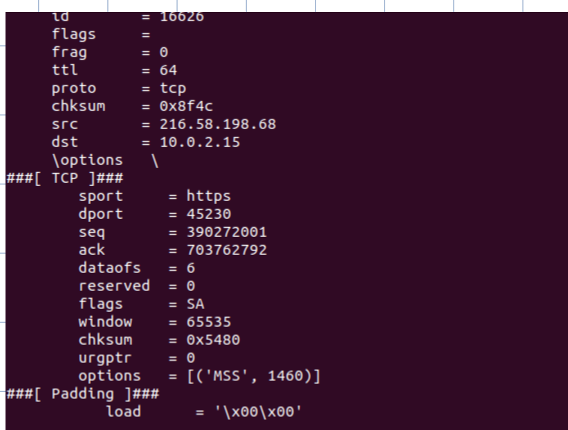
פרויקט גמר

* כל הקודים מצורפים בתיקייה לכל סעיף יהיה כתוב שם הקובץ הרלוונטי.

A1.1: קובץ הקוד בשם Sniff.py

כאשר הרצנו את הקובץ Sniffer עם פקודת Sudo המכונה פעלה וה Sniffer החל בהסנפת פאקטות (נכנס למצב "האזנה", למעשה ממתינה לכל פאקטה שנמסרה).



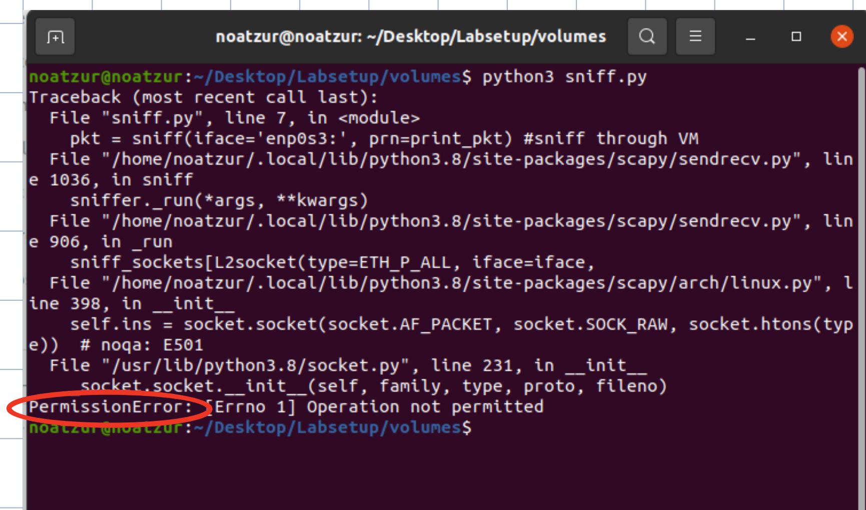
```
    id         = 16626
    flags      =
    frag       = 0
    ttl        = 64
    proto      = tcp
    chksum     = 0x8f4c
    src        = 216.58.198.68
    dst        = 10.0.2.15
    \options   \
###[ TCP ]###
       sport     = https
       dport     = 45230
       seq       = 390272001
       ack       = 703762792
       dataofs   = 6
       reserved  = 0
       flags     = SA
       window    = 65535
       chksum    = 0x5480
       urgptr    = 0
       options   = [('MSS', 1460)]
###[ Padding ]###
          load      = '\x00\x00'
```

דוגמא לפאקטת TCP "שנמסרה" ע"י הרצת הקובץ עם Sudo:

אך כאשר הרצנו את הפקודה ללא Sudo, קפצה שגיאת הרשאה, זאת כי רק משתמש בעל הרשאות Admin יכול להריץ את המכונה של ה Sniffer, זאת כי פונקציות בספריה של Scapy אשר בעזרתן מבצעים את ההסנפה, דורשים הרשאה זו.

כמו שלמדנו במעבדה, ה interface מוכנס למצב "מוניטור" (פרוץ) ולא נרצה שכל אחד עם גישה למחשב שהוא לא Admin יוכל לעשות זאת.

בפונ' של Scapy כמו פונקציה ה Sniff שדורשת הרשאה של מנהל
כדי שלא כל משתמש רגיל יוכל להסתכל על העבודה/ליצור פאקטות
מניפולציות/לגשת לשדות מסוימים בפאקטות.
הפונ' האלו ניגשות לכרטיס הרשת ובעצם פותחות raw socket
שמאפשר למשתמש לעשות עבודה שלא עם (נצרך הרשאות מנהל.
(ניתן לראות את הפונקציה עפ סדר קריאות שגרמו שגיאה עקב כך שלא
הרצנו עם סודו.)

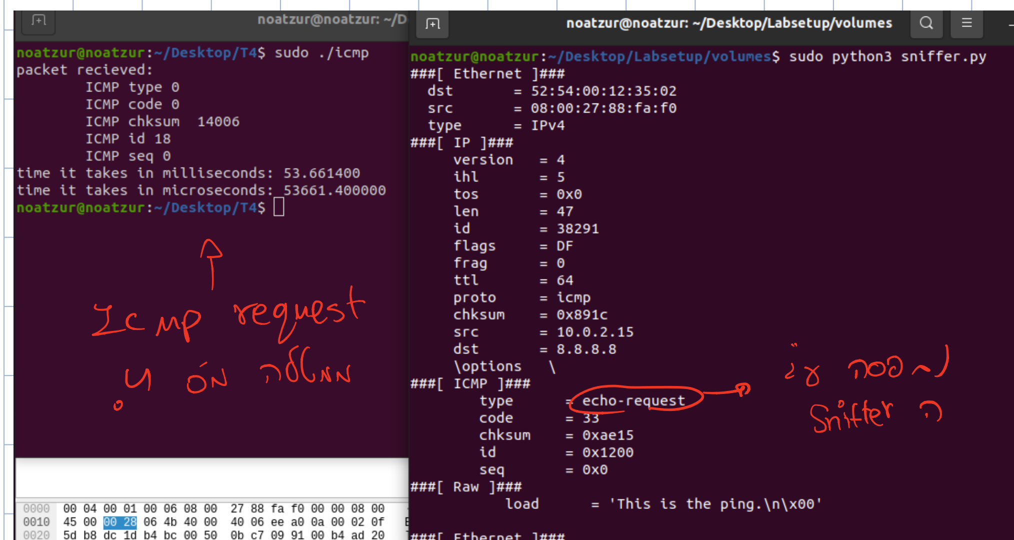


1.1B: Scapy's filter הקוד בקובץ בשם Sniffer.py

• ICMP - ע"י הכנסת filter לפונק' Sniff:

Sniff (iface='enp0s3', filter='icmp', prn=print_pkt)

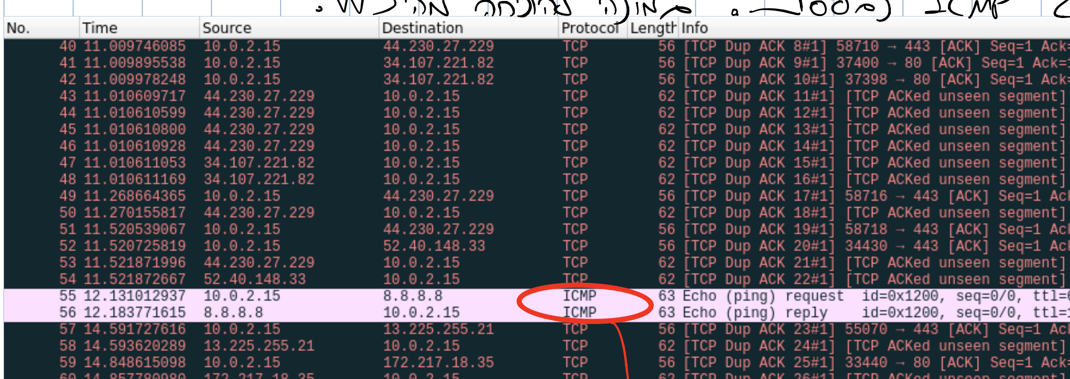
השלמנו שליחת Icmp request מהשאלה הקודמת.
הפעלנו את הSniffer ושלחנו את הIcmp request. נ"ין



noatzur@noatzur: ~/D
noatzur@noatzur:~/Desktop/T4$ sudo ./icmp
packet recieved:
ICMP type 0
ICMP code 0
ICMP chksum 14006
ICMP id 18
ICMP seq 0
time it takes in milliseconds: 53.661400
time it takes in microseconds: 53661.400000
noatzur@noatzur:~/Desktop/T4$
Icmp request
נשלחה מכאן
noatzur@noatzur: ~/Desktop/Labsetup/volumes
noatzur@noatzur:~/Desktop/Labsetup/volumes$ sudo python3 sniffer.py
###[Ethernet]###
dst = 52:54:00:12:35:02
src = 08:00:27:88:fa:f0
type = IPv4
###[IP]###
version = 4
ihl = 5
tos = 0x0
len = 47
id = 38291
flags = DF
frag = 0
ttl = 64
proto = icmp
chksum = 0x891c
src = 10.0.2.15
dst = 8.8.8.8
\options \
###[ICMP]###
type = echo-request
code = 33
chksum = 0xae15
id = 0x1200
seq = 0x0
###[Raw]###
load = 'This is the ping.\n\x00'
###[Ethernet]###
נתפסה ע"י הSniffer

(נתפסה גם הפקטה הreply שהתקבלה עבור הrequest)

בנוסף, דפדפן אינטרנט היה פתוח במקביל כדי לוודא שרק פאקטים מסוג ICMP נתפסו. תמונה להוכחה מהWS:



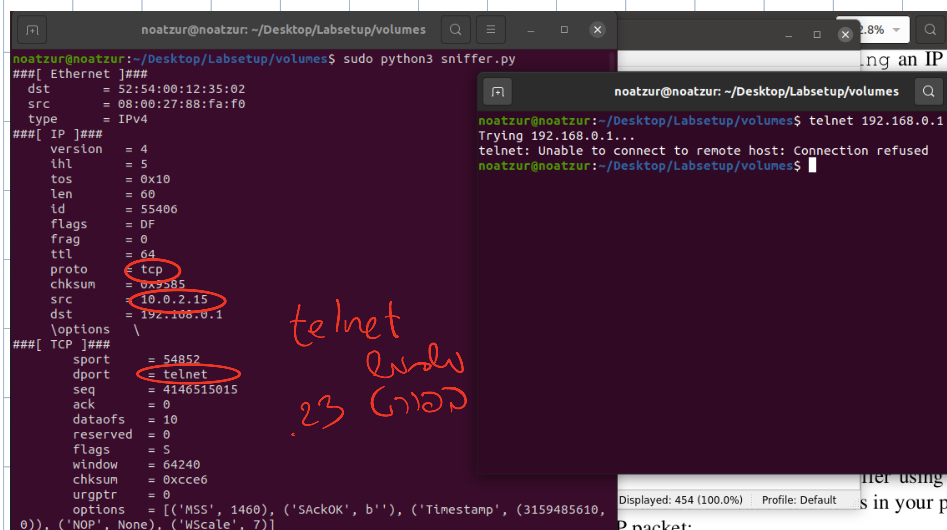
No. Time Source Destination Protocol Length Info
40 11.009746085 10.0.2.15 44.230.27.229 TCP 56 [TCP Dup ACK 8#1] 58710 → 443 [ACK] Seq=1 Ack=
41 11.009895538 10.0.2.15 34.107.221.82 TCP 56 [TCP Dup ACK 9#1] 37400 → 80 [ACK] Seq=1 Ack=1
42 11.009978248 10.0.2.15 34.107.221.82 TCP 56 [TCP Dup ACK 10#1] 37398 → 80 [ACK] Seq=1 Ack=
43 11.010609717 44.230.27.229 10.0.2.15 TCP 62 [TCP Dup ACK 11#1] [TCP ACKed unseen segment]
44 11.010610599 44.230.27.229 10.0.2.15 TCP 62 [TCP Dup ACK 12#1] [TCP ACKed unseen segment]
45 11.010610800 44.230.27.229 10.0.2.15 TCP 62 [TCP Dup ACK 13#1] [TCP ACKed unseen segment]
46 11.010610928 44.230.27.229 10.0.2.15 TCP 62 [TCP Dup ACK 14#1] [TCP ACKed unseen segment]
47 11.010611053 34.107.221.82 10.0.2.15 TCP 62 [TCP Dup ACK 15#1] [TCP ACKed unseen segment]
48 11.010611169 34.107.221.82 10.0.2.15 TCP 62 [TCP Dup ACK 16#1] [TCP ACKed unseen segment]
49 11.268664365 10.0.2.15 44.230.27.229 TCP 56 [TCP Dup ACK 17#1] 58716 → 443 [ACK] Seq=1 Ack
50 11.270155817 44.230.27.229 10.0.2.15 TCP 62 [TCP Dup ACK 18#1] [TCP ACKed unseen segment]
51 11.520539067 10.0.2.15 44.230.27.229 TCP 56 [TCP Dup ACK 19#1] 58718 → 443 [ACK] Seq=1 Ack
52 11.520725819 10.0.2.15 52.40.148.33 TCP 56 [TCP Dup ACK 20#1] 34430 → 443 [ACK] Seq=1 Ack
53 11.521871996 44.230.27.229 10.0.2.15 TCP 62 [TCP Dup ACK 21#1] [TCP ACKed unseen segment]
54 11.521872667 52.40.148.33 10.0.2.15 TCP 62 [TCP Dup ACK 22#1] [TCP ACKed unseen segment]
55 12.131012937 10.0.2.15 8.8.8.8 ICMP 63 Echo (ping) request id=0x1200, seq=0/0, ttl=6
56 12.183771615 8.8.8.8 10.0.2.15 ICMP 63 Echo (ping) reply id=0x1200, seq=0/0, ttl=1
57 14.591727616 10.0.2.15 13.225.255.21 TCP 56 [TCP Dup ACK 23#1] 55070 → 443 [ACK] Seq=1 Ack
58 14.593620289 13.225.255.21 10.0.2.15 TCP 62 [TCP Dup ACK 24#1] [TCP ACKed unseen segment]
59 14.848615098 10.0.2.15 172.217.18.35 TCP 56 [TCP Dup ACK 25#1] 33440 → 80 [ACK] Seq=1 Ack=
60 14.857780980 172.217.18.35 10.0.2.15 TCP 62 [TCP Dup ACK 26#1] [TCP ACKed unseen segment]

רק 2 אלו הופיעו.

- כאן התבקשנו לבצע פילטר של פאקט TCP אשר מגיע מ- IP מסוים ומפורט 23. (telnet)

Sniff(iface='enp0s3', filter='tcp and host 10.0.2.15 and dst port 23', prn=print_pkt)

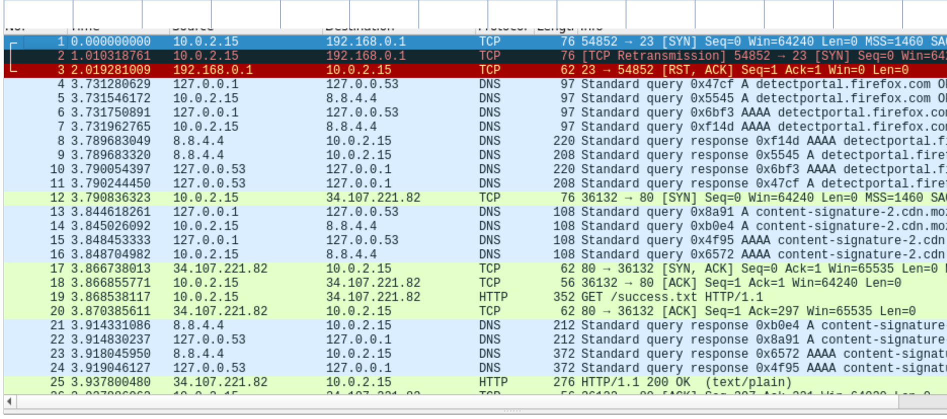
לפני שנבדוק את עצמנו, בדקנו את השלבים הבאים:
כאשר הפעלנו את ה Sniffer, פתחנו גם דפדפן אינטרנט.
שום פאקטה לא נתפסה ע"י ה Sniffer.
פתחנו terminal נוסף! ביצענו שיחה ע"י פרוטוקול telnet.
פרוטוקול telnet אשר שימוש בפורט 23,
ויוצר חיבור TCP.



```
noatzur@noatzur:~/Desktop/Labsetup/volumes$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst       = 52:54:00:12:35:02
  src       = 08:00:27:88:fa:f0
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 60
     id        = 55406
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x9585
     src       = 10.0.2.15
     dst       = 192.168.0.1
     \options   \
###[ TCP ]###
        sport     = 54852
        dport     = telnet
        seq       = 4146515015
        ack       = 0
        dataofs   = 10
        reserved  = 0
        flags     = S
        window    = 64240
        chksum    = 0xcce6
        urgptr    = 0
        options   = [('MSS', 1460), ('SAckOK', b''), ('Timestamp', (3159485610, 0)), ('NOP', None), ('WScale', 7)]
```

```
noatzur@noatzur:~/Desktop/Labsetup/volumes$ telnet 192.168.0.1
Trying 192.168.0.1...
telnet: Unable to connect to remote host: Connection refused
noatzur@noatzur:~/Desktop/Labsetup/volumes$
```


רק פאקטות מ IP 10.0.2.15 שהעברו ל dport 23 נתפסו:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	192.168.0.1	TCP	76	54852 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SAC
2	1.010318761	10.0.2.15	192.168.0.1	TCP	76	[TCP Retransmission] 54852 → 23 [SYN] Seq=0 Win=642
3	2.019281009	192.168.0.1	10.0.2.15	TCP	62	23 → 54852 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4	3.731280629	127.0.0.1	127.0.0.53	DNS	97	Standard query 0x47cf A detectportal.firefox.com OP
5	3.731546172	10.0.2.15	8.8.4.4	DNS	97	Standard query 0x5545 A detectportal.firefox.com OP
6	3.731750891	127.0.0.1	127.0.0.53	DNS	97	Standard query 0x6bf3 AAAA detectportal.firefox.com
7	3.731962765	10.0.2.15	8.8.4.4	DNS	97	Standard query 0xf14d AAAA detectportal.firefox.com
8	3.789683049	8.8.4.4	10.0.2.15	DNS	220	Standard query response 0xf14d AAAA detectportal.fi
9	3.789683320	8.8.4.4	10.0.2.15	DNS	208	Standard query response 0x5545 A detectportal.firef
10	3.790054397	127.0.0.53	127.0.0.1	DNS	220	Standard query response 0x6bf3 AAAA detectportal.fi
11	3.790244450	127.0.0.53	127.0.0.1	DNS	208	Standard query response 0x47cf A detectportal.firef
12	3.790836323	10.0.2.15	34.107.221.82	TCP	76	36132 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SAC
13	3.844618261	127.0.0.1	127.0.0.53	DNS	108	Standard query 0x8a91 A content-signature-2.cdn.moz
14	3.845026092	10.0.2.15	8.8.4.4	DNS	108	Standard query 0xb0e4 A content-signature-2.cdn.moz
15	3.848453333	127.0.0.1	127.0.0.53	DNS	108	Standard query 0x4f95 AAAA content-signature-2.cdn.
16	3.848704982	10.0.2.15	8.8.4.4	DNS	108	Standard query 0x6572 AAAA content-signature-2.cdn.
17	3.866738013	34.107.221.82	10.0.2.15	TCP	62	80 → 36132 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 M
18	3.866855771	10.0.2.15	34.107.221.82	TCP	56	36132 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
19	3.868538117	10.0.2.15	34.107.221.82	HTTP	352	GET /success.txt HTTP/1.1
20	3.870385611	34.107.221.82	10.0.2.15	TCP	62	80 → 36132 [ACK] Seq=1 Ack=297 Win=65535 Len=0
21	3.914331086	8.8.4.4	10.0.2.15	DNS	212	Standard query response 0xb0e4 A content-signature-
22	3.914830237	127.0.0.53	127.0.0.1	DNS	212	Standard query response 0x8a91 A content-signature-
23	3.918045950	8.8.4.4	10.0.2.15	DNS	372	Standard query response 0x6572 AAAA content-signatu
24	3.919046127	127.0.0.53	127.0.0.1	DNS	372	Standard query response 0x4f95 AAAA content-signatu
25	3.937800480	34.107.221.82	10.0.2.15	HTTP	276	HTTP/1.1 200 OK (text/plain)

הפאקטה הראשונה היא מפורט 23 כפי שניתן לראות את IP 10.0.2.15 אך ישנם גם פאקטות נוספות מפורטים אחרים וIP שונים שלא הוספנו.

• בחרנו particular Subnet 128.230.0.0/16.

* לקחתי מאתר ipinfo.io אתר של אוניברסיטת Syracuse
עם IP <- 128.230.18.123 כדי לגלוש לאתר ב-Subnet המצוין מלמעלה.

Sniff (iface ='enp0s3', filter ='net 128.230.0.0/16', prn=print-Pkt)

```
###[ Ethernet ]###
  dst       = 52:54:00:12:35:02
  src       = 08:00:27:88:fa:f0
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 14743
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x61c9
     src       = 10.0.2.15
     dst       = 128.230.18.123
     \options   \
###[ TCP ]###
        sport     = 58870
        dport     = https
        seq       = 1149736888
        ack       = 481152953
        dataofs   = 5
        reserved  = 0
        flags     = A
        window    = 63986
        chksum    = 0x9f8a
        urgptr    = 0
        options   = []
```

אתר האוניברסיטה ב Subnet המתאים.

הסנפה - פאקטה ע"י ה Sniffer

תמונה מ-WireShark אודות הסנפה:

No.	Time	Source	Destination	Protocol	Length	Info
246	7.627144522	216.58.205.196	10.0.2.15	TLSv1.2	2909	Application Data, Application Data, Application Data, Applica…
247	7.627196557	10.0.2.15	216.58.205.196	TCP	54	55128 → 443 [ACK] Seq=710 Ack=13249 Win=62480 Len=0
248	7.628415106	128.230.18.63	10.0.2.15	TCP	60	443 → 41768 [ACK] Seq=129972 Ack=2012 Win=65535 Len=0
249	7.636230395	10.0.2.15	216.58.205.196	TLSv1.2	93	Application Data
250	7.638268243	216.58.205.196	10.0.2.15	TCP	60	443 → 55128 [ACK] Seq=13249 Ack=749 Win=65535 Len=0
251	7.654356881	69.25.199.41	10.0.2.15	TLSv1.3	214	Server Hello
252	7.654430099	10.0.2.15	69.25.199.41	TCP	54	46260 → 443 [ACK] Seq=514 Ack=161 Win=64080 Len=0
253	7.657573723	69.25.199.41	10.0.2.15	TLSv1.3	4314	Change Cipher Spec
254	7.657635223	10.0.2.15	69.25.199.41	TCP	54	46260 → 443 [ACK] Seq=514 Ack=4421 Win=62480 Len=0
255	7.658155907	8.8.4.4	10.0.2.15	DNS	132	Standard query response 0xf38c AAAA syracuse.edu SOA ns1.syr…
256	7.658717424	69.25.199.41	10.0.2.15	TLSv1.3	1153	Application Data
257	7.658756384	10.0.2.15	69.25.199.41	TCP	54	46260 → 443 [ACK] Seq=514 Ack=5520 Win=61381 Len=0
258	7.665592056	69.25.199.41	10.0.2.15	TLSv1.3	648	Application Data
259	7.665652276	10.0.2.15	69.25.199.41	TCP	54	46260 → 443 [ACK] Seq=514 Ack=6114 Win=63900 Len=0
260	7.677081332	10.0.2.15	69.25.199.41	TLSv1.3	134	Change Cipher Spec, Application Data
261	7.677690018	10.0.2.15	69.25.199.41	TLSv1.3	413	Application Data
262	7.678631391	69.25.199.41	10.0.2.15	TCP	60	443 → 46260 [ACK] Seq=6114 Ack=594 Win=65535 Len=0
263	7.678631760	69.25.199.41	10.0.2.15	TCP	60	443 → 46260 [ACK] Seq=6114 Ack=953 Win=65535 Len=0
264	7.731173737	128.230.18.123	10.0.2.15	TLSv1.2	89	Application Data
265	7.731237052	10.0.2.15	128.230.18.123	TCP	54	58834 → 443 [ACK] Seq=471 Ack=193 Win=63900 Len=0
266	7.746661101	10.0.2.15	172.217.171.232	TLSv1.2	147	Application Data
267	7.748230440	172.217.171.232	10.0.2.15	TCP	60	443 → 58252 [ACK] Seq=164 Ack=244 Win=65535 Len=0

רק הפאקטה שמגיעה ל IP הזה הוסנפה.

1.2: spoof

הקוד בקובץ בשם spoofICMP.py

בסעיף זה זייפנו פקטת ICMP פשוטה (request) וקיבלנו עבורה reply.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_88:fa:f0	Broadcast	ARP	42	Who has 10.0.2.3? Tell 10.0.2.15
2	0.000461128	RealtekU_12:35:03	PcsCompu_88:fa:f0	ARP	60	10.0.2.3 is at 52:54:00:12:35:03
3	0.031056211	10.0.2.15	10.0.2.3	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply
4	0.031326155	10.0.2.3	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (reque

ה request - הפקטה ששלחנו וקיבלנו reply חזרה

שליחת הפקטה שזייפנו ע"י קוד בפייתון (בקובץ המצורף)

```
noatzur@noatzur: ~/Desktop/Labsetup/volumes
noatzur@noatzur:~/Desktop/Labsetup/volumes$ sudo python3 spoofICMP.py
[sudo] password for noatzur:
.
Sent 1 packets.
noatzur@noatzur:~/Desktop/Labsetup/volumes$ sudo python3 spoofICMP.py
.
Sent 1 packets.
noatzur@noatzur:~/Desktop/Labsetup/volumes$
```

הדרך לו כדי שנוכל לראות מאור של המבנה של IP object שנוצר.

```
3 a=IP()
4 a.dst = '10.0.2.3'
5 b = ICMP()
6 p=a/b
7 send(p)
8 print("lis(a) : ")
9 ls(a)
```

```
^Z
[1]+  Stopped                 sudo python3 sniffer.py
noatzur@noatzur:~/Desktop/Labsetup/volumes$ sudo python3 spoofICMP.py
.
Sent 1 packets.
lis(a) :
version    : BitField  (4 bits)                = 4               (4)
ihl        : BitField  (4 bits)                = None            (None)
tos        : XByteField                        = 0               (0)
len        : ShortField                        = None            (None)
id         : ShortField                        = 1               (1)
flags      : FlagsField  (3 bits)              = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField  (13 bits)               = 0               (0)
ttl        : ByteField                         = 64              (64)
proto      : ByteEnumField                     = 0               (0)
chksum     : XShortField                       = None            (None)
src        : SourceIPField                     = '10.0.2.15'     (None)
dst        : DestIPField                       = '10.0.2.3'      (None)
options    : PacketListField                   = []              ([])
noatzur@noatzur:~/Desktop/Labsetup/volumes$
```

1.3: traceroute

קובץ בשם ttl.py

בסעיף זה, שלחנו פקטת ICMP לdestination IP - אוניברסיטת אריאל.

ורצינו לבדוק מה ה-ttl, כלומר כמה hops (routers) הפקטה מעבירה עד שמגיע יעדה. ביצענו בהתחלה לולאת for שמריצה את ה שליחה וכל פעם משנה את ה ttl ומעלה אותו ב-1 מה ttl הקודם.

שמנו את הלולאה בהרחבה שרצה 50 פעמים.
ב-wireshark ראינו שכאשר ה-ttl הגיע ל-15
קיבלנו reply בפעם הראשונה. מכיון (ניתן להבין שאחרי שהפקטה
עברה כ- 15 hops, היא הגיעה ל-destination

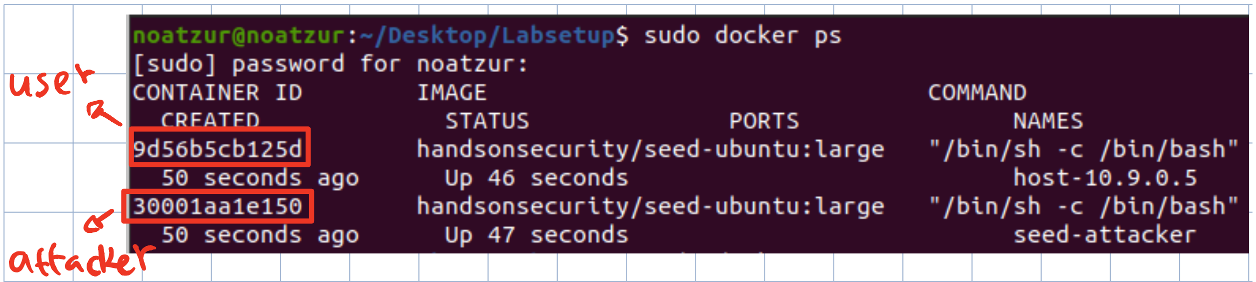
0-15

No.	Time	Source	Destination	Protocol	Length	Info
5	0.056847681	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=0 (no response f…
6	0.057008557	10.0.2.2	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
7	0.096977099	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=1 (no response f…
8	0.097166992	10.0.2.2	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
9	0.140823626	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=2 (no response f…
10	0.144041676	192.168.0.1	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
11	0.179104207	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=3 (no response f…
12	0.181271716	192.168.1.1	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
13	0.216481892	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=4 (no response f…
14	0.227649339	10.174.128.1	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
15	0.253163583	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=5 (no response f…
16	0.292638658	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=6 (no response f…
17	0.314294796	172.17.3.101	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
18	0.328786915	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=7 (no response f…
19	0.365740462	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=8 (no response f…
20	0.378454252	212.25.116.149	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
21	0.405161187	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=9 (no response f…
22	0.420958402	10.25.19.10	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
23	0.440497930	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=10 (no response …
24	0.451857419	212.25.77.14	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
25	0.480602529	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=11 (no response …
26	0.491926007	10.90.99.13	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
27	0.517199443	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=12 (no response …
28	0.560780487	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=13 (no response …
29	0.569996997	74.125.51.88	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
30	0.600897579	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=14 (no response …
31	0.614191776	74.125.244.225	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
32	0.641453005	10.0.2.15	34.96.118.58	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=15 (reply in 34)
33	0.651938050	72.14.234.95	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
34	0.695357519	34.96.118.58	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=111 (request in …

reply

1.4 Sniff & spoof בקובץ Sniff_spoof.py

בסעיף זה השתמשנו ב-2 מכונות או ב docker שהפעל במכונה 1.
שימוש ב docker נעשה ע"י הפקודות build docker-compose
ואז up. פתחנו 2 טרמינלים ב vm, בראשון הפעלנו את הcontainer
של המתקף ע"י פקודות ps ו exec (ציינו מה id של ה-container הרלוונטי
ופתחנו אותו.)
בטרמינל השני פתחנו ע"י אותם צעדים את ה Container
של ה user "הרגיל".



```
noatzur@noatzur:~/Desktop/Labsetup$ sudo docker ps
[sudo] password for noatzur:
CONTAINER ID        IMAGE                                 COMMAND
  CREATED             STATUS              PORTS               NAMES
9d56b5cb125d        handsonsecurity/seed-ubuntu:large   "/bin/sh -c /bin/bash"
  50 seconds ago      Up 46 seconds                           host-10.9.0.5
30001aa1e150        handsonsecurity/seed-ubuntu:large   "/bin/sh -c /bin/bash"
  50 seconds ago      Up 47 seconds                           seed-attacker
```


בקובץ ה Sniff_spoof שינינו את פונ' ה Sniff כך שיקבל
כפרמטר של iface את השם של ה interface של ה container
כמו שהסברנו בחלק המשימה (מתחיל ב -br)
ואז הפונ' שמופעלת ברגע שהוספנו פאקטה מסוג ICMP.
הפונ' החדשה מקבלת את הפאקטה שנתפסה, ויוצרת פאקטה
חדשה מסוג ICMP. הפאקטה החדשה מורכבת מהשדות הבאים:

dest IP - לאן הפאקטה נועדה

Src IP - ממי נשלחה.

Seq - מס' הפאקטה מתוך סדר הפאקטות שנשלחו (על מנת
לשמור על סדר שליחה וקבלה).

load - המידע שמכילה הפאקטה.

נצור מכל שדה אלו פאקטה חדשה, כך שה dst IP שלה יהיה
ה-src של הפאקטה שהתקבלה (שלחה הping); src IP שלה יהיה
ה-dst של הפאקטה שנשלחה. כמו כן ה type של ה Icmp
הוא 0 כלו' מסוג reply (request=8) וכך בעצם יצרנו הודעת
reply מזויפת ל IP שאינו קיים כלל.

ב Container של ה User בצענו את פקודת ה ping עם הIP
המבוקש וב Container של המתקיף, הרצנו את הקובץ Sniff_spoof

מתקיין ה- Volumes המשפר.

• ping 1.2.3.4

user מבצע ping ל 1.2.3.4

```
root@9d56b5cb125d:/# 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=80.2 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=18.9 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=26.8 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=30.7 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=28.8 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=28.9 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=40.2 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=30.2 ms
64 bytes from 1.2.3.4: icmp_seq=9 ttl=64 time=24.4 ms
64 bytes from 1.2.3.4: icmp_seq=10 ttl=64 time=31.2 ms
64 bytes from 1.2.3.4: icmp_seq=11 ttl=64 time=18.2 ms
^Z
[11]+  Stopped                 ping 1.2.3.4
root@9d56b5cb125d:/#
```

הכתובת הזו לא תקינה, כלו' לא קיים host אשר מחזיק בה. ובכל זאת התקבלו פונגים כאילו באמת host מסוים שיש לו את ה IP הזה קיבל את ה request והחזיר reply. תמונה מה-wireshark של אותה הרצה:

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

No.	Time	Source	Destination	Protocol	Length	Info
74	23.015656423	10.0.2.15	52.35.31.120	TCP	56	42678 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
75	23.016715500	10.0.2.15	52.35.31.120	TLSv1.2	269	Client Hello
76	23.017377296	52.35.31.120	10.0.2.15	TCP	62	443 → 42678 [ACK] Seq=1 Ack=214 Win=65535 Len=0
77	23.242578324	52.35.31.120	10.0.2.15	TLSv1.2	1516	Server Hello
78	23.242624848	10.0.2.15	52.35.31.120	TCP	56	42678 → 443 [ACK] Seq=214 Ack=1461 Win=63900 Len=0
79	23.243575163	52.35.31.120	10.0.2.15	TLSv1.2	1997	Certificate, Server Key Exchange, Server Hello Done
80	23.243626104	10.0.2.15	52.35.31.120	TCP	56	42678 → 443 [ACK] Seq=214 Ack=3402 Win=62480 Len=0
81	23.249054309	10.0.2.15	52.35.31.120	TLSv1.2	182	Client Key Exchange, Change Cipher Spec, Encrypted Handshake …
82	23.249364020	52.35.31.120	10.0.2.15	TCP	62	443 → 42678 [ACK] Seq=3402 Ack=340 Win=65535 Len=0
83	23.251283117	10.0.2.15	52.35.31.120	TLSv1.2	2892	Application Data
84	23.251621684	52.35.31.120	10.0.2.15	TCP	62	443 → 42678 [ACK] Seq=3402 Ack=1800 Win=65535 Len=0
85	23.251621904	52.35.31.120	10.0.2.15	TCP	62	443 → 42678 [ACK] Seq=3402 Ack=3176 Win=65535 Len=0
86	23.351946293	10.9.0.5	1.2.3.4	ICMP	100	Echo (ping) request id=0x0018, seq=10/2560, ttl=64 (no respo…
87	23.351946293	10.9.0.5	1.2.3.4	ICMP	100	Echo (ping) request id=0x0018, seq=10/2560, ttl=64 (reply in…
88	[illegible]	10.0.2.15	1.2.3.4	ICMP	100	Echo (ping) request id=0x0018, seq=10/2560, ttl=63 (no respo…
89	23.383066671	1.2.3.4	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0018, seq=10/2560, ttl=64 (request …
90	23.383092523	1.2.3.4	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0018, seq=10/2560, ttl=64
91	[illegible]	52.35.31.120	10.0.2.15	TLSv1.2	107	Change Cipher Spec, Encrypted Handshake Message
92	23.473940668	10.0.2.15	52.35.31.120	TCP	56	42678 → 443 [ACK] Seq=3176 Ack=3453 Win=63900 Len=0
93	23.538839979	52.35.31.120	10.0.2.15	TLSv1.2	326	Application Data
94	23.538886135	10.0.2.15	52.35.31.120	TCP	56	42678 → 443 [ACK] Seq=3176 Ack=3723 Win=63900 Len=0
95	24.353006739	10.9.0.5	1.2.3.4	ICMP	100	Echo (ping) request id=0x0018, seq=11/2816, ttl=64 (no respo…
96	24.353006739	10.9.0.5	1.2.3.4	ICMP	100	Echo (ping) request id=0x0018, seq=11/2816, ttl=64 (reply in…
97	24.353153733	10.0.2.15	1.2.3.4	ICMP	100	Echo (ping) request id=0x0018, seq=11/2816, ttl=63 (no respo…
98	24.371106024	1.2.3.4	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0018, seq=11/2816, ttl=64 (request …
99	24.371132955	1.2.3.4	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0018, seq=11/2816, ttl=64

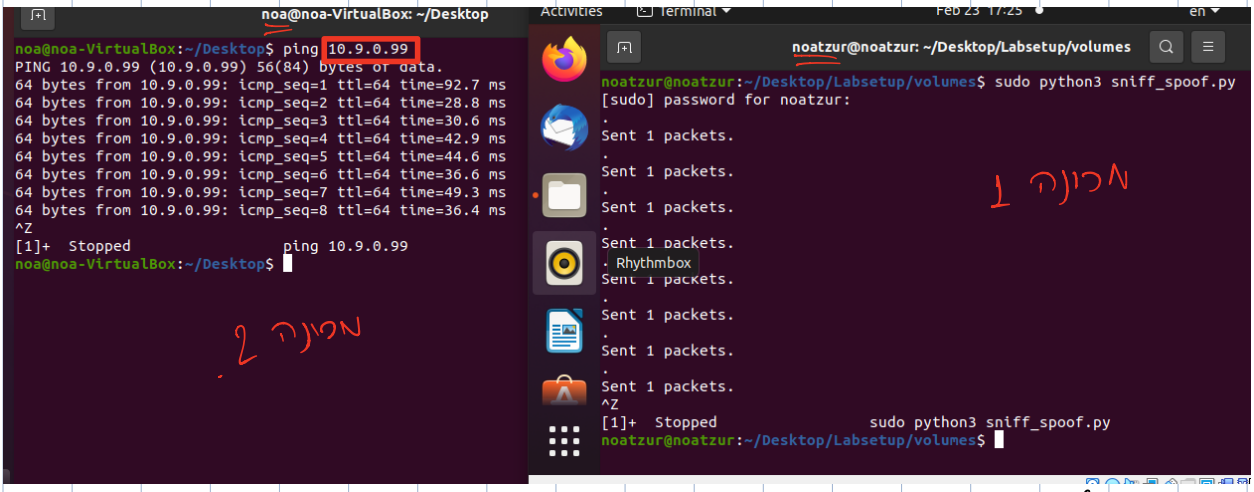
הודגש - ה reply המצופה -

• ביצוע ping ל- 10.9.0.99.

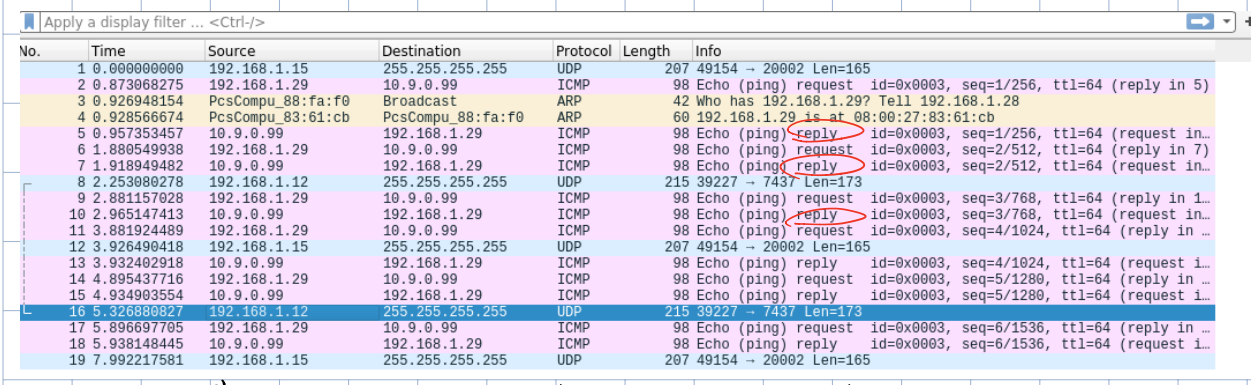
במכונה 1 הפעלנו את סקריפט ה- Sniff & Spoof שלנו.

במכונה השניה שלחנו ping ל IP המבוקש.

ניתן לראות שאכן התקבלו הודעות reply מהכתובת שזו כתובת שלא משויך אליה שום מכונה ולכן לא אמורים לקבל מענה מלכתחילה.



בנוסף - תמונה מ-WS שתוכל במקביל:



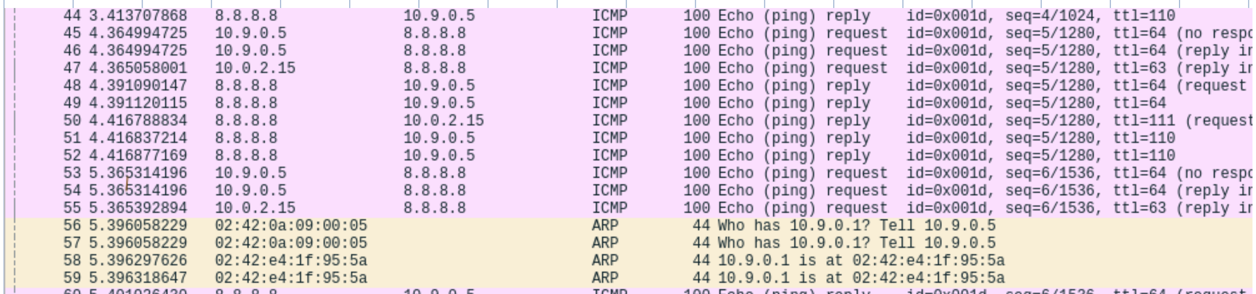
ניתן לראות שעבור כל request שנשלח ע"י ה ping התקבלה reply מה ICMP.

- ping 8.8.8.8

כתובת ה IP הזו תקינה ומשוייכת לserver של גוגל.
במקרה כזה, ה user אמור לקבל הודעת echo reply
באופן תקין. נזכיר שה sniff_spoof שלנו עדיין פועל בצד של
התוקף. במקרה כזה, נוצרת הודעת echo reply מזויפת מצד התוקף.
ה user מקבל 2 הודעות reply זהות, עם אותם פרטים בדיוק (כל
הודעה (למשל אותו seq number אשר אמור להיות ייחודי עבור תגובת
ping מסוימת) במקרה כזה ה user יראה כפילים של הודעות
ה - echo reply.

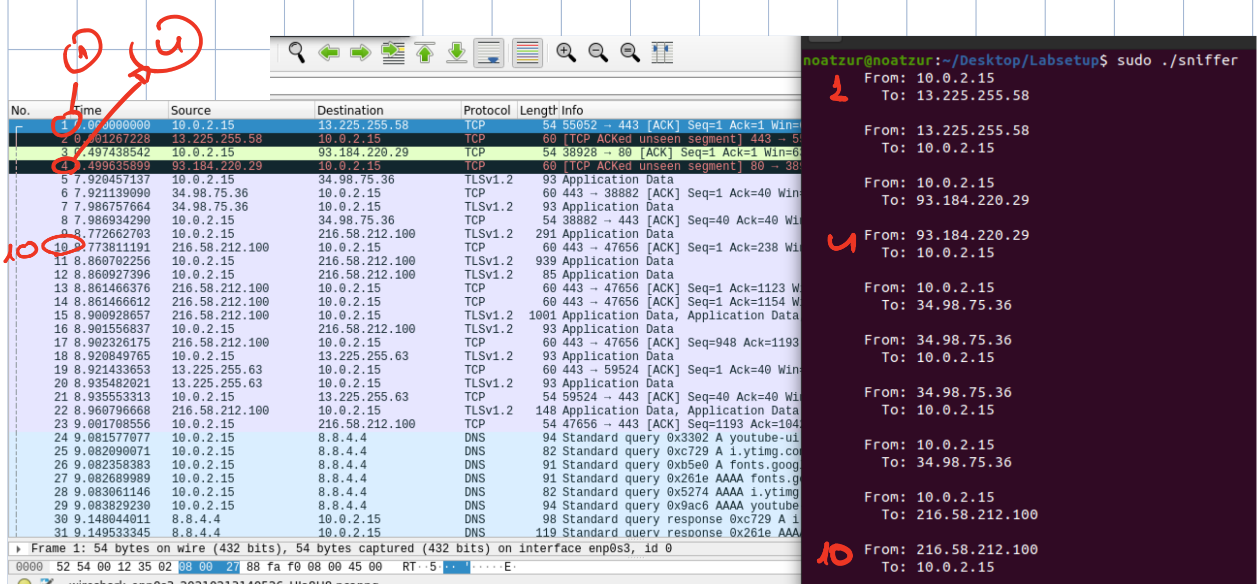
:Duplicates

```
[14]+  Stopped                 ping 8.8.8.8
root@9d56b5cb125d:/# 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=110 time=53.8 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=64 time=70.7 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=24.7 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=110 time=57.1 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=19.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=110 time=52.4 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=27.1 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=110 time=52.2 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=5 ttl=64 time=26.2 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=110 time=51.9 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=6 ttl=64 time=35.8 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=110 time=53.0 ms (DUP!)
^Z
[15]+  Stopped                 ping 8.8.8.8
root@9d56b5cb125d:/#
```


44 3.413707868 8.8.8.8 10.9.0.5 ICMP 100 Echo (ping) reply id=0x001d, seq=4/1024, ttl=110
45 4.364994725 10.9.0.5 8.8.8.8 ICMP 100 Echo (ping) request id=0x001d, seq=5/1280, ttl=64 (no respo
46 4.364994725 10.9.0.5 8.8.8.8 ICMP 100 Echo (ping) request id=0x001d, seq=5/1280, ttl=64 (reply in
47 4.365058001 10.0.2.15 8.8.8.8 ICMP 100 Echo (ping) request id=0x001d, seq=5/1280, ttl=63 (reply in
48 4.391090147 8.8.8.8 10.9.0.5 ICMP 100 Echo (ping) reply id=0x001d, seq=5/1280, ttl=64 (request
49 4.391120115 8.8.8.8 10.9.0.5 ICMP 100 Echo (ping) reply id=0x001d, seq=5/1280, ttl=64
50 4.416788834 8.8.8.8 10.0.2.15 ICMP 100 Echo (ping) reply id=0x001d, seq=5/1280, ttl=111 (request
51 4.416837214 8.8.8.8 10.9.0.5 ICMP 100 Echo (ping) reply id=0x001d, seq=5/1280, ttl=110
52 4.416877169 8.8.8.8 10.9.0.5 ICMP 100 Echo (ping) reply id=0x001d, seq=5/1280, ttl=110
53 5.365314196 10.9.0.5 8.8.8.8 ICMP 100 Echo (ping) request id=0x001d, seq=6/1536, ttl=64 (no respo
54 5.365314196 10.9.0.5 8.8.8.8 ICMP 100 Echo (ping) request id=0x001d, seq=6/1536, ttl=64 (reply in
55 5.365392894 10.0.2.15 8.8.8.8 ICMP 100 Echo (ping) request id=0x001d, seq=6/1536, ttl=63 (reply in
56 5.396058229 02:42:0a:09:00:05 ARP 44 Who has 10.9.0.1? Tell 10.9.0.5
57 5.396058229 02:42:0a:09:00:05 ARP 44 Who has 10.9.0.1? Tell 10.9.0.5
58 5.396297626 02:42:e4:1f:95:5a ARP 44 10.9.0.1 is at 02:42:e4:1f:95:5a
59 5.396318647 02:42:e4:1f:95:5a ARP 44 10.9.0.1 is at 02:42:e4:1f:95:5a

:2.1A הקוד בקובץ Sniff_921.C

בסעיף זה כתבנו Sniffer בשפת C ללא פילטרים.


noatzur@noatzur:~/Desktop/Labsetup$ sudo ./sniffer
From: 10.0.2.15
To: 13.225.255.58
From: 13.225.255.58
To: 10.0.2.15
From: 10.0.2.15
To: 93.184.220.29
From: 93.184.220.29
To: 10.0.2.15
From: 10.0.2.15
To: 34.98.75.36
From: 34.98.75.36
To: 10.0.2.15
From: 34.98.75.36
To: 10.0.2.15
From: 10.0.2.15
To: 34.98.75.36
From: 10.0.2.15
To: 216.58.212.100
From: 216.58.212.100
To: 10.0.2.15
Frame 1: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s3, id 0
wireshark_enp0s3_20210213140526_HJa8H8.pcapng

בתמונה משמאל ניתן לראות את ההרצה של קובץ Sniff_921.C ובמקביל הרצה של wireshark. ניתן לראות שהפאקטות שהתקבלו ב-WS הוספנו ע"י ה Sniffer והודפסו ה dst ! src איפי של כל אחת מהם (לפי הסדר (למשל ① זה הפאקטה שהודפסה ראשונה בטרמינל וכן הלאה...)

Question 1:

① pcap_lookupdev(errbuf);

הפונ' מחפשת הכרטיס רשת הדיפולטיבי (ברירת מחדל) אשר דרכו נרצה לראות את התעבורה (לתפוס פאקטות...) כמו כן, אם ישנם כמה כרטיסי רשת כאלו, הפונ' מחזיר רשימה של כרטי' הרשת.

כמו כן הפונ' מחזיר פוינטר ל-String שמייצג את שם כרטיס הרשת (אם זו רשימה של כרטיסי רשת אז לראשון).

② pcap_open_live(myDEV, BUFSIZ, 1, 1000, errbuf);

פותחת את כרטיס הרשת למצב "תפיסת" פאקטות.

הפונ' מקבלת את הפרמטרים הבאים:

(Device) myDEV - מקבלת את שם כרטיס הרשת שמצאנו קודם שאותו נפעיל כך שיהיה במצב של תפיסת פאקטות

(snaplen) BUFSIZ - מגדיר את גודל התור (buffer). התור יכיל פאקטות שנתפסו. (כמות הפאקטות שנשמרות בתור)

(promisc) - אם 1 אז מפעיל את המצב של promiscuous - פרוץ.

0 ← המצב הפרוץ כבוי.

(to_ms — time out) 1000 - (to_ms) - כמה זמן כל פאקטה נמצאת בבאפר

errbuf - פוינטר לבאפר שיחזיק שגיאות במידה ויצרכו,

③ pcap_loop(handle, -1, got_Packet, NULL) -

מהפונקצ'יה הקודמת ②

הפונ' מקבלת פוינטר מסוג pcap_t שזה ה-session שפתחנו אל מול כרטיס הרשת הנבחר ועבור כל פאקטה שנתפסת קוראת לפונ' מסוימת שמטפלת בפאקטה, במקרה שלנו פונ' בשם got-packet שהגדרנו מעלה. הפונ' got_packet מבצעת את הפעולות הרצויות עם המידע שהתקבל.

-1 : מגדיר ל pcap_loop לרוץ בלולאה אינסופית עד שגורם חיצוני יעצור את הלולאה ולהמשיך לתפוס פאקטות.

④ pcap_close (handle)-

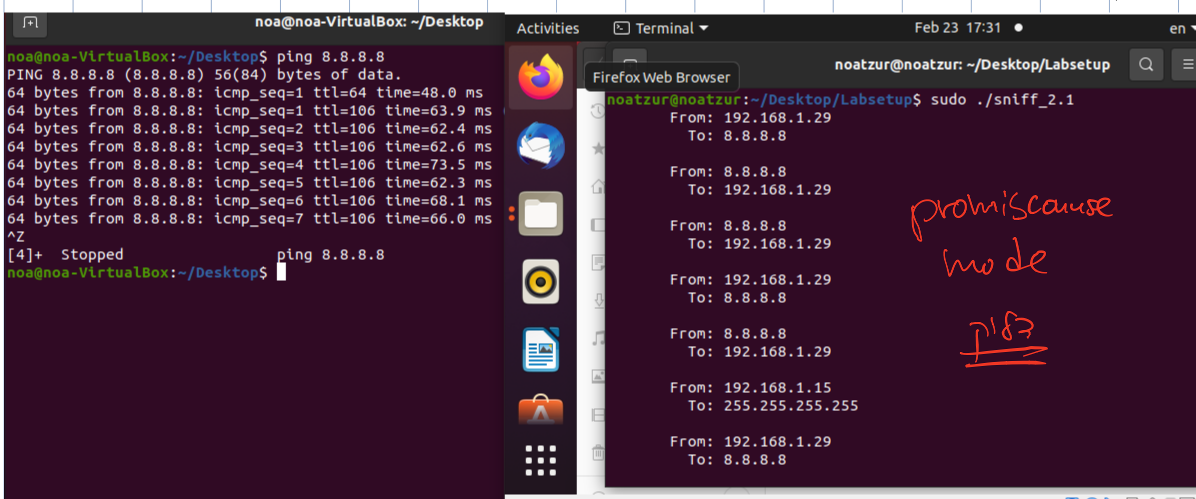
סוגרת את ה session שנפתח ע"י פונ' מס' ②. מוחקת את הקבצים והזכרון שהוקצה עבור ה session הזה.

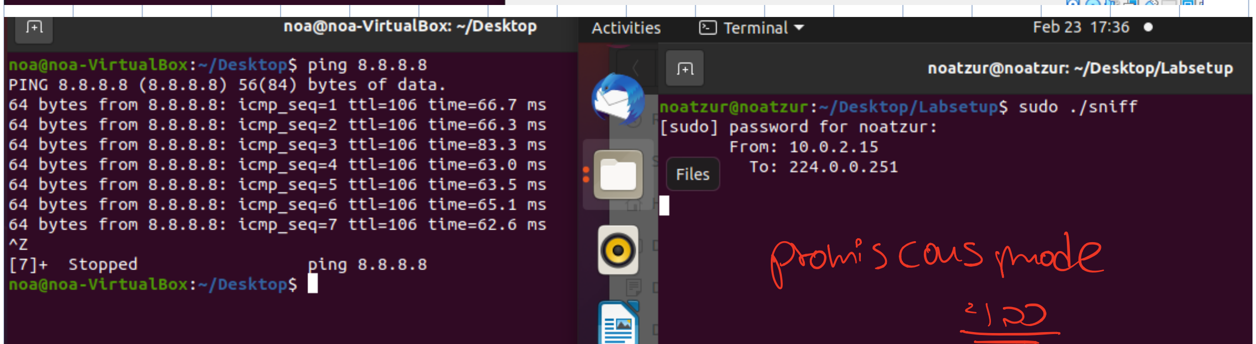
Question 2:

כדי לפתוח raw socket ובכך לאפשר לנו לקבל כל פאקטה שעוברת דרך כרטיס הרשת הנבחר, ולאו דווקא פאקטות המיועדות אלינו, צריך הרשאות admin. אם כל משתמש היה יכול לפתוח raw socket ולהפעיל את ה- promiscouse mode, הוא היה יכול לבחון את התעבורה ברשת שעוברת דרך כרטיס הרשת ולאו דווקא מיועדת לכרטיס הספציפי.

ולכן העברת ה Sniffer שלנו למצב שבו הוא תופס פאקטות שלא מיועדות אליו דווקא, ידרוש root privilege. התוכנה תזרוק שגיאת segmentation fault אם לא נשתמש בהרשאת הroot. בנוסף הפונ' שמגדירה גישה לכרטיס רשת דורשת root privilege. מגדירים זאת בפונ' pcap_open_live.

Question 3: כאשר promis mode דלוק, אנחנו בעצם מכניסים את
ה-sniffer שלנו למצב "פרוץ" האומר לכרטיס הרשת - אל תבצע סינון
של הפאקטות שנכנסות דרכך. כל פאקטה שעוברת - תתפוס אותה. כאשר המצב
לא דלוק, קורה הפעולה הרגילה שקורית ביומיום כאשר אנו משתמשים באינטרנט -
כרטיס הרשת מכניס אליו אך ורק פקטות המיועדות לו, עם זאת, כדי שנבחין את
ה-sniffer שלנו כאשר המצב דלוק/כבוי, פתחנו מכונה נוספת (שלה Ip
שונה מהמכונה ה-1) וממנה הפעלנו פינג ל-8.8.8.8.
כאשר המצב היה דלוק, ניתן לראות שהפאקטות נתפסו והודפסו למסך,
אך המצב היה כבוי, ניתן לראות שלא נתפסו פאקטות ולא הודפס דבר:





* המכונות (שתיהן) שינו בהגדרות ה network ל- Allow vm בשדה של
promiscouse mode. רק כך הצלחנו לשנות בקוד מ-0 ל-1 ולהפך
ולראות את השינוי.

2.1B : filters הקוד בקובץ sniff_filter.c ו sniff_filter2.c

- sniff_filter.c - מפיס פאקטות מסוג ICMP מ-destination ו source ספציפיים:

עשינו זאת ע"י הגדרת הבאפר filter-exp עם הביטוי

"Icmp and Src host 10.0.2.15 and dst host 8.8.8.8"

ע"י פונק pcap_compile ממירים את ה String שלנו ושומרים את הגרסא המקומפלת של הפילטר במבנה מיוחד לכך (bpf_program).

ע"י הפונק pcap_setfilter מפעילים את ה filter שלנו על ה Session.

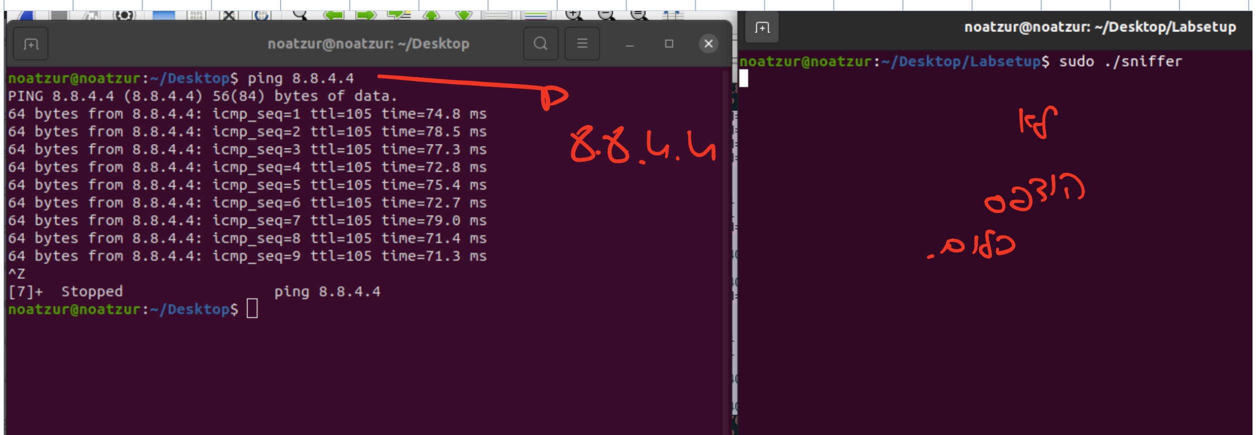
תמונה 1 - הדפסה של הפאקטות שנתפסו, ניתן לראות שבכך רק את IP המצוין, ורק מסוג ICMP:

```
noatzur@noatzur:~/Desktop/Labsetup$ sudo ./sniffer
        From: 10.0.2.15
          To: 8.8.8.8
im an ICMP packet !!!

        From: 10.0.2.15
          To: 8.8.8.8
im an ICMP packet !!!

        From: 10.0.2.15
          To: 8.8.8.8
im an ICMP packet !!!

        From: 10.0.2.15
          To: 8.8.8.8
im an ICMP packet !!!

        From: 10.0.2.15
          To: 8.8.8.8
im an ICMP packet !!!

        From: 10.0.2.15
          To: 8.8.8.8
im an ICMP packet !!!
```

תמונה 2 מה wireshark שהופעל במקביל לפעולת ה Sniffer ניתן לראות שהתקבלו פאקטות מכל מיני סוגים אך נתפסו רק אלו של ה IP המצוינים ומסוג ICMP:

No.	Time	Source	Destination	Protocol	Length	Info
31	0.570543984	192.114.46.176	10.0.2.15	TCP	60	443 → 38406 [FIN, ACK] Seq=1 Ack=26 Win=65535 Len=0
32	0.570742509	10.0.2.15	192.114.46.176	TCP	54	38406 → 443 [ACK] Seq=26 Ack=2 Win=63900 Len=0
33	0.570544259	192.114.46.176	10.0.2.15	TCP	60	443 → 38404 [FIN, ACK] Seq=1 Ack=26 Win=65535 Len=0
34	0.570814929	10.0.2.15	192.114.46.176	TCP	54	38404 → 443 [ACK] Seq=26 Ack=2 Win=63900 Len=0
35	1.373876530	10.0.2.15	65.9.109.34	TLSv1.2	93	Application Data
36	1.376565611	65.9.109.34	10.0.2.15	TCP	60	443 → 43760 [ACK] Seq=1 Ack=40 Win=65535 Len=0
37	1.392919336	65.9.109.34	10.0.2.15	TLSv1.2	93	Application Data
38	1.393303208	10.0.2.15	65.9.109.34	TCP	54	43760 → 443 [ACK] Seq=40 Ack=40 Win=63900 Len=0
39	2.376865568	10.0.2.15	91.228.74.198	TLSv1.2	100	Application Data
40	2.377197446	10.0.2.15	172.217.21.14	TLSv1.2	93	Application Data
41	2.379465778	91.228.74.198	10.0.2.15	TCP	60	443 → 55138 [ACK] Seq=1 Ack=47 Win=65535 Len=0
42	2.379466659	172.217.21.14	10.0.2.15	TCP	60	443 → 56094 [ACK] Seq=1 Ack=40 Win=65535 Len=0
43	2.445061257	172.217.21.14	10.0.2.15	TLSv1.2	93	Application Data
44	2.445174633	10.0.2.15	172.217.21.14	TCP	54	56094 → 443 [ACK] Seq=40 Ack=40 Win=63900 Len=0
45	2.445061861	91.228.74.198	10.0.2.15	TLSv1.2	100	Application Data
46	2.445391840	10.0.2.15	91.228.74.198	TCP	54	55138 → 443 [ACK] Seq=47 Ack=47 Win=63900 Len=0
47	2.550998145	10.0.2.15	192.114.43.110	TLSv1.2	78	[TCP Previous segment not captured] , Application Data
48	2.551171779	10.0.2.15	192.114.43.110	TCP	54	45538 → 443 [FIN, ACK] Seq=26 Ack=1 Win=63900 Len=0
49	2.554431903	192.114.43.110	10.0.2.15	TCP	60	[TCP ACKed unseen segment] 443 → 45538 [ACK] Seq=1 Ack=26 Win…
50	2.554432578	192.114.43.110	10.0.2.15	TCP	60	443 → 45538 [ACK] Seq=1 Ack=27 Win=65535 Len=0
51	2.567673281	192.114.43.110	10.0.2.15	TCP	60	443 → 45538 [FIN, ACK] Seq=1 Ack=27 Win=65535 Len=0
52	2.568186567	10.0.2.15	192.114.43.110	TCP	54	45538 → 443 [ACK] Seq=27 Ack=2 Win=63900 Len=0
53	2.804684001	10.0.2.15	192.114.43.110	TCP	54	45540 → 443 [ACK] Seq=1 Ack=1 Win=65535 Len=0
54	2.806717468	192.114.43.110	10.0.2.15	TCP	60	[TCP ACKed unseen segment] 443 → 45540 [ACK] Seq=1 Ack=2 Win=…
55	3.283470108	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request id=0x0010, seq=1/256, ttl=64 (reply in 5…
56	3.350200167	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0010, seq=1/256, ttl=107 (request i…
57	3.380806754	10.0.2.15	172.217.19.34	TLSv1.2	93	Application Data
58	3.381829778	10.0.2.15	172.217.19.34	TLSv1.2	93	Application Data
59	3.382173672	10.0.2.15	108.177.15.154	TLSv1.2	93	Application Data
60	3.382973004	172.217.19.34	10.0.2.15	TCP	60	443 → 45018 [ACK] Seq=1 Ack=40 Win=65535 Len=0
61	3.382973929	172.217.19.34	10.0.2.15	TCP	60	443 → 45020 [ACK] Seq=1 Ack=40 Win=65535 Len=0

תמונה 3 - עוד בדיקה שהבצעה לבדיקת הפילטר של הקוד:



הפילטר הוא ל dst IP 8.8.8.8 ולא 8.8.4.4, ולכן ה Sniffer לא הדפיס כלום.

- פילטר TCP - sniff_filter2.c

פילטר מסנן TCP פורטים 10-100.
אותם שלבים בדיוק כמו בסעיף הקודם, רק שהבצעה הביטוי המסנן הוא: "tcp and dst portrange 10-100"

על מנת לבדוק את הפילטר הזה, מצאנו בויקיפדיה הסבר על פרוטוקול telnet - הפרוטוקול משמש בחיבור לפורט 23 (שהוא בטווח המסנן), והוא יוצר חיבור TCP.

```
noatzur@noatzur:~/Desktop/Labsetup$ telnet 10.0.6.24
Trying 10.0.6.24...
```

כאן ניתן לראות שהפסו את הפקטות TCP
בטווח של 100-10:

```
noatzur@noatzur: ~/Desktop/Labsetup
noatzur@noatzur:~/Desktop/Labsetup$ sudo ./sniffer
[sudo] password for noatzur:
       From: 10.0.2.15
         To: 10.0.6.24
im an TCP packet !!!

       From: 10.0.2.15
         To: 10.0.6.24
im an TCP packet !!!

       From: 10.0.2.15
         To: 10.0.6.24
im an TCP packet !!!

       From: 10.0.2.15
         To: 10.0.6.24
im an TCP packet !!!

       From: 10.0.2.15
         To: 34.107.221.82
im an TCP packet !!!

       From: 10.0.2.15
         To: 34.107.221.82
im an TCP packet !!!

       From: 10.0.2.15
         To: 34.107.221.82
im an TCP packet !!!

       From: 10.0.2.15
```

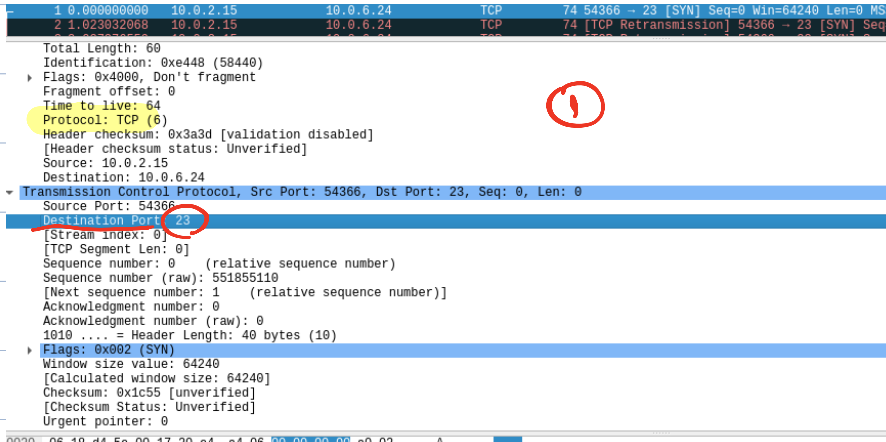
① ניתן לראות כי שהוא נשלח לפורט 23

② ניתן לראות כי שהוא נשלח לפורט 80.

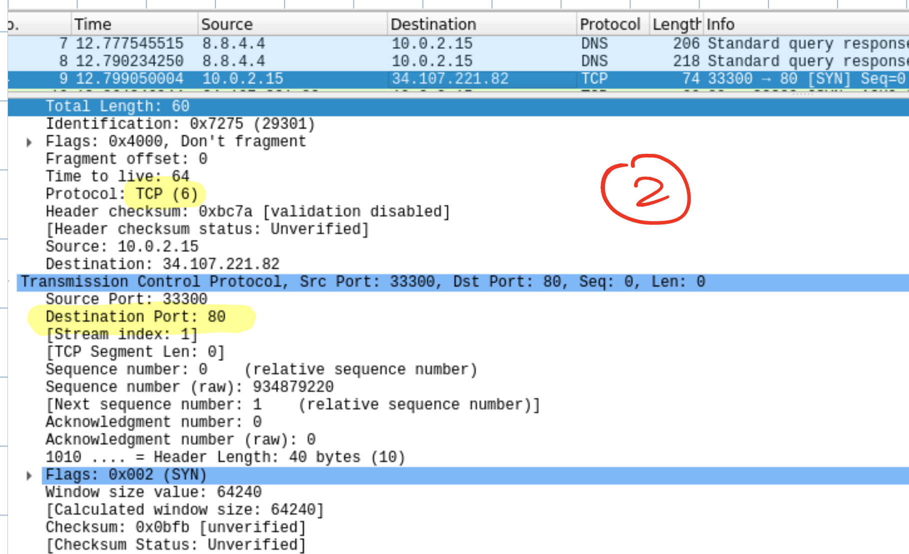
שניהם בטווח.

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	10.0.6.24	TCP	74	54366 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T…
2	1.023032068	10.0.2.15	10.0.6.24	TCP	74	[TCP Retransmission] 54366 → 23 [SYN] Seq=0 Win=64240 Len=0 M…
3	3.037376559	10.0.2.15	10.0.6.24	TCP	74	[TCP Retransmission] 54366 → 23 [SYN] Seq=0 Win=64240 Len=0 M…
4	7.229806345	10.0.2.15	10.0.6.24	TCP	74	[TCP Retransmission] 54366 → 23 [SYN] Seq=0 Win=64240 Len=0 M…
5	12.709770056	10.0.2.15	8.8.4.4	DNS	95	Standard query 0xe012 A detectportal.firefox.com OPT
6	12.710085976	10.0.2.15	8.8.4.4	DNS	95	Standard query 0x34c6 AAAA detectportal.firefox.com OPT
7	12.777545515	8.8.4.4	10.0.2.15	DNS	206	Standard query response 0xe012 A detectportal.firefox.com CNA…
8	12.790234250	8.8.4.4	10.0.2.15	DNS	218	Standard query response 0x34c6 AAAA detectportal.firefox.com …
9	12.799050000	10.0.2.15	34.107.221.82	TCP	74	33300 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T…
10	12.864246944	34.107.221.82	10.0.2.15	TCP	60	80 → 33300 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
11	12.864356038	10.0.2.15	34.107.221.82	TCP	54	33300 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
12	12.864549563	10.0.2.15	34.107.221.82	HTTP	350	GET /success.txt HTTP/1.1
13	12.865268073	34.107.221.82	10.0.2.15	TCP	60	80 → 33300 [ACK] Seq=1 Ack=297 Win=65535 Len=0
14	12.932918025	34.107.221.82	10.0.2.15	HTTP	274	HTTP/1.1 200 OK (text/plain)
15	12.932994138	10.0.2.15	34.107.221.82	TCP	54	33300 → 80 [ACK] Seq=297 Ack=221 Win=64020 Len=0
16	12.998538915	10.0.2.15	8.8.4.4	DNS	82	Standard query 0xa412 A example.org OPT
17	13.012384684	10.0.2.15	8.8.4.4	DNS	82	Standard query 0xefd0 AAAA example.org OPT
18	13.012968541	10.0.2.15	8.8.4.4	DNS	84	Standard query 0x2bd4 A ipv4only.arpa OPT
19	13.013694636	10.0.2.15	8.8.4.4	DNS	84	Standard query 0x7ba6 AAAA ipv4only.arpa OPT
20	13.018650942	10.0.2.15	8.8.4.4	DNS	106	Standard query 0xaed6 A content-signature-2.cdn.mozilla.net O…
21	13.018906963	10.0.2.15	8.8.4.4	DNS	106	Standard query 0xc1cf AAAA content-signature-2.cdn.mozilla.ne…
22	13.024058936	10.0.2.15	34.107.221.82	TCP	74	33302 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T…



```
1 0.000000000   10.0.2.15   10.0.6.24   TCP   74 54366 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS
2 1.023032068   10.0.2.15   10.0.6.24   TCP   74 [TCP Retransmission] 54366 → 23 [SYN] Seq=
    Total Length: 60
    Identification: 0xe448 (58440)
  ▸ Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x3a3d [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.2.15
    Destination: 10.0.6.24
▾ Transmission Control Protocol, Src Port: 54366, Dst Port: 23, Seq: 0, Len: 0
    Source Port: 54366
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Sequence number (raw): 551855110
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 0
    Acknowledgment number (raw): 0
    1010 .... = Header Length: 40 bytes (10)
  ▸ Flags: 0x002 (SYN)
    Window size value: 64240
    [Calculated window size: 64240]
    Checksum: 0x1c55 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
```

No.	Time	Source	Destination	Protocol	Length	Info
7	12.777545515	8.8.4.4	10.0.2.15	DNS	206	Standard query response
8	12.790234250	8.8.4.4	10.0.2.15	DNS	218	Standard query response
9	12.799050004	10.0.2.15	34.107.221.82	TCP	74	33300 → 80 [SYN] Seq=0

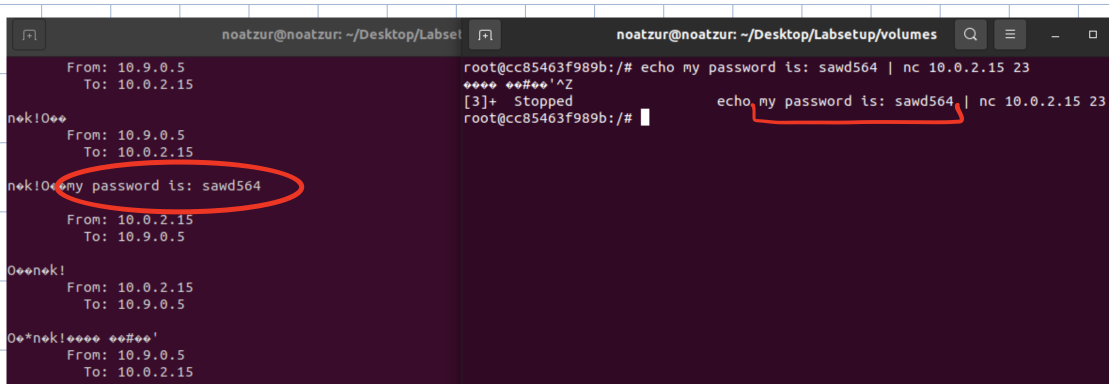
```
    Total Length: 60
    Identification: 0x7275 (29301)
  ▸ Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0xbc7a [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.2.15
    Destination: 34.107.221.82
 Transmission Control Protocol, Src Port: 33300, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 33300
    Destination Port: 80
    [Stream index: 1]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Sequence number (raw): 934879220
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 0
    Acknowledgment number (raw): 0
    1010 .... = Header Length: 40 bytes (10)
  ▸ Flags: 0x002 (SYN)
    Window size value: 64240
    [Calculated window size: 64240]
    Checksum: 0x0bfb [unverified]
    [Checksum Status: Unverified]
```

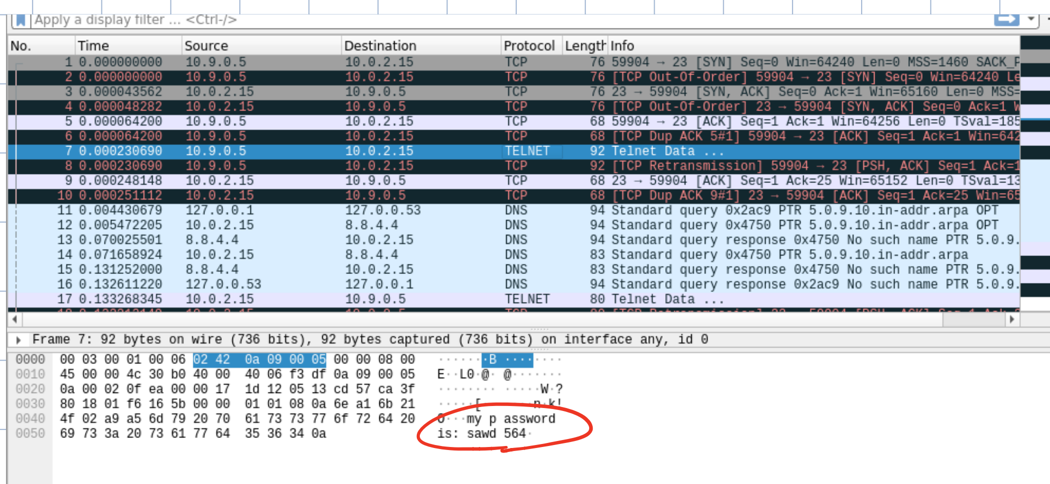

2.1c:

בסעיף זה התבקשנו להדפיס סיסמא של user אשר משתמש בפרוטוקול telnet, פרוטוקול זה משתמש ב port 23 ואינו מאובטח ולכן ניתן לראות את ה-data שהמשתמש מזין.

בתמונה מטה הדפסתי למסך את כל ה-data שהתקבל והסיסמא מסומנת בצורה ידנית.

בנוסף צירפנו גם תמונה של WS שם ניתן לראות את הdata. 18 מנות נשלחו ע"י telnet





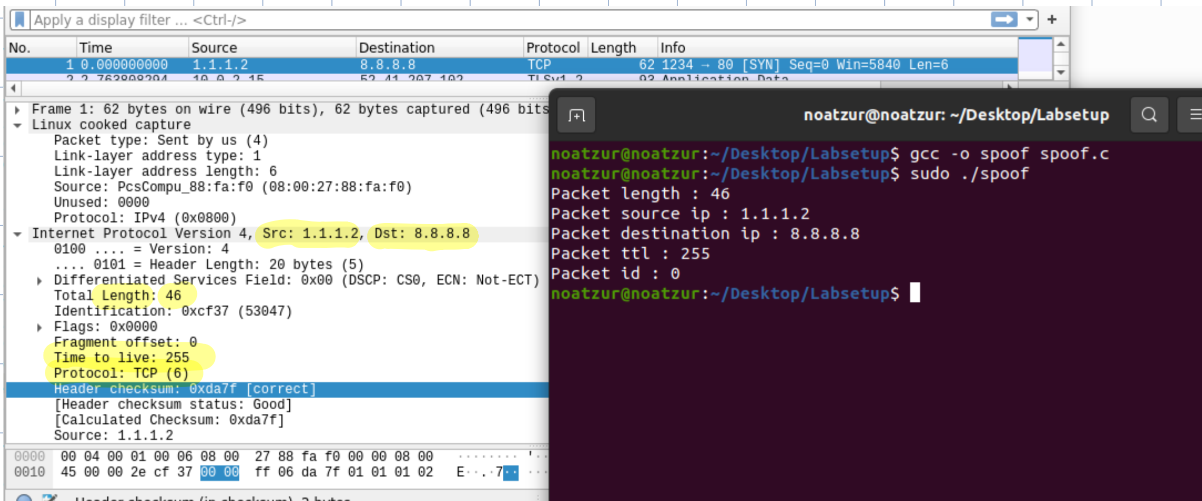
2.2A:

spoof.c קוד Spoof

בסעיף זה התבקשנו לכתוב את תוכנית ה-Spoof שלנו.

כתבנו תוכנית שיוצרת פקטה מסוג TCP, שה-IP שלה הוא 8.8.8.8 וה-IP source שלה 1.1.1.2 (IP שלא קיים בכלל...)

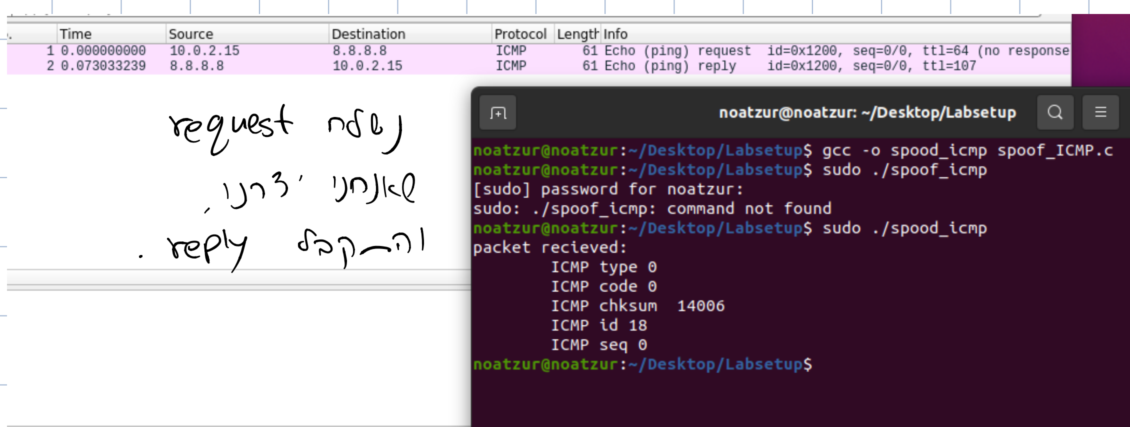
בתמונה המצורפת ניתן לראות את הפקטה שיצרנו ואת פרטיה-
וב WS ניתן לראות שאכן, נשלחה הפקטה שלנו.



2.2B: Spoof ICMP request

קובץ spoof_ICMP.c

בסעיף זה התבקשנו לכתוב תכנית spoof לייצור פקטה
מסוג ICMP request (id=8), השתמשנו בקוד שכתבנו
במטלה הקודמת כי הוא עונה על צרכים אלו בדיוק.



נשלח request
שאנחנו יצרנו,
והתקבל reply.

בנוסף בסעיף זה התבקשנו לענות על 3 שאלות:

Question 4:

לאחר ניסוי קצר שעשינו בקובץ spoof.c, שבו שינינו את גודל ה ip header לגלות כי לא משנה מה הגודל שנגדיר לו, כל עוד הערך גדול מ-20 כי זה הגודל המינימלי.
(וכן לשים לב גם ערכים גדולים, למשל 100, הפקטה נשלחה באופן תקין ו"מרפד" את ה data ב-אפסים

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

No.	Time	Source	Destination	Protocol
1	0.000000000	1.1.1.2	8.8.8.8	TCP
2	5.250797530	PcsCompu_88:fa:f0	RealtekU_12:35:02	ARP
3	5.251586175	RealtekU_12:35:02	PcsCompu_88:fa:f0	ARP

```
.... 0101 = Header Length: 20 bytes (5)
0000  52 54 00 12 35 02 08 00  27 88 fa f0 08 00 45 00   RT··5···'·····E·
0010  00 64 b9 0b 00 00 ff 06  f0 75 01 01 02 08 08      ·d······ ·u······
0020  08 08 04 d2 00 50 00 00  00 00 00 00 00 00 50 02   ·····P·· ······P·
0030  16 d0 50 0a 00 00 00 00  00 00 00 00 00 00 00 00   ··P····· ········
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0070  00 00                                              ··
```

ריפוד באפסים ← כאשר הגדרנו את הערך ל-100.

```
noatzur@noatzur:~/Desktop/Labsetup$ gcc -o spoof spoof.c
noatzur@noatzur:~/Desktop/Labsetup$ sudo ./spoof
[sudo] password for noatzur:
sendto failed: Invalid argument
noatzur@noatzur:~/Desktop/Labsetup$
```

השגיאה שנוצרה כאשר הגדרנו את הגודל עם 10.

Question 5:

גם בסעיף זה בדקנו ניסוי קטן כדי לבחון את
מחקנו את השורה שמחשבת checksum עבור השדה המתאים ב Iph ושמנו רק 0, שליחת הפקטה עברה באופן תקין, וגם ws ניתן היה לראות שה header checksum היה correct וה status הוא good. מכאן הבנו שה raw socket מחשב עבורנו את ה checksum ולכן אין הכרח שאנו נעשה את החישוב.

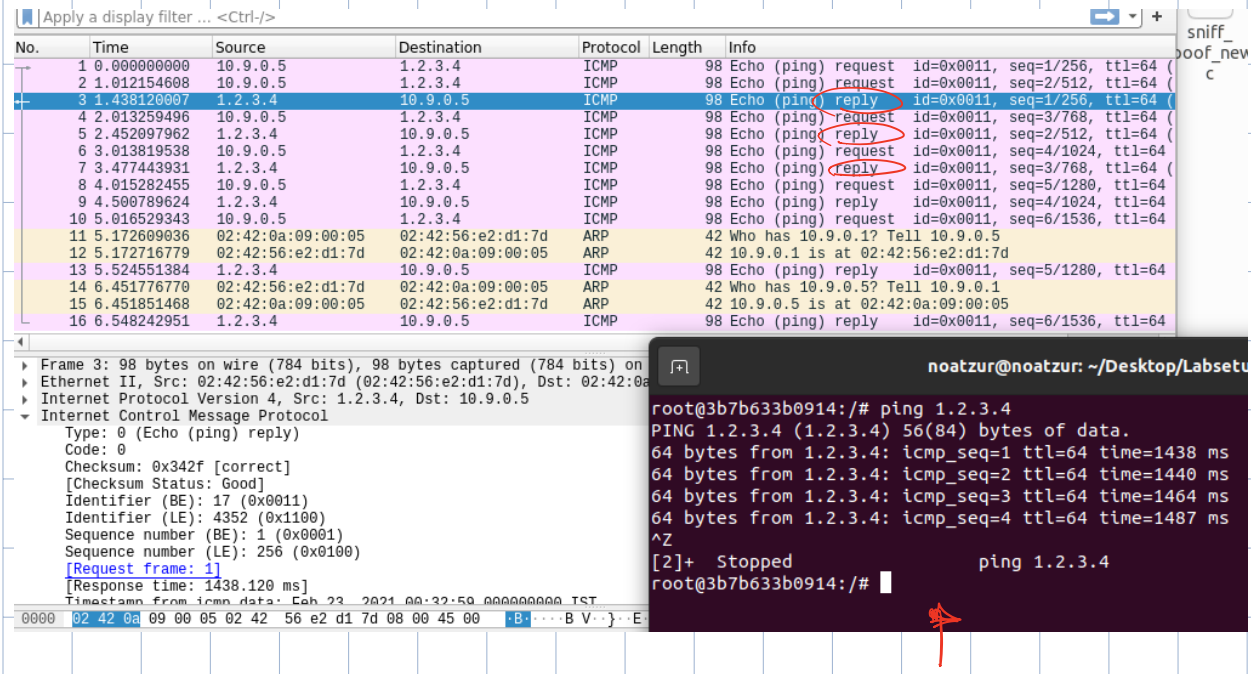
Question 6:

על מנת שנוכל לייצר פאקטה משלנו, לגשת ולשנות שדות ב-headers (נצטרך הרשאות מנהל), לא כל משתמש רגיל יכול לעשות שינויים כאלה. כמו כן, גם כדי להסניף פאקטות שלא מיועדות לכרטיס הרשת הספציפי דרכו אנו עובדים, נצטרך root privilege. הפונ' של ספריית pcap דורשות זאת כמובן. על מנת לבצע את כל האמור מעלה, תכניות שננסה להריץ ללא הרשאה מתאימה לא ירוצו ויזרקו שגיאת Permission שכן לא נוכל לייצר פאקטות ובכך לעשות את עיקר התוכנית - spoof.

2.3: Sniff & spoof

sniff_spoof_new.c

בסעיף זה התבקשנו "לחבר" את תכנית ה Sniff שכתבנו עם תכנית ה Spoof.
ב Container של המשתמש הרגיל שלחנו ping ל IP שלא קיים 1.2.3.4 כאשר במקביל, בContainer של הattacker הרצנו את הSniff & spoof שלנו.
התכנית, ע"י פונ' ה Sniff שכתבנו, זיהתה את ה ping שנשלח מהמשתמש הרגיל (ICMP) ויצרה פאקטה ICMP reply מזויפת ושלחה לאותו IP ממנו התקבל ה ping.
כך בעצם "יצרנו מצב" שבו המשתמש הרגיל האמין שהוא קיבל תשובה מ IP שלא קיים, כאילו היה קיים...



ראה ש 1.2.3.4 אין קיים ומתקבל reply.