

תקשורת ומחשוב מטלה 6

מגישות:

רז אלבז ת"ז: 207276775

ליאור נגר ת"ז: 209399294

Lab Task Set 1: Using Scapy to Sniff and Spoof Packets

Task 1.1: Sniffing Packets

המטרה של משימה זו היא ללמוד כיצד להשתמש ב-Scapy כדי לבצע רחרח מנות ב-Python.

התבקשנו לממש את הקוד הבא:

```
#!/usr/bin/env python3
from scapy.all import *

def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface='br-c93733e9f913', filter='icmp', prn=print_pkt)
```

Task 1.1A: (sniffer.py)

במשימה זו הפעלנו את התוכנית בפעם הראשונה עם הרשאות root ראינו כי הוא ממתין ל"הסניף" חבילות (ראו תמונה מספר 1), ומתחיל להסניף אותן(ראו תמונה מספר 2+3). לאחר מכן הפעלנו שוב את התוכנית אבל מבלי להשתמש בהרשאות ה-root וקיבלנו שגיאה.

להלן צילומי מסך:

עם הרשאות :

תמונה מספר 1:

```
raz@raz-VirtualBox: ~/PycharmProjects/Ex6
raz@raz-VirtualBox:~/PycharmProjects$ cd Ex6
raz@raz-VirtualBox:~/PycharmProjects/Ex6$ sudo chmod a+x sniffer.py
[sudo] password for raz:
raz@raz-VirtualBox:~/PycharmProjects/Ex6$ sudo python3 sniffer.py
```

תמונה מספר 2+3: (סוגים שונים של פרוטוקולים):

No.	Time	Source	Destination	Protocol	Length	Info
72	31.734200736	172.217.169.67	10.0.2.15	UDP	184	443 → 41058 Len=142
73	31.734772184	172.217.169.67	10.0.2.15	UDP	70	443 → 41058 Len=28
74	31.734892400	10.0.2.15	172.217.169.67	UDP	83	41058 → 443 Len=41
75	31.755710943	10.0.2.15	172.217.169.67	UDP	74	41058 → 443 Len=32
76	31.836116977	172.217.169.67	10.0.2.15	UDP	70	443 → 41058 Len=28
77	34.156644871	142.250.187.202	10.0.2.15	TLSv1.2	139	Application Data
78	34.204724766	10.0.2.15	142.250.187.202	TCP	54	46454 → 443 [ACK] Seq=
79	37.801295352	10.0.2.15	142.250.178.14	TLSv1.2	93	[TCP Previous segment
80	37.802241229	142.250.178.14	10.0.2.15	TCP	60	[TCP Acked unseen seq
81	37.803491925	10.0.2.15	142.250.178.14	TLSv1.2	138	Application Data, App
82	37.804261066	142.250.178.14	10.0.2.15	TCP	60	443 → 44848 [ACK] Seq=
83	37.806320135	10.0.2.15	142.250.178.14	TLSv1.2	93	Application Data
84	37.807079863	142.250.178.14	10.0.2.15	TCP	60	443 → 44848 [ACK] Seq=
85	37.808302215	10.0.2.15	142.250.178.14	TLSv1.2	78	Application Data
86	37.808398215	10.0.2.15	142.250.178.14	TCP	54	44848 → 443 [FIN, ACK
87	37.809074687	142.250.178.14	10.0.2.15	TCP	60	443 → 44848 [ACK] Seq=
88	37.809075275	142.250.178.14	10.0.2.15	TCP	60	443 → 44848 [ACK] Seq=
89	37.877426464	142.250.178.14	10.0.2.15	TLSv1.2	93	Application Data

```

raz@raz-VirtualBox: ~/PycharmProjects/Ex6

##[ Ethernet ]##
dst = 52:54:00:12:35:02
src = 08:00:27:2c:09:e1
type = IPv4
##[ IP ]##
version = 4
ihl = 5
tos = 0x0
len = 60
id = 44246
flags = DF
frag = 0
ttl = 64
proto = tcp
chksum = 0x48ce
src = 10.0.2.15
dst = 142.250.178.14
\options \
##[ TCP ]##
sport = 44848
dport = https
seq = 423108876
ack = 0
dataofs = 10
reserved = 0
flags = S
window = 64240
chksum = 0x4d46
urgptr = 0
options = [('MSS', 1460), ('SACKOK', b''), ('Timestamp', (411390689, 0)), ('NOP', None), ('WScale', 7)]

##[ Ethernet ]##
dst = 08:00:27:2c:09:e1
src = 52:54:00:12:35:02
type = IPv4
##[ IP ]##
version = 4
ihl = 5
tos = 0x0
len = 1385
id = 64476
flags = 0
frag = 0
ttl = 64

```

בלי הרשאות:

```

raz@raz-VirtualBox: ~/PycharmProjects/Ex6

raz@raz-VirtualBox:~/PycharmProjects/Ex6$ chmod a+x sniffer.py
raz@raz-VirtualBox:~/PycharmProjects/Ex6$ ./sniffer.py
Traceback (most recent call last):
  File "./sniffer.py", line 6, in <module>
    pkt = sniff(iface='enp0s3', filter='icmp', prn=print_pkt)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1263, in sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1127, in _run
    sniff_sockets[L2socket(type=ETH_P_ALL, iface=iface,
  File "/usr/local/lib/python3.8/dist-packages/scapy/arch/linux.py", line 486, in __init__
    self.ins = socket.socket(
  File "/usr/lib/python3.8/socket.py", line 231, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
raz@raz-VirtualBox:~/PycharmProjects/Ex6$

```

(sniffer.py) :Task 1.1B

במשימה זו התבקשנו לבצע סינונים שונים ב"הרחה".

כעת, נראה את

:Capture only the ICMP

בכדי לבדוק פרוטוקול ICMP ביצענו מטרמינל נוסף ping לכתובת של Google :
8.8.8.8 .

כפי שניתן לראות מצילום המסך של הטרמינל הפרוטוקול אשר מופיע הוא ICMP.

The screenshot shows a network analysis tool interface. The left pane displays the packet details for an ICMP Echo (ping) request. The right pane shows a list of captured packets, with the selected packet (Frame 250) highlighted. The packet details show the source IP as 10.0.2.15 and the destination IP as 8.8.8.8. The packet list shows multiple ICMP Echo (ping) requests and replies.

Packet Details (Left Pane):

```

type      = IPv4
[[[ IP ]]]
version   = 4
ihl       = 5
tos       = 0x0
len       = 84
id        = 31866
flags     = DF
frag      = 0
ttl       = 64
proto     = icmp
chksum    = 0xaz10
src       = 10.0.2.15
dst       = 8.8.8.8
\options
[[[ ICMP ]]]
type      = echo-request
code      = 0
chksum    = 0xc2a1
id        = 0x6
seq       = 0x94
unused    = ''

[[[ Raw ]]]
load      = '\x03'\xcba\x00\x00\x00\x00\x9ch\x0b\x00\x00\x00\x0
0x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#%&
() +,-./:01234567'

[[[ Ethernet ]]]
dst       = 08:00:27:2c:09:e1
src       = 52:54:00:12:35:02
type      = IPv4

[[[ IP ]]]
version   = 4
ihl       = 5
tos       = 0x0
len       = 84
id        = 5951
flags     = 0
frag      = 0
ttl       = 65
proto     = icmp
chksum    = 0x564c
src       = 8.8.8.8
dst       = 10.0.2.15
\options

```

Packet List (Right Pane):

No.	Time	Source	Destination	Protocol	Length	Info
370	198.672108966	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
371	198.764669078	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
372	199.674584943	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
373	199.748019707	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
376	200.677590138	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
377	200.747879628	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
378	201.679549919	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
379	201.772869337	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
380	202.681522430	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
381	202.755268266	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
382	203.687393315	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
383	203.771618730	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
384	204.689016216	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
385	204.760827713	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
386	205.691589195	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
387	205.763045923	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
388	206.693477154	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
389	206.770483234	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
407	207.696030973	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
408	207.789679884	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
409	208.699681870	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
410	208.773344931	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
411	209.701528055	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
412	209.774505956	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
415	210.708375424	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
416	210.781286661	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
432	211.708869391	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
433	212.182098771	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
434	212.710456113	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
435	212.785487803	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
436	213.718228272	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
437	213.794894536	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
438	214.720054259	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
439	214.790502387	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
440	215.726852314	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
441	215.798543974	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply

Frame 250: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0

:Capture any TCP packet and destination port number 23

ביצענו מטרמינל נוסף : telnet 142.250.187.202

פרוטוקול telnet משמש בעיקר משתמשים המעוניינים להתחבר באמצעות שורת הפקודה בין מחשבים ברשת, פרוטוקול זה מדמה מחשב המחובר למחשב אחר ועובד עליו למעשה. (ויקיפדיה) על כן נעזרנו בפרוטוקול זה בכדי לבדוק את הפרוטוקול של TCP. כפי שניתן לראות בצילום המסך מטה הפרוטוקול אשר מופיע הוא TCP. כמו כן, בצילום ה-Wireshark ניתן לראות DST PORT:23 כפי שהתבקשנו.

The left screenshot shows a terminal window with the following output:

```
###[ Ethernet ]###
dst      = 52:54:00:12:35:02
src      = 08:00:27:2c:09:e1
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x10
len      = 60
id       = 54060
flags    = DF
frag     = 0
ttl      = 64
proto    = tcp
chksum   = 0x10ac
src      = 10.0.2.15
dst      = 142.250.187.202
options  = \
###[ TCP ]###
sport    = 42834
dport    = telnet
seq      = 2560163844
ack      = 0
dataoffs = 10
reserved = 0
flags    = S
window   = 64240
chksum   = 0x5702
urgptr   = 0
options  = [('MSS', 1460), ('SackOK', b''), ('Timestamp', (391629740, 1, 0)), ('NOP', None), ('WScale', 7)]
###[ Ethernet ]###
dst      = 52:54:00:12:35:02
src      = 08:00:27:2c:09:e1
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x10
len      = 60
id       = 54061
flags    = DF
frag     = 0
ttl      = 64
proto    = tcp
```

The right screenshot shows the Wireshark interface with the following table of captured packets:

No.	Time	Source	Destination	Protocol	Length	Info
16	3.276265647	142.250.187.202	10.0.2.15	TLSv1.2	93	Application
18	3.276796854	10.0.2.15	142.250.187.202	TLSv1.2	93	Application
8	3.090112223	10.0.2.15	213.57.2.5	DNS	102	Standard qu
9	3.090450676	10.0.2.15	213.57.2.5	DNS	102	Standard qu
23	3.452213576	142.250.187.202	10.0.2.15	TLSv1.2	102	Application
12	3.112251765	213.57.2.5	10.0.2.15	DNS	118	Standard qu
22	3.451912865	142.250.187.202	10.0.2.15	TLSv1.2	121	Application
14	3.273808798	142.250.187.202	10.0.2.15	TLSv1.2	123	Application
4	3.079347485	142.250.187.202	10.0.2.15	TLSv1.2	124	Application
13	3.114697901	213.57.2.5	10.0.2.15	DNS	130	Standard qu
2	3.076655580	142.250.187.202	10.0.2.15	TLSv1.2	138	Application
25	23.543606429	142.250.187.202	10.0.2.15	TLSv1.2	139	Application
10	3.099863530	10.0.2.15	142.250.187.202	TLSv1.2	287	Application
20	3.279442588	10.0.2.15	142.250.187.202	TLSv1.2	491	Application
27	28.626113899	10.0.2.15	216.58.212.206	TLSv1.2	93	Application
28	28.626992832	216.58.212.206	10.0.2.15	TCP	60	443 → 59174
29	28.703930625	216.58.212.206	10.0.2.15	TLSv1.2	93	Application
30	28.753937447	10.0.2.15	216.58.212.206	TCP	54	59174 → 443
31	33.791195523	PcsCompu_2c:09:e1	RealtekU_12:35:02	ARP	42	Who has 10.
32	33.791858309	RealtekU_12:35:02	PcsCompu_2c:09:e1	ARP	60	10.0.2.2 is
33	34.051573351	10.0.2.15	142.250.187.202	TCP	74	[TCP Retran

The detailed view of the selected packet (No. 33) shows:

```
Frame 33: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_2c:09:e1 (08:00:27:2c:09:e1), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 142.250.187.202
Transmission Control Protocol, Src Port: 42834, Dst Port: 23, Seq: 0, Len: 0
```

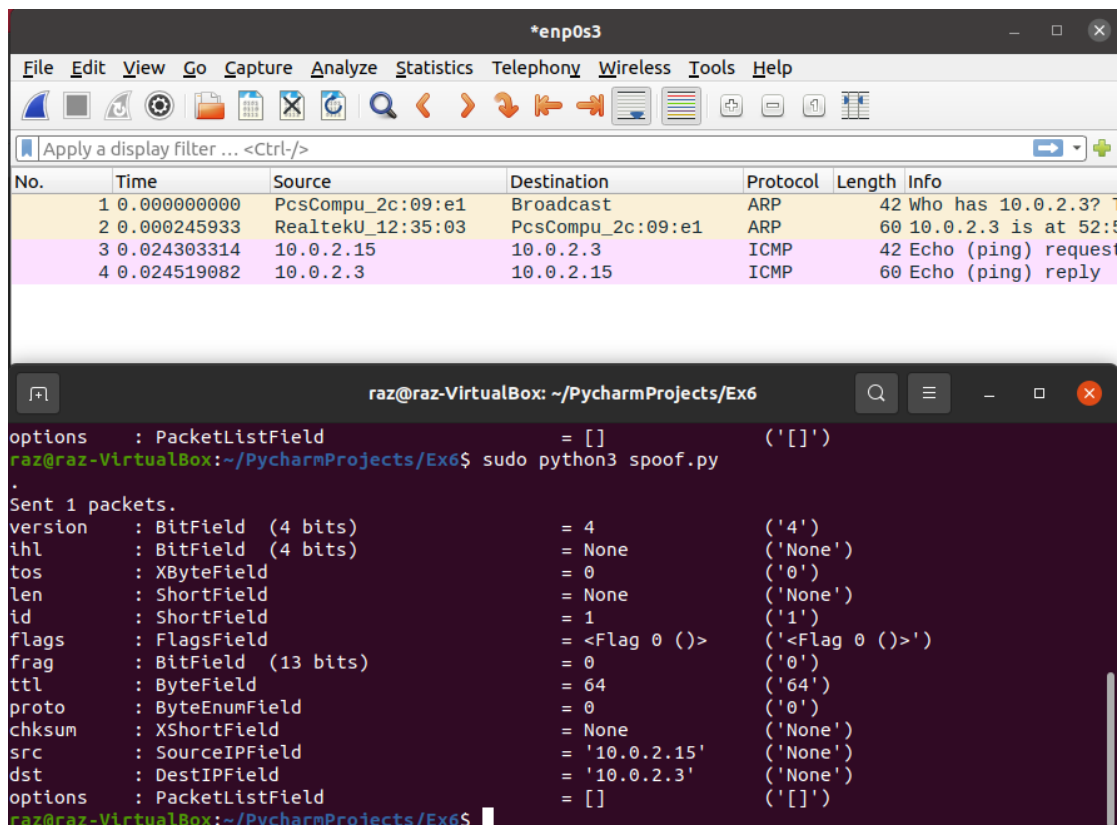
:Capture packets comes from or to go to a particular subnet

בשביל סינון זה בחרנו ב: 8.0.0.0/24 (היה ניתן לבחור כרצוננו), בכדי לבדוק שאכן מתבצע סינון ביצענו בשורת הפקודה ping לכתובת : 8.0.0.7 . ניתן לראות שאכן התבצע סינון של כתובות subnet. כמו כן, במקום הימני ביותר יכולנו לבחור כל מספר בתחום בין 0 לבין 24.

The screenshot shows a network traffic analysis tool (Wireshark) displaying a packet capture. The packet list on the left shows a packet from 10.0.2.15 to 8.0.0.7. The packet details pane shows the ICMP Echo request structure. The packet bytes pane shows the raw data. The packet list pane shows the packet details.


(spoof.py) : Task 1.2: Spoofing ICMP Packets

שלחנו חבילת ICMP כפי שקיבלנו בדוגמה לקוד זיוף מנות ICMP. שלחנו ping request לIP מזויף: 10.0.2.3 והראינו בעזרת ה-Wireshark שמתקבל ממנו ping reply. הוספנו הדפסה של פירוט החבילה על ידי הוספת השורה: ls(a) להלן צילומי מסך:



(TTL.py) :Task 1.3: Traceroute

במשימה זו התבקשנו לשלוח חבילות ICMP ליעד מסוים, אנחנו שלחנו ל- Instagram בכתובת: 157.240.221.174, בדיקת ה-IP נעשתה על ידי הרצה של הפקודה: `nslookup www.instagram.com` ב-CMD, והתבקשנו לבדוק מהו שדה ה-TTL-TIME TO LIVE של החבילה הרצויה עד אשר תגיע ליעד. בכדי לבצע את המשימה, על ידי לולאה FOR ביצענו שליחת חבילות ICMP החל מ-1 עד 20, כך שבכל פעם שדה ה-TTL ישתנה בהתאם לריצת הלולאה. ראינו כי מתקבלת תגובה ראשונה כאשר TTL=15 כפי שניתן לראות בצילומי המסך מטה.



```
C:\Users\97252>nslookup www.instagram.com
Server: ns2-cache.hotnet.net.il
Address: 213.57.22.5

Non-authoritative answer:
Name: z-p42-instagram.c10r.instagram.com
Addresses: 2a03:2880:f258:e0:face:b00c:0:4420
157.240.221.174
Aliases: www.instagram.com
geo-p42.instagram.com

C:\Users\97252>
```

```
raz@raz-VirtualBox:~/PycharmProjects/Ex6$ sudo python3 TTL.py
```

[illegible]

```
raz@raz-VirtualBox:~/PycharmProjects/Ex6$
```

No.	Time	Source	Destination	Protocol	Length	Info
5	2.268524721	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=1 (no response f...
6	2.268985613	10.0.2.2	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
7	2.357680488	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=0 (no response f...
8	2.358025669	10.0.2.2	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
9	2.401589064	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=1 (no response f...
10	2.401926958	10.0.2.2	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
11	2.453264876	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=2 (no response f...
12	2.480360997	192.168.1.1	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
13	2.528560817	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=3 (no response f...
14	2.543856485	10.158.48.1	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
15	2.563875345	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=4 (no response f...
16	2.606476302	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=5 (no response f...
17	2.622992770	172.17.5.126	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
18	2.654995793	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=6 (no response f...
19	2.671779295	172.17.3.10	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
20	2.688213005	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=7 (no response f...
21	2.736451548	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=8 (no response f...
22	2.770879070	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=9 (no response f...
23	2.810785214	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=10 (no response f...
24	2.851821344	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=11 (no response f...
25	2.887502221	157.240.68.86	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
26	2.888917972	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=12 (no response f...
27	2.927614896	129.134.44.198	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
28	2.934781137	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=13 (no response f...
29	2.968079675	129.134.54.231	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
30	2.979617942	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=14 (no response f...
31	3.010223328	173.252.67.159	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
32	3.023081129	10.0.2.15	157.240.221.174	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=15 (reply in 33)

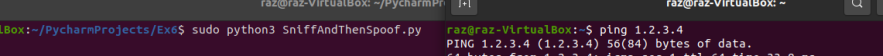
(SnifferAndThenPoof.py) :Task 1.4: Sniffing and-then Spoofing

במשימה זו יש תוקף ויש נתקף. התוקף מבצע הפעלה של התוכנית SniffAndThenSpoof.py בכדי לגנוב את המידע של הנתקף, והנתקף מצפה לתגובה מה-host האמיתי, אך לא כך יהיה הדבר.

בחלק זה של המטלה יצרנו פונקציה דומה לזו של sniffer.py אך השינויים הם בכך שכאשר נשלחת חבילה, התוכנית שכתבנו תיקח את החבילה אשר נשלחה ותיצור חבילת ICMP חדשה. נגדיר את השדות כך: ה-IP SRC של החבילה החדשה יהיה ה-IP DEST של החבילה שנשלחה ו ה-IP DEST של החבילה החדשה יהיה ה-IP SRC שנשלחה. ה-TYPE יהיה שווה 0 מכיוון שזה ה-TYPE של הודעת REPLY. ה-SEQ וה-IHL יישארו זהים.

הנתקף שולח פינג לכתובות אשר התבקשו : 8.8.8.8, 10.9.0.99, 1.2.3.4 ונראה שעל אף שלשתי הכתובות הראשונות שציינו לא קיים host אשר מחזיק אותן-משמע הן לא קיימות, בכל זאת מתקבלים נתונים באופן תקין כאילו הן כן. למעשה, מה שקורה הוא שהתוקף הוא זה שמחזיר לנתקף את הודעת reply. ניכר לראות כי הכתובת 8.8.8.8 היא כתובת אשר קיים לה host והוא Google, לכן מה שקורה הוא כפי שיוצג בצילום המסך מטה הוא שמתקבלות 2 תגובות reply להודעה אחת, הן מ-host האמיתי והן מהתוקף.

PING 1.2.3.4



The screenshot shows a Kali Linux terminal window with the title bar "Kali Linux - Terminal". The terminal prompt is "root@kali:~#". The user has entered the command "ping 1.2.3.4". The output shows the first eight ping results, each with a 64-byte packet size and a time of approximately 27-28 ms. The statistics at the bottom indicate 8 packets transmitted, 8 received, 0% packet loss, and a round-trip time of 7022ns.

```
root@kali:~# ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data:
64 bytes from 1.2.3.4: icmp_seq=1 ttl=64 time=33.8 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=11.8 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=26.4 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=22.7 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=22.4 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=34.3 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=28.6 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=27.6 ms
^C
--- 1.2.3.4 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7022ns
rtt min/avg/max/mdev = 11.752/25.926/34.251/6.756 ms
```

The screenshot shows the Wireshark network protocol analyzer interface. The title bar indicates the file name is *any. The menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The toolbar contains various icons for file operations, capture, analysis, and display. The status bar at the top shows 'Apply a display filter ... <Ctrl-/>'. The main packet list pane displays 18 captured packets, all of which are ICMP Echo (ping) requests or replies. The columns shown are No., Time, Source, Destination, Protocol, Length, and Info. The packets alternate between requests (No. 1, 3, 5, 7, 9, 11, 13, 15, 17) and replies (No. 2, 4, 6, 8, 10, 12, 14, 16, 18). All traffic is between the source IP 10.0.2.15 and the destination IP 10.0.2.15.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	1.2.3.4	ICMP	100	Echo (ping) request
2	0.025036417	1.2.3.4	10.0.2.15	ICMP	100	Echo (ping) reply
3	1.002371065	10.0.2.15	1.2.3.4	ICMP	100	Echo (ping) request
4	1.039335459	1.2.3.4	10.0.2.15	ICMP	100	Echo (ping) reply
5	2.004137137	10.0.2.15	1.2.3.4	ICMP	100	Echo (ping) request
6	2.048126996	1.2.3.4	10.0.2.15	ICMP	100	Echo (ping) reply
7	3.005591748	10.0.2.15	1.2.3.4	ICMP	100	Echo (ping) request
8	3.035865843	1.2.3.4	10.0.2.15	ICMP	100	Echo (ping) reply
9	4.009155715	10.0.2.15	1.2.3.4	ICMP	100	Echo (ping) request
10	4.036403734	1.2.3.4	10.0.2.15	ICMP	100	Echo (ping) reply
11	5.010891585	10.0.2.15	1.2.3.4	ICMP	100	Echo (ping) request
12	5.027256454	1.2.3.4	10.0.2.15	ICMP	100	Echo (ping) reply
13	6.012569988	10.0.2.15	1.2.3.4	ICMP	100	Echo (ping) request
14	6.041385555	1.2.3.4	10.0.2.15	ICMP	100	Echo (ping) reply
15	7.028352128	10.0.2.15	1.2.3.4	ICMP	100	Echo (ping) request
16	7.058424198	1.2.3.4	10.0.2.15	ICMP	100	Echo (ping) reply
17	8.031899568	10.0.2.15	1.2.3.4	ICMP	100	Echo (ping) request
18	8.055208367	1.2.3.4	10.0.2.15	ICMP	100	Echo (ping) reply

PING 10.9.0.99

```

raz@raz-VirtualBox: ~/PycharmProjects/Ex6
Sent 1 packets.
^Craz@raz-VirtualBox:~/PycharmProjects/Ex6$ sudo python3 SniffAndThenSpoof.py
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.

raz@raz-VirtualBox: ~$ ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data:
64 bytes from 10.9.0.99: icmp_seq=1 ttl=64 time=25.6 ms
64 bytes from 10.9.0.99: icmp_seq=2 ttl=64 time=34.8 ms
64 bytes from 10.9.0.99: icmp_seq=3 ttl=64 time=21.5 ms
64 bytes from 10.9.0.99: icmp_seq=4 ttl=64 time=19.2 ms
64 bytes from 10.9.0.99: icmp_seq=5 ttl=64 time=38.4 ms
64 bytes from 10.9.0.99: icmp_seq=6 ttl=64 time=20.2 ms
64 bytes from 10.9.0.99: icmp_seq=7 ttl=64 time=17.8 ms
64 bytes from 10.9.0.99: icmp_seq=8 ttl=64 time=104 ms
64 bytes from 10.9.0.99: icmp_seq=9 ttl=64 time=42.0 ms
64 bytes from 10.9.0.99: icmp_seq=10 ttl=64 time=58.9 ms
64 bytes from 10.9.0.99: icmp_seq=11 ttl=64 time=34.0 ms
64 bytes from 10.9.0.99: icmp_seq=12 ttl=64 time=33.9 ms
64 bytes from 10.9.0.99: icmp_seq=13 ttl=64 time=104 ms
64 bytes from 10.9.0.99: icmp_seq=14 ttl=64 time=24.1 ms
64 bytes from 10.9.0.99: icmp_seq=15 ttl=64 time=29.0 ms
64 bytes from 10.9.0.99: icmp_seq=16 ttl=64 time=29.9 ms

```

Task 1.4_10.9.0.99.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

icmp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	10.9.0.99	ICMP	100	Echo (ping) request id=0x0
2	0.019125757	10.9.0.99	10.0.2.15	ICMP	100	Echo (ping) reply id=0x0
3	1.001517399	10.0.2.15	10.9.0.99	ICMP	100	Echo (ping) request id=0x0
4	1.039858394	10.9.0.99	10.0.2.15	ICMP	100	Echo (ping) reply id=0x0
8	2.003303131	10.0.2.15	10.9.0.99	ICMP	100	Echo (ping) request id=0x0
9	2.023415775	10.9.0.99	10.0.2.15	ICMP	100	Echo (ping) reply id=0x0
12	3.005440139	10.0.2.15	10.9.0.99	ICMP	100	Echo (ping) request id=0x0
13	3.023200975	10.9.0.99	10.0.2.15	ICMP	100	Echo (ping) reply id=0x0
14	4.015138095	10.0.2.15	10.9.0.99	ICMP	100	Echo (ping) request id=0x0
15	4.119475975	10.9.0.99	10.0.2.15	ICMP	100	Echo (ping) reply id=0x0
16	5.016779372	10.0.2.15	10.9.0.99	ICMP	100	Echo (ping) request id=0x0
17	5.058721488	10.9.0.99	10.0.2.15	ICMP	100	Echo (ping) reply id=0x0
18	6.038998146	10.0.2.15	10.9.0.99	ICMP	100	Echo (ping) request id=0x0
19	6.097719562	10.9.0.99	10.0.2.15	ICMP	100	Echo (ping) reply id=0x0
20	7.044513102	10.0.2.15	10.9.0.99	ICMP	100	Echo (ping) request id=0x0
21	7.078452393	10.9.0.99	10.0.2.15	ICMP	100	Echo (ping) reply id=0x0
22	8.046460564	10.0.2.15	10.9.0.99	ICMP	100	Echo (ping) request id=0x0
23	8.080277131	10.9.0.99	10.0.2.15	ICMP	100	Echo (ping) reply id=0x0
24	9.048235640	10.0.2.15	10.9.0.99	ICMP	100	Echo (ping) request id=0x0
25	9.068031325	10.9.0.99	10.0.2.15	ICMP	100	Echo (ping) reply id=0x0
26	10.050730554	10.0.2.15	10.9.0.99	ICMP	100	Echo (ping) request id=0x0
27	10.074814558	10.9.0.99	10.0.2.15	ICMP	100	Echo (ping) reply id=0x0
28	11.051350201	10.0.2.15	10.9.0.99	ICMP	100	Echo (ping) request id=0x0

PING 8.8.8.8

```

raz@raz-VirtualBox: ~/PycharmProjects/Ex6
Sent 1 packets.
raz@raz-VirtualBox:~/PycharmProjects/Ex6$ sudo python3 SniffAndThenSpoof.py
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.
Sent 1 packets.

raz@raz-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=64 time=31.3 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=55 time=75.9 ms (DUPLICATE)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=13.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=55 time=79.9 ms (DUPLICATE)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=22.1 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=55 time=93.6 ms (DUPLICATE)
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=21.1 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=55 time=71.4 ms (DUPLICATE)
64 bytes from 8.8.8.8: icmp_seq=5 ttl=64 time=24.0 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=55 time=80.6 ms (DUPLICATE)
64 bytes from 8.8.8.8: icmp_seq=6 ttl=64 time=26.2 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=55 time=72.1 ms (DUPLICATE)
64 bytes from 8.8.8.8: icmp_seq=7 ttl=64 time=28.2 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=55 time=85.7 ms (DUPLICATE)
64 bytes from 8.8.8.8: icmp_seq=8 ttl=64 time=56.3 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=55 time=475 ms (DUPLICATE)
64 bytes from 8.8.8.8: icmp_seq=9 ttl=64 time=27.4 ms

```

Task 1.4_8.8.8.8.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	8.8.8.8	ICMP	100	Echo (ping) request
2	0.031231581	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
3	0.075823442	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
4	1.001972619	10.0.2.15	8.8.8.8	ICMP	100	Echo (ping) request
5	1.015351322	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
6	1.081801869	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
7	2.003278708	10.0.2.15	8.8.8.8	ICMP	100	Echo (ping) request
8	2.025381853	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
9	2.096905219	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
10	3.005443217	10.0.2.15	8.8.8.8	ICMP	100	Echo (ping) request
11	3.026440968	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
12	3.076768736	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
13	4.006913708	10.0.2.15	8.8.8.8	ICMP	100	Echo (ping) request
14	4.030881933	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
15	4.087535720	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
16	5.008140942	10.0.2.15	8.8.8.8	ICMP	100	Echo (ping) request
17	5.034250258	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
18	5.080235207	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
19	5.122859264	PcsCompu_d4:f0:9a		ARP	44	Who has 10.0.2.2? Tel
20	5.123196422	RealtekU_12:35:02		ARP	62	10.0.2.2 is at 52:54:0
21	6.010438762	10.0.2.15	8.8.8.8	ICMP	100	Echo (ping) request
22	6.030608486	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
23	6.096048577	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
24	7.013228778	10.0.2.15	8.8.8.8	ICMP	100	Echo (ping) request
25	7.069488315	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
26	7.488210552	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
27	8.019682828	10.0.2.15	8.8.8.8	ICMP	100	Echo (ping) request
28	8.047061327	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
29	8.109439962	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply

Frame 1: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0

Lab Task Set 2: Writing Programs to Sniff and Spoof Packets

Task 2.1A

- Question 1. Please use your own words to describe the sequence of the library calls that are essential for sniffer programs. This is meant to be a summary, not detailed explanation like the one in the tutorial or book.
- Question 2. 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?
- Question 3. Please turn on and turn off the promiscuous mode in your sniffer program. The value 1 of the third parameter in pcap open live() turns on the promiscuous mode (use 0 to turn it off). Can you demonstrate the difference when this mode is on and off? Please describe how you can demonstrate this. You can use the following command to check whether an interface's promiscuous mode is on or off (look at the promiscuity's value).

פתרון שאלה 1:

ראשית, השימוש בפונקציה `pcap_open_live` השייכת לספריית `pcap` תאפשר לנו לראות את כל תעבורת הרשת ותעזור לנו ללכוד מנות ברשת. שנית, בכדי להשיג את המידע המבוקש, נפעיל את הסינונים אותם נרצה על ידי שימוש בפונקציות `pcap_compile` אשר תשמש בשביל לקמפל מחרוזת לתוך תוכנית הסינון וב- `pcap_setfilter` על מנת לסנן פרוטוקולים ספציפיים. לבסוף, נרצה ללכוד את המנות, ולכן תוכנת ההסנפה תסניף את החבילות המסוננות ותשתמש בפונקציית `pcap_loop` על מנת ללכוד מנות בלולאה אינסופית עד אשר נקבל את המידע.

פתרון שאלה 2:

נצטרך הרשאות מנהל על מנת להריץ את תוכנית ההסנפה מכיוון שזה נדרש בעת פתיחת `raw socket` המאפשר לקבל כל חבילה אשר עוברת בכרטיס רשת מסוים. במידה ולא נשתמש בהרשאות מנהל נקבל שגיאה בעת ההרצה היות וזה נועד למנוע את האפשרות שכל משתמש יפתח `raw socket` ויבחן את החבילות אשר עוברות ברשת. שגיאה כזו התקבלה כאשר ניסינו להריץ את התוכנית בtask 1.1A. מצורפת תמונה לתזכורת:

```

raz@raz-VirtualBox: ~/PycharmProjects/Ex6
raz@raz-VirtualBox:~/PycharmProjects/Ex6$ chmod a+x sniffer.py
raz@raz-VirtualBox:~/PycharmProjects/Ex6$ ./sniffer.py
Traceback (most recent call last):
  File "./sniffer.py", line 6, in <module>
    pkt = sniff(iface='enp0s3', filter='icmp', prn=print_pkt)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1263, in sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1127, in _run
    sniff_sockets[L2socket(type=ETH_P_ALL, iface=iface,
  File "/usr/local/lib/python3.8/dist-packages/scapy/arch/linux.py", line 486, in __init__
    self.ins = socket.socket(
  File "/usr/lib/python3.8/socket.py", line 231, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
raz@raz-VirtualBox:~/PycharmProjects/Ex6$

```

פתרון שאלה 3: (sniffer.c)

המודם promiscuous בתוכנית שלנו sniffer.c נדלק ונכבה על ידי השמת "1" ו-"0" בהתאמה במשתנה השלישי בפונקציה pcap_open_live() כאשר הפעלנו את המודם promiscuous כל חבילה אשר עברה בכרטיס הרשת נתפסה. בנוסף, בעזרת טרמינל נוסף ביצענו פינג ל 8.8.8.8 וניתן לראות שהתבצעו הדפסות על המסך. כאשר כיבינו את המודם promiscuous לא נתפסו חבילות.

ON

```

raz@raz-VirtualBox: ~/CLionProjects/ex6
raz@raz-VirtualBox:~/CLionProjects/ex6$ sudo ./sniffer
Got a packet
SRC: 39.212.240.154
DEST: 82.84.0.18
Got a packet
SRC: 39.212.240.154
DEST: 82.84.0.18
Got a packet
SRC: 39.212.240.154
DEST: 82.84.0.18
Got a packet
SRC: 39.212.240.154
DEST: 82.84.0.18
Got a packet
SRC: 39.212.240.154
DEST: 82.84.0.18
Got a packet
SRC: 39.212.240.154
DEST: 82.84.0.18
^C
raz@raz-VirtualBox:~/CLionProjects/ex6$

raz@raz-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=108 time=73.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=108 time=71.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=108 time=71.4 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=108 time=73.0 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=108 time=73.6 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 71.373/72.594/73.561/0.929 ms
raz@raz-VirtualBox:~$

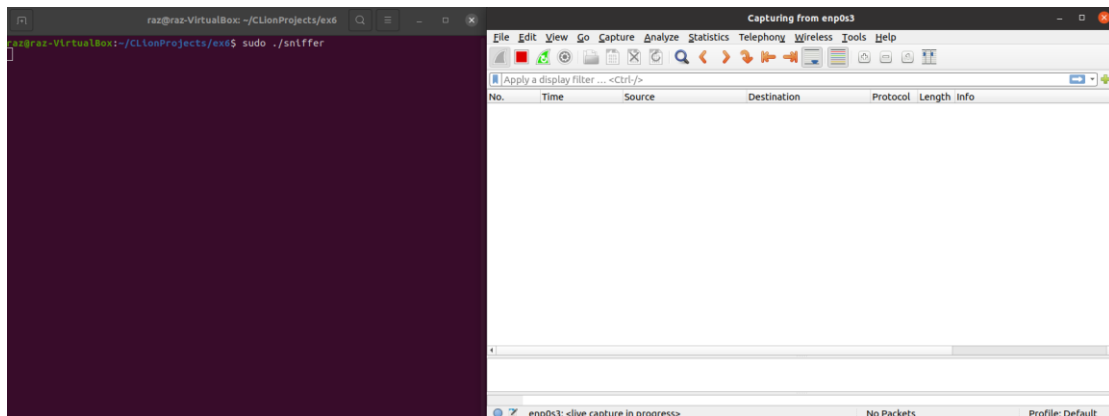
```

Task 2.1A_on.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
36	24.988825591	10.0.2.15	192.168.1.1	DNS	76	Standard query 0x9541
37	24.993814451	192.168.1.1	10.0.2.15	DNS	76	Standard query response
38	26.039044926	10.0.2.15	35.224.170.84	TCP	74	[TCP Retransmission]
39	28.055669866	10.0.2.15	35.224.170.84	TCP	74	[TCP Retransmission]
40	28.111273244	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
41	28.184663714	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
42	29.113020608	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
43	29.184576461	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
44	30.102826047	10.0.2.15	35.224.170.84	TCP	74	[TCP Retransmission]
45	30.114410372	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
46	30.164930133	35.224.170.84	10.0.2.15	TCP	60	80 → 39666 [SYN, ACK]
47	30.164977287	10.0.2.15	35.224.170.84	TCP	54	39666 → 80 [ACK] Seq=
48	30.165150178	10.0.2.15	35.224.170.84	HTTP	141	GET / HTTP/1.1
49	30.165419635	35.224.170.84	10.0.2.15	TCP	60	80 → 39666 [ACK] Seq=
50	30.185759187	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
51	30.369493475	35.224.170.84	10.0.2.15	HTTP	202	HTTP/1.1 204 No Conte
52	30.369540377	10.0.2.15	35.224.170.84	TCP	54	39666 → 80 [ACK] Seq=
53	30.369493759	35.224.170.84	10.0.2.15	TCP	60	80 → 39666 [FIN, ACK]
54	30.369709959	10.0.2.15	35.224.170.84	TCP	54	39666 → 80 [FIN, ACK]
55	30.369840274	35.224.170.84	10.0.2.15	TCP	60	80 → 39666 [ACK] Seq=
56	31.116566530	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface enp0s3, id 0
 Ethernet II, Src: PcsCompu_d4:f0:9a (08:00:27:d4:f0:9a), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
 Internet Protocol Version 4, Src: 10.0.2.15, Dst: 35.224.170.84

Frame (frame), 74 bytes Packets: 59 - Displayed: 59 (100.0%) Profile: Default



Task 2.1B

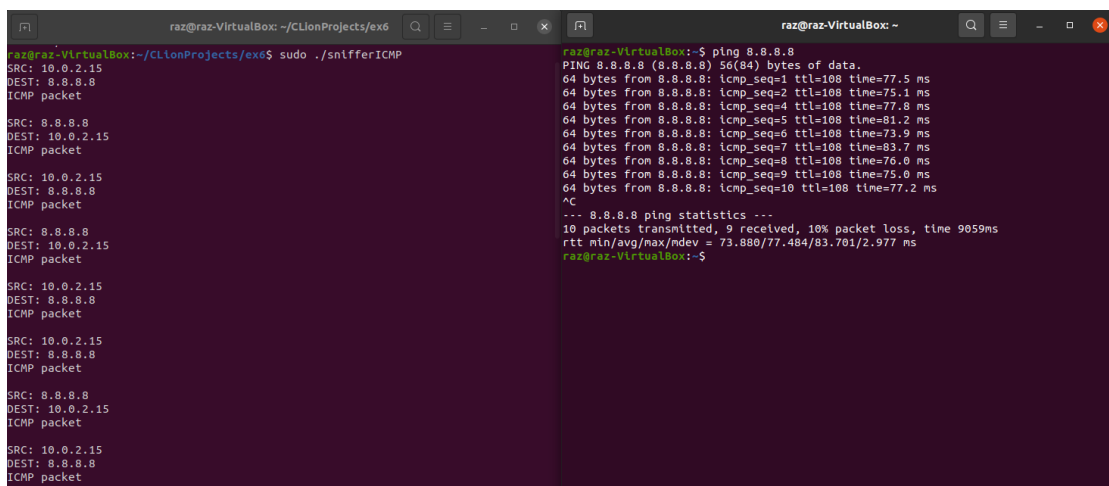
במשימה זו התבקשנו לכתוב תוכנית הדומה לתוכנית אשר כתבנו במשימה הקודמת, רק כשבמשימה זו התבקשנו ליצור סינון עבור לכידות מסוימות.

לכידת מנות ICMP בין שני מארחים ספציפיים: (snifferICMP.c)

נעשה על ידי הסינון הבא:

"ip proto icmp and host 10.0.2.15 and host 8.8.8.8"

ניתן לראות בצילום ה-wireshark כי החבילות אשר סוננו הן אכן חבילות אשר שייכות לכתובות ה-IP שהגדרנו והן מסוג ICMP.



Task 2.1B_ICMP.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request id=0x6
2	0.077437472	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply id=0x6
3	1.001923225	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request id=0x6
4	1.077010246	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply id=0x6
5	2.013750182	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request id=0x6
6	3.039722001	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request id=0x6
7	3.117489053	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply id=0x6
8	4.039890564	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request id=0x6
9	4.121047726	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply id=0x6
10	5.041904386	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request id=0x6
11	5.115665043	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply id=0x6
12	5.151647985	PcsCompu_d4:f0:9a	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2? Tell 10.0
13	5.151984318	RealtekU_12:35:02	PcsCompu_d4:f0:9a	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
14	6.044779036	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request id=0x6
15	6.128425489	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply id=0x6
16	7.046424173	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request id=0x6
17	7.122274969	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply id=0x6
18	8.052821172	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request id=0x6
19	8.127760532	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply id=0x6

Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0
 Ethernet II, Src: PcsCompu_d4:f0:9a (08:00:27:d4:f0:9a), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
 Internet Protocol Version 4, Src: 10.0.2.15, Dst: 8.8.8.8

Frame (frame), 98 bytes Packets: 19 · Displayed: 19 (100.0%) Profile: Default

לכידת מנות ה-TCP עם מספר יציאת יעד בטווח שבין 10 ל-100:
 (snifferTCP.c)

נעשתה על ידי הסינון הבא:

"tcp and dst portrange 10-100"

ניתן לראות בצילום האreshark כי החבילות אשר סוגנו הן אכן חבילות מסוג TCP עם DEST PORT בין 10-100.

Task 2.1B_TCP.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
734	16.579004187	185.199.108.154	10.0.2.15	TLSv1.3	113	Application Data
735	16.579034347	10.0.2.15	185.199.108.154	TCP	54	50580 → 443 [ACK] Seq=
736	16.581033686	185.199.108.154	10.0.2.15	TLSv1.3	4422	Application Data, App=
737	16.581068968	10.0.2.15	185.199.108.154	TCP	54	50580 → 443 [ACK] Seq=
738	16.583224983	185.199.108.154	10.0.2.15	TLSv1.3	4422	Application Data, App=
739	16.583228047	10.0.2.15	185.199.108.154	TCP	54	50580 → 443 [ACK] Seq=
740	16.587941254	185.199.108.154	10.0.2.15	TLSv1.3	1693	Application Data, App=
741	16.587988146	10.0.2.15	185.199.108.154	TCP	54	50580 → 443 [ACK] Seq=
742	16.587941569	34.120.208.123	10.0.2.15	TLSv1.2	92	Application Data
743	16.588028819	10.0.2.15	34.120.208.123	TCP	54	35436 → 443 [RST] Seq=
744	16.594867243	10.0.2.15	52.35.145.245	TLSv1.2	687	Application Data
745	16.594866369	10.0.2.15	185.199.108.154	TLSv1.3	85	Application Data
746	16.595082098	52.35.145.245	10.0.2.15	TCP	60	443 → 39454 [ACK] Seq=
747	16.595139829	185.199.108.154	10.0.2.15	TCP	60	443 → 50580 [ACK] Seq=
748	16.618630612	142.250.185.74	10.0.2.15	TCP	60	443 → 44062 [FIN, ACK]
749	16.618676855	10.0.2.15	142.250.185.74	TCP	54	44062 → 443 [ACK] Seq=
750	16.621368219	34.120.208.123	10.0.2.15	TLSv1.2	512	Application Data
751	16.622128168	10.0.2.15	34.120.208.123	TCP	54	35440 → 443 [ACK] Seq=
752	16.622999323	10.0.2.15	34.120.208.123	TLSv1.2	100	Application Data
753	16.623293042	34.120.208.123	10.0.2.15	TCP	60	443 → 35440 [ACK] Seq=
754	16.625363965	34.120.208.123	10.0.2.15	TLSv1.2	128	Application Data

Frame 1: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface enp0s3, id 0
 Ethernet II, Src: PcsCompu_d4:f0:9a (08:00:27:d4:f0:9a), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
 Internet Protocol Version 4, Src: 10.0.2.15, Dst: 192.168.1.1

Frame (frame), 84 bytes Packets: 1082 · Displayed: 1082 (100.0%) Profile: Default

Task 2.1B_TCP.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp

No.	Time	Source	Destination	Protocol	Length	Info
965	17.476655798	10.0.2.15	93.184.220.29	TCP	54	37498 → 80 [FIN, ACK]
966	17.476679632	10.0.2.15	93.184.220.29	TCP	54	37496 → 80 [FIN, ACK]
967	17.476690607	140.82.121.5	10.0.2.15	TCP	60	443 → 47978 [ACK] Seq=
968	17.476702378	10.0.2.15	185.199.108.154	TLSv1.3	78	Application Data
969	17.476690686	3.233.111.32	10.0.2.15	TCP	60	443 → 57838 [ACK] Seq=
970	17.476750366	10.0.2.15	185.199.108.154	TCP	54	50580 → 443 [FIN, ACK]
971	17.476790057	93.184.220.29	10.0.2.15	TCP	60	80 → 37498 [ACK] Seq=
972	17.476790131	93.184.220.29	10.0.2.15	TCP	60	80 → 37496 [ACK] Seq=
973	17.476790168	185.199.108.154	10.0.2.15	TCP	60	443 → 50580 [ACK] Seq=
974	17.476964905	185.199.108.154	10.0.2.15	TCP	60	443 → 50580 [ACK] Seq=
975	17.478540321	10.0.2.15	34.120.208.123	TLSv1.2	85	Encrypted Alert
976	17.478554633	10.0.2.15	34.120.208.123	TCP	54	35440 → 443 [FIN, ACK]
977	17.478643884	10.0.2.15	93.184.220.29	TCP	54	37500 → 80 [FIN, ACK]
978	17.478658493	10.0.2.15	172.217.16.131	TCP	54	58938 → 80 [FIN, ACK]
979	17.478676764	10.0.2.15	54.184.117.141	TLSv1.2	85	Encrypted Alert

Frame 965: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s3, id 0

- Ethernet II, Src: PcsCompu_d4:f0:9a (08:00:27:d4:f0:9a), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
- Internet Protocol Version 4, Src: 10.0.2.15, Dst: 93.184.220.29
- Transmission Control Protocol, Src Port: 37498, Dst Port: 80, Seq: 845, Ack: 1599, Len: 0

Frame (frame), 54 bytes Packets: 1082 · Displayed: 892 (82.4%) · Dropped: 0 (0.0%) Profile: Default

(snifferPassword.c) Task 2.1C

במשימה זו התבקשנו ליצור תוכנית אשר השתמשנו בתוכנת ה sniffer כדי ללכוד סיסמאות ברשת כאשר מישהו משתמש בtelnet ברשת שאנחנו משתמשים בה.

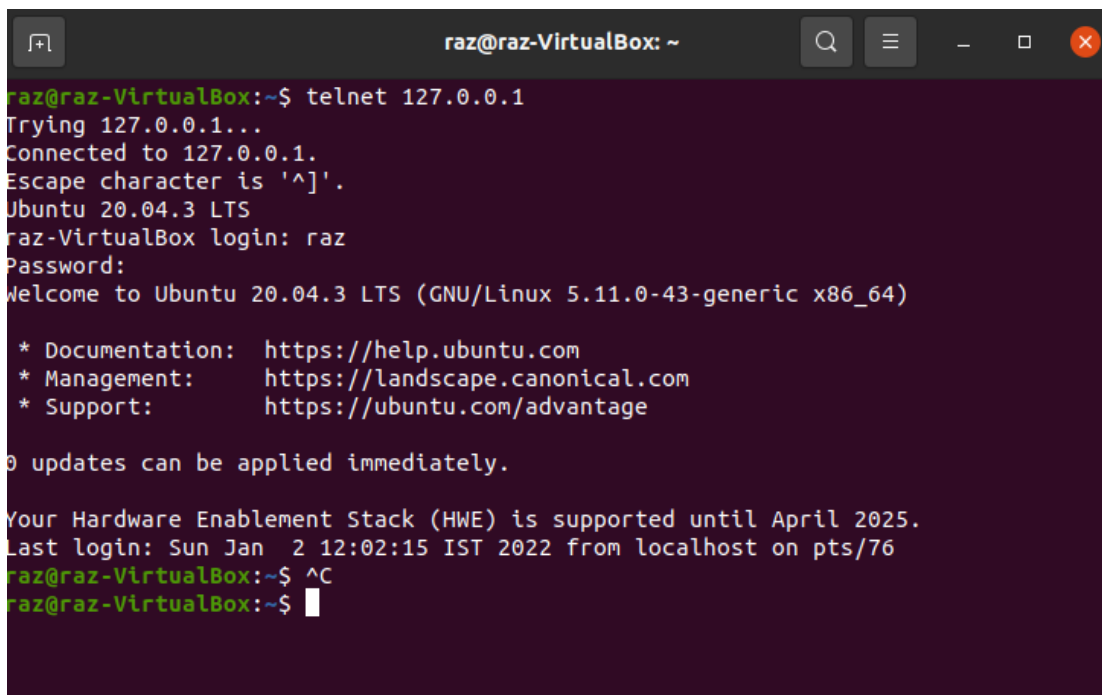
בטרמינל אחד הפעלנו את התוכנית בשביל ללכוד את הסיסמא-התוקף, בטרמינל השני ביצענו פקודה : telnet 127.0.0.1 - הנתקף ולכדנו מנות באינטרפייס lo בwireshark.

כאשר הנתקף הכניס את השם משתמש והסיסמא שלו התוקף אסף את הנתונים.

שם משתמש של הנתקף : raz

סיסמא: 207276775

ניתן לראות בהדפסות את נתוני הנתקף בצילומי המסך הבאים:
ביצוע ההכנסה מהטרמינל של הנתקף:



```

raz@raz-VirtualBox: ~$ telnet 127.0.0.1
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^['.
Ubuntu 20.04.3 LTS
raz-VirtualBox login: raz
Password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-43-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 updates can be applied immediately.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sun Jan  2 12:02:15 IST 2022 from localhost on pts/76
raz@raz-VirtualBox:~$ ^C
raz@raz-VirtualBox:~$

```

ההדפסות על גבי המסך של התוקף:

שם המשתמש:

```

raz@raz-VirtualBox: ~/CLionProjects/ex6
SRC: 10.0.2.15
DEST: 10.0.2.15
SRC: 10.0.2.15
DEST: 10.0.2.15
SRC: 10.0.2.15
DEST: 10.0.2.15
SRC: 10.0.2.15
DEST: 10.0.2.15
SRC: 10.0.2.15
DEST: 10.0.2.15
SRC: 10.0.2.15
DEST: 10.0.2.15
SRC: 10.0.2.15
DEST: 10.0.2.15
SRC: 10.0.2.15
DEST: 10.0.2.15
SRC: 10.0.2.15
DEST: 10.0.2.15

```

```

raz@raz-VirtualBox: ~/CLionProjects/ex6
SRC: 127.0.0.1
DEST: 127.0.0.1
SRC: 127.0.0.1
DEST: 127.0.0.1
SRC: 127.0.0.1
DEST: 127.0.0.1
SRC: 127.0.0.1
DEST: 127.0.0.1
SRC: 127.0.0.1
DEST: 127.0.0.1
SRC: 127.0.0.1
DEST: 127.0.0.1
SRC: 127.0.0.1
DEST: 127.0.0.1
SRC: 127.0.0.1
DEST: 127.0.0.1
SRC: 127.0.0.1
DEST: 127.0.0.1

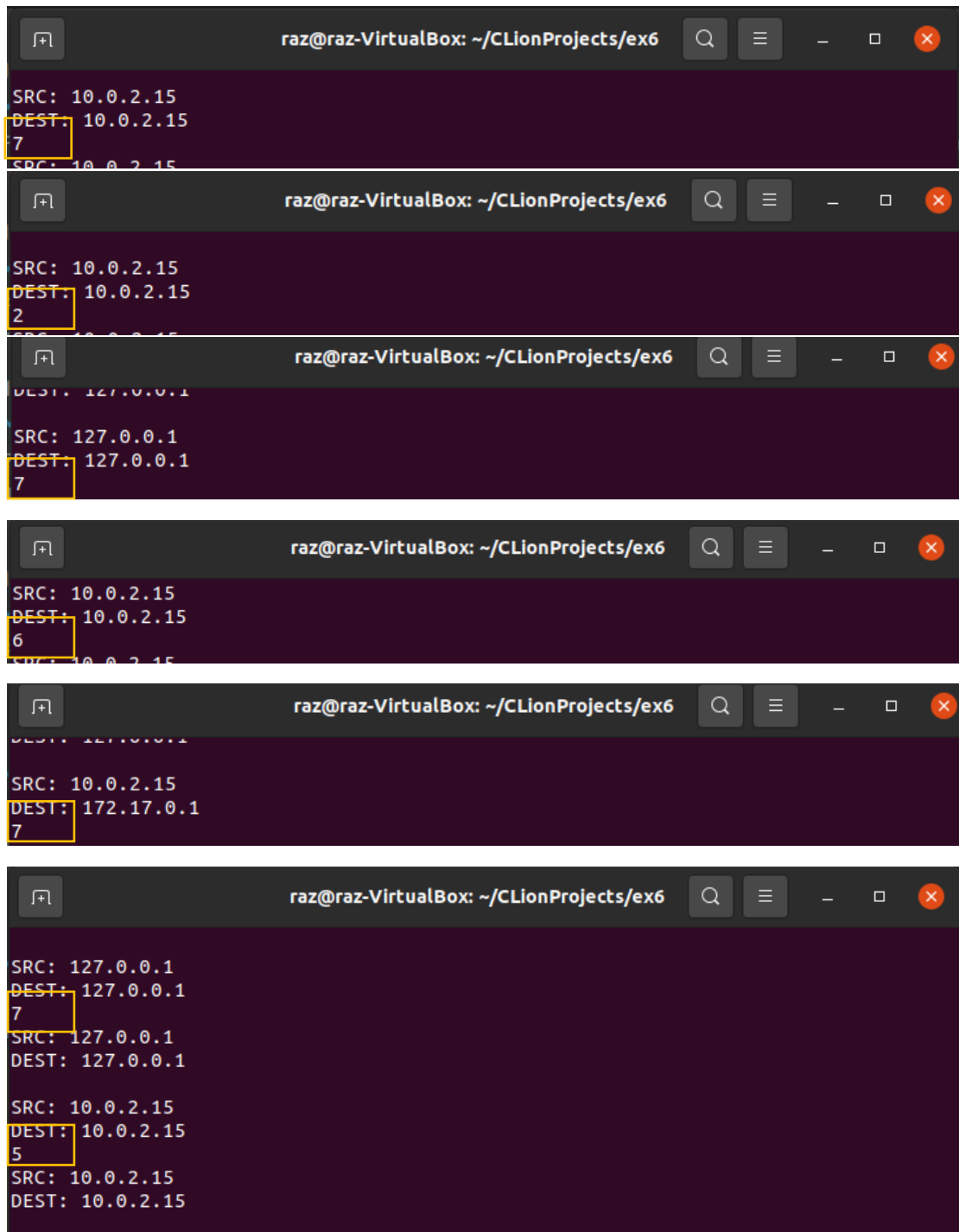
```

היסטוריה:

```

raz@raz-VirtualBox: ~/CLionProjects/ex6
SRC: 10.0.2.15
DEST: 10.0.2.15
2
raz@raz-VirtualBox: ~/CLionProjects/ex6
SRC: 10.0.2.15
DEST: 10.0.2.15
0

```

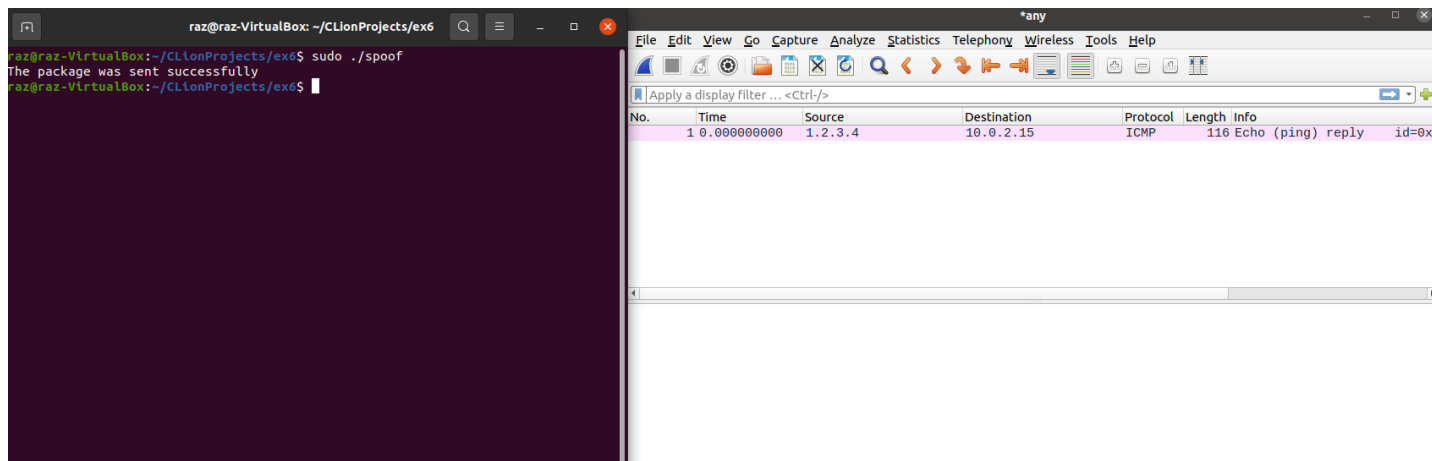


הובא צילום של כל ספרה היות ובהדפסה זה מודפס מספר פעמים.

Task 2.2: Spoofing

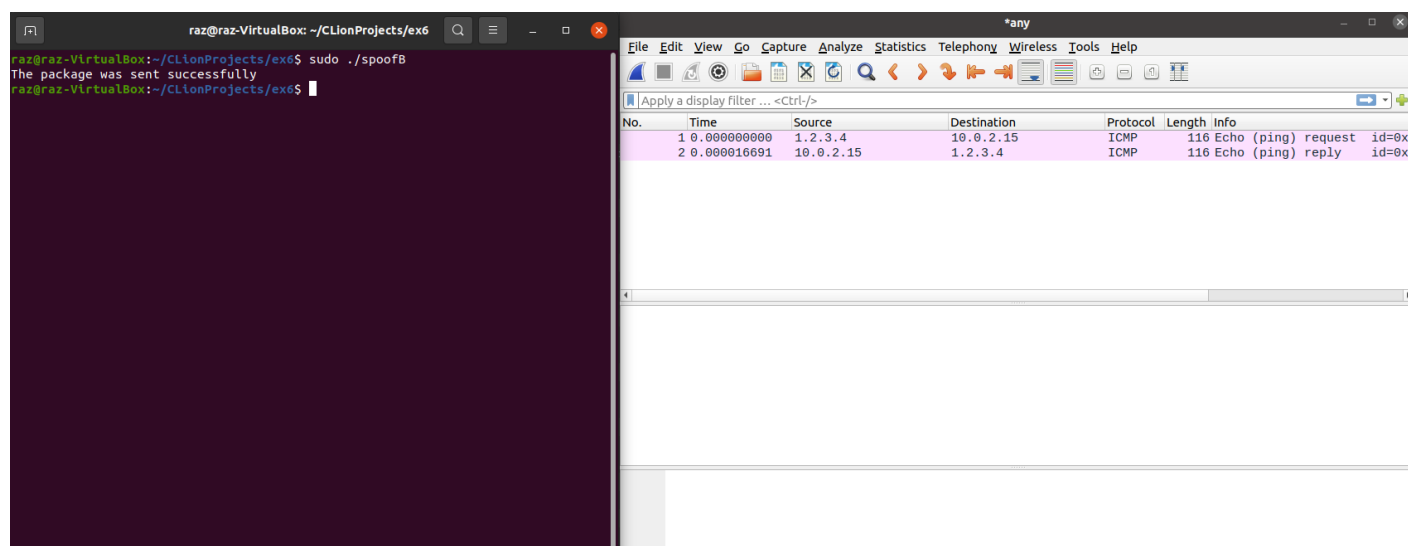
(spoof.c) Task 2.2A

במשימה זו התבקשנו לכתוב תוכנית זיוף בשפת C ולהראות שהתוכנית שולחת בהצלחה חבילות IP מזויפות. התוכנית שלחה חבילה מזויפת מכתובת "1.2.3.4" אשר איננה קיימת וראינו שהתקבלה מכתובת זו תגובה.



(spoofB.c) Task 2.2B

במשימה זו התבקשנו לכתוב תוכנית דומה לזו אשר כתבנו בסעיף הקודם רק שזו צריכה גם לייצר חבילת ICMP echo request. המשימה התבצעה על ידי לקיחת התוכנית מהסעיף הקודם והוספת השינוי `icmp->icmp_type=8` מ-0 ל-8 היות וכאשר ה-ICMP TYPE הוא 8 זה request וכאשר הוא 0 זה reply.



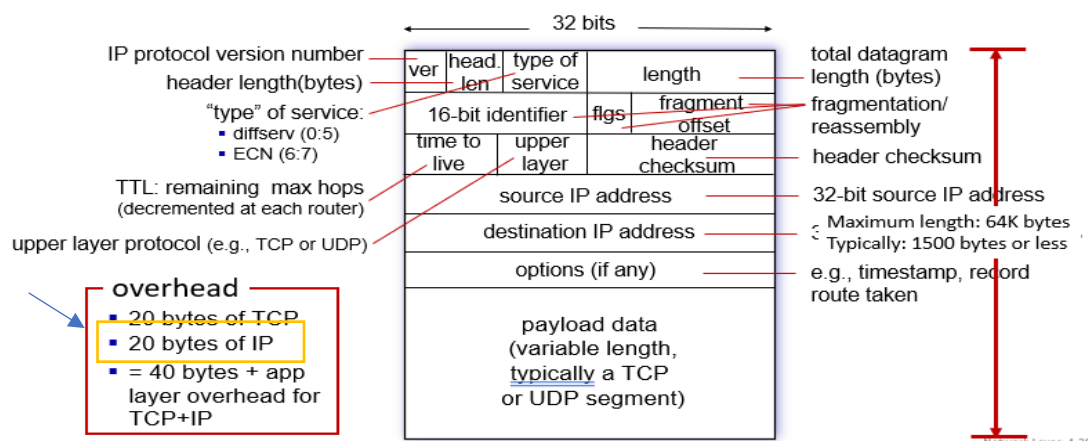
- Question 4. Can you set the IP packet length field to an arbitrary value, regardless of how big the actual packet is?
- Question 5. Using the raw socket programming, do you have to calculate the checksum for the IP header?
- Question 6. Why do you need the root privilege to run the programs that use raw sockets? Where does the program fail if executed without the root privilege?

פתרון שאלה 4:

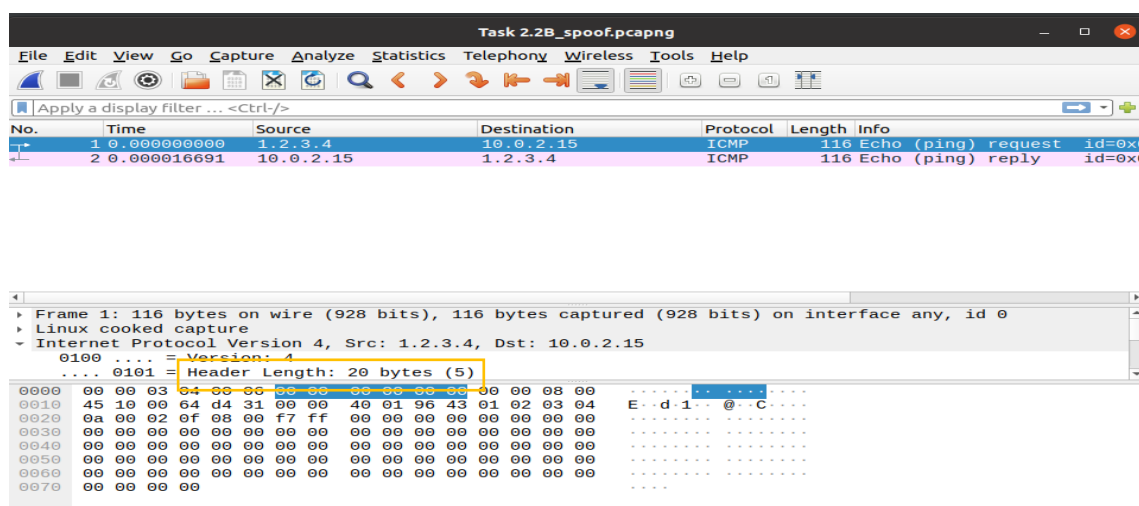
אנו הגדרנו בתוכנית spoofB.c את גודל חבילת ה-IP ל-100 באופן שרירותי מתוך למידה מההרצאות כי גודל של IP HEADER הוא 20 bytes ולכן גודל חבילת IP חייבת להיות גדולה מגודל זה.

השקופית מההרצאה:

IP Datagram format



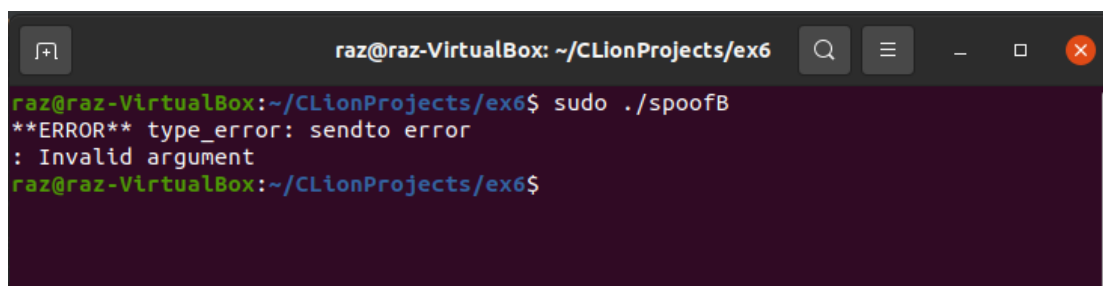
ניתן לראות בצילום המסך הבא מהתוכנית שלנו את הגודל המינימלי :



בשביל לבדוק האם זה אכן הגודל המינימלי, ואם הקטנת הגודל תשפיע על שליחת החבילה, נגדיר את גודל החבילה לגודל קטן מ-20 ונבדוק את השליחה:

```
ip->tot_len= htons( hostshort: 15);
```

שינינו את הגודל ל-15 וקיבלנו שגיאה בשלב של השליחה כפי שניתן לראות בצילום המסך הבא:



```
raz@raz-VirtualBox: ~/CLionProjects/ex6
raz@raz-VirtualBox:~/CLionProjects/ex6$ sudo ./spoofB
**ERROR** type_error: sendto error
: Invalid argument
raz@raz-VirtualBox:~/CLionProjects/ex6$
```

מכאן נסיק כי ניתן להגדיר את גודל חבילת IP לכל ערך אשר גדול מ-20.

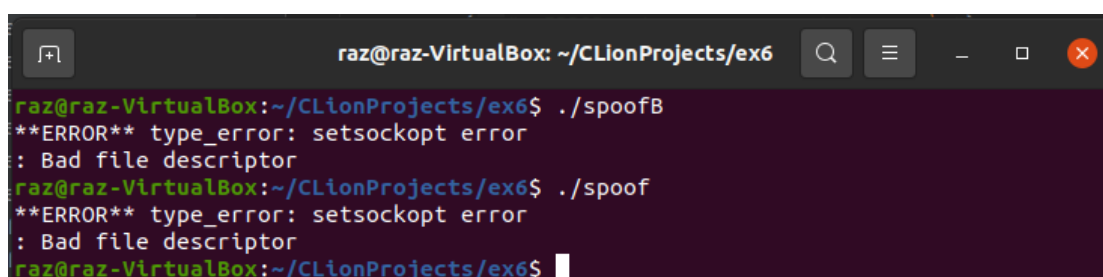
פתרון שאלה 5:

אין צורך בחישוב checksum ב- IP header היות והוא מחושב באופן אוטומטי על ידי מערכת ההפעלה. ניתן לראות בפתרון לשאלה 4, בתיעוד מהמצגת של ההרצאה כי ישנן 16 סיביות המיועדות עבור ה-checksum. מסיבה זו, לא מימשנו בקוד פונקציה אשר תחשב checksum עבור ה-IP header.

פתרון שאלה 6:

כל פתיחה של raw socket דורשת הרשאת מנהל כדי להפעיל את התוכנות המשתמשות בו. כאשר נשתמש בהרשאת מנהל נוכל לבצע שינויים ב-headers ויימנעו השינויים בהם מפני אנשים אשר לא מורשים לכך.

במשימות **Task 2.2A Task 2.2B** פתחתנו raw socket. כעת, נבדוק באיזה שלב התוכנית נכשלת אם היא לא מופעלת עם הרשאות מנהל:



```
raz@raz-VirtualBox: ~/CLionProjects/ex6
raz@raz-VirtualBox:~/CLionProjects/ex6$ ./spoofB
**ERROR** type_error: setsockopt error
: Bad file descriptor
raz@raz-VirtualBox:~/CLionProjects/ex6$ ./spoof
**ERROR** type_error: setsockopt error
: Bad file descriptor
raz@raz-VirtualBox:~/CLionProjects/ex6$
```

כפי שניתן לראות ההרצה נכשלה בשלב יצירת raw socket.

(SniffAndThenSpoof.c) Task 2.3

במשימה זו התבקשנו לבצע איחוד של sniffing and spoofing ולכן ביצענו חיבור של התוכנית sniff עם התוכנית spoof.

בטרמינל אחד הרצנו את התוכנית אשר כתבנו, בטרמינל שני ביצענו פינג לכתובת המזויפת "1.2.3.4" שהיא איננה קיימת. התוכנית שלנו זיהתה את ה ICMP echo request packet , הפיקה זיוף של ICMP echo reply packet ושלחה בחזרה למשתמש. ניתן לראות שבכל פעם שמתבצעת בקשה, התוכנית מגיבה מיד ושולחת תגובה.

בצילום המסך הבא הוכחנו כי על אף של-IP שהמשתמש שלח אליו פינג אין host המשתמש בכל זאת מקבל תשובה כאילו host קיים.

