

מסמך Pdf למטלה 5 ברשתות תקשורת

Task A

ראשית, נענה על השאלה הנשאלת בחלק זה עליה התבקשנו לענות.

תזכורת לשאלה:

Why do you need the root privilege to run a sniffer program? Where does the program fail if it is executed without the root privilege?

תשובה: תוכנית להסנפת פקטות כמו זאת שכתבנו דורשת הרשאות אדמין או root כיוון שעליה להפוך את כרטיס הרשת ל mode promiscuous במצב זה כרטיס הרשת קורא את כל התנועה שהוא מזהה ברשת, ולא רק את מה שממוען אליו. על מנת לשנות את כרטיס הרשת למצב זה עלינו לגשת להגדרות low-level של הרשת ולכן נשתמש ב sudo כלומר הרשאות admin. בנוסף, הסיבה שדרוש הרשאות promiscuous-mode היא כיוון שאנו לא רוצים שכל בן-אדם עם מחשב ו תוכנת wireshark יוכל לראות את התעבורה של הפקטות אצלי במחשב, לכן נידרש להרשאות אלו.

אם לא ניתן הרשאות Admin לתוכנית, התוכנית לא תוכל להעביר את כרטיס הרשת ל promiscuous-mode וכתוצאה מכך התוכנית לא תוכל להסניף פקטות, בנוסף סבירות גבוהה שהתוכנית לא תוכל להסניף פקטות אפילו מרשת ספציפית ללא הרשאות מנהל והתוכנית לא תבצע את מבוקשה.

לאחר שענינו על השאלה, נראה שימוש של ה sniffer ב wireshark ונסביר על דרך ההרצה.

ראשית, כתבנו קובץ Makefile המשמש ל Task A ו Task B כך שעל ידי הפקודה make all יתבצע קמפול של הקבצים.

לאחר שהקבצים יתקמפלו, נריץ את ה sniffer על ידי הרשאות מנהל, כלומר בצורה הבאה.

```
yehonatan@yehonatan-VirtualBox:~/Documents/NetWork5$ sudo ./sniffer
Finding available devices ... Done
Available Devices are :
1. enp0s3 - (null)
2. any - Pseudo-device that captures on all interfaces
3. lo - (null)
4. bluetooth-monitor - Bluetooth Linux Monitor
5. nflog - Linux netfilter log (NFLOG) interface
6. nfqueue - Linux netfilter queue (NFQUEUE) interface
7. dbus-system - D-Bus system bus
8. dbus-session - D-Bus session bus
Enter the number of the device you want to sniff : 1
```

לאחר שנריץ את הפקודה, נקבל בחזרה את המכשירים הפנויים אליהם אנחנו יכולים להקשיב, בתוכנית שכתבנו הסניפר יסיים את פעולתו אם נזין לו מכשיר שלא מקבל פקטות אטרנט, במשימה זאת אנו נרצה להאזין ללופ בק ולכן נבחר באופציה 3 (בחירה באופציה 1 תאפשר לנו להאזין לתעבורת הרשת החיצונית של המכשיר), לכן הזנו 3 (בצילום המסך הזנו 1 ע"מ לבצע בדיקה להסנפה אחרת). לאחר שנזין מכשיר חוקי להאזנה הסניפר יתחיל את פעולתו ויצור קובץ txt שאליו הוא ידפיס את המבוקש ממנו.

כעת ה - sniffer שלנו פועל, הוא יסנן פקטות לפי פרוטוקול Tcp ויכתוב את הפרטים – ההדרים, הדגלים והדאטה שהתבקשנו, כל זה בקובץ שנוצר בתחילת ההרצה. הסניפר יפסיק רק אם נפסיק אותו ידנית.

נצרך תמונה של הקובץ טקסט שנוצר כתוצאה מההרצה של ה sniffer



318813367_
207689647.
txt

הסימן של המנעול מופיע של הקובץ טקסט כיוון שהרצנו את התוכנית עם הרשאות מנהל.

```
1 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 45956, dest_port: 9999, timestamp: 1674064318, total_length: 74, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: }
2 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 9999, dest_port: 45956, timestamp: 1674064318, total_length: 74, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: }
3 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 45956, dest_port: 9999, timestamp: 1674064318, total_length: 66, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: H }
4 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 45956, dest_port: 9999, timestamp: 1674064318, total_length: 753, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: cE1Y4ÿ }
5 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 9999, dest_port: 45956, timestamp: 1674064318, total_length: 66, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: 0 }
6 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 9999, dest_port: 45956, timestamp: 1674064318, total_length: 630, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: cE1Y4ÿ }
7 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 45956, dest_port: 9999, timestamp: 1674064318, total_length: 66, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: }
8 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 45956, dest_port: 9999, timestamp: 1674064318, total_length: 66, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: }
9 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 9999, dest_port: 45956, timestamp: 1674064318, total_length: 66, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: }
10 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 45956, dest_port: 9999, timestamp: 1674064318, total_length: 66, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: }
11 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 45972, dest_port: 9999, timestamp: 1674064319, total_length: 74, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: }
12 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 9999, dest_port: 45972, timestamp: 1674064319, total_length: 74, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: }
13 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 45972, dest_port: 9999, timestamp: 1674064319, total_length: 66, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: H }
14 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 45972, dest_port: 9999, timestamp: 1674064319, total_length: 753, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: cE1Y4ÿ }
15 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 9999, dest_port: 45972, timestamp: 1674064319, total_length: 66, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: 0 }
16 { source_ip: 127.0.0.1, dest_ip: 127.0.0.1, source_port: 9999, dest_port: 45972, timestamp: 1674064319, total_length: 630, cache_flag: 0, steps_flag: 0, type_flag: 0, status_code: 0, cache_control: 0, data: cE1Y4ÿ }
```

התמונה האחרונה שצירפנו זה הפרטים שנתבקשנו להסניף מכל פקטה, ההדורים ולכתוב אותם בקובץ טקסט.

בנוסף נסביר וננתח מה קורה ב wireshark.

*Loopback: lo					
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help					
tcp					
No.	Time	Source	Destination	Protocol	Length Info
515	782367359	127.0.0.1	127.0.0.1	TCP	74 59150 → 9999 [SYN, Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=884308919
615	782384945	127.0.0.1	127.0.0.1	TCP	74 9999 → 59150 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=
715	782393904	127.0.0.1	127.0.0.1	TCP	66 59150 → 9999 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=884308919 TSecr=884308919
815	782612626	127.0.0.1	127.0.0.1	TCP	753 59150 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=687 TSval=884308919 TSecr=884308919
915	782623816	127.0.0.1	127.0.0.1	TCP	66 9999 → 59150 [ACK] Seq=1 Ack=688 Win=64896 Len=0 TSval=884308919 TSecr=884308919
1015	783591676	127.0.0.1	127.0.0.1	TCP	630 9999 → 59150 [PSH, ACK] Seq=1 Ack=688 Win=65536 Len=564 TSval=884308920 TSecr=884308920
1115	783640265	127.0.0.1	127.0.0.1	TCP	66 59150 → 9999 [ACK] Seq=688 Ack=565 Win=65024 Len=0 TSval=884308920 TSecr=884308920
1215	783749904	127.0.0.1	127.0.0.1	TCP	66 59150 → 9999 [FIN, ACK] Seq=688 Ack=565 Win=65536 Len=0 TSval=884308920 TSecr=884308920
1315	783794493	127.0.0.1	127.0.0.1	TCP	66 9999 → 59150 [FIN, ACK] Seq=565 Ack=689 Win=65536 Len=0 TSval=884308920 TSecr=884308920
1415	783798910	127.0.0.1	127.0.0.1	TCP	66 59150 → 9999 [ACK] Seq=689 Ack=566 Win=65536 Len=0 TSval=884308920 TSecr=884308920
44340	343110922	127.0.0.1	127.0.0.1	TCP	74 41810 → 9999 [SYN, Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=884333480
44440	343145086	127.0.0.1	127.0.0.1	TCP	74 9999 → 41810 [SYN, ACK] Seq=0 Ack=1 Win=65493 Len=0 MSS=65495 SACK_PERM=1 TSval=
44540	343154967	127.0.0.1	127.0.0.1	TCP	66 41810 → 9999 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=884333480 TSecr=884333480
44640	343551868	127.0.0.1	127.0.0.1	TCP	753 41810 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=687 TSval=884333480 TSecr=884333480
Frame 5: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface lo, id 0					
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)					
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1					
Transmission Control Protocol, Src Port: 59150, Dst Port: 9999, Seq: 0, Len: 0					

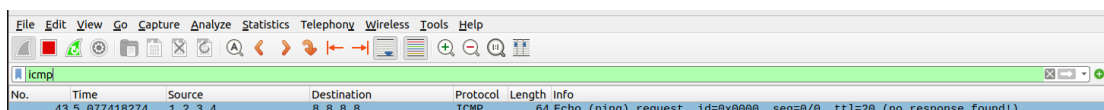
באשר ל wireshark לא נראה את הפעולה של ה sniffer. אלא רק נראה את הפקטות ש ה sniffer הסניף, לכן צירפנו תמונה של wireshark.

Task B

במשימה כתבנו קוד של spoofer.

מטרת ה spoofer היא לבצע מיסוך לכתובת ה קא שממנה אנו שולחים פקטות, ובעצם להתחזות לשולח אחר.

בקוד שלנו נתנו ל spoofer כתובת קא שאינה קיימת – 1.2.3.4 ושלחנו ping כלומר ICMP_REQUEST לכתובת ה קא של גוגל. מכיוון שהכתובת שממנה שלחנו, כאשר מי ששלחנו אליו יחזיר את התשובה, לא נוכל לראות זאת.



No.	Time	Source	Destination	Protocol	Length	Info
43	5.077418274	1.2.3.4	8.8.8.8	ICMP	64	Echo (ping) request id=0x0000, seq=0/0, ttl=20 (no response found!)

בכדי שהקוד שלנו יעשה spoofing לפרוטוקולים אחרים עם שינויים קטנים, נשנה איפה שכתוב בקוד פרוטוקול ICMP נביא לו את הפרוטוקול שאנו רוצים, בנוסף נשנה את ה struct של ה icmp ל struct של הפרוטוקול שנרצה לעשות לו spoofing.

```
// TCP header struct
struct tcphdr {
    u_int16_t source;
    u_int16_t dest;
    u_int32_t seq;
    u_int32_t ack_seq;
    u_int16_t res1:4;
    u_int16_t doff:4;
    u_int16_t fin:1;
    u_int16_t syn:1;
    u_int16_t rst:1;
    u_int16_t psh:1;
    u_int16_t ack:1;
    u_int16_t urg:1;
    u_int16_t res2:2;
    u_int16_t window;
    u_int16_t check;
    u_int16_t urg_ptr;
};
```

במקרה שנרצה לעשות spoofing לפרוטוקול מסוג Tcp אז נשתמש ב struct של ה tcphdr. מצ"ב תמונה.

```
if ((sock = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP)) == -1) {
```

בנוסף, נרצה לשנות היכן שכתוב IPPROTO_ICMP לפרוטוקול של Tcp. בנוסף להסבר על מה זה spoofer ואיך הוא עובד ומטרתו, נשאלנו בחלק זה שתי שאלות.

תזכורת –

Question 1

Can you set the IP packet length field to an arbitrary value, regardless of how big the actual packet

?is

.Question 2

Using the raw socket programming, do you have to calculate the checksum
for the IP header

תשובה:

שאלה 1: ניתן להגדיר את אורך ה IP של החבילה להיות גדול מגודל הפאקטת אך יש לקחת בחשבון כי כאשר עושים זאת עלולים לגרום למספר בעיות, בעיקר במערכת שליחת הפקטת, משום שהרשת במספר תהליכים משתמשת באורך ה IP של החבילה על מנת להחליט על הוראות הניתוב וכן לתהליכים נוספים. בעקבות כך, שינוי זה עלול לגרום להפרעות בשליחת הפקטת ואיננו מומלץ. בנוסף, לעיתים מערכת ההפעלה תתערב ותעדכן את אורך ה IP להיות האורך המתאים בפקטת הנשלחת.

שאלה 2: כאשר יוצרים raw socket על מנת לשלוח פקטות IP, מערכת ההפעלה בדרך כלל תחשב את ה checksum ותכניס או ל IP header באופן אוטומטי, לעומת זאת יש מקרים בהם יש צורך לחשב באופן ידני את ה checksum לטובת ה IP header ולהכניס אותו. מאחר שאנו משתמשים במערכת הפעלה linux או ubuntu וכן כותבים בשפת C, ברצוננו להגיע לשליטה מלאה על הקוד, ועל כל מרכיביו, ועל כן נבצע את החישוב באופן ידני בקוד שלנו על מנת לוודא כי החישוב מתבצע כהלכה והפקטות נשלחות כראוי.

Task C

ראשית, נצרף צילומי מסך לכל חלק במשימה (א', ב' ו ג') ונסביר במהלך צילומי המסך מה אנו רואים ולמה, ובנוסף נסביר לאחר צילומי המסך מה עשינו בחלק זה של המטלה, וכיצד הרצנו את הבדיקות.

בחלק זה כתבנו את הקוד של ה-snoofer בקובץ C הנקרא snoofer-1.c, ע"מ לקמפל את הקובץ של snoofer.c יש להריץ את הפקודה הבאה: gcc -o snoofer snoofer.c -lpcap כלומר יש להריץ עם הספריה lpcap, ולאחר הרצת הפקודה נקבל קובץ מקומפל בשם snoofer-1.

תחילה, נסביר מה עשינו בחלק זה,

ב TaskA כתבנו קוד של sniffer, בנוסף ב TaskB כתבנו קוד של spoofer – על שני הקודים האלו הסברנו לעיל, במשימה זאת כתבנו את קוד ה-snoofer שמטרתו היא לשלב את TaskA ו TaskB לקובץ אחד אשר מבצע את פעולת הספויפינג לכל פקטת icmp שהוא קולט. כלומר, הוא גם עושה סניפינג – הוא יאזין לפקטות ICMP REQUEST ברשת, והוא גם יתחזה ל ip אליו שולחים ICMP REQUEST וישלח ICMP REPLY במקביל אליו (ללא תלות אם ה ip קיים). ועל כן, במשימה זאת התבקשנו להריץ כמה דברים, ובמקביל להריץ את קובץ ה-snoofer שלנו, ולראות שאכן מתקבלות ICMP REPLY אצל המבקש, כלומר מתחזה למי שאני שולח אליו ICMP REQUEST.

בנוסף, בחלק זה למדנו איך להרים מספר דוקרים ואיך להשתמש בהם במקביל, אנו הרמנו דוקרים עם HostA HostB attacker כלומר בעצם ברשת הפנימית שלנו יש שלושה קונטיינרים.

בחלק א של TaskC התבקשנו לשלוח בקשת ping מ HostA ל HostB ובמקביל נרצה להריץ את ה-snoofer שלנו ולראות שאכן הוא מתחזה ל HostB ושולח תשובה במקביל אליו.

בחלק ב של TaskC התבקשנו לשלוח בקשות ping מ HostA ל DNS חיצוני קיים לדוגמא ה DNS של גוגל (8.8.8.8), ובמקביל נריץ את ה-snoofer שלנו ונרצה לראות שהוא מתחזה ל DNS של גוגל והוא שולח ICMP REPLY.

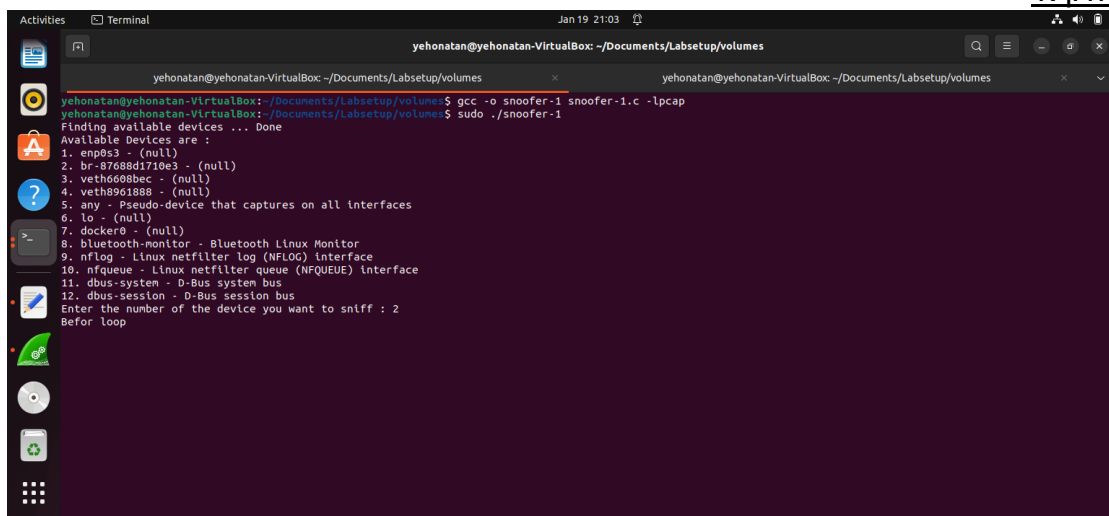
בחלק ג של TaskC התבקשנו לשלוח בקשות ping ל ip מזויף, ובמקביל נריץ את ה-snoofer ונרצה ש ה-snoofer יתחזה ל ip המזויף וישלח לנו ICMP REPLY בחלק זה ידענו כי לא יתקבלו פקטות כלל ולכן ע"מ לוודא שהסנוופר שלנו עובד הרצנו את בקשת הפינג בעזרת הפינג שיצרנו במטלה 4 ובעזרתו קיבלנו גם לראות את קבלת פקטת התשובה שהסנוופר חולל (שימוש בפינג הדיפולטיבי לא היה מאפשר זאת בגלל אורך הדריים שונה בין הפינגים).

הסבר על סביבת הבדיקה.

כאשר ניסינו להשתמש בדוקר שקיבלנו נתקלנו בבעיה שלא הצלחנו להתגבר עליה במשך שעות רבות, לאחר ההקמה של הקונטיינרים מצאנו שרק הקונטיינר בעל הרשאות הרשת host מחובר לאינטרנט (כלומר ה-attacker) ואילו שני הקונטיינרים האחרים לא מחוברים לאינטרנט כלל (HostA ו-HostB), לכן ע"מ לקבל סביבת מעבדה נאמנה לצורך שלנו הרצנו את snoofer.c על ה-VM עצמו כלומר על סביבת האובונטו, באופן זה הוא מתפקד ב host עבור הקונטיינרים הנמצאים בדוקר.

כאשר התבקשנו לשלוח פקטות אל שרת DNS קיים השתמשנו בקונטיינר המחובר לאינטרנט. בהרצות של הרשת הפנימית (HostA to HostB) השתמשנו ב attacker ב HostA ו HostB כמו שהוא היה אמור להיות.

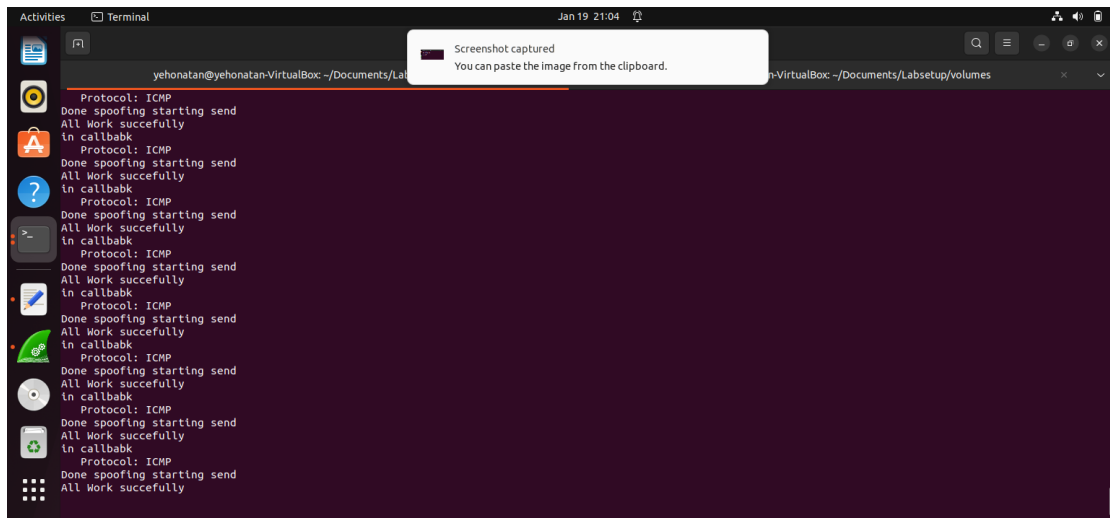
חלק א'



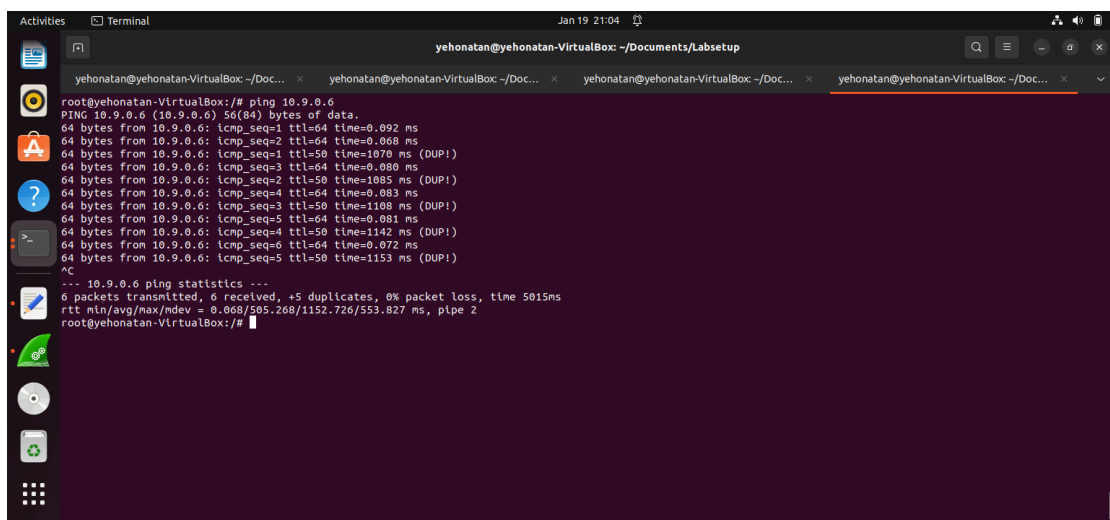
```
yehonatan@yehonatan-VirtualBox: ~/Documents/Labsetup/volumes
yehonatan@yehonatan-VirtualBox:~/Documents/Labsetup/volumes$ gcc -o snoofer-1 snoofer-1.c -lpcap
yehonatan@yehonatan-VirtualBox:~/Documents/Labsetup/volumes$ sudo ./snoofer-1
Finding available devices ... Done
Available Devices are :
1. enp0s3 - (null)
2. br-87688d1710e3 - (null)
3. veth6608b8ec - (null)
4. veth8961888 - (null)
5. any - Pseudo-device that captures on all interfaces
6. lo - (null)
7. docker0 - (null)
8. bluetooth-monitor - Bluetooth Linux Monitor
9. nflog - Linux netfilter log (NFLOG) interface
10. nfqueue - Linux netfilter queue (NFQUEUE) interface
11. dbus-system - D-Bus system bus
12. dbus-session - D-Bus session bus
Enter the number of the device you want to sniff : 2
Before loop
```

ראשית, הרצנו את ה snoofer ברקע, ע"מ שיאזין לפקטות ICMP. לאחר שנקמפל ונריץ את ה snoofer – כמובן שההרצה מתבצעת עם הרשאות מנהל (sudo), תפתח האפשרות לאיזה מכשיר נרצה להאזין, (באופן זה בנינו את התוכנית) בחלק זה נרצה לבצע האזנה למכשיר מספר 2 (הדוקר עצמו) זאת מכיוון שהשליחה של הפקטות מתבצעת ברשת הפנימית.

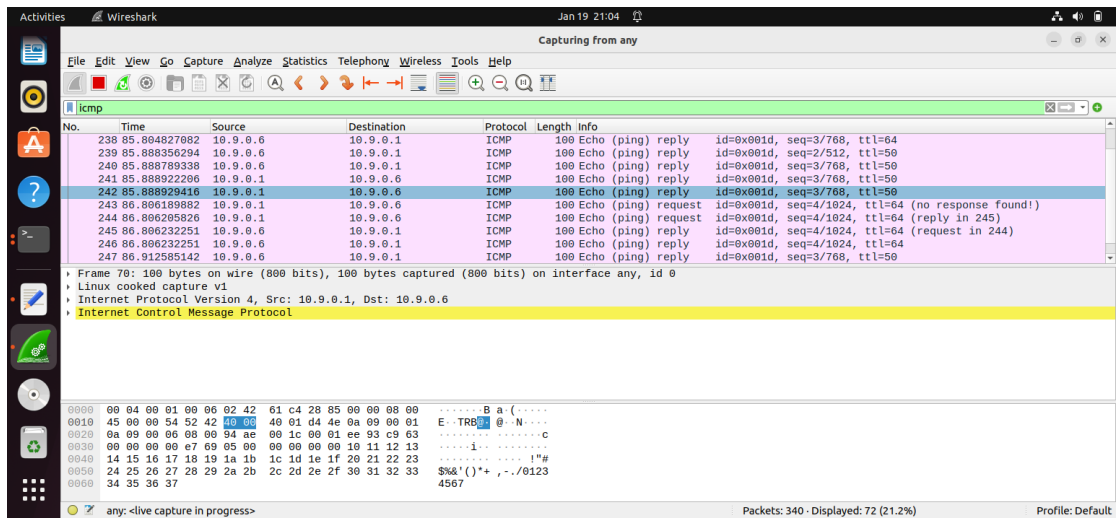
בצילומי המסך הבאים ניתן לראות את פלט הסנופר שלנו אשר מכריז על הגעת של פקטת icmp



בצילום המסך הבא ניתן לראות את טרמינל המבקש, ניתן לשים לב בנקל כי ישנם שש תגובות לבקשות וחמש דופילק אשר נוצרו בעקבות שליחת הסנופר את הריפלי.

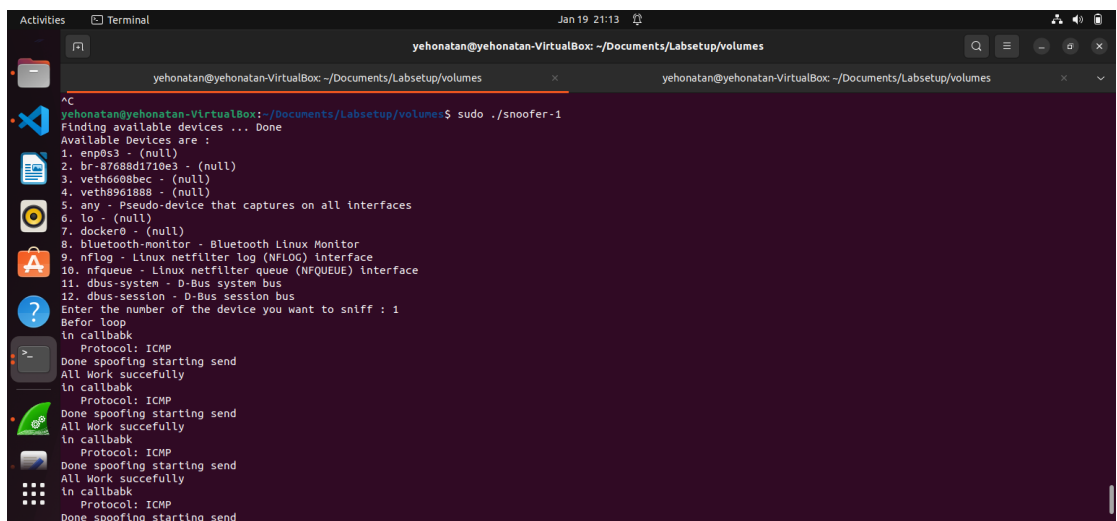


ניתן לראות בוורשארק כי קיימות שתי פקטות תגובה לכל בקשת icmp ניתן להבחין בין התגובות של הסנופר לבין התגובות של HostB לפי ה TTL של כל אחת אנו הגדרנו 50=TTL ואילו המקורי הגדיר TTL=64



חלק ב

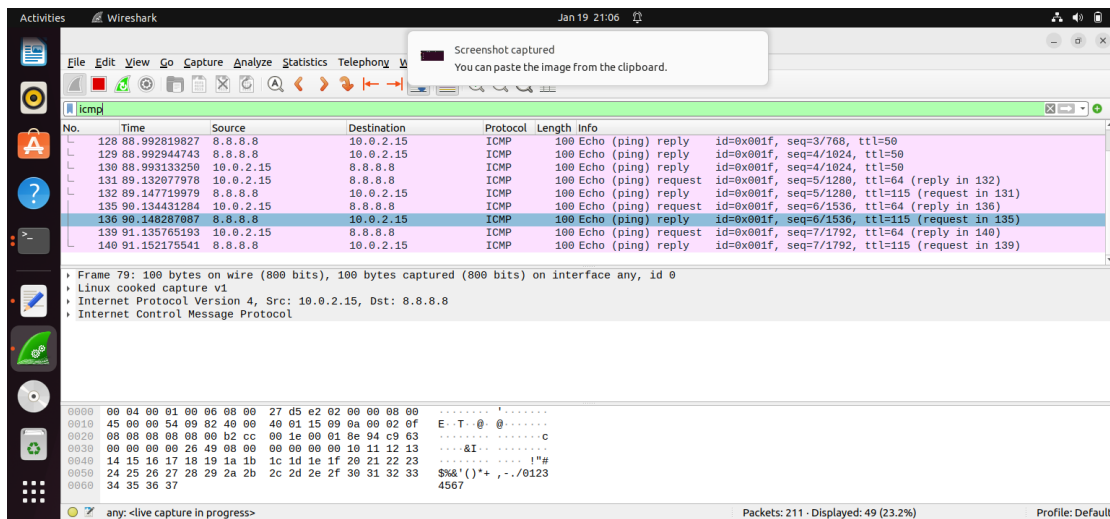
בשלב זה נתבקשנו לשלוח בקשת icmp לשרת קיים, בחרנו בשרת של גוגל, ע"מ להצליח לתפוס את הפקטות העברנו את הסנופר שלנו להאזין לתעבורה העוברת על ה VM (באופן זה קיבלנו את הבקשות מהשרת המהימן) ניתן לראות בצילום המסך את הבחירה בסנופפר ואת ההכרזות שלו על תפיסת פקטת icmp.



נראה את ההרצה מה attacker המתפקד כ HostA


```
Activities Terminal Jan 19 21:06 yehonatan@yehonatan-VirtualBox: ~/Documents/Labsetup
yehonatan@yehonatan-VirtualBox: ~/Documents/Labsetup
64 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=14.8 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=115 time=15.7 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=115 time=14.8 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=115 time=14.0 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=115 time=10.4 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=115 time=16.2 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=115 time=10.8 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=115 time=15.5 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=115 time=15.0 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=115 time=15.7 ms
64 bytes from 8.8.8.8: icmp_seq=12 ttl=115 time=15.1 ms
^C
--- 8.8.8.8 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11019ms
rtt min/avg/max/mdev = 10.819/14.715/16.368/1.512 ms
root@yehonatan-VirtualBox: /# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=115 time=16.6 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=15.0 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=115 time=13.6 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=4 ttl=50 time=1838 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=5 ttl=115 time=14.3 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=50 time=1863 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=7 ttl=115 time=15.7 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=115 time=13.9 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=115 time=16.5 ms
^C
--- 8.8.8.8 ping statistics ---
7 packets transmitted, 7 received, +3 duplicates, 0% packet loss, time 6007ms
rtt min/avg/max/mdev = 13.635/162.154/1862.960/835.731 ms, pipe 2
root@yehonatan-VirtualBox: /#
```

גם בהקלטת הווירשרק המצולמת להלן ניתן להבחין בשתי התגובות המתקבלות לכל בקשה גם כאן ניתן לראות את ההבדל ב TTL בין הפקטות שהסנופר יצר עם 50 לעות המקורי של גוגל עם 115 (ההבדל המשמעותי בין הגודל של ה TTL שגוגל נתנה לפקטות שלה אל מול הפקטות ששלח HostB בחלק הקודם נובע מכמות הנתבים והמרחק בין המבקש לשולח, כאשר המבקש והשולח הם באותה הראשת המספר יהיה נמוך יחסית אל מול שולח מרוחק).



חלק ג

בחלק זה התבקשנו לשלוח בקשת icmp לכתובת קו שאינה קיימת, דבר שבאופן ברור לא אמור להניב תגובת icmp אך בעקבות כך שהסנופר שלנו לא מתוכנת לבדוק אם ה קו שאליו נשלחה הבקשה הינו קו פעיל כאשר הרצנו את הבקשה הסנופר החזיר תגובת icmp וקיבלנו תגובה למרות שהיא לא הייתה אמורה להגיע ברשת ללא סנופר.

נבצע האזנה על כלל הפקטות העוברות ב VM על מנת שנראה אם מתבצעת בקשה ונוכל לבצע שליחת תגובה.

```
Activities Terminal Jan 19 21:13 yehonatan@yehonatan-VirtualBox: ~/Documents/Labsetup/volumes
yehonatan@yehonatan-VirtualBox: ~/Documents/Labsetup/volumes
yehonatan@yehonatan-VirtualBox:~/Documents/Labsetup/volumes$ sudo ./snoofer-1
^C
Finding available devices ... Done
Available Devices are :
1. enp0s3 - (null)
2. br-87688d1710e3 - (null)
3. veth6d08bec - (null)
4. veth8961888 - (null)
5. any - Pseudo-device that captures on all interfaces
6. lo - (null)
7. docker0 - (null)
8. bluetooth-monitor - Bluetooth Linux Monitor
9. nflog - Linux netfilter log (NFLOG) interface
10. nfqueue - Linux netfilter queue (NFQUEUE) interface
11. dbus-system - D-Bus system bus
12. dbus-session - D-Bus session bus
Enter the number of the device you want to sniff : 1
Before loop
in callback
Protocol: ICMP
Done spoofing starting send
All Work successfully
in callback
Protocol: ICMP
Done spoofing starting send
All Work successfully
in callback
Protocol: ICMP
Done spoofing starting send
All Work successfully
in callback
Protocol: ICMP
Done spoofing starting send
```

ניתן לראות בצילום המסך להלן את תגובת קו 1.2.3.4 למרות שאין קו פעל כזה. תגובות אלו הן תגובות אשר חולל הסנופר.

```
Activities Terminal Jan 19 21:13 yehonatan@yehonatan-VirtualBox: ~/Documents/Labsetup
yehonatan@yehonatan-VirtualBox: ~/Documents/Labsetup
yehonatan@yehonatan-VirtualBox:~/Documents/Labsetup$ ./ping 1.2.3.4
String: 1.2.3.4
1.2.3.4: seq=0 time=501.581970ms
1.2.3.4: seq=1 time=21.132000ms
1.2.3.4: seq=2 time=23.697001ms
1.2.3.4: seq=3 time=23.650000ms
1.2.3.4: seq=4 time=33.894001ms
1.2.3.4: seq=5 time=12.368000ms
1.2.3.4: seq=6 time=35.205002ms
1.2.3.4: seq=7 time=11.134000ms
1.2.3.4: seq=8 time=35.150002ms
1.2.3.4: seq=9 time=12.584000ms
1.2.3.4: seq=10 time=33.465000ms
1.2.3.4: seq=11 time=16.597000ms
1.2.3.4: seq=12 time=19.432983ms
^C
root@yehonatan-VirtualBox:~/Documents/Labsetup$
```

בהקלטת הווירשק ניתן לראות כי אין עוד פקטות תגובה והפקטות היחידות שנמצאות מלבד פקטות הבקשה הן הפקטות שהסנופר חולל.

Activities Wireshark Jan 19 21:16 TaskC-3.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

icmp

No.	Time	Source	Destination	Protocol	Length	Info
131	50.410767109	10.0.2.15	1.2.3.4	ICMP	63	Echo (ping) request id=0x1200, seq=0/0, ttl=64 (no response found!)
132	50.977125960	1.2.3.4	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=0/0, ttl=50
133	51.978816460	10.0.2.15	1.2.3.4	ICMP	63	Echo (ping) request id=0x1200, seq=256/1, ttl=64 (no response found!)
134	51.999715690	1.2.3.4	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=256/1, ttl=50
135	53.000252080	10.0.2.15	1.2.3.4	ICMP	63	Echo (ping) request id=0x1200, seq=512/2, ttl=64 (no response found!)
136	53.023771282	1.2.3.4	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=512/2, ttl=50
137	54.024331302	10.0.2.15	1.2.3.4	ICMP	63	Echo (ping) request id=0x1200, seq=768/3, ttl=64 (no response found!)
138	54.047853355	1.2.3.4	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=768/3, ttl=50
139	55.048781439	10.0.2.15	1.2.3.4	ICMP	63	Echo (ping) request id=0x1200, seq=1024/4, ttl=64 (no response found!)
140	55.082205099	1.2.3.4	10.0.2.15	ICMP	63	Echo (ping) reply id=0x1200, seq=1024/4, ttl=50

Frame 131: 63 bytes on wire (504 bits), 63 bytes captured (504 bits) on interface any, id 0

- Linux cooked capture v1
- Internet Protocol Version 4, Src: 10.0.2.15, Dst: 1.2.3.4
- Internet Control Message Protocol

0000 00 04 00 01 00 06 00 00 27 d5 e2 02 00 00 00 00
0010 45 00 00 2f c3 36 40 00 40 01 67 83 0a 00 02 0f E.../.00.g....
0020 01 02 03 04 08 00 ae 36 12 00 00 00 54 68 69 736...This
0030 20 69 73 20 74 68 65 20 70 09 6e 67 2e 0a 00 .. is the ping...

Internet Control Message Protocol: Protocol Packets: 245 - Displayed: 26 (10.6%) Profile: Default