**עבודת גמר – sniffing and spoofing**

**מגישים:** אריאל יפעי ומוטי דהרי.

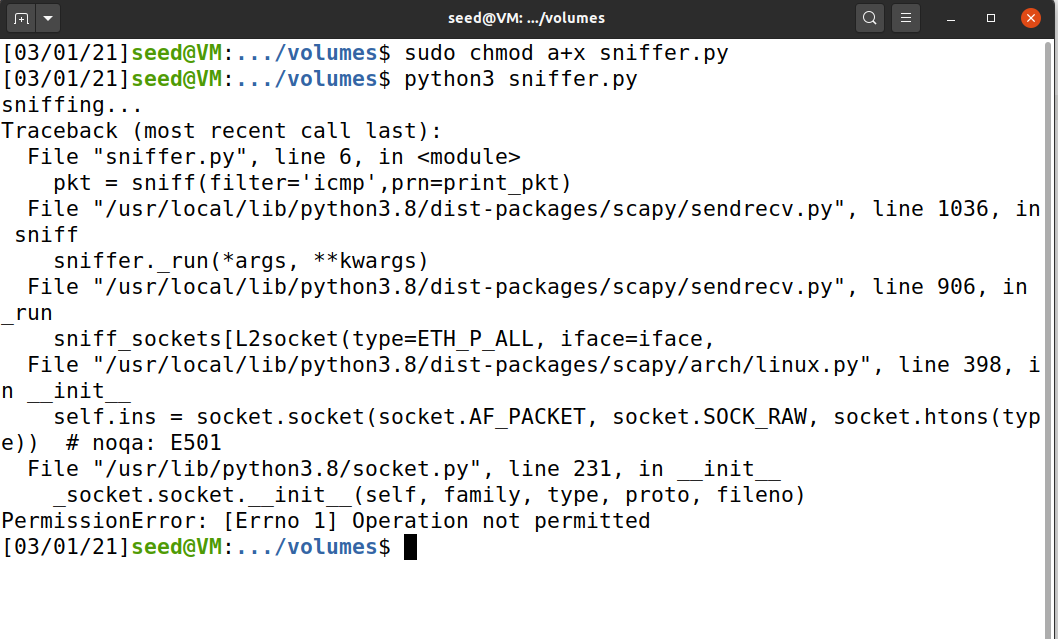
התקנו את SEEDvm בvirtualbox עדכנו את כל התוספים שהיה צריך ולאחר מכן בנינו סניפר בפייתון.

1.1A: התבקשנו להריץ את הסניפר בלי הרשאות אדמין ועם הרשאות אדמין ולראות מה קורה, הרצנו בטרמינל אחד את הסניפר שלנו ובשני את הפקודת ping לאתר של גוגל.

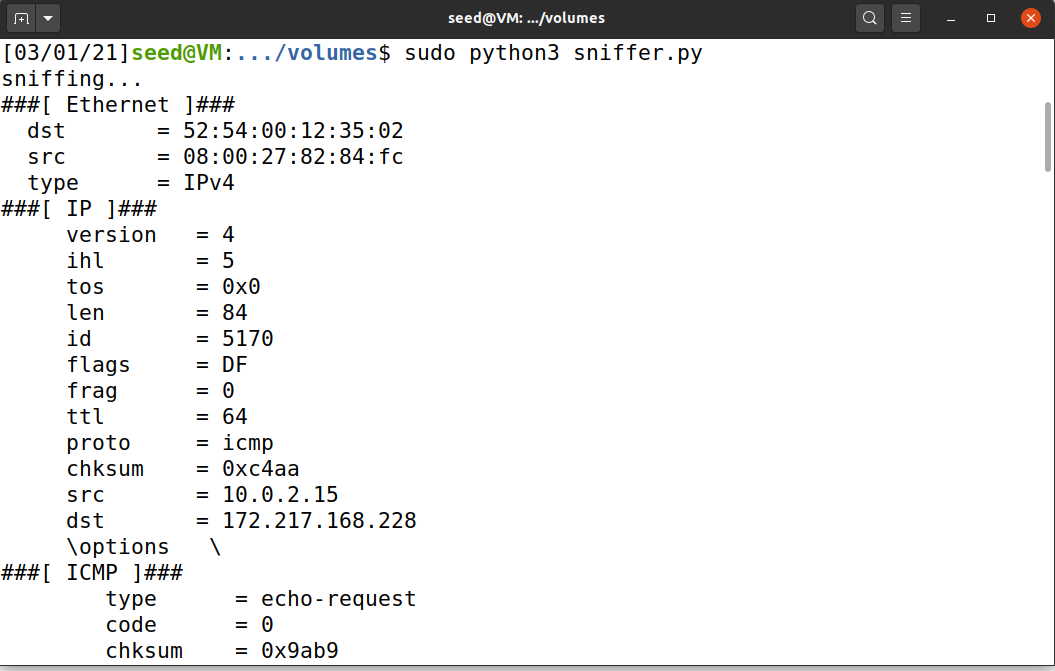
הסניפר:



הרצה בלי הרשאות אדמין:

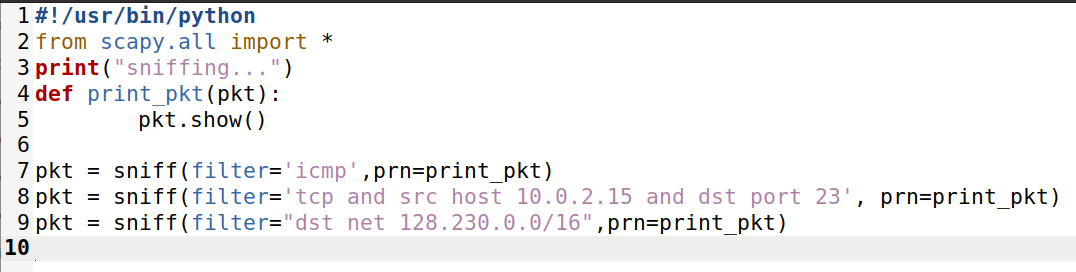


הרצה עם הרשאות:

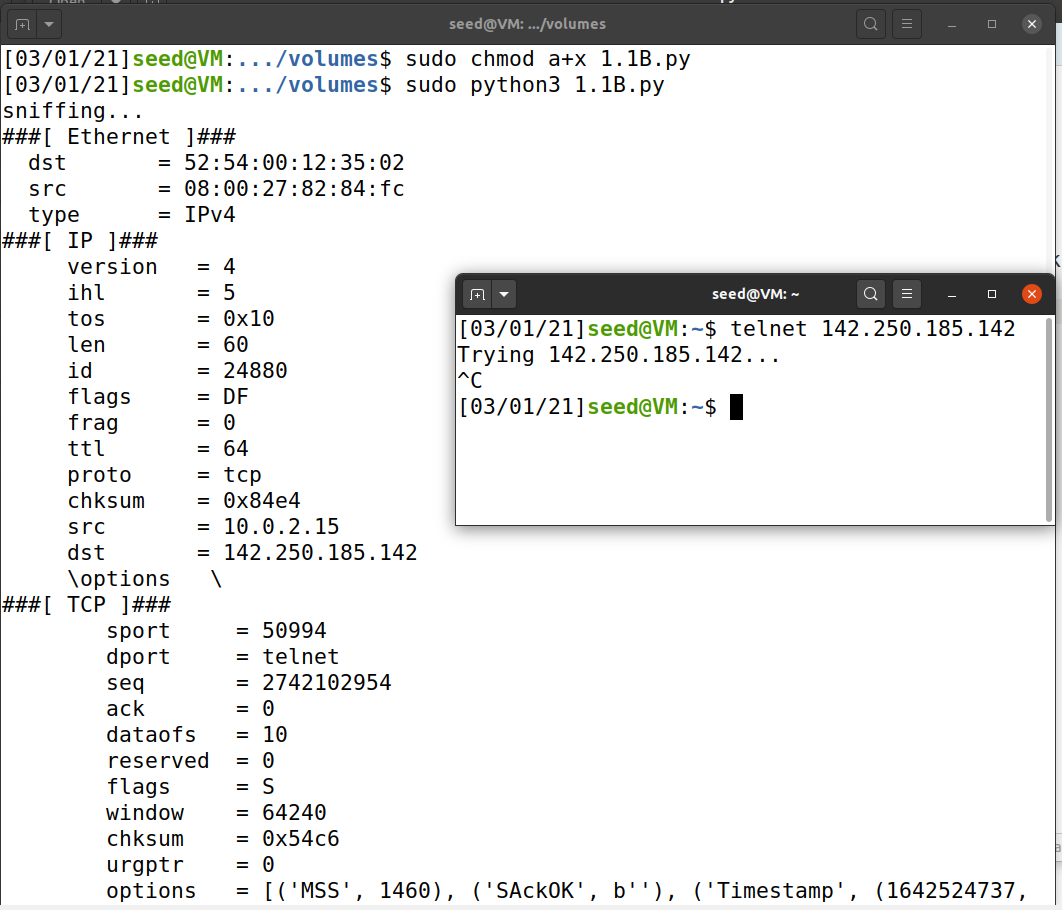


הצלחנו לתפוס פקאטות!

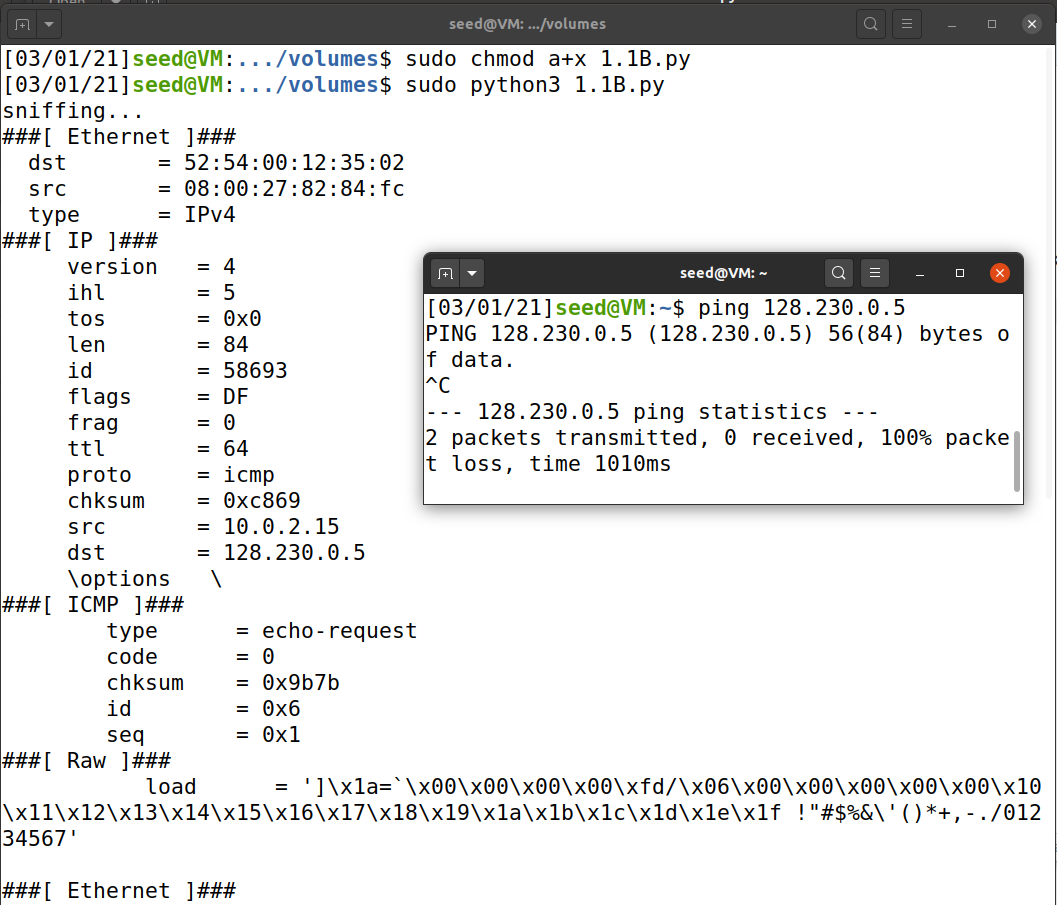
1.1B: התבקשנו להוסיף עוד פילטרים לסניפר שלנו.



תפיסת פקטת TCP:



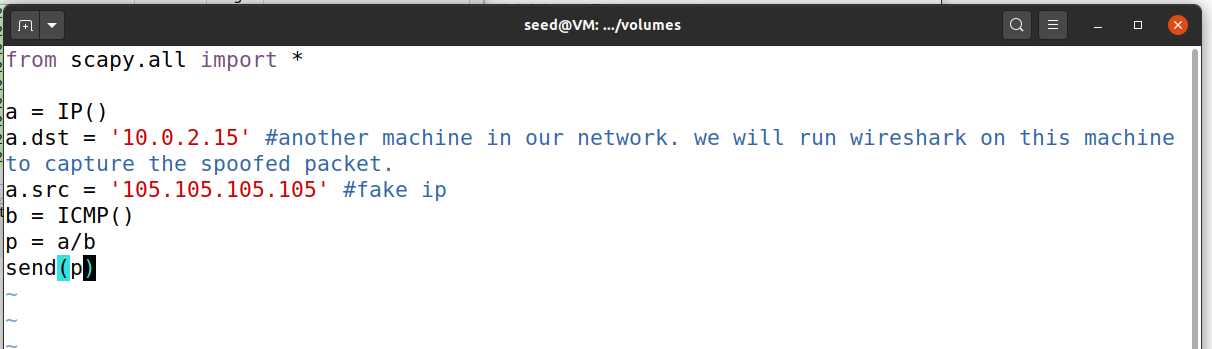
תפיסת פקטה מ subnet:



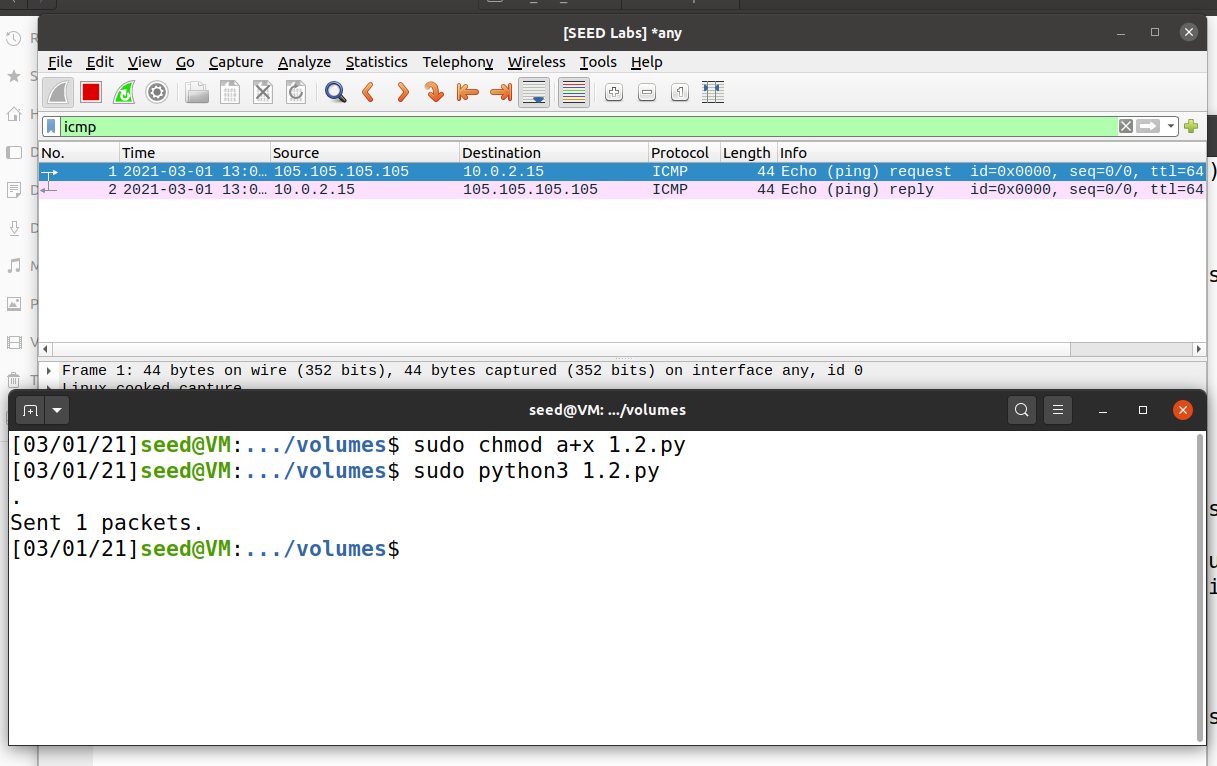
1.2:

אנחנו צריכים לעשות spopfing לפקטת icmp לכן אנו נשלח באמצעות scapy פאקטה מip שהוא לא של האינטרפייס שאנחנו משתמשים בו לאינטרפייס אחר ברשת שלנו ואנו נבדוק בwireshark שאכן האינטרפייס השני לא רואה שהפקטה נשלחה מהאינטרפייס הראשון.

הקוד:



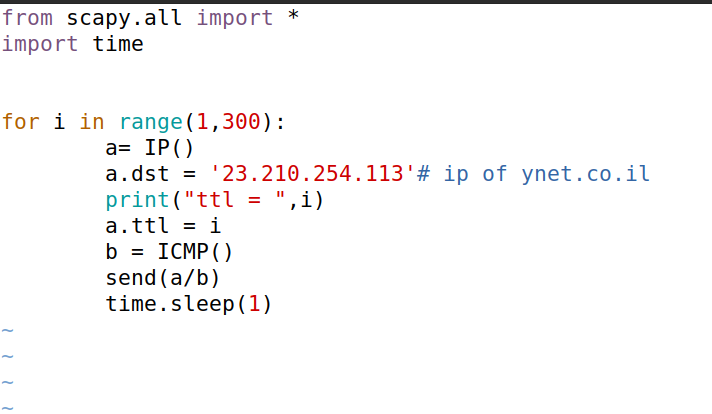
התהליך:



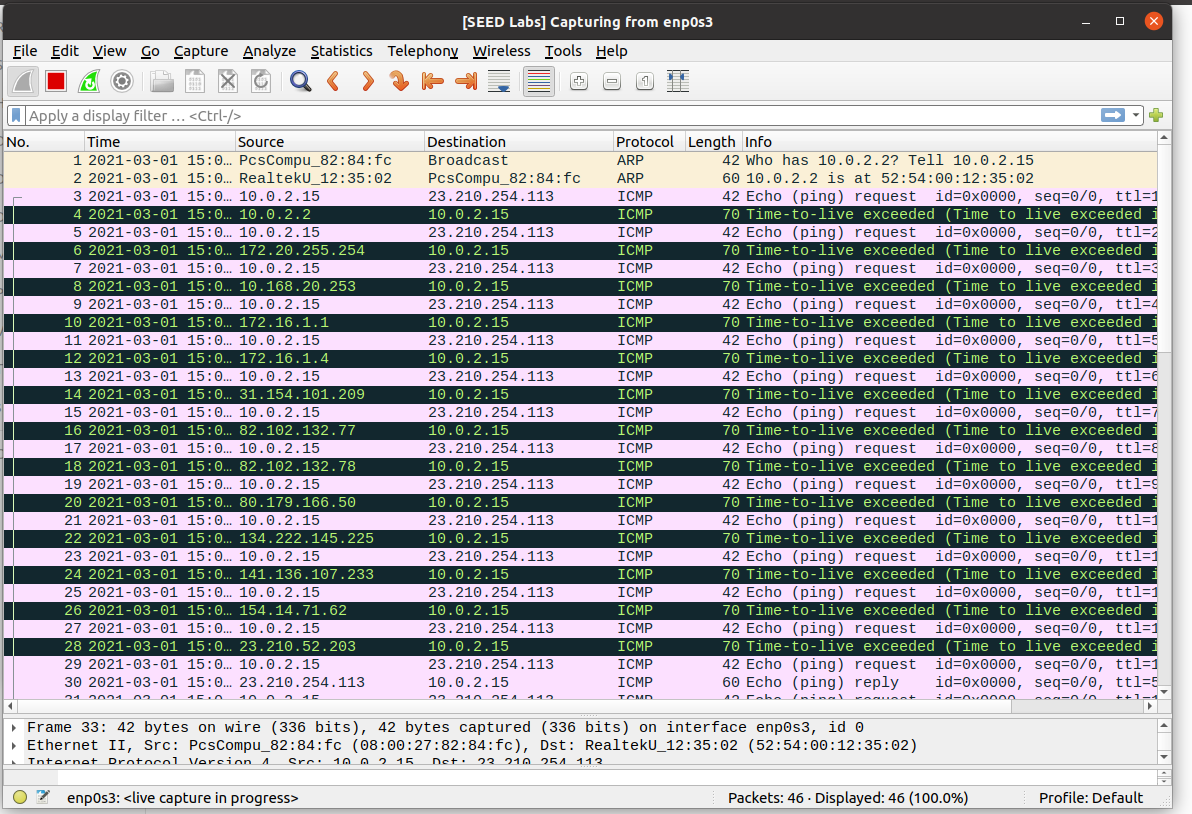
ואכן ה-request נשלח מהip המזוייף ונשלח reply למזוייף.

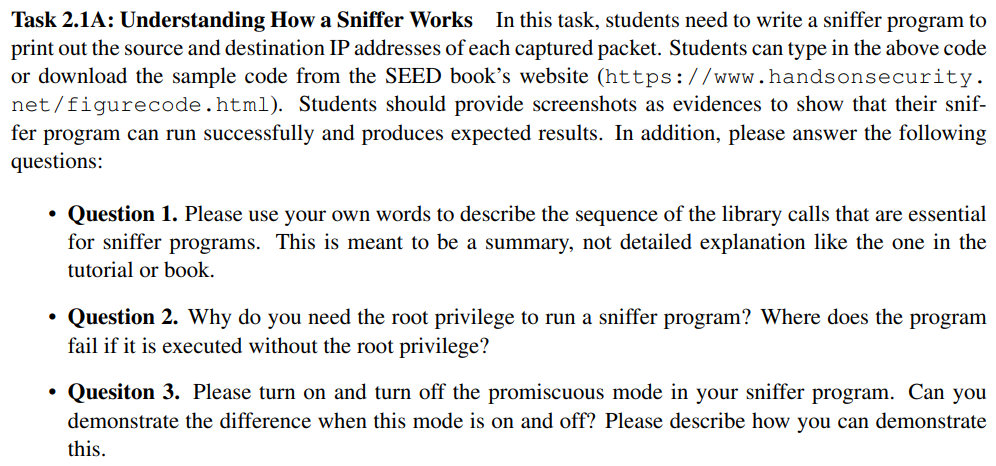
1.3:

התבקשנו לבדוק כמה ראוטרים אנו עוברים עד שאנו מגיעים ליעד,ע"פ ההדרכה עלינו להגביל את זמן החיים של הפקטה ולראות בכל פעם שיש לה error מסוג time-to-live has exceeded (הזמן שלה נגמר אך ללא השלמת המשימה) נבדוק באיזה ip של ראוטר היא עצרה, נרוץ בלולאה וכל פעם נעלה את הttl באחד ונספור כמה ip שונים אנו עוברים עד שאנחנו כבר לא מקבלים שגיאה.



קיבלנו 13 הודעות שגיאה מ13 ראוטרים שונים כלומר עברנו 13 ראוטרים עד היעד.



**Task 2.1A:   
**

שאלה 1:

רצף הקריאות בספריה pcap שנחוצות בכדי להסניף פאקטות:  
1. נקרא לפונקציה pcap\_open\_live שתאזין לאינטרפייס מסוים שנגדיר (המשתנה הראשון בפונקציה).  
המשתנה השני - מקסימום בתים שיתפסו על ידי pcap.  
המשתנה השלישי - הוא האם להפעיל במוד של promiscuous או לא (אם הוא מוגדר ל-0 זה אפשרי במקרים ספציפיים).  
המשתנה הרביעי - הוא זמן לקריאה עד שהפאקטה תמות.  
המשתנה החמישי - ebuf הוא מצביע מסוג סטרינג שבמידה ויש שגיאה כלשהי אז ירשום לשם את השגיאה ונוכל לצורך העניין לפלוט את זה בקובץ שגיאות שניצור.  
הפונקציה מחזירה pcap\_t \* אם הצליח או NULL  אם נכשל ואם הוחזר NULL אז השגיאה תיכנס לתוך המשתנה errorMSG .

2. נקרא לפונקציה pcap\_compile שמשמש לקימפול הסטרינג לפילטר התכנית.   
המשתנה הראשון - מחזיק את ה session שלנו   
המשתנה השני - מצביע למקום שנאחסן את הגרסה המסודרת של המסנן שלנו.   
המשתנה השלישי - הביטוי עצמו, בפורמט מחרוזת רגיל.  
המשתנה הרביעי - מספר שלם שמחליט אם הביטוי צריך להיות " optimized" (אופטימיזציה) או לא (0 false, 1 true).  
המשתנה החמישי - מסכת הרשת של הרשת שהמסנן חל עליה  
מחזירה 0 אם זה הקימפול של המחרוזת הצליח, ו-Pcap\_Error אם זה נכשל אם חזר Pcap\_Error ניתן לקרוא עם p כארגומנט לאחזור או הצגת טקסט השגיאה שהתקבל.

3. נקרא לפונקציה pcap\_setfilter שמשמשת להגדרת המסנן (filter) של התכנית.  
fp – הוא פויינטר למבנה bpf\_program.  
מחזירה 0 אם זה הצליח, ו-Pcap\_Error אם זה נכשל אם חזר Pcap\_Error ניתן לקרוא עם p כארגומנט לאחזור או הצגת טקסט השגיאה שהתקבל.

4. נקרא לפונקציה pcap\_loop שמעבדת חבילות מלכידה שנתפסת בלייב או saveFile עד שמעבדי החבילות cnt עוברים, בסוף saveFile מגיעה שקוראים מתוך מ pcap\_breakloop,saveFileאו שמתרחשת שגיאה.  
זה לא חוזר כאשר מתרחש פסק זמן של הבאפר מתרחש.  
ערך של 1- או 0 עבור cnt שווה לערך אינסוף, ככה שחבילות מעובדות עד שיתרחש תנאי הסיום.   
הפונקציה מחזירה 0 אם cnt מיצה או אם בזמן קריאה מsaveFile אין יותר חבילות זמינות, הוא מחזיר PCAP\_ERROR אם התרחשה שגיאה או PCAP\_ERROR\_BREAK אם הלולאה הסתיימה עקב קריאה ל- pcap\_breakloop לפני שעובדו חבילות כלשהם.  
זה לא חוזר כאשר מסתיים הפסק זמן או של המנות בשידור החי, במקום זה הוא ינסה לקרוא עוד חבילות.

5. נקרא לפונקציה pcap\_close שסוגרת את הקובצים המשוייכים לקריאה ומשחררת את המשאבים.

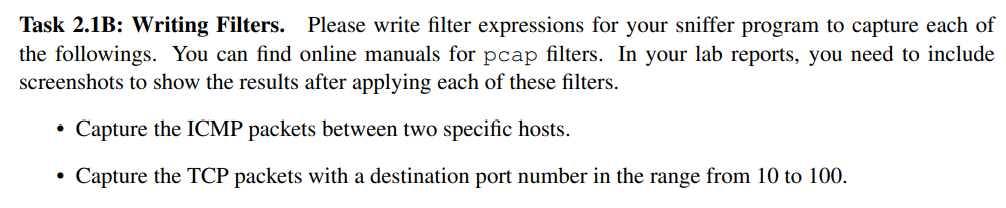
שאלה 2:

אנחנו צריכים להריץ את התוכנית של הסנפת פאקטות (חבילות) בהרשאת מנהל (administrator) מכיוון שהאזנה לאינטרפייס מסויים דורשת גישה לאינטרפייס וגישה כזו מחייבת אותנו בהרשאת מנהל (administrator).  
במידה ולא נפעיל את התוכנית עם הרשאות של מנהל (administrator) היא תיכשל בשלב של הפונקציה ה pcap\_open\_live.

שאלה 3:

כאשר נשנה את promiscuous mode ל- OFF בעזרת הפונקציה pcap\_open\_live בארגומנט השלישי – נזין שם ערך שהוא 0.  
עכשיו אנחנו נאזין לתעבורה שעוברת אך ורק במכונה הוירטואלית שאיתה אנחנו מפעילים את התוכנית ולא על כל התעבורה ברשת.

**Task 2.1B:**

****