

מטלה 5 רשתות תקשורת

מה זה sniffer?

רחרחן (sniffer) הוא תוכנה המאפשרת להאזין ולתעד תקשורת מחשבים העוברת בנקודה כלשהי ברשת. הרחרחן קולט את חבילות המידע היוצרות את התקשורת (דרך כרטיס הרשת במחשב למשל), מנתח אותן בהתאם ל-RFC הרלוונטי ומציג אותן למשתמש לאחר הניתוח.

שימושים:

- הבנת אופן הפעולה של תקשורת מחשבים באמצעות ניתוח תוצאות הרחרחן.
- ניתוח בעיות תקשורת ברשת המחשבים.
- זיהוי ניסיונות חדירה לרשת על מנת לחסום אותם.
- ניפוי שגיאות באפליקציות רשת דוגמת שרת-לקוח.

Task A

איך להריץ את התוכנה?

```
nitay@ubuntu:~/Desktop/Networks EX/Networks-Matala5$ make runsn
sudo ./Sniffer
[sudo] password for nitay:
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 : █
```

אנו נקליד make runsn, נכניס סיסמא ונבחר את המכשיר שנרצה להאזין לתקשורת העוברת דרכו.

איזה חבילות ניתן להסניף?

```
switch (iph->protocol) // Check the Protocol and do accordingly...
{
case 0: // ICMP Protocol
    ++icmp;
    print_icmp_packet(packet, size);
    break;

case 1: // ICMP Protocol
    ++icmp;
    print_icmp_packet(packet, size);
    break;

case 6: // TCP Protocol
    ++tcp;
    print_tcp_packet(packet, size, header);
    break;
```

התוכנה תזהה חבילות של פרוטוקול ICMP ו-TCP.

כיצד ישמר המידע של החבילות?

```
*****TCP Packet*****
|-Packet No.      : 1
|-Source IP       : 127.0.0.1
|-Destination IP  : 127.0.0.1
|-Source Port     : 37202
|-Destination Port : 9999
|-Timestamp      : Wed Jan 18 08:00:11 2023

|-Total_length    : 2983
|-C_flag          : 0
|-S_flag          : 0
|-T_flag          : 0
|-Status_code     : 99
|-Cache_control   : 532

| | | | | DATA
IP Header
 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00 .....E.
 00 3C 5C BE .....<\.

TCP Header
 40 00 40 06 DF FB 7F 00 00 01 7F 00 00 01 91 52 @.@...0x00...0x00....R
 27 0F D0 5F E0 F1 00 00 00 00 A0 02 FF D7 FE 30 '._.....0
 00 00 02 04 FF D7 04 02 .....

Data Payload
```

אנו שומרים בקובץ טקסט את כתובות ה-IP ו-Port של כל צד. בנוסף, את ה-Unix Timestamp, גודל החבילה, את הדגלים, סטטוס החבילה, מידע מהמטמון ומידע כולל של החבילה.

חסרונות ה-Sniffer

- (1) רחרחן יכול להסניף רק מידע של תקשורת בתוך רשת משנה נתונה. לכן לא ניתן להסניף מידע שעובר ברשת חיצונית שאינך מחובר אליה.
- (2) עליך לקבל הרשאה של מנהל המחשב כדי להפעיל אותו.

למה צריך הרשאת מנהל (root) ומה יקרה אם נפעיל בלעדיו?

כשאנחנו מפעילים רחרחן, אנו יוצרים raw socket ומשנים את מתאם הרשת למצב promiscuous, ולשם כך דרוש הרשאות root. אם לא, כל משתמש יוכל לשלוט באופן מלא במתאם הרשת ולראות את התעבורה המלאה על אותו בקר, כולל כל התעבורה של משתמשים אחרים. לכן, כדי לא לחשוף מידע רגיש אנו מגבילים את הגישה הזו. אם ננסה להפעילו ללא הרשאה זו אנו נקבל שגיאה ב-

```
handle = pcap_open_live ("eth3", BUFSIZ, 1, 1000, errbuf);
```

```
Couldn't open device: lo.
```

```
nitay@ubuntu:~/Desktop/Networks EX/Networks-Matala5$
```

```
// Step 1: Open live pcap session on NIC
handle = pcap_open_live(devname, BUFSIZ, 1, 1000, errbuf);
if (handle == NULL)
{
    fprintf(stderr, "Couldn't open device: %s.\n", devname);
    return (2);
}
```

יצירת מסננים (Task2.1B)

הוספנו ל-sniffer שלנו כמה מסננים עבור מצבים בהם נרצה לקבל לזהות חבילות מסוימות.

```
//#define FILTER ""  
// #define FILTER "tcp portrange 9998-9999"  
// #define FILTER "tcp dst portrange 10-100"  
#define FILTER "icmp and host 8.8.8.8 and host 10.0.2.15"
```

למשל כדי לסנן חבילות של פרוטוקול ICMP בין שני מחשבים, למשל בין google והמחשב שלנו, ניתן להשתמש במסנן שבתמונה.

Wireshark:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
2	0.006321728	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
3	1.000675816	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
4	1.007642197	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
5	2.001903576	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
6	2.008661541	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
7	3.004503861	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
8	3.011791236	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
9	4.007238466	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
10	4.014345582	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply

ה-sniffer שלנו:

```
*****ICMP Packet*****  
|-Source IP      : 10.0.2.15  
|-Destination IP : 8.8.8.8  
|-Type : 8 (ICMP Echo Request)  
|-Code : 0  
|-Checksum : 52540  
|-ID       : 2  
|-Sequence : 1  
  
DATA  
IP Header  
 52 54 00 12 35 02 08 00 27 15 5A A0 08 00 45 00  
 00 54 9E 74  
ICMP Header  
 40 00 40 01 80 16 0A 00  
Data Payload  
 B7 D2 C7 63 00 00 00 00 E1 B6 0B 00 00 00 00 00  
 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  
 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F  
 30 31 32 33 34 35 36 37  
*****ICMP Packet*****  
RT..5...'.Z...E.  
.T.t  
@.@.0000...  
...C.....  
.....  
!"#$%&'()*+,-./  
01234567
```

(ראה קובץ pcap וקובץ טקסט)

בנוסף, ניתן לסנן חבילות בפרוטוקול TCP שנשלחות ב-Port בין 10 ל-100. למשל, כאשר אנו גולשים באינטרנט יש לחבילות פרוטוקול 80.

```
//#define FILTER ""
// #define FILTER "tcp portrange 9998-9999"
#define FILTER "tcp dst portrange 10-100"
//#define FILTER "icmp and host 8.8.8.8 and host 10.0.2.15"
```

:Wireshark

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	34.107.221.82	TCP	74	46170 → 80 [SYN]
2	0.006891684	10.0.2.15	34.107.221.82	TCP	54	46170 → 80 [ACK]
3	0.007226378	10.0.2.15	34.107.221.82	HTTP	355	GET /canonical.ht
4	0.015182658	10.0.2.15	34.107.221.82	TCP	54	46170 → 80 [ACK]
5	0.041366431	10.0.2.15	34.107.221.82	TCP	74	46180 → 80 [SYN]
6	0.047857998	10.0.2.15	34.107.221.82	TCP	54	46180 → 80 [ACK]
7	0.047966164	10.0.2.15	34.107.221.82	HTTP	357	GET /success.txt?
8	0.054732540	10.0.2.15	34.107.221.82	TCP	54	46180 → 80 [ACK]
9	0.115354428	10.0.2.15	82.166.201.162	TCP	74	39122 → 80 [SYN]
10	0.121608742	10.0.2.15	82.166.201.162	TCP	54	39122 → 80 [ACK]
11	0.121706740	10.0.2.15	82.166.201.162	OCSP	477	Request
12	0.129360011	10.0.2.15	82.166.201.162	TCP	54	39122 → 80 [ACK]
13	0.219310739	10.0.2.15	82.166.201.162	OCSP	477	Request
14	0.226428999	10.0.2.15	82.166.201.162	TCP	54	39122 → 80 [ACK]
15	0.596525258	10.0.2.15	93.184.220.29	TCP	74	47380 → 80 [SYN]
16	0.653011480	10.0.2.15	93.184.220.29	TCP	54	47380 → 80 [ACK]
17	0.653129189	10.0.2.15	93.184.220.29	OCSP	478	Request
18	0.708900148	10.0.2.15	93.184.220.29	TCP	54	47380 → 80 [ACK]
19	1.295630961	10.0.2.15	93.184.220.29	OCSP	478	Request
20	1.349426502	10.0.2.15	93.184.220.29	TCP	54	47380 → 80 [ACK]
21	7.297824130	10.0.2.15	93.184.220.29	TCP	54	47380 → 80 [FIN,
22	7.298175884	10.0.2.15	82.166.201.162	TCP	54	39122 → 80 [FIN,
23	7.298242154	10.0.2.15	34.107.221.82	TCP	54	46180 → 80 [FIN,
24	7.298601405	10.0.2.15	34.107.221.82	TCP	54	46170 → 80 [FIN,
25	7.306743512	10.0.2.15	82.166.201.162	TCP	54	39122 → 80 [ACK]
26	7.306755833	10.0.2.15	34.107.221.82	TCP	54	46180 → 80 [ACK]
27	7.306767922	10.0.2.15	34.107.221.82	TCP	54	46170 → 80 [ACK]
28	7.359273477	10.0.2.15	93.184.220.29	TCP	54	47380 → 80 [ACK]

ה-sniffer שלנו:

```
*****TCP Packet*****
1  | -Packet No.      : 1
2  |
3  | -Source IP       : 10.0.2.15
4  | -Destination IP  : 34.107.221.82
5  | -Source Port     : 46170
6  | -Destination Port: 80
7  | -Timestamp      : Wed Jan 18 11:21:27 2023
8  |
9  | -Total_length    : 14294
10 | -C_flag          : 0
11 | -S_flag          : 0
12 | -T_flag          : 0
13 | -Status_code     : 99
14 | -Cache_control   : 55140
15 |
16 | | | | | DATA
17 | IP Header
18 | 52 54 00 12 35 02 08 00 27 15 5A A0 08 00 45 00      RT...'.Z...E.
19 | 00 3C 43 4E                                           .<CN
20 | TCP Header
21 | 40 00 40 06 EB A1 0A 00 02 0F 22 6B DD 52 B4 5A      @.@....."k.R.Z
22 | 00 50 D1 4B 45 37 00 00 00 00 A0 02 FA F0 0B FB      .P.KE7.....
23 | 00 00 02 04 05 B4 04 02                               .....
24 | Data Payload
25 | *****TCP Packet*****
```

(ראה קובץ pcap וקובץ טקסט)

הסנפת סיסמאות (Task2.1C)

ע"י הסנפת חבילות TCP במכשיר loopback לאחר שהתחברנו למחשב שלנו והקלדנו את הסיסמא ניתן לראות את האותיות שבהם השתמשנו באזור ה-Payload:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
```

```
nitay@ubuntu:~/Desktop/Networks EX/Networks-Ma
tala5$ make runsn
sudo ./Sniffer
[sudo] password for nitay:
Finding available devices ... Done
Available Devices are :
1. enp0s3 - (null)
2. any - Pseudo-device that captures on all in
terfaces
3. lo - (null)
4. bluetooth-monitor - Bluetooth Linux Monitor
5. nflog - Linux netfilter log (NFLOG) interfa
ce
6. nfqueue - Linux netfilter queue (NFQUEUE) i
nterface
7. dbus-system - D-Bus system bus
8. dbus-session - D-Bus session bus
Enter the number of the device you want to sni
ff : 3
Device: lo
█
```

```
Connected to 127.0.0.1.
Escape character is '^]'.
Ubuntu 22.04.1 LTS
ubuntu login: nitay
Password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.1
5.0-58-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonic
al.com
 * Support:       https://ubuntu.com/advant
age

28 updates can be applied immediately.
To see these additional updates run: apt lis
t --upgradable

Last login: Wed Jan 18 13:52:00 IST 2023 fro
m localhost on pts/1
nitay@ubuntu:~$ █
```

בקשה להכניס סיסמא (חבילה 37)

```

A4 A0 C8 86 16 DB 6B 28 EE C8 80 18 02 00 FE 32      .....k(..0....2
Data Payload
50 61 73 73 77 6F 72 64 3A 20                        Password:
*****TCP_Rocket*****

```

ובחבילות הבאות יופיעו האותיות

```

Data Payload
  69
*****TCP Packet*****

Data Payload
  79
*****TCP Packet*****

```

וכן הלאה עד להודעה שההתחברות הושלמה בהצלחה.

(ראה קובץ טקסט מצורף)

מה זה spoofer?

כאשר משתמש רגיל שולח חבילה, מערכות הפעלה בדרך כלל אינן מאפשרות למשתמש להגדיר את כל השדות בכתובות הפרוטוקול (כגון כתובות TCP, UDP ו-IP). מערכת הפעלה תגדיר את רוב השדות, תוך מתן אפשרות רק למשתמשים להגדיר כמה שדות, כגון כתובת ה-IP של היעד, מספר יציאת היעד וכו'. עם זאת, אם למשתמש יש הרשאת root, הוא יכול להגדיר כל שדה שרירותי בחבילה. זה נקרא זיוף חבילות (spoofing), וניתן לעשות זאת באמצעות raw socket. בדרך זו ניתן למתכנתים את השליטה המוחלטת על בניית החבילה ואפשר לבנות כל חבילה שרירותית, כולל הגדרת שדות הכותרת והמטען (payload).

האם ניתן להגדיר את שדה אורך חבילת ה-IP לערך שרירותי, ללא קשר לגודל החבילה בפועל?

לא, מכיוון שבעת שליחת החבילה באמצעות sendto הפונקציה תדרוס את ערך הגודל שנגדיר אלא אם נעביר לה את ip->iph_len ואז ניתן להגדיר את מספר הביטים שישלחו. אולם אם אורך ה-IP מתחת ל-20 החבילה תחשב כפגומה ונקבל שגיאה.

למשל, נגדיר את אורך ה-IP כ-1500:

```
ip->iph_checksum = htons(0);  
ip->iph_len = htons(1500);
```

וב-Wireshark אכן ניתן לראות length = 1514 וכן len = 1460 כאשר

	Source	Destination	Protocol	Length	Info
000000	10.0.2.15	127.0.0.1	TCP	1514	12345 → 9090 [ACK, ECN] Seq=1 Ack=1 Win=512 Len=1460

כאשר משתמשים ב-Raw Socket, האם עלינו לחשב את ה-checksum עבור כותרת ה-IP?

ה-checksum משמש לבדיקת שגיאות של כותרת ה-IP. הוא מחושב אוטומטית דרך הפונקציה sendto וה-kernal עושה את החישוב בעצמו.

איך להריץ את התוכנה?

```
runspICMP:
    sudo ./Spoofer ICMP

runspUDP:
    sudo ./Spoofer UDP

runspTCP:
    sudo ./Spoofer TCP
```

לאחר שערכנו את כתובת ה-IP ליעד השליחה ושינינו את כתובת השולח ל-IP שלנו, של מחשב אחר, או IP שגוי, נקליד את הקיצורים של ה-makefile או נכתוב <protocol ./Spoofer sudo>.

Task2.2A

למשל, נשלח חבילה בפרוטוקול ICMP לגוגל כאשר ה-IP הוא 1.2.3.4.

```
ip->iph_sourceip.s_addr = inet_addr("1.2.3.4");
ip->iph_destip.s_addr = inet_addr("8.8.8.8");
```

```
nitay@ubuntu:~/Desktop/Networks EX/Networks-Matala5$ make runspICMP
sudo ./Spoofer ICMP
[sudo] password for nitay:
nitay@ubuntu:~/Desktop/Networks EX/Networks-Matala5$
```

Source	Destination	Protocol	Length	Info
1.2.3.4	8.8.8.8	ICMP	42	Echo (ping) request

מכיוון שנתנו IP source שלא שלנו, לא קיבלנו פינג בחזרה.

(ראה קובץ pcap)

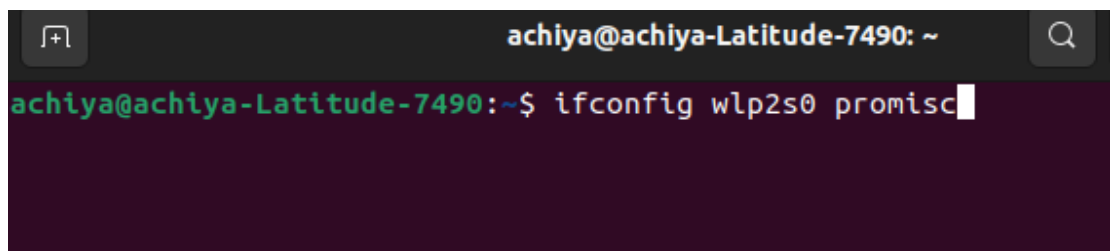
Task C

איך התוכנה של sniff and spoof עובדת: קודם התוכנה מסניפה בעזרת פילטר את הפאקטות שאנו רוצים.

לאחר שהיא הסניפה פאקטה מתאימה התוכנה שולחת spoof ל ip source של הפאקטה שהיא קיבלה אשר מכילה icmp echo replay מזויף מהשרת ip destination של הפאקטה.

א.

בדרך שלנו לבצע משימה זאת הייתה בעזרת 3 מחשבים בייתיים הנמצאים באותה רשת. על מנת שנוכל לעקוב אחרי שני המחשבים האחרים עם המחשב השלישי נצטרך להפעיל את הפירמיסקיוס מוד בצורה כזאת:



```
achiya@achiya-Latitude-7490: ~  
achiya@achiya-Latitude-7490:~$ ifconfig wlp2s0 promisc
```

מה שמצב זה עושה זה שכרטיס הרשת שלנו יוכל להציג את כל המידע שעובר דרך הרשת wlp2s0 ולא רק מה שמיועד לאותו כרטיס רשת.

הרצנו את התוכנה של sniff_and_spoof על המחשב השלישי בצורה כזאת:

```

sudo ./snoof
Finding available devices ... Done
Available Devices are :
1. wlp2s0 - (null)
2. any - Pseudo-device that captures on all interfaces
3. lo - (null)
4. enp0s31f6 - (null)
5. docker0 - (null)
6. br-cb0b6aeb9c9a - (null)
7. bluetooth0 - Bluetooth adapter number 0
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
Device: wlp2s0
^Cmake: *** [makefile:28: runsno] Interrupt

```

נשלח את הפינג בצורה כזאת:

```

vboxuser@Achiya:~$ ping 192.168.1.227
PING 192.168.1.227 (192.168.1.227) 56(84) bytes of data.
64 bytes from 192.168.1.227: icmp_seq=1 ttl=63 time=64.9 ms
64 bytes from 192.168.1.227: icmp_seq=2 ttl=63 time=71.1 ms
64 bytes from 192.168.1.227: icmp_seq=3 ttl=63 time=92.8 ms
64 bytes from 192.168.1.227: icmp_seq=4 ttl=63 time=113 ms
64 bytes from 192.168.1.227: icmp_seq=5 ttl=63 time=69.9 ms
64 bytes from 192.168.1.227: icmp_seq=6 ttl=63 time=83.2 ms
64 bytes from 192.168.1.227: icmp_seq=7 ttl=63 time=86.6 ms
64 bytes from 192.168.1.227: icmp_seq=8 ttl=63 time=95.2 ms
64 bytes from 192.168.1.227: icmp_seq=9 ttl=63 time=115 ms
64 bytes from 192.168.1.227: icmp_seq=10 ttl=63 time=36.5 ms
64 bytes from 192.168.1.227: icmp_seq=11 ttl=63 time=38.0 ms
64 bytes from 192.168.1.227: icmp_seq=12 ttl=63 time=106 ms
^C
--- 192.168.1.227 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11223ms
rtt min/avg/max/mdev = 36.495/81.053/115.331/25.024 ms
vboxuser@Achiya:~$

```

הווירשארק נראה כך:

icmp						
No.	Time	Source	Destination	Protocol	Length	Info
8	3.476120145	192.168.1.129	192.168.1.227	ICMP	100	Echo (ping) request id=0x0001, seq=165/42240, ttl=63 (reply in 9)
9	3.476215985	192.168.1.227	192.168.1.129	ICMP	100	Echo (ping) reply id=0x0001, seq=165/42240, ttl=64 (request in 8)
10	3.784162986	192.168.1.227	192.168.1.129	ICMP	44	Echo (ping) reply id=0x1200, seq=0/0, ttl=20
11	4.500045438	192.168.1.129	192.168.1.227	ICMP	100	Echo (ping) request id=0x0001, seq=166/42496, ttl=63 (reply in 12)
12	4.500104571	192.168.1.227	192.168.1.129	ICMP	100	Echo (ping) reply id=0x0001, seq=166/42496, ttl=64 (request in 11)
13	4.808040485	192.168.1.227	192.168.1.129	ICMP	44	Echo (ping) reply id=0x1200, seq=0/0, ttl=20
15	5.523812482	192.168.1.129	192.168.1.227	ICMP	100	Echo (ping) request id=0x0001, seq=167/42752, ttl=63 (reply in 16)
16	5.523873515	192.168.1.227	192.168.1.129	ICMP	100	Echo (ping) reply id=0x0001, seq=167/42752, ttl=64 (request in 15)
17	5.832135255	192.168.1.227	192.168.1.129	ICMP	44	Echo (ping) reply id=0x1200, seq=0/0, ttl=20
22	6.553130121	192.168.1.129	192.168.1.227	ICMP	100	Echo (ping) request id=0x0001, seq=168/43008, ttl=63 (reply in 23)
23	6.553195076	192.168.1.227	192.168.1.129	ICMP	100	Echo (ping) reply id=0x0001, seq=168/43008, ttl=64 (request in 22)
24	6.856028864	192.168.1.227	192.168.1.129	ICMP	44	Echo (ping) reply id=0x1200, seq=0/0, ttl=20
27	7.575838820	192.168.1.129	192.168.1.227	ICMP	100	Echo (ping) request id=0x0001, seq=169/43264, ttl=63 (reply in 28)
28	7.575906299	192.168.1.227	192.168.1.129	ICMP	100	Echo (ping) reply id=0x0001, seq=169/43264, ttl=64 (request in 27)
29	7.880150879	192.168.1.227	192.168.1.129	ICMP	44	Echo (ping) reply id=0x1200, seq=0/0, ttl=20

ניתן לראות שהתוכנה שלנו עובדת טוב כי לאחר קבלה של ריקווסט
משרת 192.168.1.129 ל 192.168.1.227 אנחנו רואים שני ריפלי
שאחד מהם הוא האמיתי עם אורך 100 ואחד הוא ה spoof שלנו
באורך 44.

ב. נריץ את התוכנה בצורה הבאה עם sudo ./snoof
המכשיר any:

```
❯ achiya@achiya-Latitude-7490:~/Desktop/Networks/Labsetup/volumes/matala5$ sudo ./snoof
Finding available devices ... Done
Available Devices are :
1. wlp2s0 - (null)
2. any - Pseudo-device that captures on all interfaces
3. lo - (null)
4. enp0s31f6 - (null)
5. docker0 - (null)
6. br-cb0b6aeb9c9a - (null)
7. bluetooth1 - Bluetooth adapter number 0
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
Device: any
```

עכשיו התוכנה שלנו מסניפה פאקטות icmp.
נשלח פינג לשרת 8.8.8.8 למשל:

```

achiya@achiya-Latitude-7490:~$ 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=119 time=15.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=119 time=16.1 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=119 time=15.9 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=119 time=15.4 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=119 time=15.4 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=119 time=17.6 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=119 time=15.3 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=119 time=15.8 ms

```

ה-Wireshark נראה כך:

No.	Time	Source	Destination	Protocol	Length	Info
2496	49.137125568	192.168.1.227	8.8.8.8	ICMP	100	Echo (ping) request id=0x0020, seq=5/1280, ttl=64 (reply in 2497)
2497	49.153173002	8.8.8.8	192.168.1.227	ICMP	100	Echo (ping) reply id=0x0020, seq=5/1280, ttl=119 (request in 2496)
2532	49.383519764	8.8.8.8	192.168.1.227	ICMP	44	Echo (ping) reply id=0xedff, seq=4608/18, ttl=20
2545	50.138704891	192.168.1.227	8.8.8.8	ICMP	100	Echo (ping) request id=0x0020, seq=6/1536, ttl=64 (reply in 2546)
2546	50.156690387	8.8.8.8	192.168.1.227	ICMP	100	Echo (ping) reply id=0x0020, seq=6/1536, ttl=119 (request in 2545)
2581	50.407310339	8.8.8.8	192.168.1.227	ICMP	44	Echo (ping) reply id=0xedff, seq=4608/18, ttl=20
2598	51.139264283	192.168.1.227	8.8.8.8	ICMP	100	Echo (ping) request id=0x0020, seq=7/1792, ttl=64 (reply in 2599)
2599	51.155188063	8.8.8.8	192.168.1.227	ICMP	100	Echo (ping) reply id=0x0020, seq=7/1792, ttl=119 (request in 2598)
2629	51.431354552	8.8.8.8	192.168.1.227	ICMP	44	Echo (ping) reply id=0xedff, seq=4608/18, ttl=20
2652	52.140581098	192.168.1.227	8.8.8.8	ICMP	100	Echo (ping) request id=0x0020, seq=8/2048, ttl=64 (reply in 2653)
2653	52.156336755	8.8.8.8	192.168.1.227	ICMP	100	Echo (ping) reply id=0x0020, seq=8/2048, ttl=119 (request in 2652)
2687	52.455190735	8.8.8.8	192.168.1.227	ICMP	44	Echo (ping) reply id=0xedff, seq=4608/18, ttl=20
2700	53.141882139	192.168.1.227	8.8.8.8	ICMP	100	Echo (ping) request id=0x0020, seq=9/2304, ttl=64 (reply in 2701)
2701	53.157742745	8.8.8.8	192.168.1.227	ICMP	100	Echo (ping) reply id=0x0020, seq=9/2304, ttl=119 (request in 2700)
2733	53.479509356	8.8.8.8	192.168.1.227	ICMP	44	Echo (ping) reply id=0xedff, seq=4608/18, ttl=20

אפשר לראות פה פינג רגיל בשתי שורות הראשונות כגון 2496 ו 2497 אך שורה אחרי פינג רגיל כמו 2532 אפשר לראות את הריפלי של הסנופר שלנו עם אורך 44. עם אותו source ו destination כמו הריפלי שקיבלנו מ 8.8.8.8 ולכן ניתן לראות שהתוכנה עובדת כמו שצריך.

ג. עכשיו נשלח פינג לשרת מזויף כגון 8.8.8.1:

```

achiya@achiya-Latitude-7490:~$ ping 8.8.8.1
PING 8.8.8.1 (8.8.8.1) 56(84) bytes of data.

```

לאחר כמה שניות אנחנו עדיין לא רואים תשובה אך אם נסתכל ב-

Wireshark:

14	0.251756740	192.168.1.227	8.8.8.1	ICMP	100 Echo (ping) request	id=0x0024, seq=21/5376, ttl=64 (no response found!)
15	0.343855368	8.8.8.1	192.168.1.227	ICMP	44 Echo (ping) reply	id=0xedff, seq=4608/18, ttl=20
61	1.271784836	192.168.1.227	8.8.8.1	ICMP	100 Echo (ping) request	id=0x0024, seq=22/5632, ttl=64 (no response found!)
62	1.367915950	8.8.8.1	192.168.1.227	ICMP	44 Echo (ping) reply	id=0xedff, seq=4608/18, ttl=20
105	2.295961898	192.168.1.227	8.8.8.1	ICMP	100 Echo (ping) request	id=0x0024, seq=23/5888, ttl=64 (no response found!)
106	2.391976900	8.8.8.1	192.168.1.227	ICMP	44 Echo (ping) reply	id=0xedff, seq=4608/18, ttl=20
147	3.319963947	192.168.1.227	8.8.8.1	ICMP	100 Echo (ping) request	id=0x0024, seq=24/6144, ttl=64 (no response found!)
148	3.416161175	8.8.8.1	192.168.1.227	ICMP	44 Echo (ping) reply	id=0xedff, seq=4608/18, ttl=20
197	4.344008719	192.168.1.227	8.8.8.1	ICMP	100 Echo (ping) request	id=0x0024, seq=25/6400, ttl=64 (no response found!)
198	4.440018206	8.8.8.1	192.168.1.227	ICMP	44 Echo (ping) reply	id=0xedff, seq=4608/18, ttl=20
246	5.367821703	192.168.1.227	8.8.8.1	ICMP	100 Echo (ping) request	id=0x0024, seq=26/6656, ttl=64 (no response found!)
247	5.463880710	8.8.8.1	192.168.1.227	ICMP	44 Echo (ping) reply	id=0xedff, seq=4608/18, ttl=20
295	6.391972550	192.168.1.227	8.8.8.1	ICMP	100 Echo (ping) request	id=0x0024, seq=27/6912, ttl=64 (no response found!)
296	6.488027637	8.8.8.1	192.168.1.227	ICMP	44 Echo (ping) reply	id=0xedff, seq=4608/18, ttl=20

ניתן לראות שאנחנו שולחים פינג ל 8.8.8.1 אך לא מקבלים תשובה
 כי כתוב no response found! אנחנו רואים שהתוכנה שלנו עובדת
 היטב כי היא עדיין שולחת ריפלי מזויף לשרת שלנו למרות ש
 8.8.8.1 לא קיים.

מה זה Gateway?

ה-gateway משמש כנקודת כניסה ויציאה לרשת מכיוון שכל הנתונים
 חייבים לעבור או לתקשר עם השער לפני הניתוב.

ה-gateway שלנו מדמה רשת לא אמינה שמאבדת נתונים
 בהסתברות של 50%.

איך להריץ את התוכנה?

לשם ההסבר, אנו הגדרנו שה-Gateway ישלח ב-Port 9999 הודעות
 שהועברו אליו אל ה-Host. ניתן להריצו ע"י make_rungate.

```
nitay@ubuntu:~/Desktop/Networks EX/Networks-Matala5$ make_rungate
./Gateway 127.0.0.1
Waiting for clients
```

כעת, כדי להראות שהוא עובד נריץ את GatewayHost אליו נשלח
 הודעות שהתקבלו בסבירות של 50%.

```
nitay@ubuntu:~/Desktop/Networks EX/Networks-
Matala5$ ./GatewayHost
Waiting for clients
```


עכשיו נריץ את GatewayClient שישלח הודעה אל ה-Gateway שלנו דרך Port 9998.

```
nitay@ubuntu:~/Desktop/Networks-Matala5$ ./GatewayClient  
nitay@ubuntu:~/Desktop/Networks-Matala5$
```

ניתן לראות שההודעה התקבלה ב-Gateway והגרלנו מספר בין 0 ל-1 שיצא מעל 0.5 ולכן ההודעה הועברה ל-GatewayHost.

```
nitay@ubuntu:~/Desktop/Networks EX/Networks-Matala5$ make rungate  
./Gateway 127.0.0.1  
Waiting for clients  
Received packet from 127.0.0.1:48954  
Data is: hello  
Random number is: 0.840188  
Packet received  
█
```

```
nitay@ubuntu:~/Desktop/Networks EX/Networks-Matala5$ ./GatewayHost  
Waiting for clients  
Received packet from 127.0.0.1:33605  
Data is: hello  
Packet received  
█
```

בפעם השנייה המספר שהגרלנו יצא מתחת ל-0.5 ולכן לא הועברה ההודעה הלאה.

```
Received packet from 127.0.0.1:58689  
Data is: hello  
Random number is: 0.394383  
Packet lost  
█
```

לצורך הדגמה, ניתן לראות הרצה של התוכנית כאשר התקבלו שלוש חבילות ב-Gateway אך רק שניים הועברו הלאה. (ראה קובץ pcap)

ב-Terminal:

```
Received packet from 127.0.0.1:58725
Data is: hello
Random number is: 0.840188
Packet received
Received packet from 127.0.0.1:34604
Data is: hello
Random number is: 0.394383
Packet lost
Received packet from 127.0.0.1:41949
Data is: hello
Random number is: 0.783099
Packet received

nitay@ubuntu:~/Desktop/Networks
├─ EX/Networks-Matala5$ ./Gateway
Host
Waiting for clients
Received packet from 127.0.0.1:49879
Data is: hello
Packet received
Received packet from 127.0.0.1:49879
Data is: hello
Packet received

nitay@ubuntu:~/Desktop/Networks
├─ EX/Networks-Matala5$ ./Gateway
Client
nitay@ubuntu:~/Desktop/Networks
├─ EX/Networks-Matala5$ ./Gateway
Client
nitay@ubuntu:~/Desktop/Networks
├─ EX/Networks-Matala5$ ./Gateway
Client
nitay@ubuntu:~/Desktop/Networks
├─ EX/Networks-Matala5$
```

ב-Wireshark ניתן לראות את שלושת החבילות שהועברו ל-gateway

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	UDP	48	58725 → 9998 Len=6
2	0.000245390	127.0.0.1	127.0.0.1	TPLINK-SMARTHOME/JSON	48	UDP Cmd:0
3	1.614792194	127.0.0.1	127.0.0.1	UDP	48	34604 → 9998 Len=6
4	2.511048181	127.0.0.1	127.0.0.1	UDP	48	41949 → 9998 Len=6
5	2.511196392	127.0.0.1	127.0.0.1	TPLINK-SMARTHOME/JSON	48	UDP Cmd:0

אך רק שניים הועברו ל-Host. (TPLINK מופיע רק פעמיים)