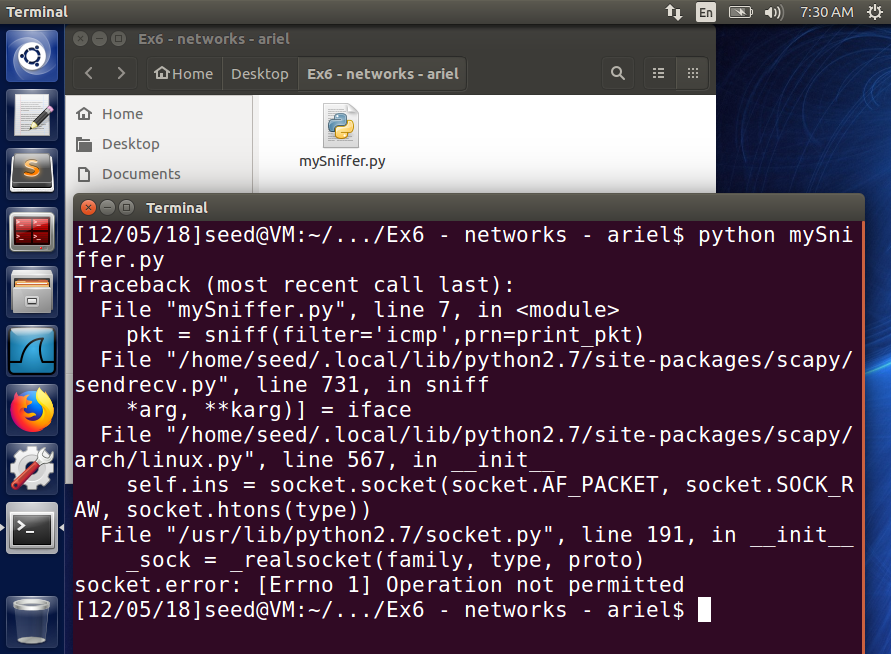
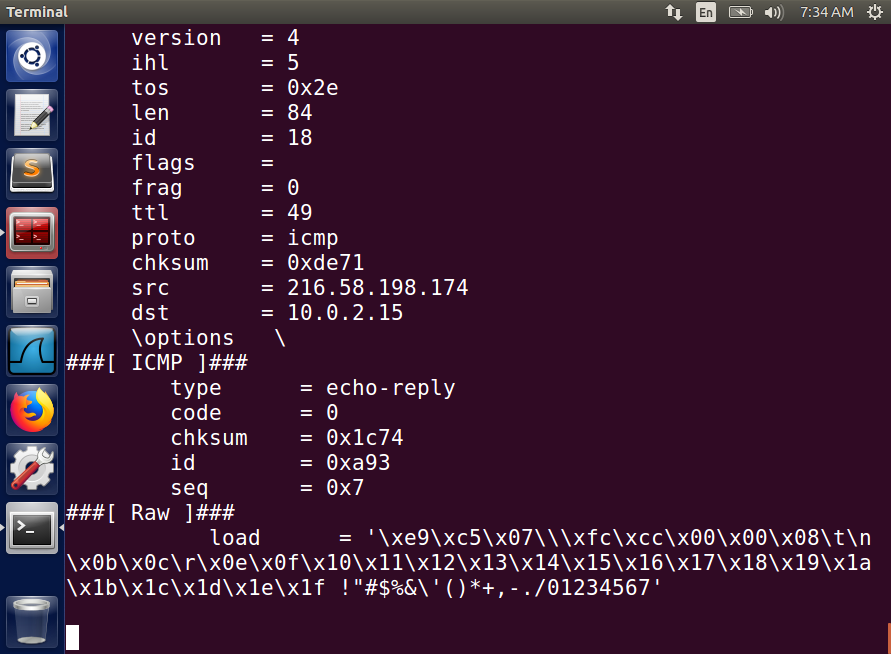
**עבודת הגשה 6 – Sniffing Spoofing**

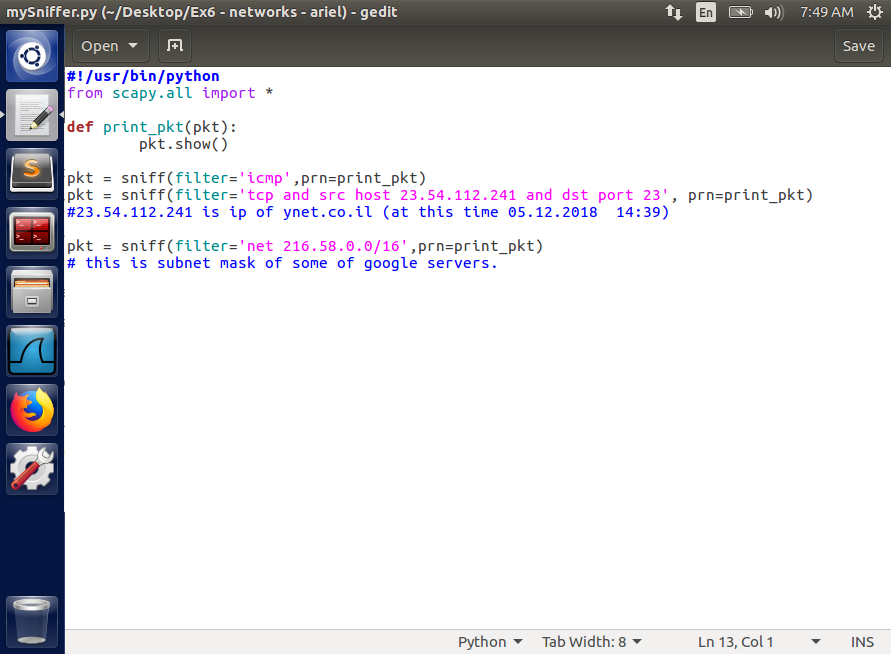
**מגישים:**ליעד כהן 316602630

תימור שרעבי 311508220

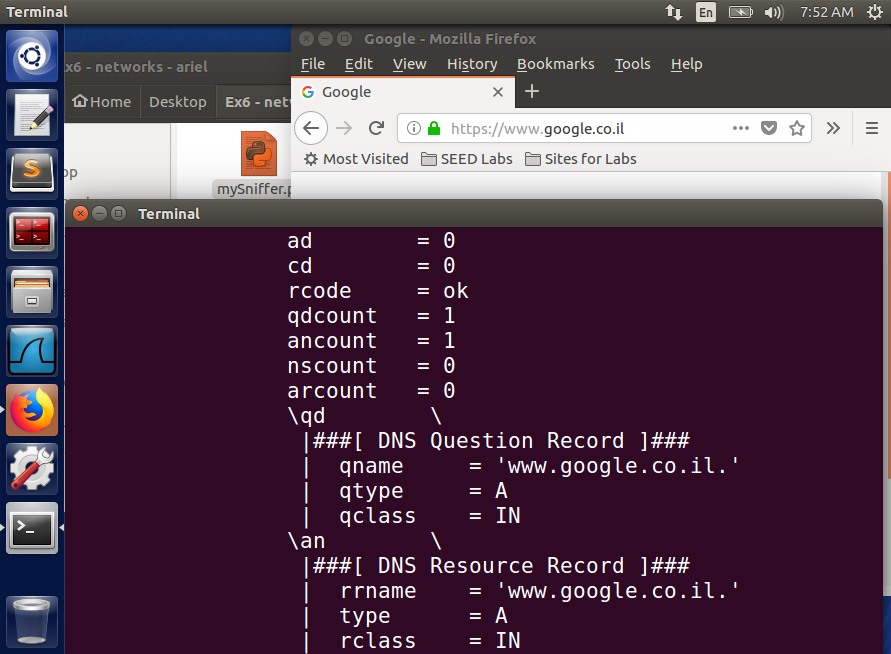
1. התקנו את ה SEEDvm והפעלנו ב virtual box .
   1. **א': Task 1.1A** ריצת קוד הסנפה ללא הרשאות אדמין:   
      קיבלנו תשובה operation not permitted.  
      



ריצת קוד הסנפה עם הרשאות אדמין: הצלחנו לתפוס פאקטה !

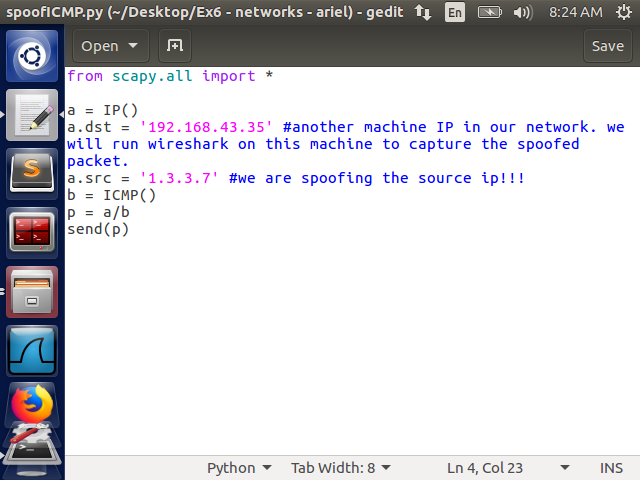
**1.1 ב': Task 1.1B** שינוי קובץ ההספנה שלנו שיכיל את הפילטרים הנוספים (כל אחד בנפרד כפי שנתבקשנו, אבל באותה תכנית).

ריצה של ההסנפה עם הפילטרים הרלוונטים, כניסה לאתר google.com (שהוא חלק מהסאבנט בפילטר השני שרשמנו) ותפיסת הפאקטה.

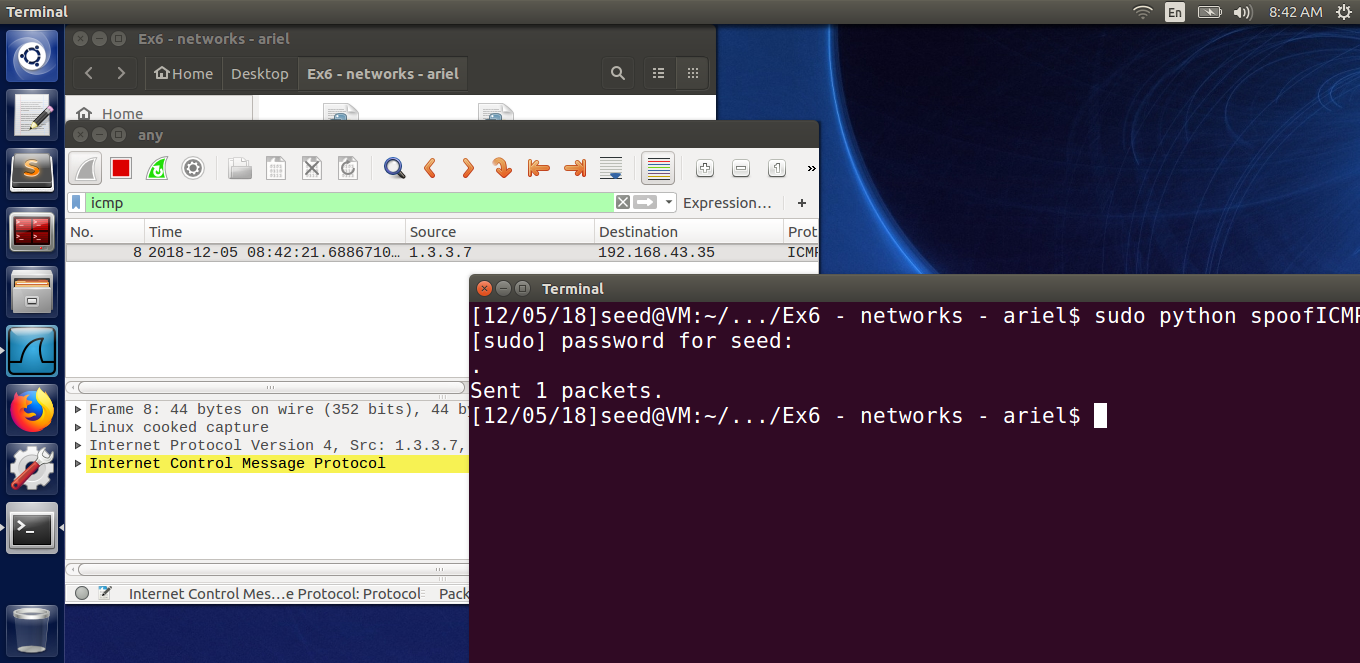


**Task 1.2: Spoofing ICMP Packets**

נשנה את הsrc ip שלנו לאייפי שונה כלשהו, נשנה את ה dst ip לכתובת IP של מחשב ברשת שלנו, ובאותו מחשב ששלחנו נפתח WIRESHARK לראות את הפאקטה שנשלחה שם – ונבדוק מה ה SOURCE שלה!



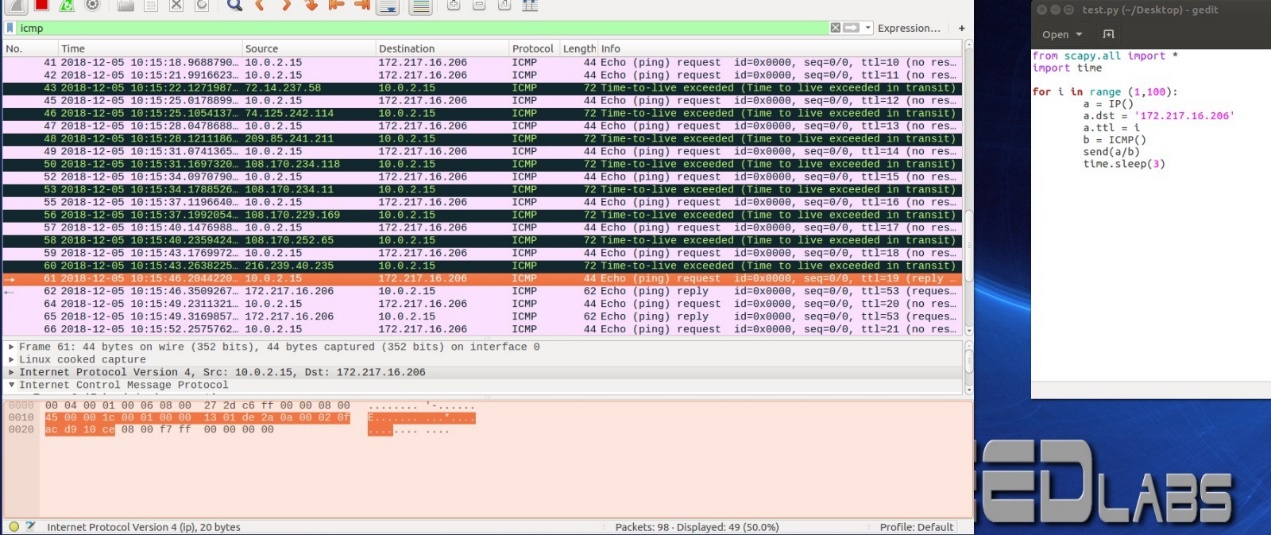
ואכן הSOURCE של הפאקטה הוא 1.3.3.7 כפי שרשמנו.



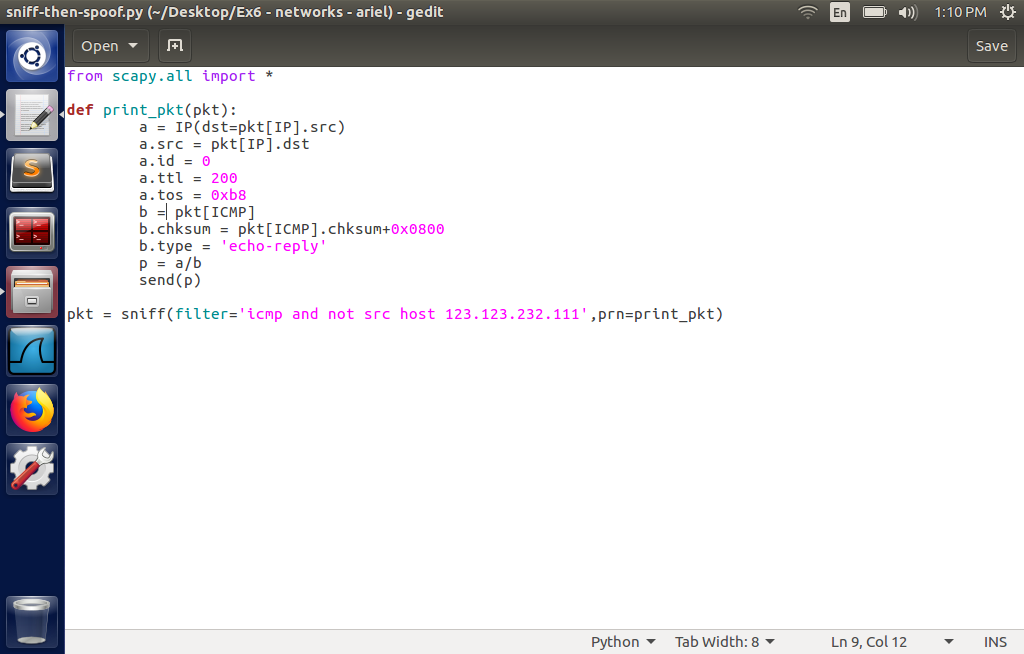
**Task 1.3: Traceroute**

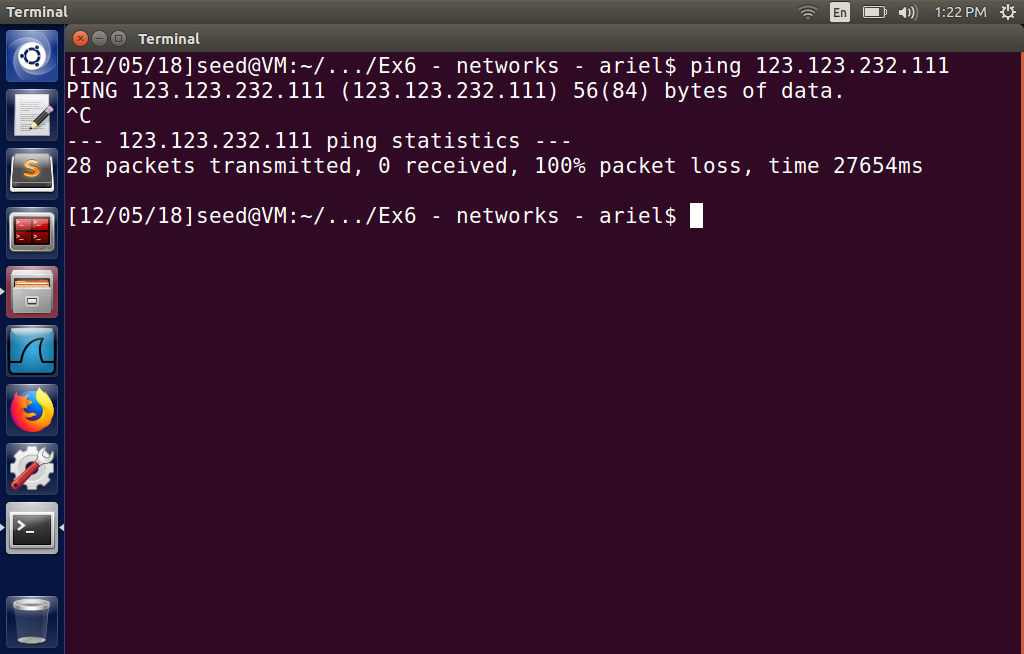
נבדוק כמה ראוטרים 'עומדים' בין ה VM שלנו לבין google.com (216.58.213.110).

רשמנו קוד שירוץ אוטומטי בפייטון.

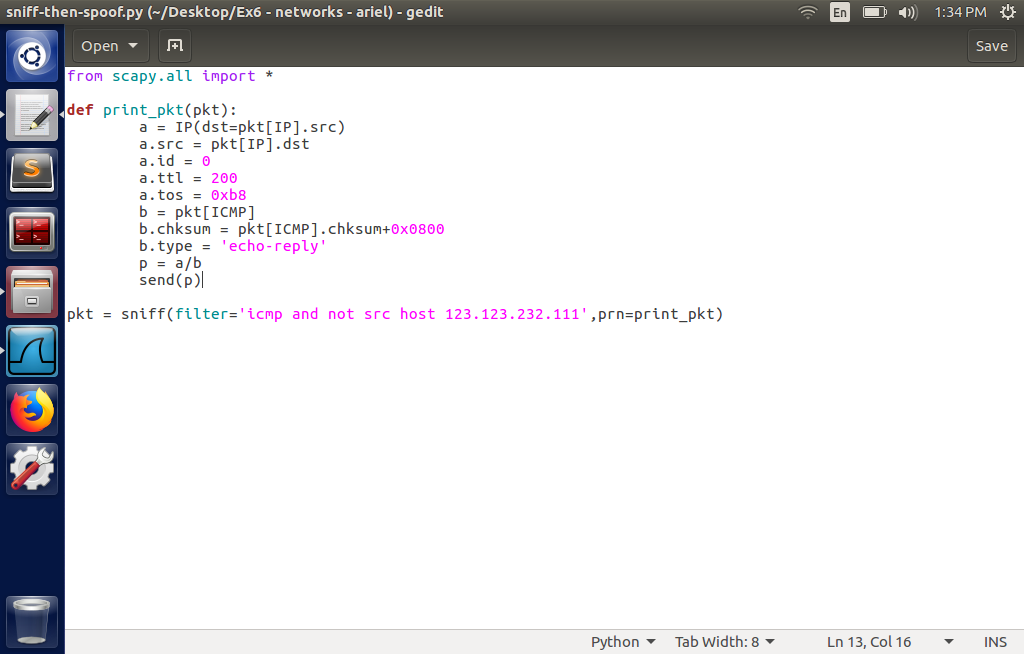
בTTL 19 כבר קיבלנו תשובה. לכן 19 ראוטרים.

**Task 1.4: Sniffing and-then Spoofing**

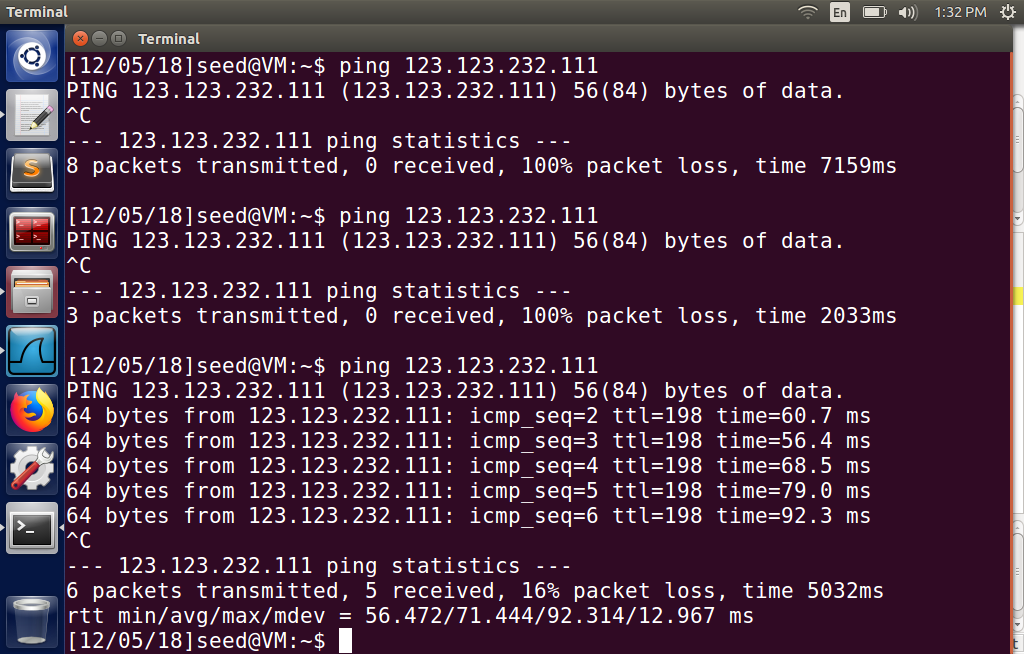
מצאנו אייפי שלא מגיע לשום מקום: 123.123.232.111. הנה : (קוד בעמוד הבא!)



בנינו קוד למכונה 2 שתתפוס פאקטות שנשלחות כ ICMP .   
ברגע שהתקבלה פאקטה , אנחנו נקרא לפונקציה חדשה שבנינו שהיא תבנה פאקטה חדשה- היעד שלה הוא הכתובת מקור של הפאקטה שתפסנו (המקורית),   
הכתובת מקור היא כתובת היעד (שלא באמת קיים).   
אנחנו מעדכנים את ה Checksum והID וה type כדי שהפאקטה תתקבל כ reply לבקשה המקורית. זה הקוד שלנו:



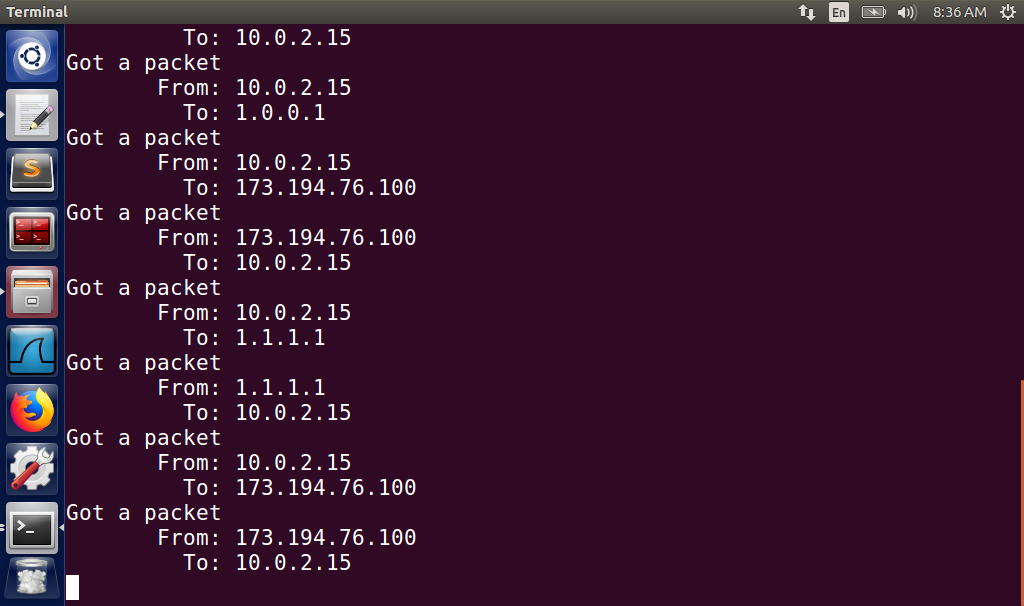
כעת הרצנו את ה sniffer-then-spoof.py במכונה 2 ובמכונה 1 עשינו פינג לכתובת שאין בה אף אחד. 123.123.232.111. אנחנו אמורים לקבל תשובה שהיעד לא נמצא, אבל מכונה 2 עושה SPOOF ומחזירה פאקטה למכונה 1, ככה שמכונה 1 רואה כאילו היעד קיים!!!



**Task 2.1A: Understanding How a Sniffer Works**

הרצנו את הקוד ותפסנו פאקטות:

צריך להראות את **האייפי ליעד** **ואייפי למקור** של הפאקטה שתפסנו. נשתמש בקוד שניתן ב PDF דרך האתר <https://www.handsonsecurity.net/example-code> , ונראה שהקוד עובד ומדפיס לנו את מה שרצינו:



• **Question 1** : רצף הקריאות בספריה pcap הנחוצות להסנפת פקטות:   
ראשית נקרא לפונקציה pcap\_open\_live שתאזין לאינטרפייס מסוים שנגדיר(ארגומנט ראשון), הארגומנט השני- המקסימום בתים שיתפסו על ידי pcap , הארגומנט השלישי הוא האם להפעיל במוד promiscuous או לא (למרות שאם מוגדר ל0 זה אפשרי תחת מקרים ספציפיים), הארגומנט הרביעי הוא זמן time out לקריאה, הארגומנט הרביעי ebuf הוא מצביע מסוג סטרינג שבמידה ויש שגיאה כלשהי אז ירשום לשם את השגיאה.

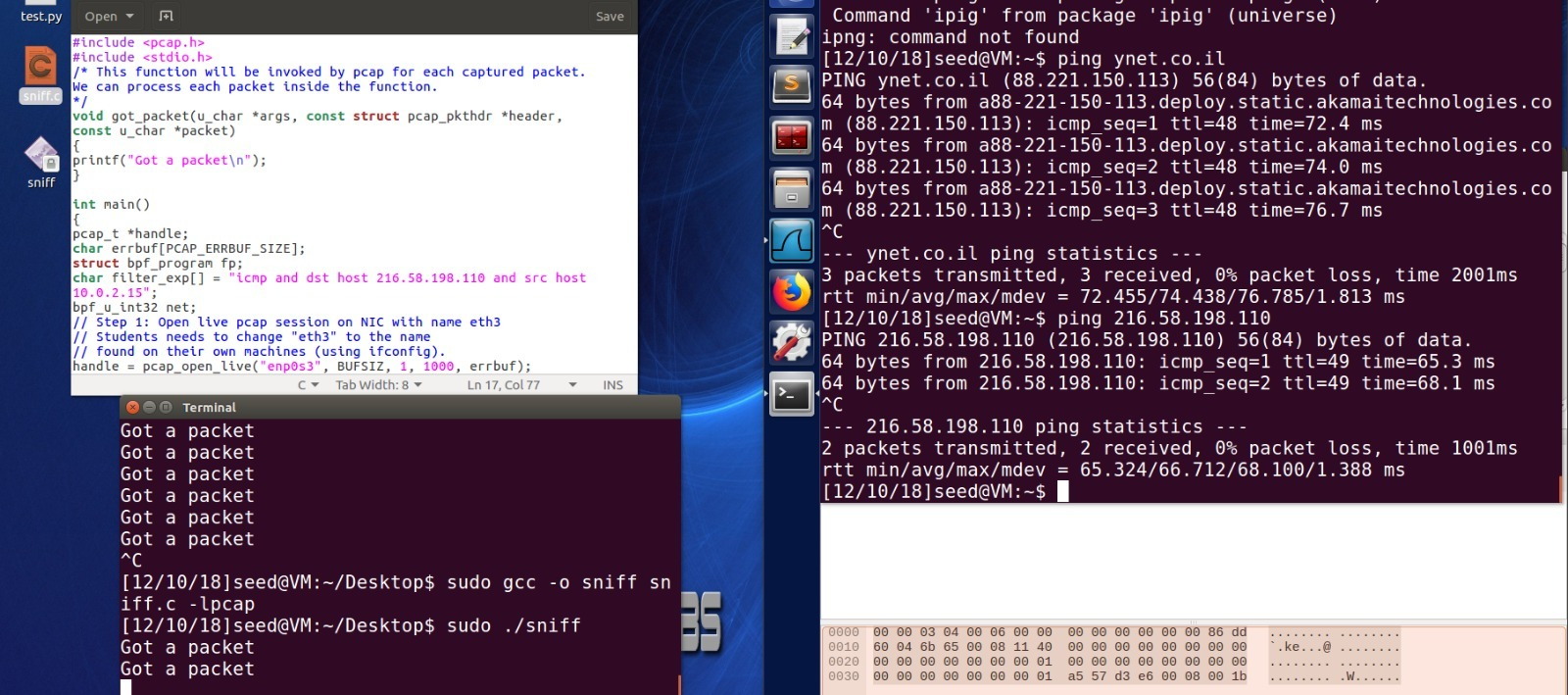
שנית נקרא לפונקציה pcap\_compile, משמש לקימפול הסטרינג לפילטר התכנית. מחזירה 0 אם הצליח, ו Pcap\_Error אם נכשל.  
ואז לפונקציה pcap\_setfilter משמש להגדרת הפילטר של התכנית. מחזיר 0 אם מצליח, ו Pcap\_Error אם נכשל.  
ולפונקציה pcap\_loop מעבדת פקטות שתופסת בלייב.  
ולפונקציה pcap\_close סוגרת את הקובץ לקריאה ומשחררת משאבים.

**• Question 2.:** אנו צריכים להריץ תכנית להסנפה בהרשאות מנהל כי האזנה לאינטרפייס מסוים דורש גישה לאינטרפייס וגישה כזו דורשת הרשאות מנהל.

אם לא נפעיל את התכנית בהרשאות מנהל היא תיפול כבר בשלב ה pcap\_open\_live .

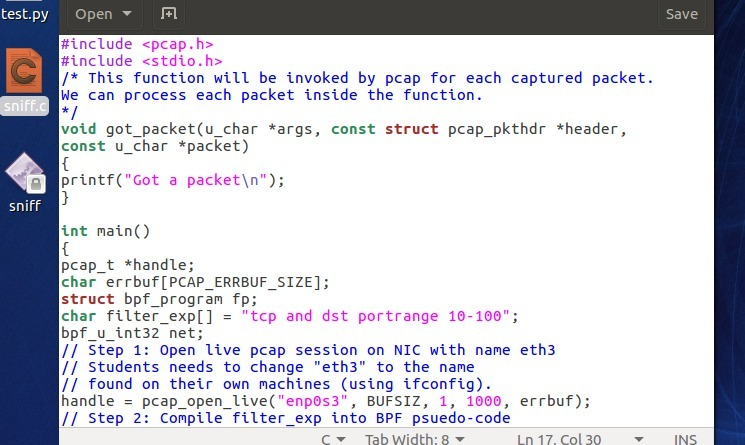
**• Quesiton 3.** נשנה את promiscuous mode ל OFF בעזרת הפונקציה pcap\_open\_live בארגומנט השלישי-נשים שם 0. המשמעות: כעת נאזין לתעבורה שעוברת רק במכונה הוירטואלית הספציפית שהפעלנו עליה את התכנית, ולא על כל התעבורה ברשת.

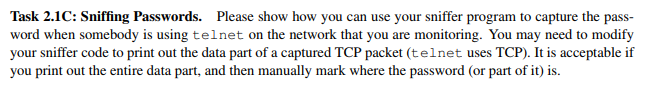
**Task 2.1B: Writing Filters**

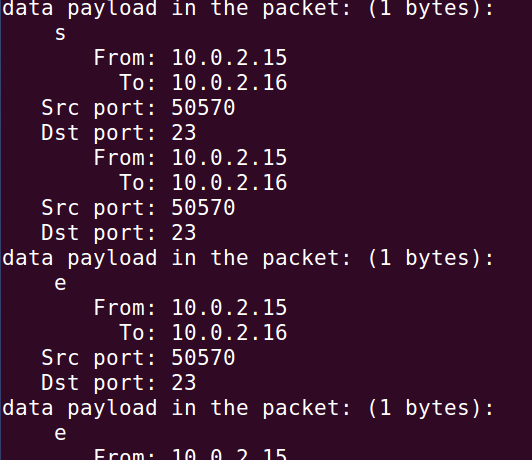
• Capture the ICMP packets between two specific hosts.

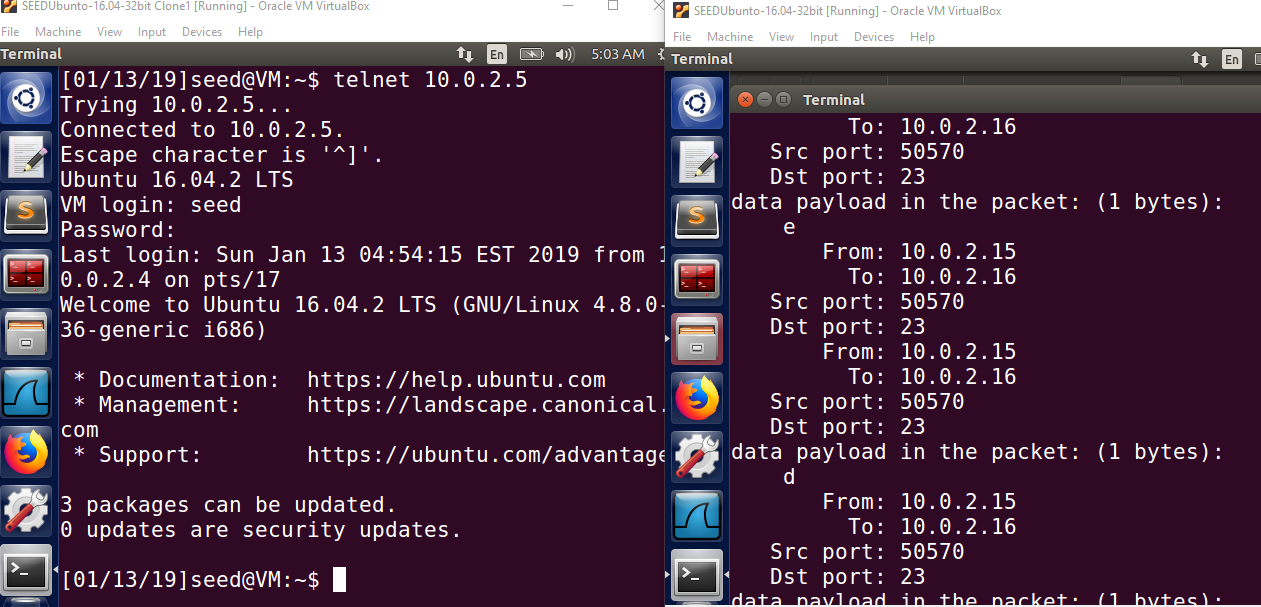
• Capture the TCP packets with a destination port number in the range from 10 to 100.

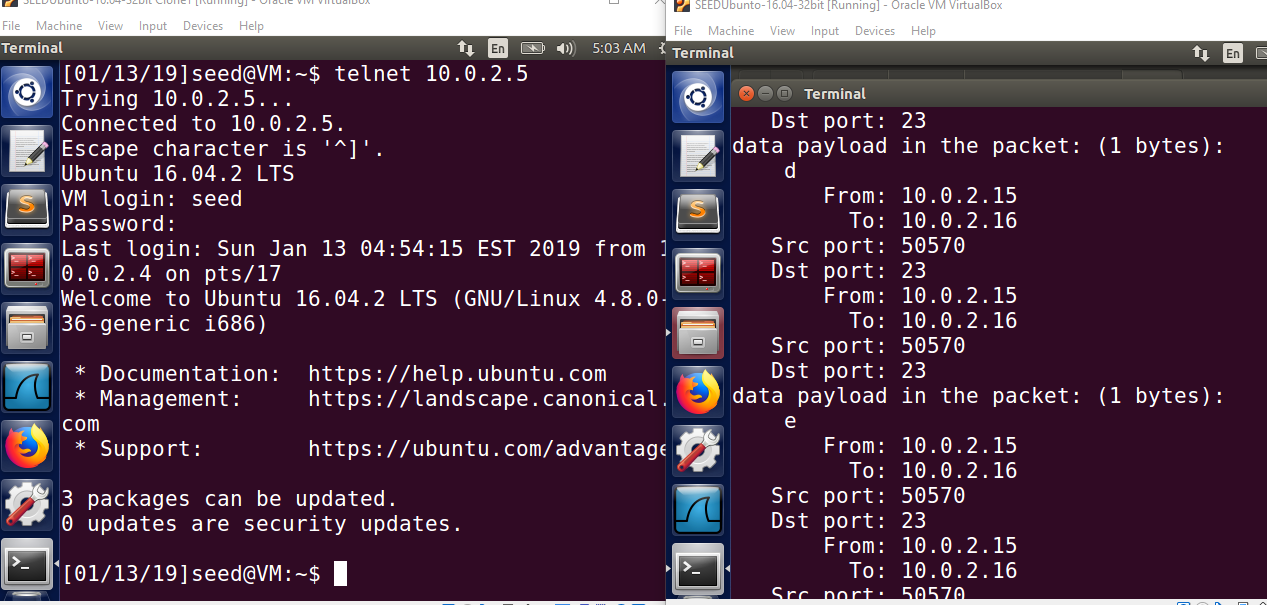
הפילטר שכתבנו:

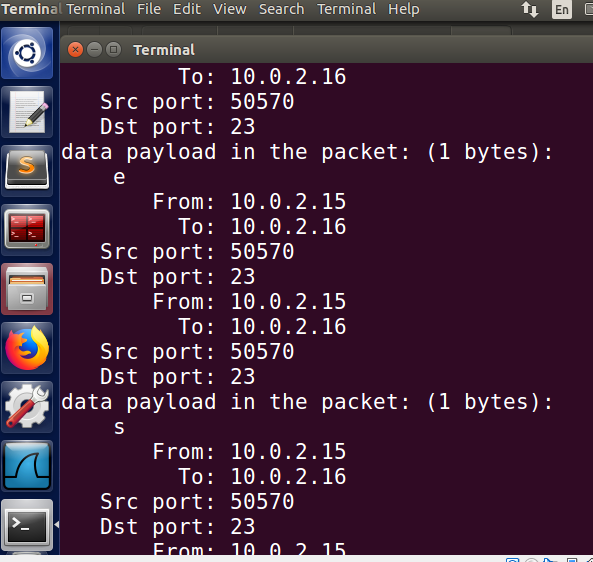








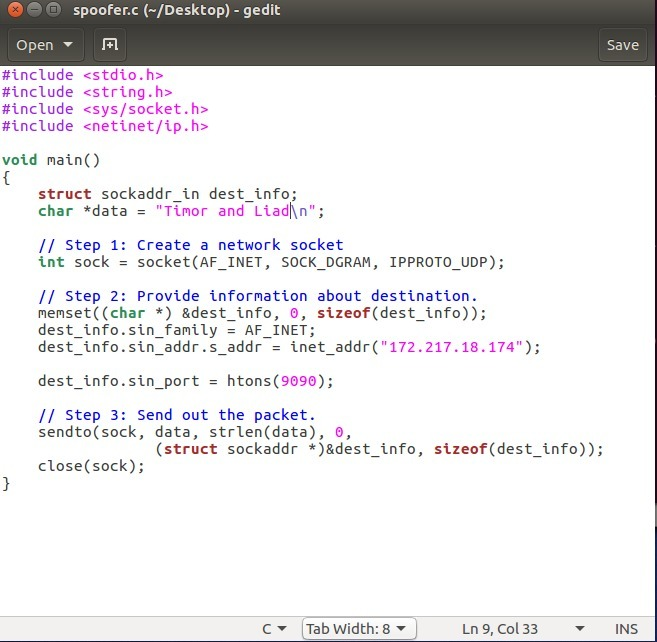
תפיסת הסיסמה:  




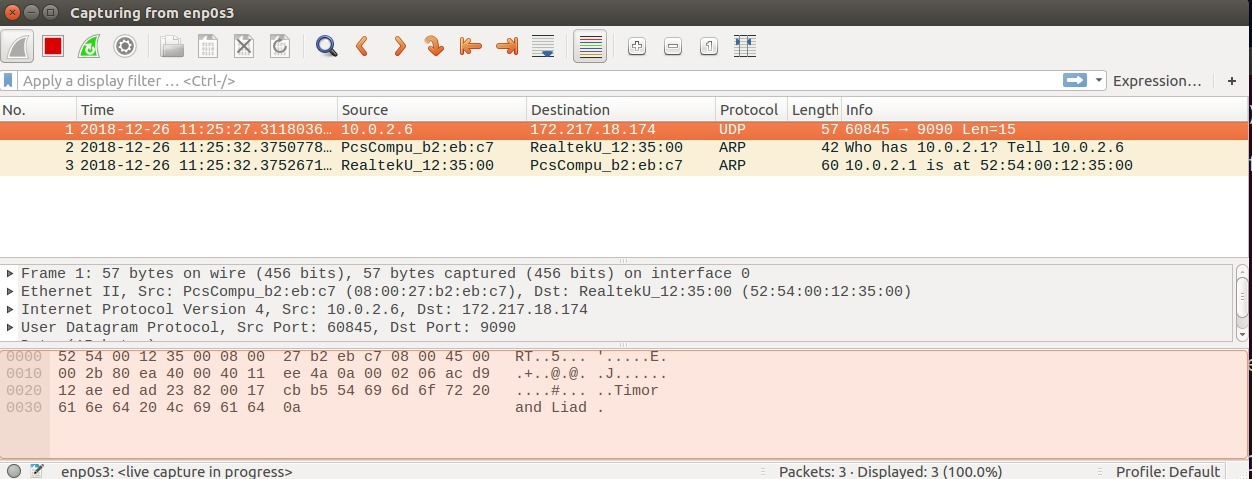
בטרמינל של ההסנפה, המידע נשלח על פורט 23 (שירות telnet) כבית-בית (ניתן לראות את היוזר SEED "נתפס" S ואז E ואז E ואז D . לאחר מכן סיסמה, באותו האופן.



הקוד שלנו ל spoof:

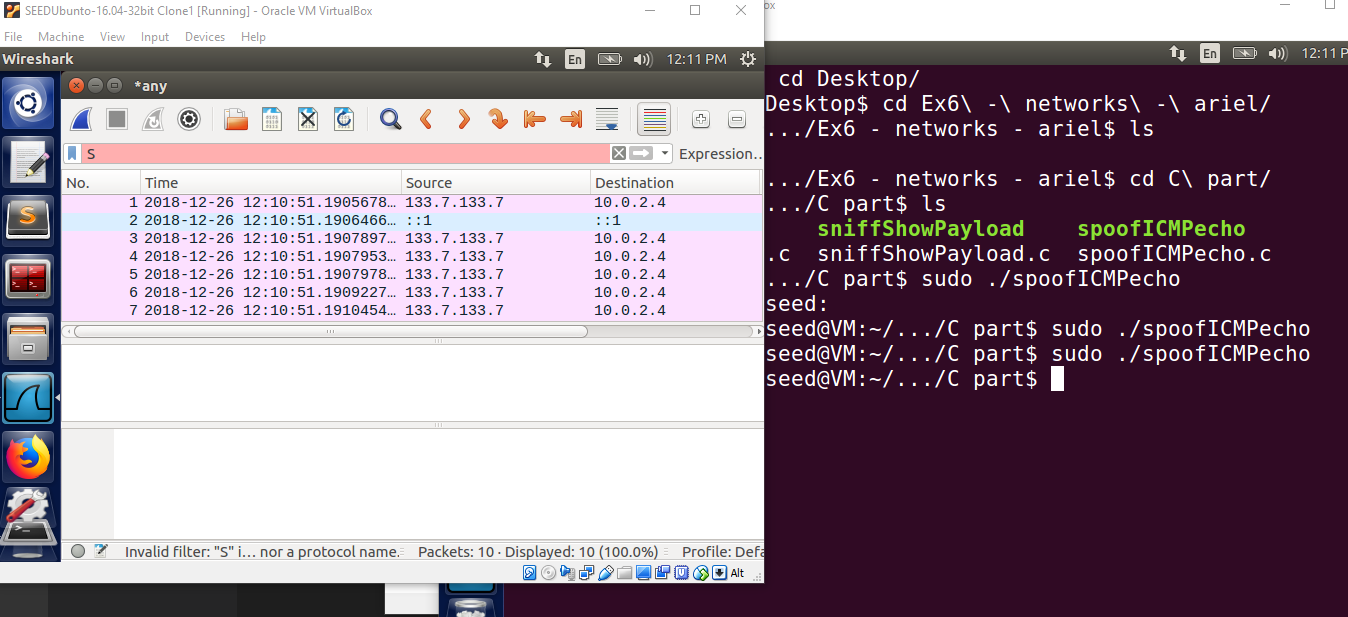


הוכחה:





שליחת הודעות ICMP כאילו אנחנו מאייפי 133.7.133.7 (אבל בעצם שלחנו ממכונה וירטואלית אחרת).





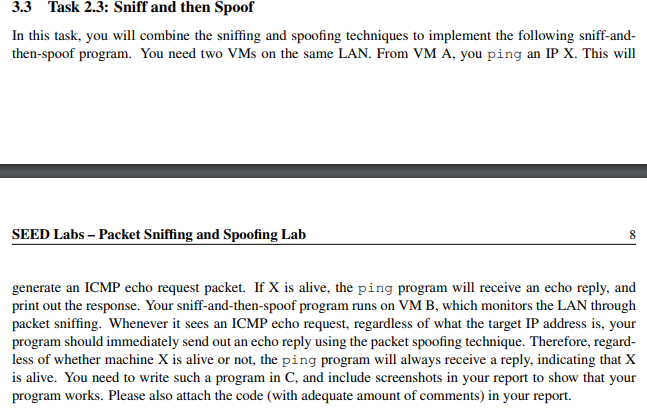
כן, טכנית ניתן , אבל **זה לא ישנה ויחזור לגודל המקורי**.



לא, את זה קובעת מערכת ההפעלה. אין אפילו שדה לקביעת הערך כאשר משתמשים ב raw socket programming .



כדי להריץ תכניות שמשתמשות ב RAW SOCKETS נצטרך הרשאות מנהל אחרת נוכל לעקוף חוקים אחרים ברשת שלא אמורה להיות לנו גישה אליהם, למשל, אנו לא יכולים לאגד תכנית מסוימת לפורט מתחת ל1024 ללא הרשאות מנהל. אולם, אם נוכל להריץ תוכנית כנ"ל ללא הרשאות מנהל, נוכל לקבוע בתכנית לסמלץ סרבר על פורט מתחת ל1024 בעזרת raw sockets.

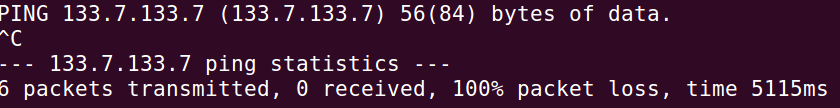


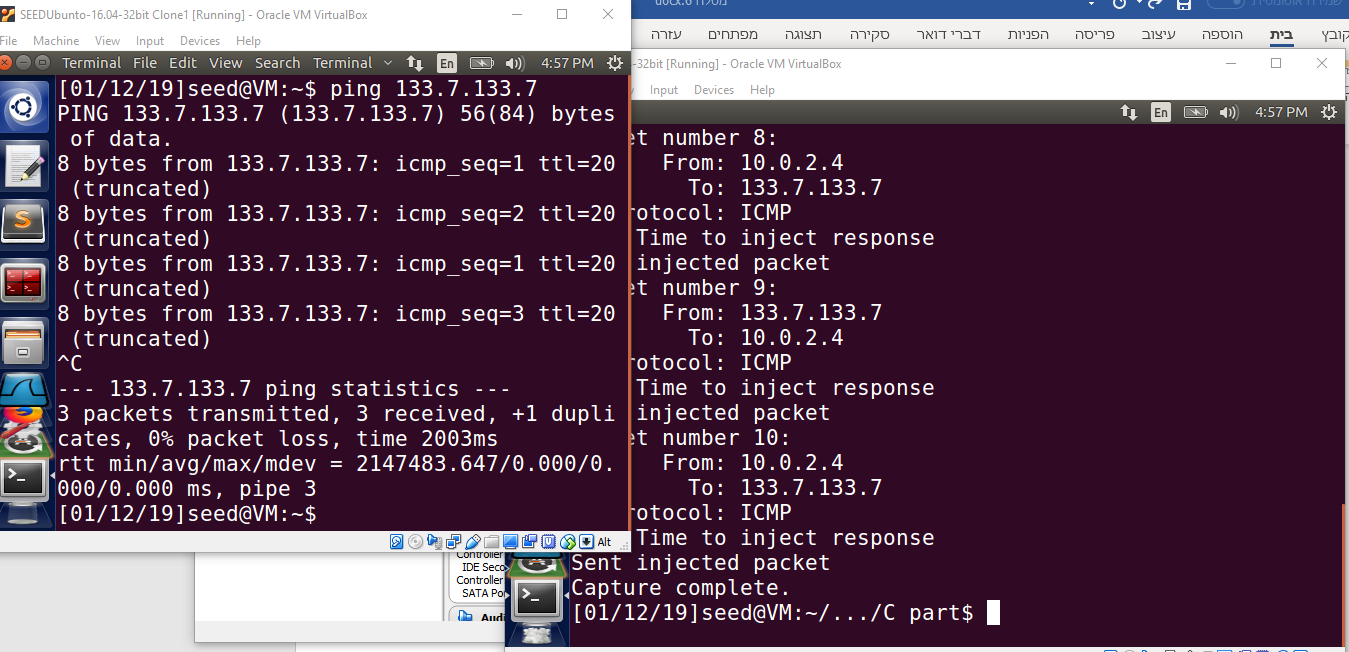
הגדרות:

10.0.2.5 מכונה A. תאזין ותשלח ECHO REPLY מיד כשתשמע.  
10.0.2.4 מכונה B. תשלח פינג לאייפי 133.7.133.7 שלא קיים.

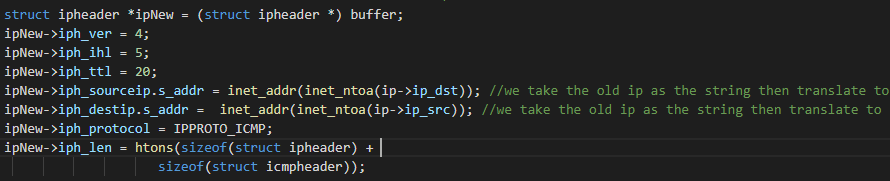
נראה שמכונה B קיבלה REPLY למרות שאייפי 133.7.133.7 לא קיים (לא צריך להחזיר תשובה).

פינג ממכונה B לכתובת 133.7.133.7 הוכחה שהכתובת לא נגישה:

  
ולאחר כתיבת הקוד והפעלה על גבי מכונה A, לאחר מכן ביצוע פינג במכונה B וקבלת הPONG :



בהתחלה יצרנו sniffing-spoofing שיעבוד עבור אייפי ספציפי שבחרנו, ולאחר מכן עדכנו את הקוד שכעת יאזין לכל פינג ברשת ויחזיר תשובה כאילו מ"המקור" .



לכן כעת יעבוד על כל ping לכתובת כלשהי, לפי הדרישה.

קוד לכל שלב במטלת SEED זו מצורף לתיקיית ההגשה.