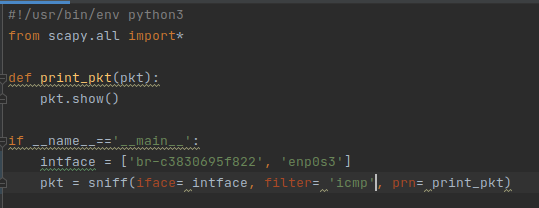
פרויקט גמר בתקשורת

**חלק 1:**

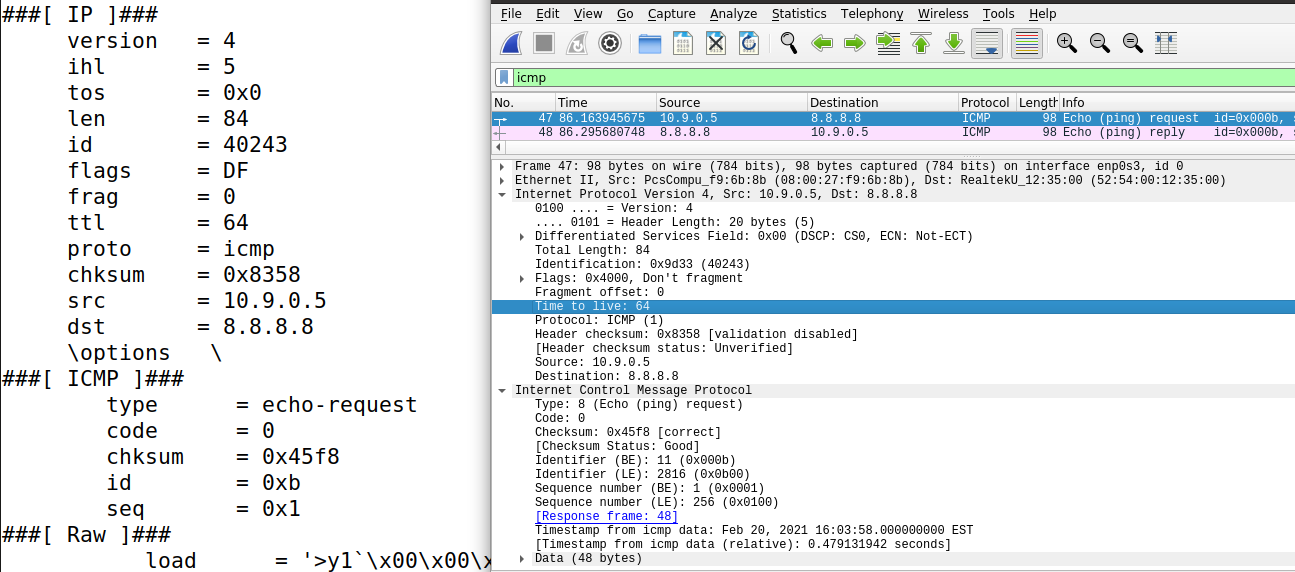
משימה 1.1:

**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:

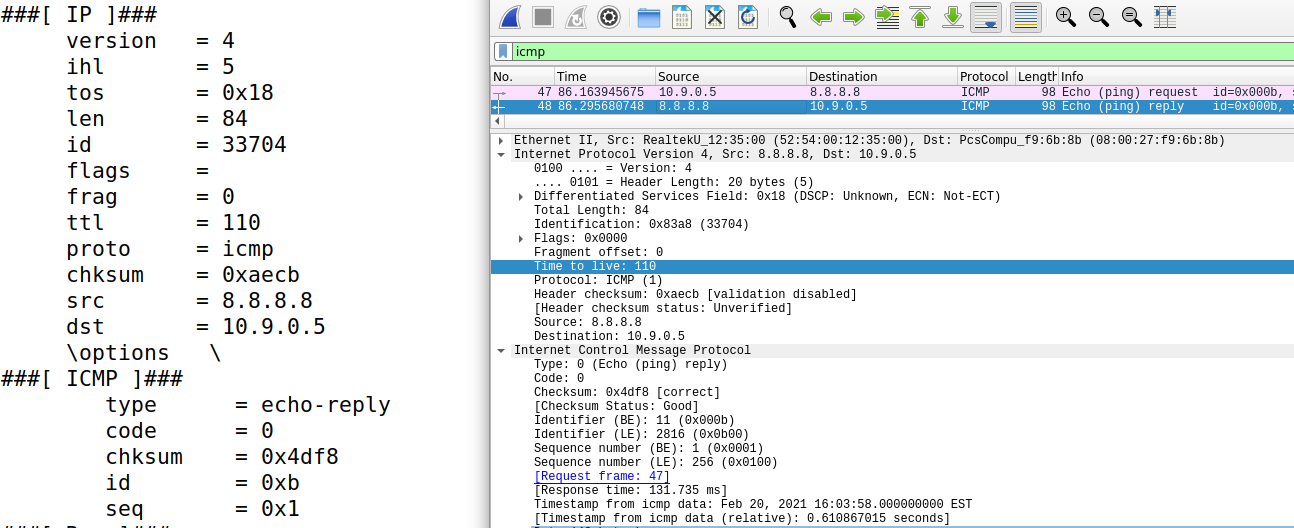
request:



terminal

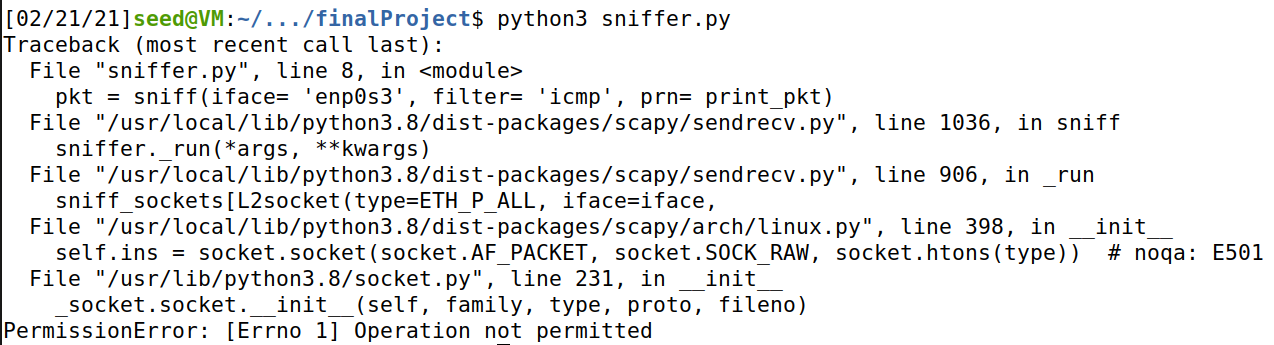
wireshark

reply:



terminal

wireshark

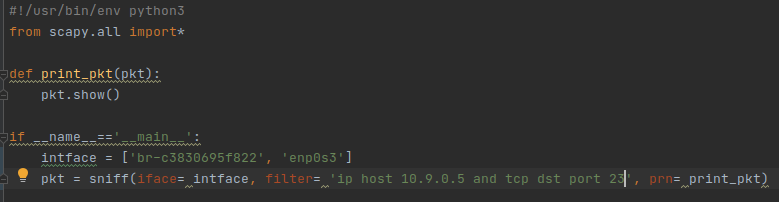
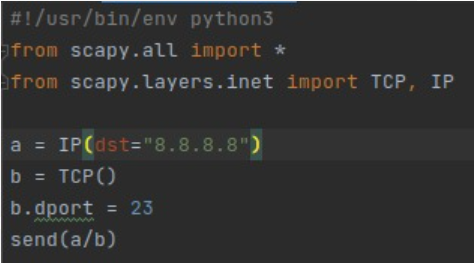
כאן ניתן לראות את תוצאת ההרצה של הקובץ בלי הרשאת גישה (sudo). הסיבה שמתקבלת שגיאה, כי האזנה לפקטות זה דבר שדורש הרשאת מנהל. בלי sudo אין הרשאת מנהל, ולכן לא ניתן להפעיל ,raw socket ולכן מתקבלת שגיאה.

**חלק B:**

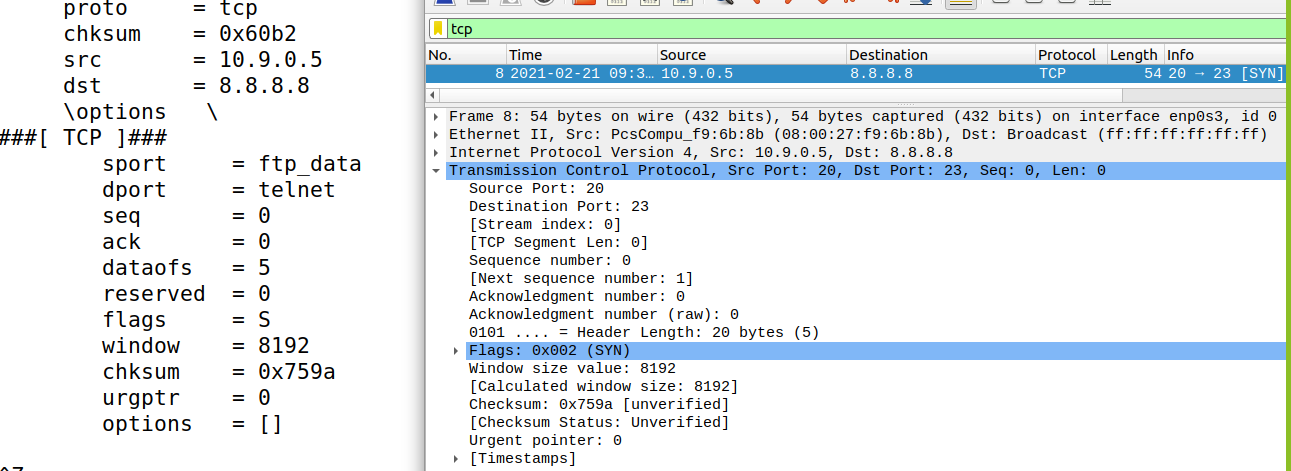
בחלק הזה נתבקשנו לפלטר את הסניפר שבנינו לפי סוגי פקטות:

-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):



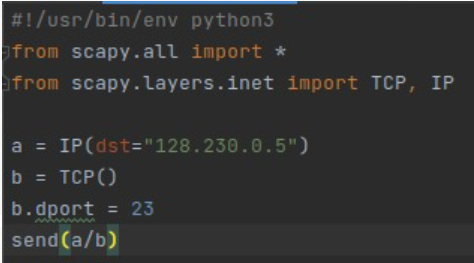
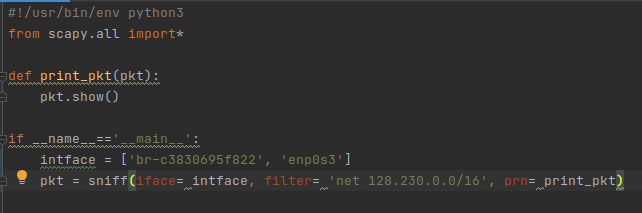
ניתן לראות שקיבלנו אותן פקטת TCP בטרמינל ובווירשארק:



terminal

wireshark

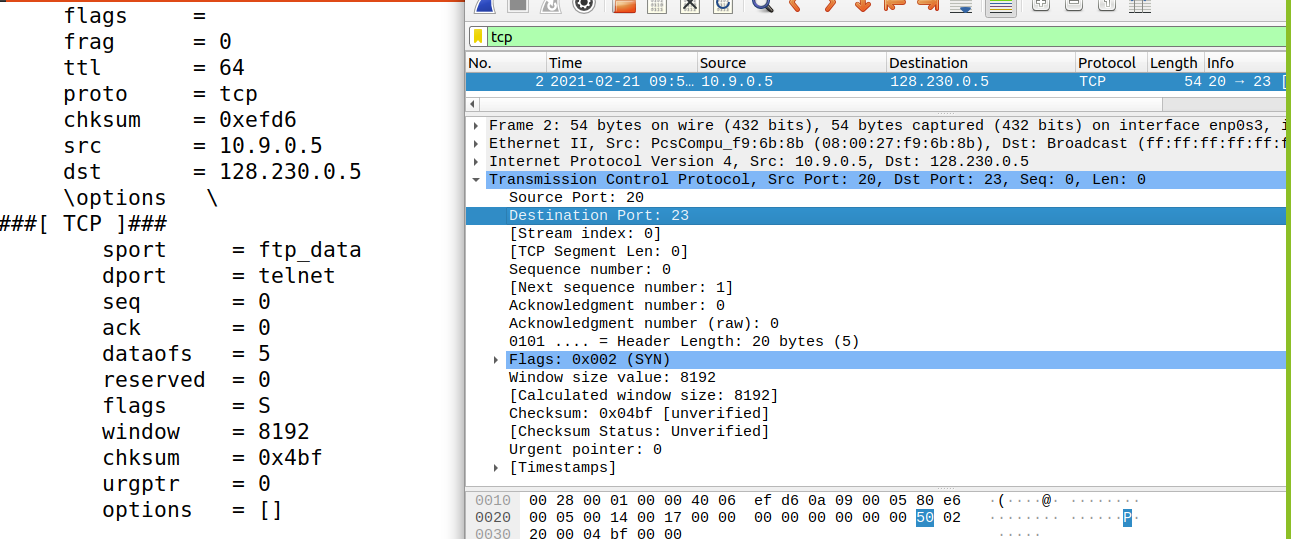
-לפי רשת פנימית: כאן נתבקשנו ללכוד פקטות שרצות ברשת פנימית מסוימת (128.230.0.0/16):



ניתן לראות שקיבלנו תוצאה זהה בטרמינל ובwireshark:

terminal

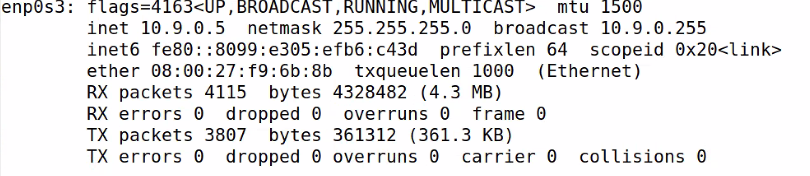
wireshark



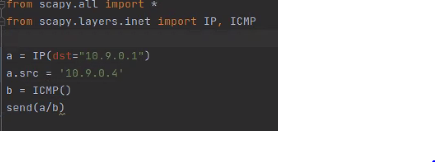
משימה 1.2:

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

ראשית, נשים לב מה הip שלנו: 10.9.0.5:



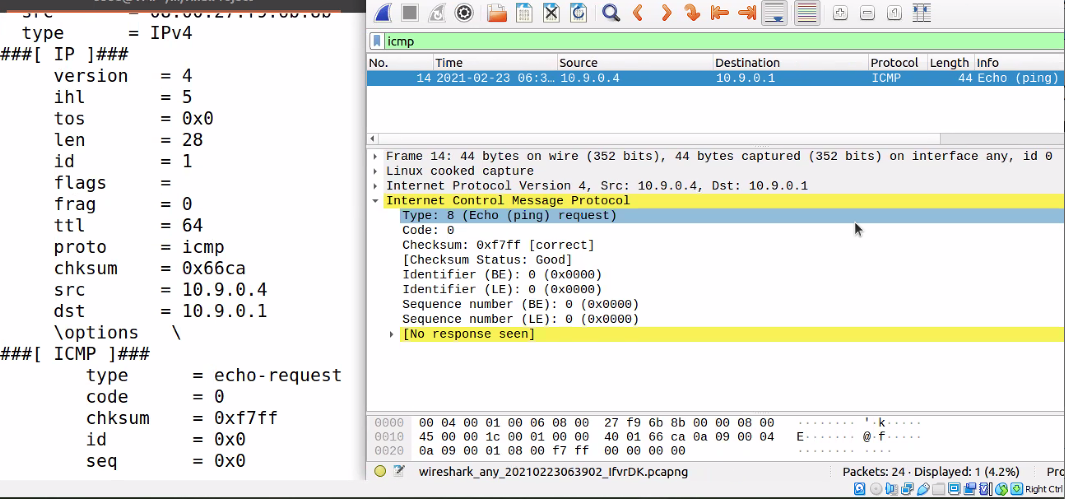
כעת, נבצע spoofing: נתחזה לip 10.9.0.4 ונשלח פאקטת icmp לip 10.9.0.1:



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):

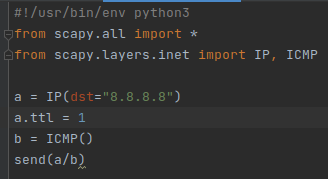
terminal

wireshark

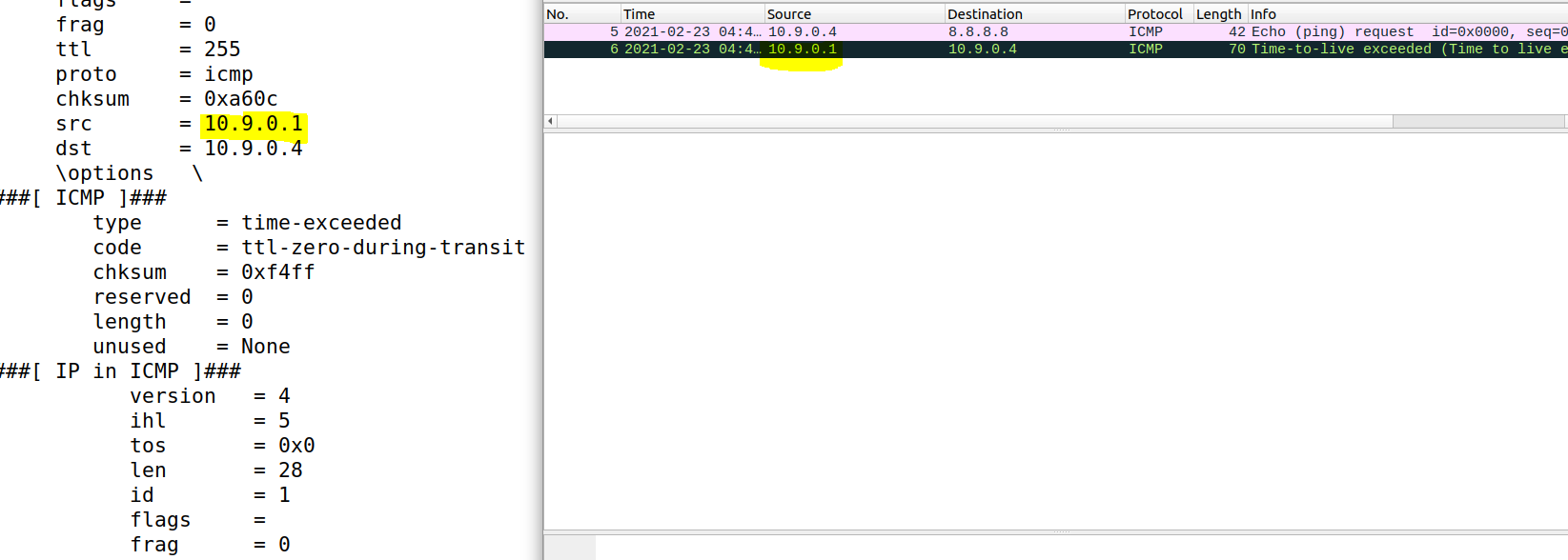


משימה 1.3:

נשלח בקשת icmp לip 8.8.8.8 ונראה כמה נתבים הפקטה צריכה לעבור בדרך על מנת להגיע ליעדה.

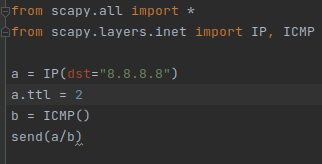
כאשר ttl = 1,

הפאקטה תגיע עד הנתב שכתובתו היא 10.9.0.1:

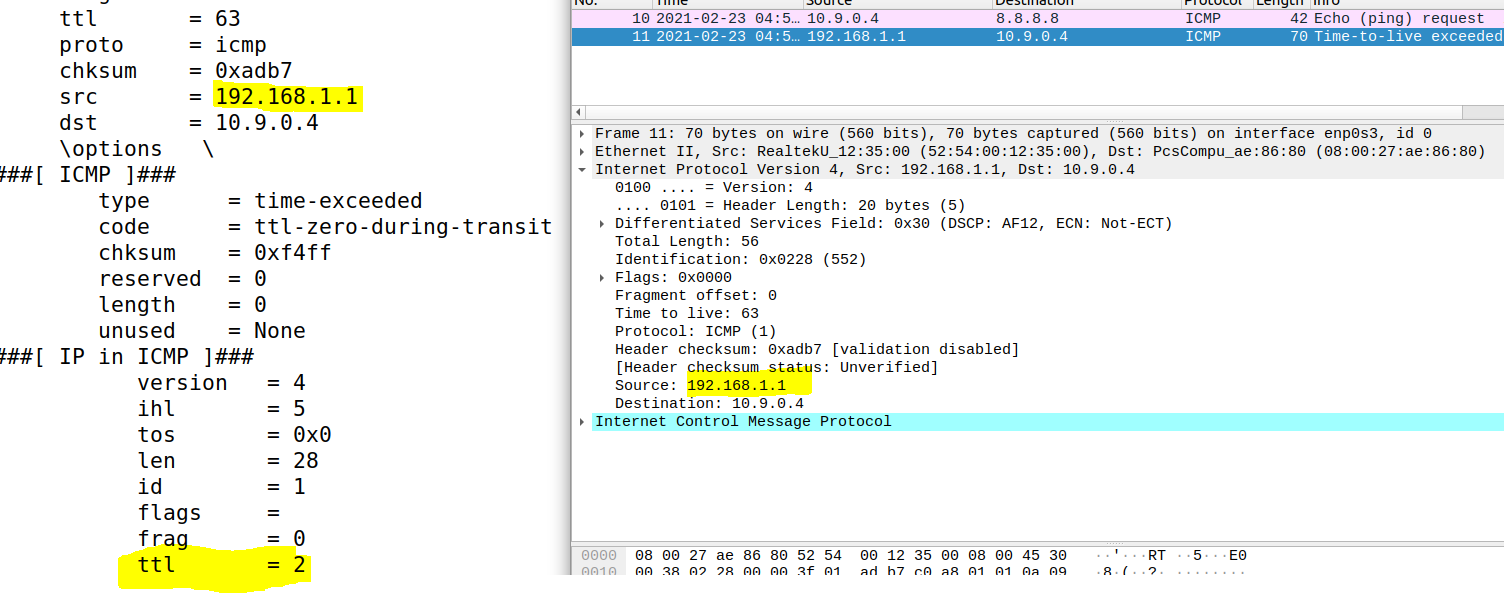


terminal

wireshark

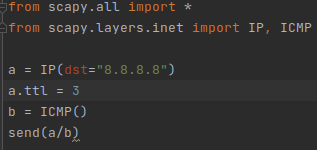
כאשר ttl = 2,

הפאקטה תגיע עד הנתב שכתובתו 192.168.1.1:

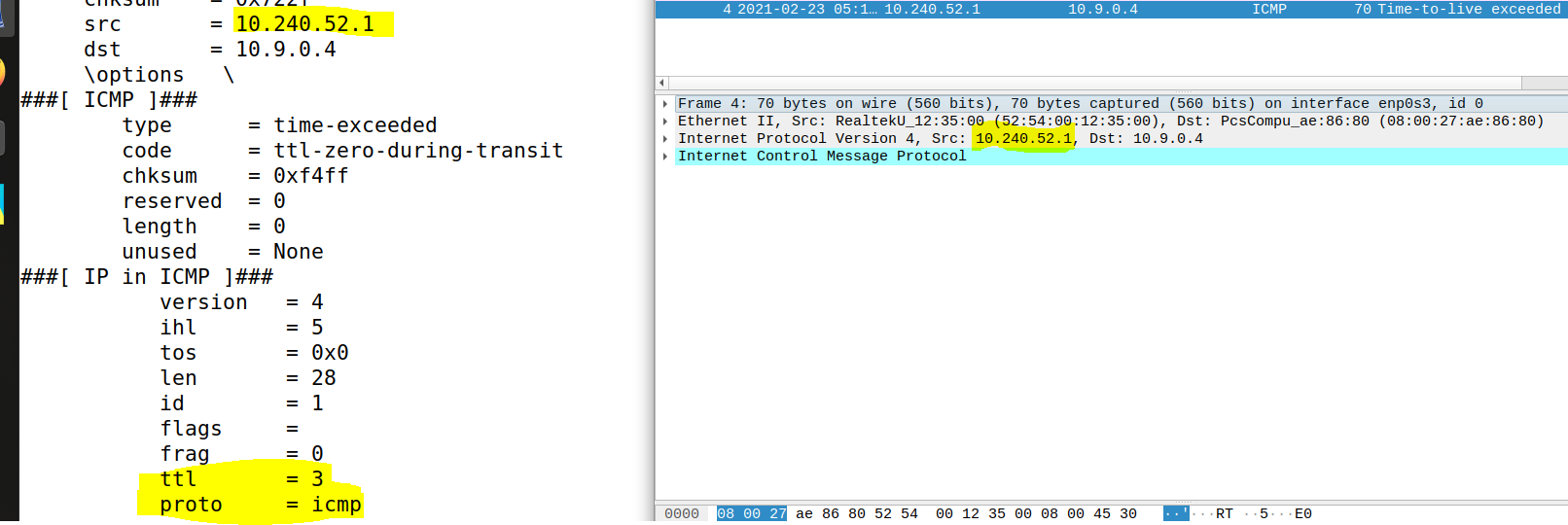


terminal

wireshark

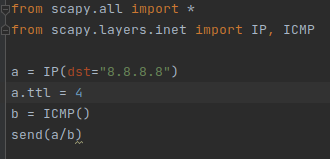
כאשר ttl = 3,

הפאקטה תגיע עד הנתב שכתובתו 10.240.52.1:

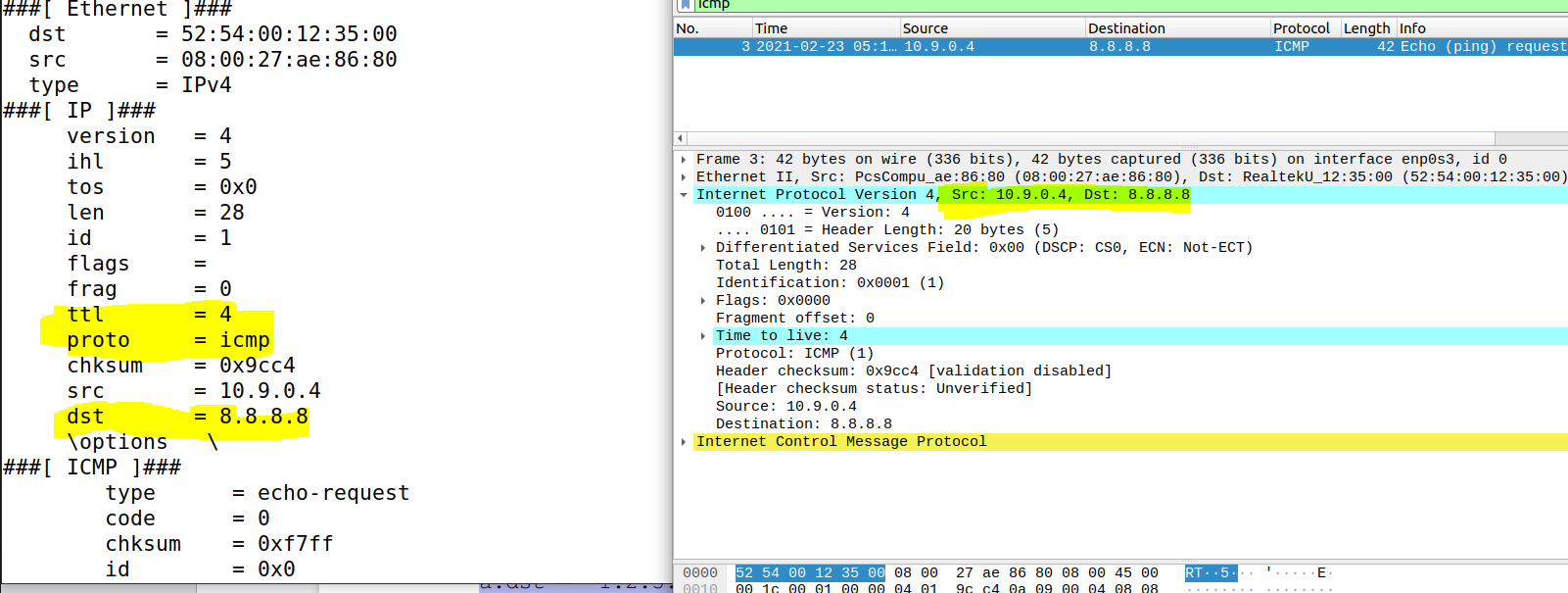


terminal

wireshark

כאשר ttl = 4,

הפאקטה מגיעה ליעד כלומר ל-8.8.8.8:



terminal

wireshark

ניתן לראות שהפעם בניגוד לפעמים הקודמות, אין הודעה על נפילת הפקטה בדרך.

משימה 1.4:

מצאנו שכדי להגיע מהhost שלנו ל8.8.8.8 הפאקטה עוברת ב4 נתבים בדרך כלומר הttl הוא 4. הנתבים בהם הפאקטה עוברת מפורטים לעיל.

משימה 1.4:

* תמונה שמכילה טקסט, צילום מסך, מקורה

  התיאור נוצר באופן אוטומטי**1.2.3.4**- כתובת לא קיימת:

ניתן לראות שהצלחנו להחזיר reply למרות שהכתובת אינה קיימת ברשת.

* תמונה שמכילה שולחן

  התיאור נוצר באופן אוטומטי**10.9.0.99**- כתובת לא קיימת בתוך הLAN:

ניתן לראות כי למרות שלא קיימת כתובת כזו בתוך הLAN הצלחנו להחזיר reply.

הודעת ping מטרתה לגלות האם הip אליו עשינו Ping קיים. כדי לדעת אם host מסויים קיים בתוך הLAN או כדי לדעת את הדרך אליו מוציאים הודעת arp. זו הסיבה שאנו רואים כאן arp ולא icmp request.

הסיבה לכך שאנו רואים פעמיים ARP כל פעם היא שהwireshark מאזין בany וקיימים 2 hosts בsubnet לכן אנו רואים ARP עבור כל host בsubnet.

* תמונה שמכילה טקסט

  התיאור נוצר באופן אוטומטי**8.8.8.8**- כתובת שקיימת ברשת:

ניתן לראות כי עבור כל request קיימים 2 replies אחד שלנו ואחד של 8.8.8.8 המקורי. שלנו מתקבל כתשובה לrequest ואילו משל 8.8.8.8 הוא מתעלם.

**חלק 2:**

משימה 2.1:

**חלק A**:

1. נסביר את הקריאות בסניפר שכתבנו:

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

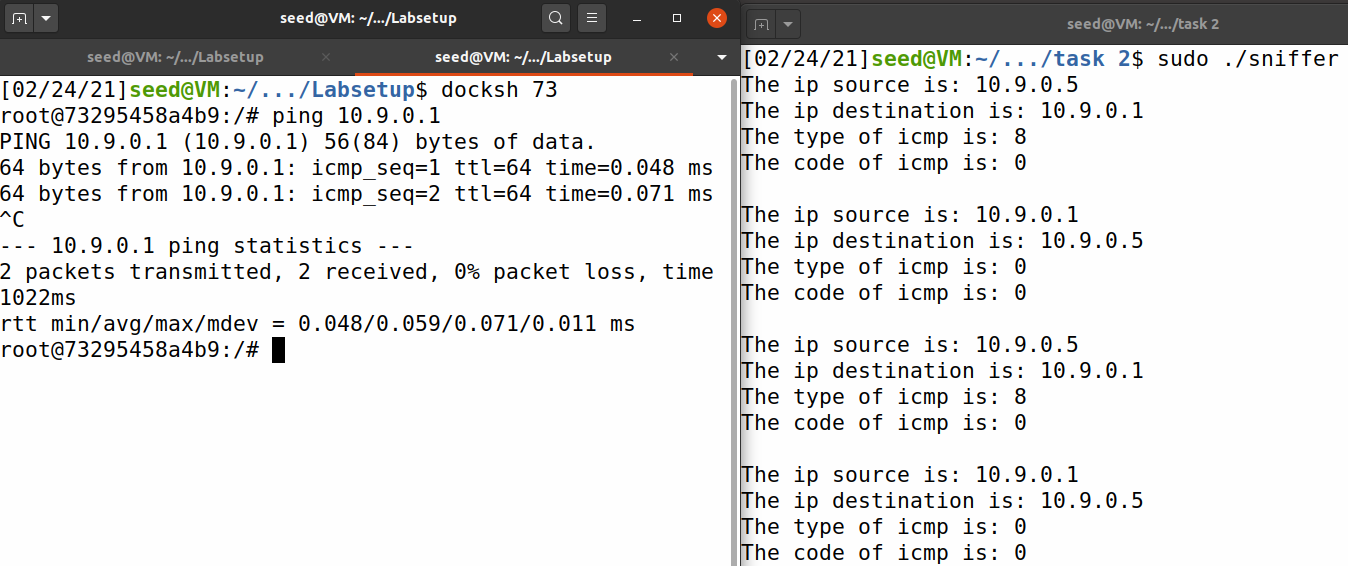
**pcap\_compile&pcap\_setfilter** – פלטור פאקטות: כלומר, אנו מגדירים לסוקט אילו פקטות להרים (פקטות מסוג icmp, או מסוג tcp, או מסוג udp וכיו"ב), ע"פ סינטקס BPF.

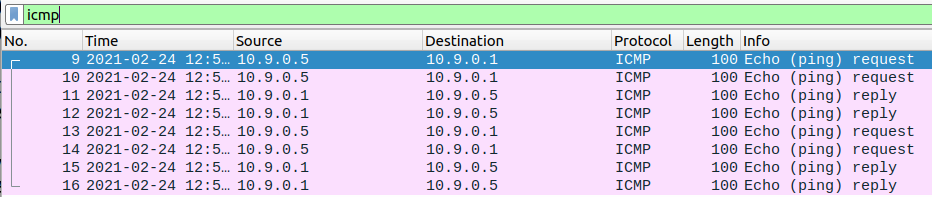
**pcap\_loop** – פקודה זו אחראית על הרמת הפאקטות ע"פ הפילטר שהגדרנו לעיל. היא רצה בלופ מספר פעמים על פי איך שמגדירים לה (1-/0 היא תריץ בלופ איסוף פעמים). כאשר מורמת פאקטה, הפקודה תפנה את הפאקטה לפונקציה שהגדרנו לה. (בקוד שלנו זו המתודה got\_packet שמדפיסה את פרטי הפאקטה).

**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.

**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.