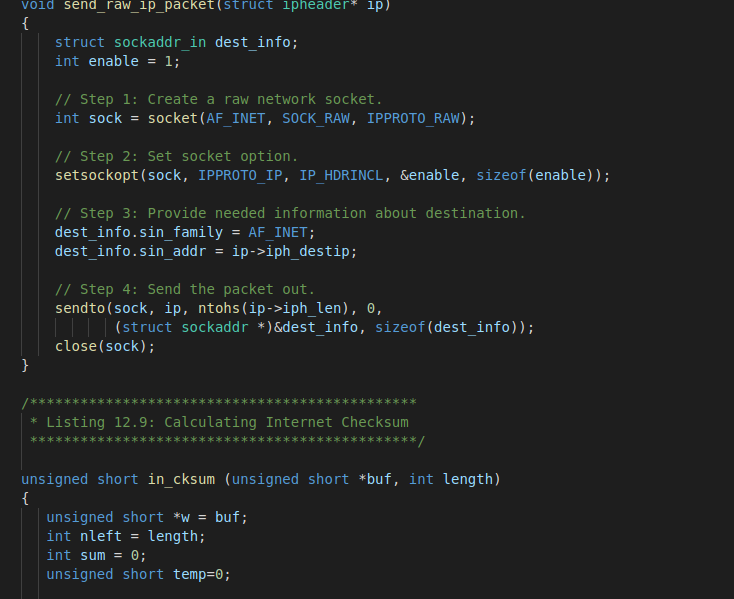


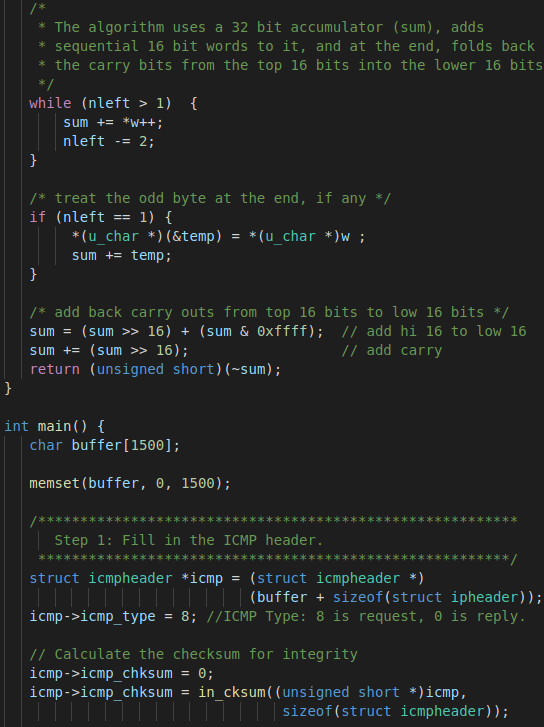
לאחר שהפעלנו את הדוקר, נפעיל את הסניפר עבור הסיסמה. בפלט משמאל ניתן לזהות את הסיסמה dees, ואכן זוהי הסיסמה של הVM:

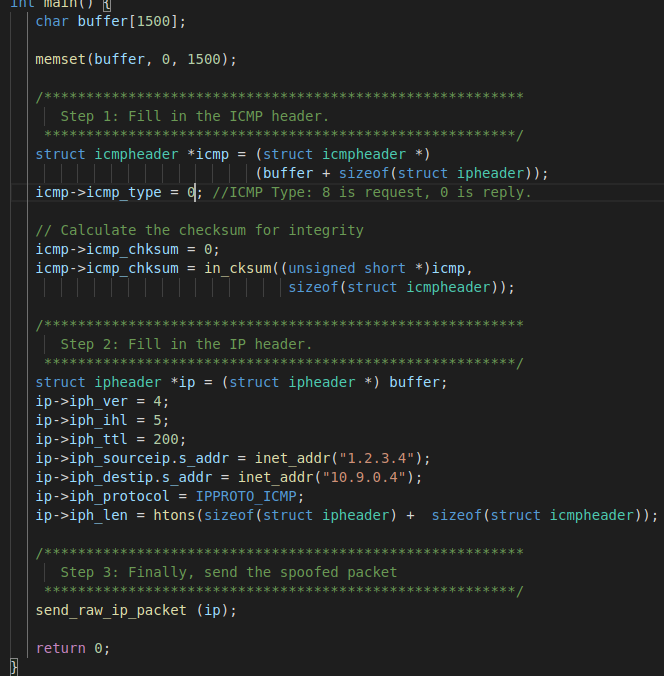
משימה 2.2:

אנחנו צריכים למעשה לזייף בקשת icmp ולזייף תגובה.

הip של הVM שלנו הוא 10.9.0.4. הip של הקורבן: 10.9.0.7. נתחזה ל1.2.3.4 ונזייף תגובה לקורבן:

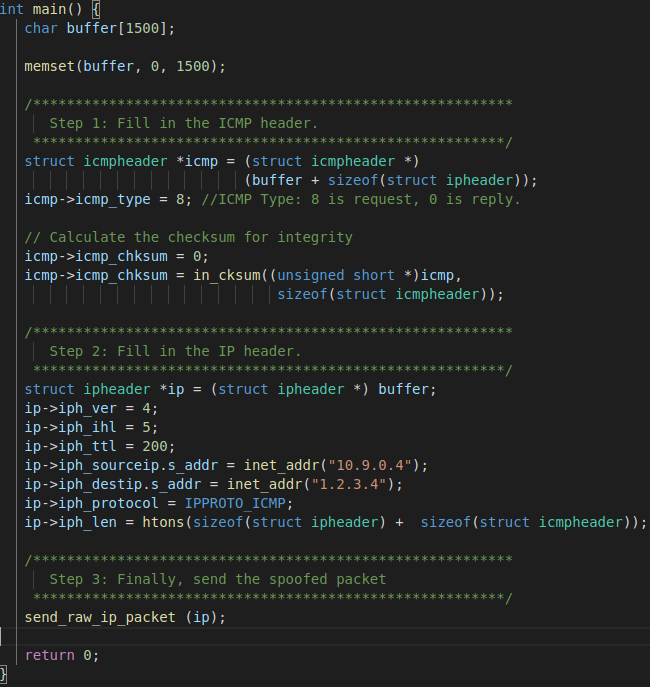
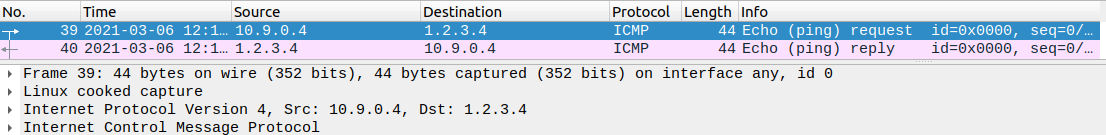
להלן הקוד עבור זיוף תגובה: type = 0:



 והנה בווירשארק, ניתן לראות רק תגובה מ1.2.3.4 ל10.9.0.4:



כעת, בשביל ליצור שאלה תמימה מ10.9.0.4 נחליף בין src לdest ונגדיר type = 8, ונקבל בווירשארק גם שאלה וגם תגובה:



שאלה 4-

שאלה 5-

אם נשנה את חישוב הchecksum, ייתכן ולא נקבל replyת

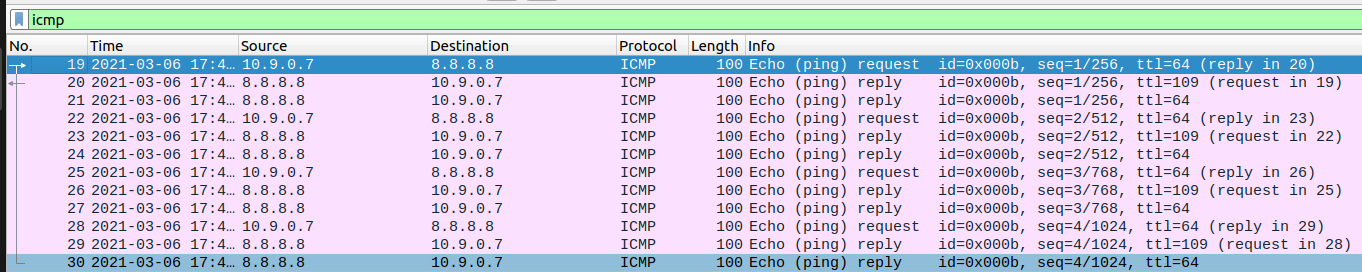
שאלה 6-

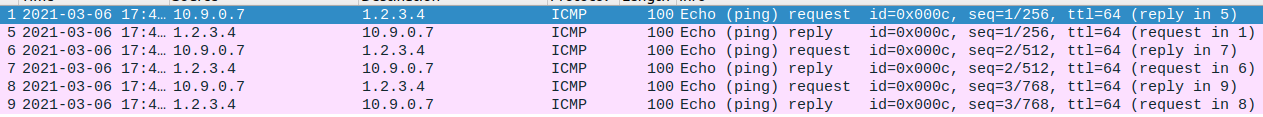
הסיבה שאנו צריכים הרשאה היא כי אנו משנים את הdata בשכבת התעבורה (כלומר, מגדירים src לפקטה שנחזיר. הפקטה נופלת למעשה ביצירת הסוקט

משימה 2.3:

בחלק זה אנחנו מתבקשים ליצור תוכנית שתרחרח בקשות icmp. לכל בקשה שכזאת עבור ip מסוים, הVM שלנו, שכתובתו 10.9.0.4 יחזיר reply בין אם הכתובת קיימת ובין אם לאו.

הקורבן שלנו הוא מכונה שכתובתה 10.9.0.7:

להלן תצלום ווירשארק עבור בקשה ל8.8.8.8: כל תגובה מופיעה פעמיים: פעם כי 8.8.8.8 מחזיר, ופעם כי 10.9.0.4 מתחזה אליו:

להלן תצלום ווירשארק של ping 1.2.3.4: הפעם כל תגובה תופיע פעם אחת (10.9.0.4 מתחזה) כי 1.2.3.4 לא קיים ברשת