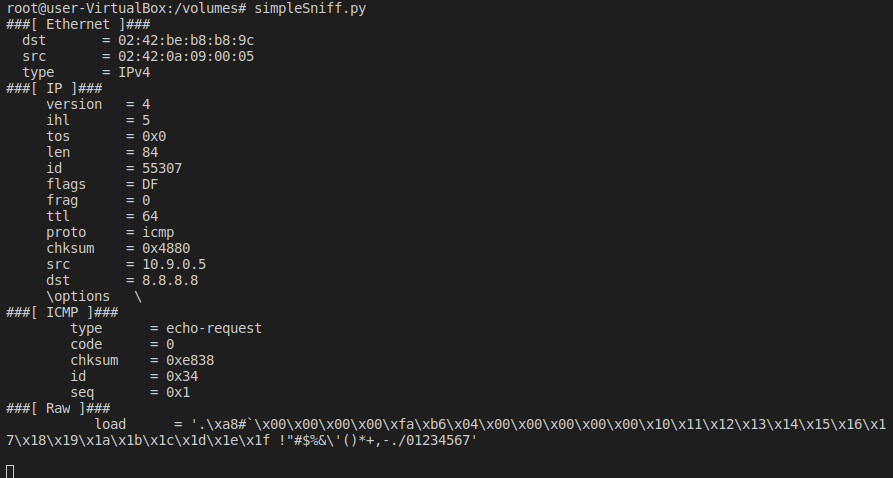
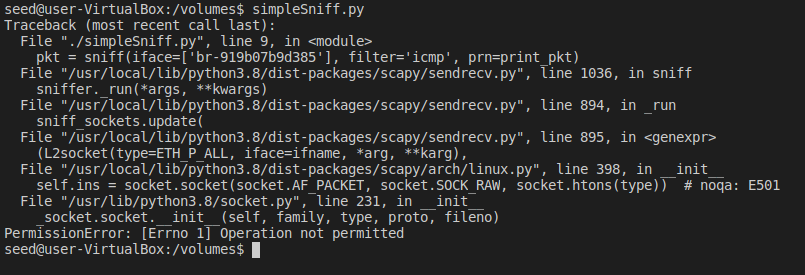
מטלה ברשתות 4

1.1A בהרצה של מנהל הכל עובד





בהרצה ללא גישת מנהל אין לו אישור לרוץ ולכן נותן שגיאת not permitted כי אין אישור לקוד לגשת לכרטיס הרשת הווירטואלי



1.1B

ICMP

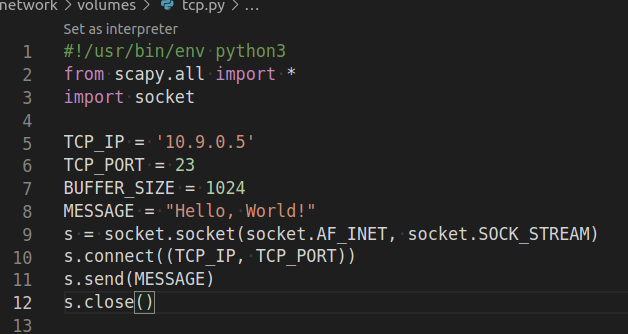


ומצורפת תמונה של הרצה בתת שאלה קודמת

TCP with ip:10.9.0.6 dest port:23

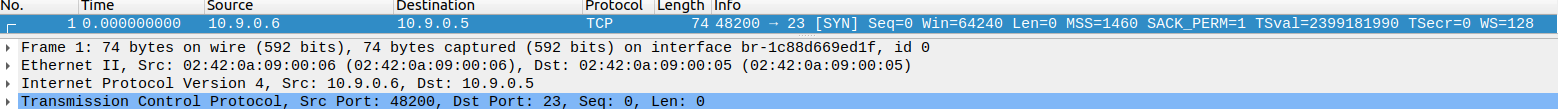
הוספנו עוד משתמש ל-containers ומשם שלחנו לאחד המשתמשים האחרים הודעת TCP בזמן שהקשבנו עם ה-attacker

דוגמת הקוד שנשלחה:

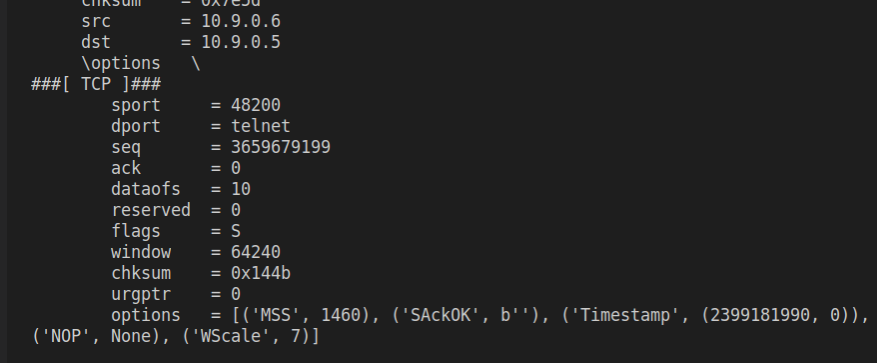


התוצאה:

ב-wireshark



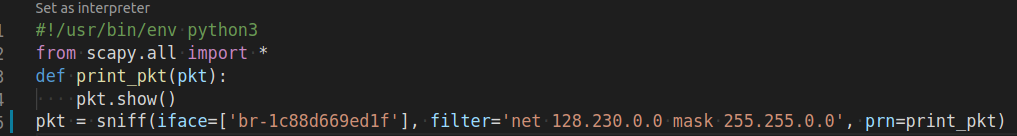
ב-attacker



Subnet:

שלחנו ping לכתובת 128.230.61.171 מכתובת 10.9.0.6

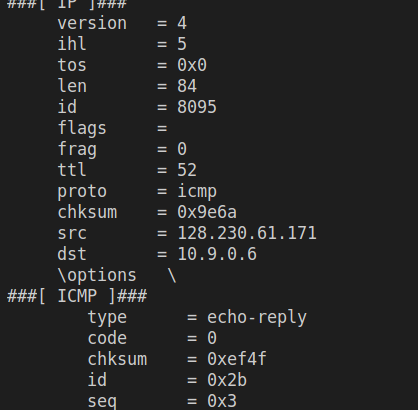
דוגמת הקוד שבצענו sniffing



התוצאה ב-wireshark



ב-attacker

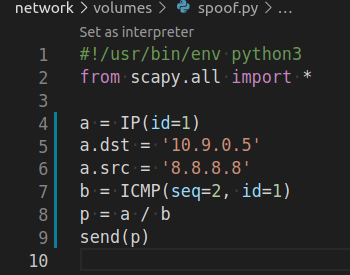


**Task 1.2**

ביצענו spoofing כך:

יצרנו פקטה b שהיא פקטת ICMP ויצרנו פקטה a שהיא פקטת IP ושינינו את הsrc שלו להיות "8.8.8.8" (גוגל) ואת ה-dest שלו להיות אחד מה-host שלנו בכתובת "10.9.0.5" ואז שלחנו אותה אל ה-host והוא הגיב בecho reply לכתובת "8.8.8.8".

הקוד:



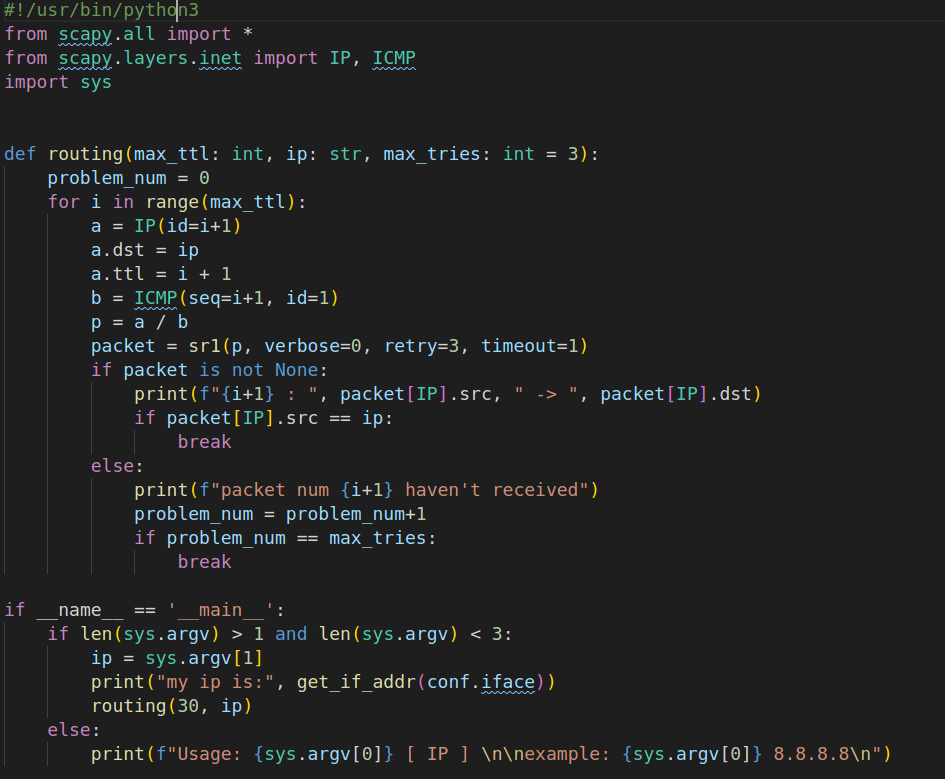
התוצאה ב-wireshark



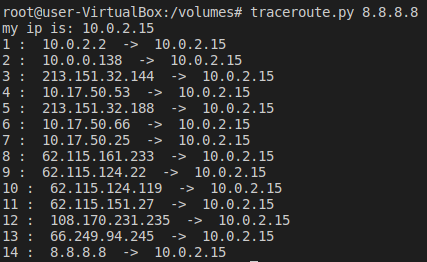
**Task 1.3**

הקוד של traceroute שלנו עובד כך: מקבל מהמשתמש כתובת IP ואז מתחיל לשלוח ICMP עם TTL עולה כל פעם, ומקבל מהמקום שבו הTTL מגיע ל0 תשובה שהICMP ששלחנו הגיע למקסימום TTL שלו, ואז מדפיסים על המסך את הכתובת ממנה קיבלנו את הICMP החוזר ואם הגענו לIP הנכסף אז מפסיק(מפסיק גם אם יש 3 שגיאות)

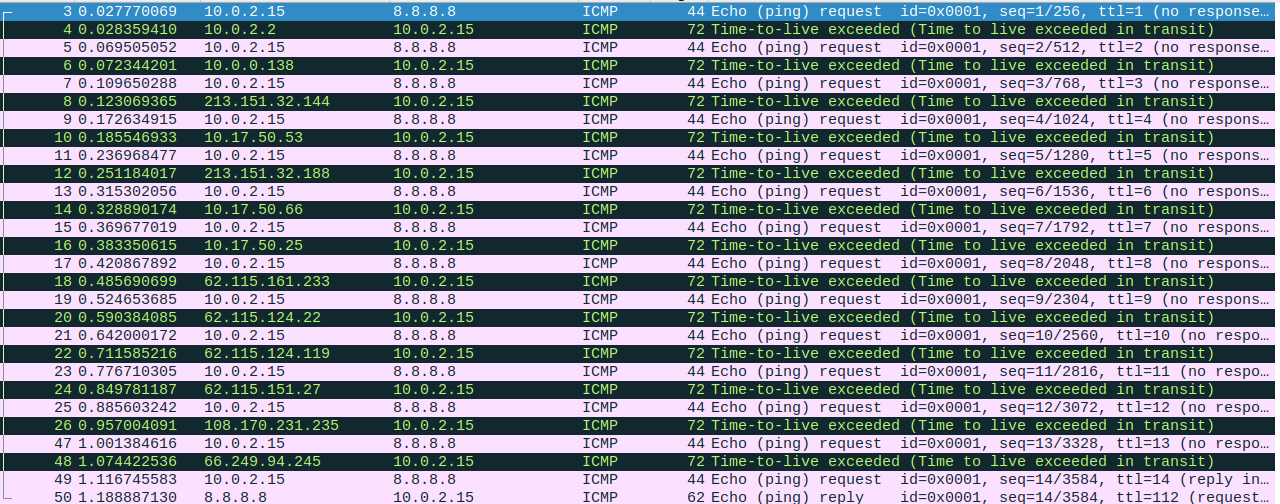
הקוד(מצורף גם הקוד עצמו בשם traceroute.py ):

****

דוגמת הרצה של הקוד ל8.8.8.8:



התוצאה ב-wireshark

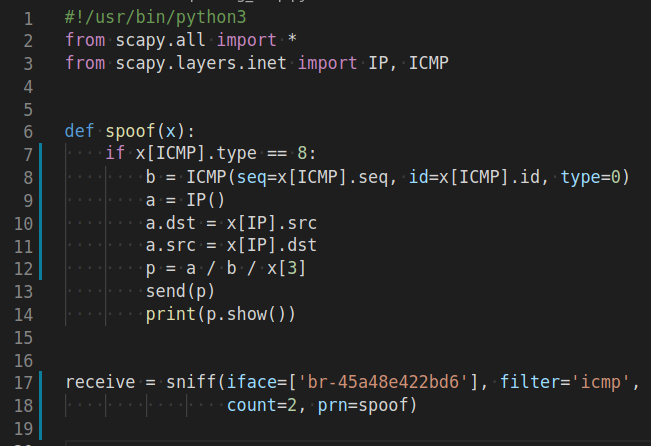


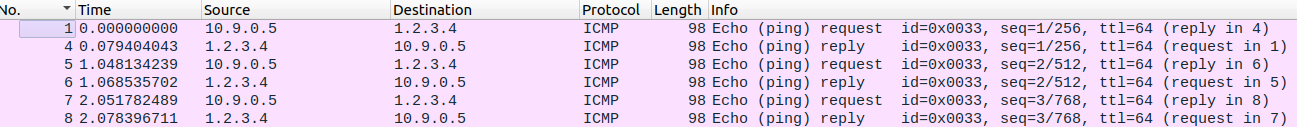
**Task 1.4**

שלחנו הודעת ICMP לכתובת 1.2.3.4 שלא קיימת מתוך ה-host שלנו, ולא קיבלנו תגובה חזרה,

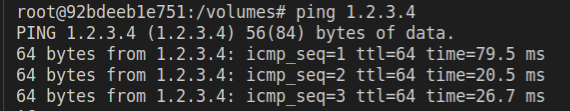
בתוך ה-Wireshark :

ואז הפעלנו את הקוד spoofing שלנו כאן:

ושלחנו שוב הודעת ICMP בעזרת ה-ping המובנה בלינוקס, והפעם קיבלנו הודעת reply שזייפנו.

תמונת -Wireshark

תמונת הקבלה מהטרמינל:



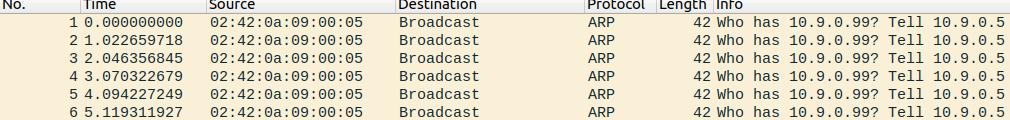
**לכתובת 10.9.0.99**

שלחנו הודעת פינג אך קיבלנו רק תשובה ש-

Destination Host Unreachable

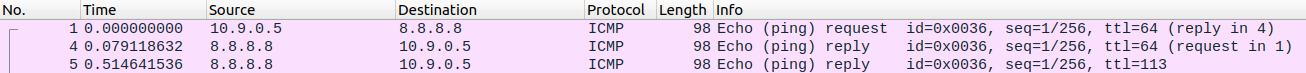
ובמבט על ה-wireshark נראה כי ה-host שולח הודעת broadcast ARP ומחפש למי יש את הכתובת הנ"ל אך ללא תשובה, וכיוון שכתובת זו צריכה להיות קיימת בתוך הרשת הפנימית בגלל ה-subnet mask שלה, היא אמורה להחזיר תשובה אך היא באמת אינה קיימת ברשת הפנימית, ולכן אינו שולח בכלל הודעת ICMP ולכן לא ניתן לבצע spoofing. (אם רוצים בכל זאת לבצע spoofing צריך לבצע זאת ע"י הקשבה להודעת ה-ARP)

התוצאה ב-wireshark:



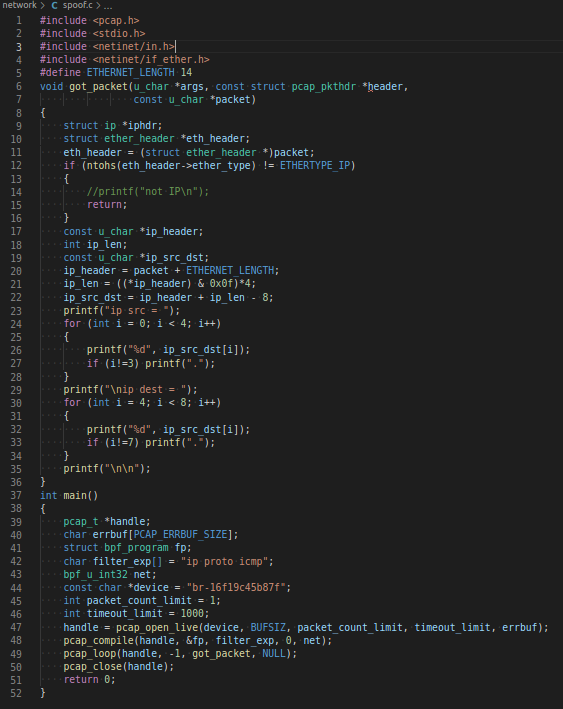
**לכתובת 8.8.8.8**

שלחנו הודעת ping וקיבלנו הודעות reply חוזרות, ומידי פעם הודעות reply משוכפלות ,מכיוון שגם מערכת ה-spoofing שלנו החזירה תשובה וגם הכתובת 8.8.8.8 החזירה תשובה.

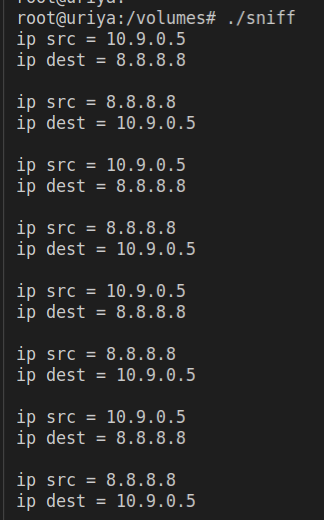
התוצאה ב-wireshark:

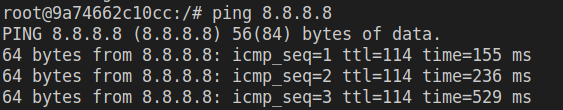
**Lab Task Set 2:**

**Task 2.1A:**

הקוד שלנו:

התוצאה בתפיסת הפקטות שנשלחות באמצעות ping ל-8.8.8.8 והקילוף שלהם עד מציאת ה-ip source וה-ip destination בקוד שלנו:



****

בתמונה מימין ניתן לראות את שליחת ה-ping לכתובת 8.8.8.8 ע"י המשתמש של ה-host

ובתמונה משמאל ניתן לראות את ההצגה של החבילות שנתפסו ע"י ה-attacker באמצעות ה-pcap ,וקולפו עד שכבת ה-ip כדי למצוא את ה-ip source וה-ip destination

**תשובות לשלושת השאלות:**

תשובה 1:

פתחנו קריאה למצביע לכרטיס הרשת הווירטואלי של הדוקר כדי שנשלח שם חבילות ונוכל לתפוס אותם.

עבור כל חבילת ip שקיבלנו, דילגנו על 14 הבתים הראשונים שהם בעצם ה- link header , וקיבלנו מצביע לתחילת הip header .

כדי לבדוק כמה הגודל שלו התעלמנו מ-4 הבתים הראשונים (שמייצגים את גרסת ה-ip (ולקחנו את 4 הבתים הבאים שבהם רשום את גודל ה-header .

וכיוון שה- ip source וה-ip destination נמצאים תמיד בסוף הפקטה קפצנו את כל הגודל וחזרנו אחורה 8 בתים כדי לקבל את שניהם והדפסנו אותם.

תשובה 2:

אם לא נשתמש בהרשאת ה-root אזי אנחנו ננסה לגשת לכרטיס רשת שמנוהל ע"י מערכת ההפעלה והגישה אליו לא ניתנת למשתמשים רגילים(כלומר ללא הרשאת מנהל) ולכן היא נחסמת!

השגיאה שניתנת - Segmentation fault (core dumped)

תשובה 3:

אצלנו זה לא משנה אם נשתמש בpromiscuous mode או לא כיוון שהחבילות של ה-host נשלחות תמיד דרך ה-VM וה-attacker מחובר על הnetwork של ה-VM ולכן כל החבילות היוצאות מה-host עוברות דרך ה-attacker וה-MAC address שלהם כאילו מיועד ל-attacker.

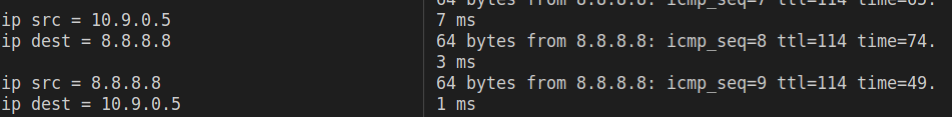
(promiscuous mode מעביר את כל החבילות הנמצאות באותה רשת לא משנה מה ה-MAC address שלהם ישירות למערכת ההפעלה.)

**Task 2.1B:**

**ICMP**

הוספנו את שורת הקוד הבאה:

ואז שלחנו ping אל 8.8.8.8 מכתובת 10.9.0.5 שהוא אחד ה-host שלנו ואכן הפילטר עבד כפי שניתן לראות בתמונה הבאה:

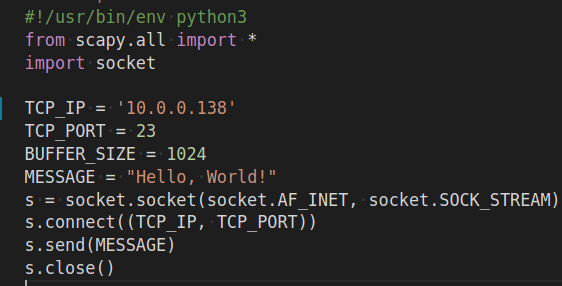
מימין שליחת ה-ping ומשמאל תפיסה עם ה-sniff והפילטר המתאים.

(כאשר שלחתי לכתובת אחרת 10.0.0.138 הוא לא תפס כלום)

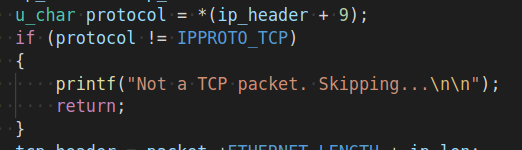
**TCP**

שינינו את הפילטר לשורת הקוד הבאה:



ושלחנו הודעת TCP באמצעות הקוד הבא ל-10.0.0.138 בתוך port 23:

תמונת השליחה ב-wireshark (כמובן ללא מענה)

 הוספנו את הקוד הבא אחרי הפילטר כדי לוודא שאכן קיבלנו חבילות של TCP (ע"מ להוכיח שהפילטר עובד).

כלומר- ווידינו שאכן זהו הפרוטוקול שהוא מתעסק איתו, וסוג הפרוטוקול מופיע בתוך ה- ip header10 בתים מההתחלה.

**Task 2.1C:**

ניסינו להתקין telnet ולהשתמש בו אך לא הצלחנו.

באופן תיאורטי: היינו צריכים לתפוס את כל ה-stream של ה-tcp של telnet ואז לחבר את כל ה-data שלהם לקובץ מילים ושם ניתן יהיה לקרוא את כל המידע כולל הסיסמא.

**Task 2.2A:**

כתבנו את הקוד (מופיע מצורף ל-pdf בשם – ip\_spoof.c ) כך ששלחנו הודעת ICMP Echo request ,כאילו מכתובת 1.2.3.4 שלא קיימת אל ה-host שלנו, וקיבלנו חזרה reply

כך נראה ב-wireshark:

**Task 2.2B:**

השתמשנו באותו קוד מהסעיף הקודם רק שהפעם שלחנו ל-8.8.8.8 מכתובת 10.9.0.5 שהיא ה-host שלנו.

התוצאה ב-wireshark:

**תשובה ל-4:**

נחלק ל2 מקרים:

אם שינינו גם את ה-total length בהתאם:

לא, כיוון שהוא לא יידע לקרוא את הפקטה כמו שצריך כך שלא יוכל לעבור אל השכבה הבאה.

דוגמא לשינוי קטן של הגודל:



ואם לא נשנה בהתאמה אלא רק את הגודל של ה-ip header:

אז לא תשלח בכלל הפקטה או שתשלח אך תהיה שגויה.

דוגמא לשגיאה שנתפסה ב-wireshark:



**תשובה ל-5:**

לא, אין חובה לחשב את ה-checksum , כי הווידוא של ה- checksum כבוי.



**תשובה ל-6:**

צריך הרשאת root כדי שנוכל אנחנו במקום מערכת ההפעלה עצמה לגשת לשכבות הנמוכות.

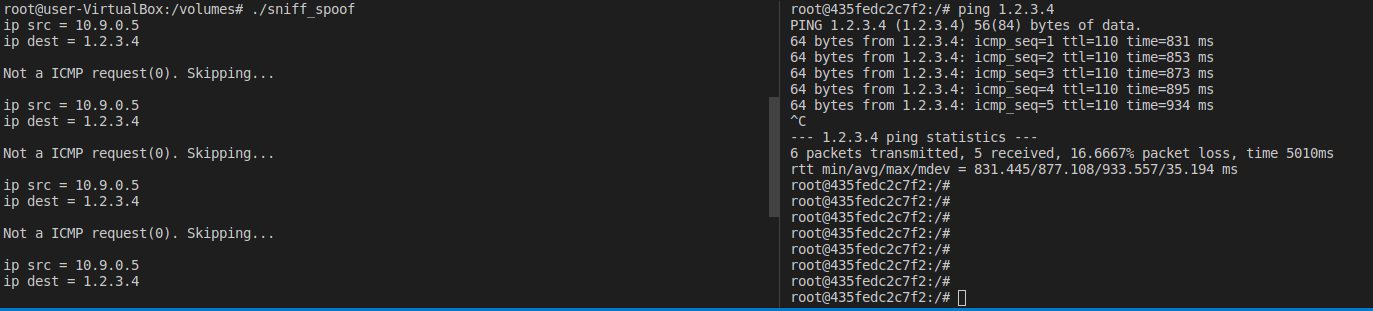
אם לא תהיה הרשאת root אזי לא נוכל לבצע פתיחת socket, כיוון שלא ניתן לפתוח raw socket ללא אישור root.



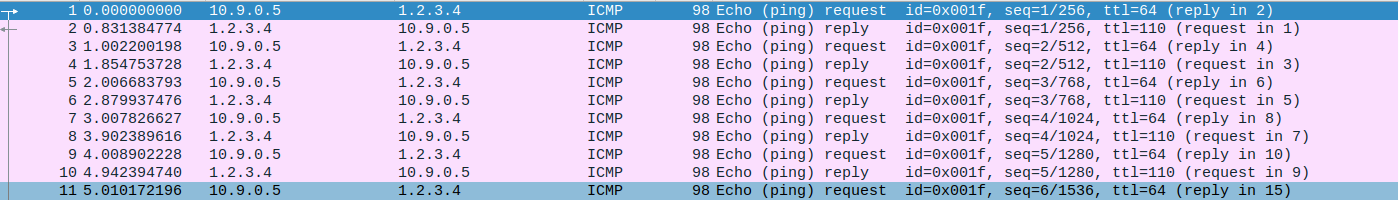
**Task 2.3:**

צילום שליחת Ping ל1.2.3.4 (כתובת שאינה קיימת) וקבלת reply על ידי הקוד שלנו שמבצע sniffing ו-spoofing

(שליחת הפינג היא מה-host והפינג החוזר נשלח מהattacker )

****

התוצאה ב-wireshark :



**הקוד מצורף בשם sniff\_spoof.c**