

תקשורת ומחשוב - מטלה שיטיתPacket Sniffing and Spoofing LabTask 1.1: Sniffing PacketsTask 1.1A:

לאחר הרצה של הקוד בשם `Task_1_1_A.py` (תמונה 1), על ידי root privilege עם הוספה של פקודת `sudo` לפני שם הקובץ, ניתן לראות את הפקטות ICMP (תמונה 2) שנשלחו מכתובת 10.0.2.6 ולשרת DNS של 8.8.8.8 (תמונה 3).

```
task_1_1_A.py
~/Desktop
Open Save
1#!/usr/bin/env python3
2from scapy.all import *
3
4def print_pkt(pkt):
5    pkt.show()
6
7pkt = sniff(iface='br-6a709a0e8789', 'enp0s3', filter='icmp', prn=print_pkt)
Python 3 Tab Width: 8 Ln 7, Col 48 INS
```

```
^[[A^C[01/01/22]seed@VM:~/Desktop$ chmod a+x task_1_1_A.py
[01/01/22]seed@VM:~/Desktop$ sudo python3 task_1_1_A.py
```

```
###[ Ethernet ]###
```

```
dst      = 52:54:00:12:35:00
src      = 08:00:27:de:a2:20
type     = IPv4
```

```
###[ IP ]###
```

```
version  = 4
ihl      = 5
tos      = 0x0
len      = 84
id       = 15909
flags    = DF
frag     = 0
ttl      = 64
proto    = icmp
chksum   = 0xe06e
src      = 10.0.2.6
dst      = 8.8.8.8
\options \
```

```
###[ ICMP ]###
```

```
type     = echo-request
code     = 0
chksum   = 0x2622
id       = 0x6
seq      = 0x1
```

```
###[ Raw ]###
```

```
load     = 'G\r\xd0a\x00\x00\x00\x00\xf2\x94\t\x00\x00\x00\x00\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#%&\'()*+,-./01234567'
```

```
[01/01/22]seed@VM:~/Desktop$ 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=117 time=43.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=50.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=44.1 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=117 time=44.2 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=117 time=44.4 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=117 time=43.9 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=117 time=44.2 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=117 time=43.8 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=117 time=44.3 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=117 time=43.5 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=117 time=44.0 ms
64 bytes from 8.8.8.8: icmp_seq=12 ttl=117 time=43.4 ms
64 bytes from 8.8.8.8: icmp_seq=13 ttl=117 time=43.9 ms
64 bytes from 8.8.8.8: icmp_seq=14 ttl=117 time=43.6 ms
64 bytes from 8.8.8.8: icmp_seq=15 ttl=117 time=43.6 ms
64 bytes from 8.8.8.8: icmp_seq=16 ttl=117 time=44.0 ms
64 bytes from 8.8.8.8: icmp_seq=17 ttl=117 time=43.0 ms
64 bytes from 8.8.8.8: icmp_seq=18 ttl=117 time=48.3 ms
64 bytes from 8.8.8.8: icmp_seq=19 ttl=117 time=43.4 ms
64 bytes from 8.8.8.8: icmp_seq=20 ttl=117 time=44.0 ms
64 bytes from 8.8.8.8: icmp_seq=21 ttl=117 time=43.7 ms
64 bytes from 8.8.8.8: icmp_seq=22 ttl=117 time=44.1 ms
64 bytes from 8.8.8.8: icmp_seq=23 ttl=117 time=46.3 ms
64 bytes from 8.8.8.8: icmp_seq=24 ttl=117 time=44.1 ms
64 bytes from 8.8.8.8: icmp_seq=25 ttl=117 time=43.6 ms
64 bytes from 8.8.8.8: icmp_seq=26 ttl=117 time=43.7 ms
64 bytes from 8.8.8.8: icmp_seq=27 ttl=117 time=46.4 ms
```

ריצת התוכנית בלי שימוש ב- *root privilege* , כלומר בלי שימוש בפקודת *sudo* גורמת לשגיאה. לכן אם רוצים להסניף פקטות אנחנו צריכים להריץ ב- *root privilege* על מנת שתהיה לנו גישה לממשק שבו אנחנו מריצים את הקוד.

```
[01/01/22]seed@VM:~/Desktop$ chmod a+x task_1_1_A.py
[01/01/22]seed@VM:~/Desktop$ python3 task_1_1_A.py
Traceback (most recent call last):
  File "task_1_1_A.py", line 7, in <module>
    pkt = sniff(iface=['br-6a709a0e8789', 'enp0s3'], filter='icmp', prn=print_pkt)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1036, in sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 894, in _run
    sniff_sockets.update(
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 895, in <genexpr>
    (L2socket(type=ETH_P_ALL, iface=iface, *arg, **karg),
  File "/usr/local/lib/python3.8/dist-packages/scapy/arch/linux.py", line 398, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type)) # noqa: E501
  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
```

Task 1.1B:

1. תפיסת פקטות ICMP בלבד ע"י שימוש ב-BPF:

```
seed@VM: ~/Desktop
PermissionError: [Errno 13] operation not permitted
[01/01/22]seed@VM:~/Desktop$ chmod +x task_1_1_A.py
[01/01/22]seed@VM:~/Desktop$ sudo python3 task_1_1_A.py

#### Ethernet ####
dst      = 52:54:00:12:35:00
src       = 08:00:27:de:a2:20
type      = IPv4

#### IP ####
version   = 4
ihl       = 5
tos       = 0x0
len       = 84
id        = 61219
flags     = DF
frag      = 0
ttl       = 64
proto     = icmp
chksum    = 0x2f70
src       = 10.0.2.6
dst       = 8.8.8.8
\options  \

#### ICMP ####
type      = echo-request
code      = 0
chksum    = 0xccaa6
id        = 0x8
seq       = 0xa

#### Raw ####
load      = '\xdf\x1a\xd0a\x00\x00\x00\x00\xb7\xf7\x05\x00\x00
\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e
\x1f !#$%&'()*+,-./01234567'

#### Ethernet ####
64 bytes from 8.8.8.8: icmp_seq=12 ttl=117 time=44.2 ms
64 bytes from 8.8.8.8: icmp_seq=13 ttl=117 time=43.5 ms
64 bytes from 8.8.8.8: icmp_seq=14 ttl=117 time=43.7 ms
64 bytes from 8.8.8.8: icmp_seq=15 ttl=117 time=43.7 ms
64 bytes from 8.8.8.8: icmp_seq=16 ttl=117 time=44.0 ms
64 bytes from 8.8.8.8: icmp_seq=17 ttl=117 time=43.3 ms
64 bytes from 8.8.8.8: icmp_seq=18 ttl=117 time=43.3 ms
64 bytes from 8.8.8.8: icmp_seq=19 ttl=117 time=45.2 ms
64 bytes from 8.8.8.8: icmp_seq=20 ttl=117 time=43.4 ms
^C
--- 8.8.8.8 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19035ms
rtt min/avg/max/mdev = 43.167/43.815/45.161/0.459 ms
[01/01/22]seed@VM:~/Desktop$ 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=117 time=43.6 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=43.9 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=44.1 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=117 time=43.9 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=117 time=44.4 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=117 time=45.3 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=117 time=44.1 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=117 time=43.3 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=117 time=43.3 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=117 time=44.2 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=117 time=44.4 ms
64 bytes from 8.8.8.8: icmp_seq=12 ttl=117 time=44.0 ms
64 bytes from 8.8.8.8: icmp_seq=13 ttl=117 time=44.8 ms
^C
--- 8.8.8.8 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12098ms
rtt min/avg/max/mdev = 43.253/44.089/45.273/0.540 ms
```

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-01-01 03:1...	10.0.2.6	8.8.8.8	ICMP	98	Echo (ping) request id=0x0006, seq=1/256, ttl=64 (reply in 2)
2	2022-01-01 03:1...	8.8.8.8	10.0.2.6	ICMP	98	Echo (ping) reply id=0x0006, seq=1/256, ttl=117 (request in 1)
3	2022-01-01 03:1...	10.0.2.6	8.8.8.8	ICMP	98	Echo (ping) request id=0x0006, seq=2/512, ttl=64 (reply in 4)
4	2022-01-01 03:1...	8.8.8.8	10.0.2.6	ICMP	98	Echo (ping) reply id=0x0006, seq=2/512, ttl=117 (request in 3)
5	2022-01-01 03:1...	10.0.2.6	8.8.8.8	ICMP	98	Echo (ping) request id=0x0006, seq=3/768, ttl=64 (reply in 6)
6	2022-01-01 03:1...	8.8.8.8	10.0.2.6	ICMP	98	Echo (ping) reply id=0x0006, seq=3/768, ttl=117 (request in 5)
7	2022-01-01 03:1...	10.0.2.6	8.8.8.8	ICMP	98	Echo (ping) request id=0x0006, seq=4/1024, ttl=64 (reply in 8)
8	2022-01-01 03:1...	8.8.8.8	10.0.2.6	ICMP	98	Echo (ping) reply id=0x0006, seq=4/1024, ttl=117 (request in 7)

2. תפיסת פקטות הבאות מכתובת IP מסוימת עם port destination 23

```
task_1_1_A.py
~/Desktop

1#!/usr/bin/env python3
2from scapy.all import *
3
4def print_pkt(pkt):
5    pkt.show()
6
7pkt = sniff(filter='tcp dst port 23 and src host 10.0.2.6', prn=print_pkt)
```

```
seed@VM: ~/Desktop$ sudo python3 task_1_1_A.py
###[ Ethernet ]###
  dst      = 52:54:00:12:35:00
  src      = 08:00:27:de:a2:20
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x10
  len      = 60
  id       = 64937
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = tcp
  checksum = 0x20ed
  src      = 10.0.2.6
  dst      = 8.8.8.8
  \options \
###[ TCP ]###
  sport    = 56154
  dport    = telnet
  seq      = 1312901517
  ack      = 0
  dataoffs = 10
  reserved = 0
  flags    = S
  window   = 64240
  checksum = 0x1c44
  urgprtr  = 0
  options  = [('MSS', 1460), ('SACKOK', b''), ('Timestamp', (33925
56547, 0)), ('NOP', None), ('WScale', 7)]

64 bytes from 8.8.8.8: icmp_seq=58 ttl=117 time=43.6 ms
64 bytes from 8.8.8.8: icmp_seq=59 ttl=117 time=43.4 ms
64 bytes from 8.8.8.8: icmp_seq=60 ttl=117 time=269 ms
64 bytes from 8.8.8.8: icmp_seq=61 ttl=117 time=43.5 ms
64 bytes from 8.8.8.8: icmp_seq=62 ttl=117 time=44.3 ms
64 bytes from 8.8.8.8: icmp_seq=63 ttl=117 time=44.4 ms
64 bytes from 8.8.8.8: icmp_seq=64 ttl=117 time=43.3 ms
64 bytes from 8.8.8.8: icmp_seq=65 ttl=117 time=43.8 ms
64 bytes from 8.8.8.8: icmp_seq=66 ttl=117 time=43.5 ms
64 bytes from 8.8.8.8: icmp_seq=67 ttl=117 time=43.3 ms
64 bytes from 8.8.8.8: icmp_seq=68 ttl=117 time=44.2 ms
64 bytes from 8.8.8.8: icmp_seq=69 ttl=117 time=43.5 ms
64 bytes from 8.8.8.8: icmp_seq=70 ttl=117 time=43.4 ms
64 bytes from 8.8.8.8: icmp_seq=71 ttl=117 time=43.1 ms
64 bytes from 8.8.8.8: icmp_seq=72 ttl=117 time=43.6 ms
64 bytes from 8.8.8.8: icmp_seq=73 ttl=117 time=43.8 ms
64 bytes from 8.8.8.8: icmp_seq=74 ttl=117 time=44.4 ms
64 bytes from 8.8.8.8: icmp_seq=75 ttl=117 time=43.8 ms
64 bytes from 8.8.8.8: icmp_seq=76 ttl=117 time=44.4 ms
64 bytes from 8.8.8.8: icmp_seq=77 ttl=117 time=43.7 ms
64 bytes from 8.8.8.8: icmp_seq=78 ttl=117 time=44.4 ms
64 bytes from 8.8.8.8: icmp_seq=79 ttl=117 time=46.2 ms
^C
--- 8.8.8.8 ping statistics ---
79 packets transmitted, 79 received, 0% packet loss, time 78443ms
rtt min/avg/max/mdev = 43.059/46.741/269.383/25.213 ms
[01/01/22]seed@VM:~/Desktop$ telnet 8.8.8.8
Trying 8.8.8.8...
^C
[01/01/22]seed@VM:~/Desktop$ telnet 8.8.8.8
Trying 8.8.8.8...
^C
```

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

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-01-01 04:2	10.0.2.6	8.8.8.8	TCP	74	56154 → 23 [SYN] Seq=1312901517 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3392549436 TSecr=0 WS=128
2	2022-01-01 04:2	10.0.2.6	8.8.8.8	TCP	74	[TCP Retransmission] 56154 → 23 [SYN] Seq=1312901517 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3392550468 TSecr=0 WS
3	2022-01-01 04:2	10.0.2.6	8.8.8.8	TCP	74	[TCP Retransmission] 56154 → 23 [SYN] Seq=1312901517 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3392552484 TSecr=0 WS

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_de:a2:20 (08:00:27:de:a2:20), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
Internet Protocol Version 4, Src: 10.0.2.6, Dst: 8.8.8.8
Transmission Control Protocol, Src Port: 56154, Dst Port: 23, Seq: 1312901517, Len: 0

3. תפיסת פקטות שבאות או נשלחות מ- subnet 128.230.0.0/16

```
task_1_1_A.py
~/Desktop

1#!/usr/bin/env python3
2from scapy.all import *
3
4def print_pkt(pkt):
5    pkt.show()
6
7pkt = sniff(filter='dst net 128.230.0.0/16', prn=print_pkt)
```

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

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-01-01 04:4	10.0.2.6	128.230.0.0	ICMP	98	Echo (ping) request id=0x0012, seq=1/256, ttl=64 (no response found!)
2	2022-01-01 04:4	10.0.2.6	128.230.0.0	ICMP	98	Echo (ping) request id=0x0012, seq=2/512, ttl=64 (no response found!)
3	2022-01-01 04:4	10.0.2.6	128.230.0.0	ICMP	98	Echo (ping) request id=0x0012, seq=3/768, ttl=64 (no response found!)
4	2022-01-01 04:4	10.0.2.6	128.230.0.0	ICMP	98	Echo (ping) request id=0x0012, seq=4/1024, ttl=64 (no response found!)
5	2022-01-01 04:4	10.0.2.6	91.189.91.157	NTP	90	NTP Version 4, client
6	2022-01-01 04:4	10.0.2.6	128.230.0.0	ICMP	98	Echo (ping) request id=0x0012, seq=5/1280, ttl=64 (no response found!)
7	2022-01-01 04:4	91.189.91.157	10.0.2.6	NTP	90	NTP Version 4, server

Frame 4: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_de:a2:20 (08:00:27:de:a2:20), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
Internet Protocol Version 4, Src: 10.0.2.6, Dst: 128.230.0.0
Internet Control Message Protocol

```
seed@VM: ~/Desktop
[01/01/22]seed@VM:~/Desktop$ chmod +x task_1_1.A.py
[01/01/22]seed@VM:~/Desktop$ sudo python3 task_1_1.A.py
###[ Ethernet ]###
dst      = 52:54:00:12:35:00
src      = 08:00:27:de:a2:20
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x0
len      = 84
id       = 22282
flags    = DF
frag     = 0
ttl      = 64
proto    = icmp
chksum   = 0x56b3
src      = 10.0.2.6
dst      = 128.230.0.0
\options \
###[ ICMP ]###
type     = echo-request
code     = 0
chksum   = 0x87e6
id       = 0x12
seq      = 0x7
###[ Raw ]###
load     = ^#\xd0a\x00\x00\x00\x00y\xa8\t\x00\x00\x00\x00\x00
00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"%$%&
')*+,-./01234567'
```

Task 1.2: Spoofing ICMP Packets

לאחר הרצה של הקוד בשם `Task_1_2.py` (תמונה 1), ניתן לראות שעשינו *spoofing* ע"י הסניפר של פקטות ICMP מהסעיף הקודם. השתמשנו ב-*source* בכתובת IP 10.2.0.6 של המכונה שאנחנו עובדים איתה וכ-*destination* את שרת ה-DNS של גוגל 8.8.8.8 (תמונה 2).

```
task_1_2.py
~/Desktop
task_1_1_A.py
task_1_2.py

1 from scapy.all import *
2
3 a = IP()
4 a.src = '10.0.2.6'
5 a.dst = '8.8.8.8'
6 b = ICMP()
7 p = a/b
8 send(p)
9 ls(a)

seed@VM: ~/Desktop
ls
seed@VM:~/Desktop$ chmod a+x task_1_2.py
seed@VM:~/Desktop$ sudo python3 task_1_2.py

packets.
: BitField (4 bits) = 4 (4)
: BitField (4 bits) = None (None)
: XByteField = 0 (0)
: ShortField = None (None)
: ShortField = 1 (1)
: FlagsField (3 bits) = <Flag 0 (>) (<Flag

: BitField (13 bits) = 0 (0)
: ByteField = 64 (64)
: ByteEnumField = 0 (0)
: XShortField = None (None)
: SourceIPField = '10.0.2.6' (None)
: DestIPField = '8.8.8.8' (None)
: PacketListField = [] ([])

seed@VM:~/Desktop$
```

No.	Time	Source	Destination	Protocol	Length	Info
6	2022-01-01 05:5...	8.8.8.8	127.0.0.1	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response ...
7	2022-01-01 05:5...	8.8.8.8	127.0.0.1	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response ...

Frame 6: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface any, id 0

- Ethernet II, Src: Linux cooked capture, Dst: 01:00:00:00:00:00
- Internet Protocol Version 4, Src: 8.8.8.8, Dst: 127.0.0.1
- Internet Control Message Protocol

בשאלה הזאת עלינו למצוא את *traceroute* כלומר כמות הנתבים שמפרדים בין הכתובת ה-IP שלי לבין הכתובת ה-IP של היעד. על מנת לחשב *traceroute* עד השרת DNS של Facebook עם הכתובת IP 129.134.31.12, הוספנו לקטע קוד שמצורף (תמונה 1) בתרגיל כמה דברים לקוד שלנו `task_1_3.py`, משתנה בוליאני שיאפשר לנו בעצם לדעת עם הגענו לכתובת היעד או שאנחנו עדיין בדרך, רצנו בלולאה בטווח עד 50 (מספר מספיק גבוה) ועבור כל איטרציה בדקנו אם הכתובת הנוכחית שווה לכתובת היעד עם *timeout* מוגדר. (תמונה 2). ניתן לראות שה-`TTT` הנדרש עד להגעה היעד הוא 9.

No.	Time	Source	Destination	Protocol	Length	Info
3	2022-01-01 08:4...	10.0.2.6	129.134.31.12	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=1 (no response found!)
4	2022-01-01 08:4...	10.0.2.6	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
5	2022-01-01 08:4...	10.0.2.6	129.134.31.12	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=2 (no response found!)
6	2022-01-01 08:4...	31.134.13.77	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
7	2022-01-01 08:4...	10.0.2.6	129.134.31.12	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=3 (no response found!)
8	2022-01-01 08:4...	212.179.16.121	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
9	2022-01-01 08:4...	10.0.2.6	129.134.31.12	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=4 (no response found!)
10	2022-01-01 08:4...	212.179.124.85	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
11	2022-01-01 08:4...	10.0.2.6	129.134.31.12	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=5 (no response found!)
12	2022-01-01 08:4...	10.91.99.6	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
13	2022-01-01 08:4...	10.0.2.6	129.134.31.12	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=6 (no response found!)
14	2022-01-01 08:4...	157.240.74.60	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
15	2022-01-01 08:4...	10.0.2.6	129.134.31.12	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=7 (no response found!)
16	2022-01-01 08:4...	129.134.36.117	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
17	2022-01-01 08:4...	10.0.2.6	129.134.31.12	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=8 (no response found!)
18	2022-01-01 08:4...	157.240.39.73	10.0.2.6	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
19	2022-01-01 08:4...	10.0.2.6	129.134.31.12	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=9 (reply in 20)
20	2022-01-01 08:4...	129.134.31.12	10.0.2.6	ICMP	62	Echo (ping) reply id=0x0000, seq=0/0, ttl=56 (request in 19)

Frame 19: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface any, id 0
 Linux cooked capture
 Internet Protocol Version 4, Src: 10.0.2.6, Dst: 129.134.31.12
 Internet Control Message Protocol

Task 1.4: Sniffing and then Spoofing

1.2.3.4: השתמשתי בשתי מכונות וירטואליות, הראשונה עם הכתובת **10.0.2.6** ש"תוקפת" ע"י *spoofing* את המכונה השנייה עם הכתובת **10.0.2.7** ששולחת ping ל-**1.2.3.4** שהוא *host* שלא קיים ברשת אנטרנט. ניתן לראות בתעבורה בוירשארק את הפרוטוקול ARP שהוא בעצם שואל ברשת למי שייך את כתובת היעד.

```
task_1_4.py
~/Desktop

task_1_4.py x task_1_2.py x task_1_3.py x task_1_4.py x *Untitled Document 1

1#!/usr/bin/python3
2from scapy.all import *
3
4def spoofing(pkt):
5    if ICMP in pkt and pkt[ICMP].type == 8: #is a request
6        print('***** BEFORE SPOOFING *****')
7        print(' >> IP SRC: ', pkt[IP].src)
8        print(' >> IP DST: ', pkt[IP].dst)
9
10       ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
11       icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
12       data = pkt[Raw].load
13       spoofed_pkt = ip/icmp/data
14
15       print('***** AFTER SPOOFING *****')
16       print(' >> IP SRC: ', spoofed_pkt[IP].src)
17       print(' >> IP DST: ', spoofed_pkt[IP].dst)
18       print("\n")
19       send(spoofed_pkt, verbose=0)
20
21pkt = sniff(interface='br-6a709a0e8789', 'enp0s3', filter='icmp and src
host 10.0.2.7', prn=spoofing)
```

```
seed@VM: ~/Desktop
[01/01/22]seed@VM:~/Desktop$ chmod a+x task_1_4.py
[01/01/22]seed@VM:~/Desktop$ sudo python3 task_1_4.py
***** BEFORE SPOOFING *****
>> IP SRC: 10.0.2.7
>> IP DST: 1.2.3.4
***** AFTER SPOOFING *****
>> IP SRC: 1.2.3.4
>> IP DST: 10.0.2.7

***** BEFORE SPOOFING *****
>> IP SRC: 10.0.2.7
>> IP DST: 1.2.3.4
***** AFTER SPOOFING *****
>> IP SRC: 1.2.3.4
>> IP DST: 10.0.2.7

***** BEFORE SPOOFING *****
>> IP SRC: 10.0.2.7
>> IP DST: 1.2.3.4
***** AFTER SPOOFING *****
>> IP SRC: 1.2.3.4
>> IP DST: 10.0.2.7
```

```
seed@VM: ~
[01/01/22]seed@VM:~$ 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=54.5 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=15.3 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=22.3 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=15.5 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=27.0 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=23.1 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=24.4 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=32.6 ms
64 bytes from 1.2.3.4: icmp_seq=9 ttl=64 time=24.2 ms
^C
--- 1.2.3.4 ping statistics ---
11 packets transmitted, 9 received, 18.181% packet loss, time 10166ms
rtt min/avg/max/mdev = 15.281/26.533/54.450/11.085 ms
```

Apply a display filter ... <Ctrl-/>					
Source	Destination	Protocol	Length	Info	
14:1... 10.0.2.7	1.2.3.4	ICMP	98	Echo (ping) request	id=0x0004, seq=1/256, ttl=64
14:1... PcsCompu_de:a2:20	Broadcast	ARP	60	Who has 10.0.2.7? Tell 10.0.2.6	
14:1... PcsCompu_4c:2b:79	PcsCompu_de:a2:20	ARP	42	10.0.2.7 is at 08:00:27:4c:2b:79	
14:1... 1.2.3.4	10.0.2.7	ICMP	98	Echo (ping) reply	id=0x0004, seq=1/256, ttl=64
14:1... 10.0.2.7	1.2.3.4	ICMP	98	Echo (ping) request	id=0x0004, seq=2/512, ttl=64
14:1... 1.2.3.4	10.0.2.7	ICMP	98	Echo (ping) reply	id=0x0004, seq=2/512, ttl=64
14:1... 10.0.2.7	1.2.3.4	ICMP	98	Echo (ping) request	id=0x0004, seq=3/768, ttl=64
14:1... 1.2.3.4	10.0.2.7	ICMP	98	Echo (ping) reply	id=0x0004, seq=3/768, ttl=64

▶ Frame 10: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0

▶ Ethernet II, Src: PcsCompu_de:a2:20 (08:00:27:de:a2:20), Dst: PcsCompu_4c:2b:79 (08:00:27:4c:2b:79)

▶ Internet Protocol Version 4, Src: 1.2.3.4, Dst: 10.0.2.7

▶ Internet Control Message Protocol

10.9.0.99: באופן דומה השתמשנו באותם מכונות וירטואליות. במקרה הזה ה- host לא קיים ב-LAN לכן לא משנה כמה ping נשלח לא נקבל תשובה כי הוא בלתי ניתן להשגה.

```
seed@VM: ~  
[01/01/22]seed@VM:~$ ping 10.9.0.99  
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.  
From 10.9.0.1 icmp_seq=1 Destination Host Unreachable  
From 10.9.0.1 icmp_seq=2 Destination Host Unreachable  
From 10.9.0.1 icmp_seq=3 Destination Host Unreachable  
From 10.9.0.1 icmp_seq=4 Destination Host Unreachable  
From 10.9.0.1 icmp_seq=5 Destination Host Unreachable  
From 10.9.0.1 icmp_seq=6 Destination Host Unreachable  
From 10.9.0.1 icmp_seq=7 Destination Host Unreachable  
From 10.9.0.1 icmp_seq=8 Destination Host Unreachable  
From 10.9.0.1 icmp_seq=9 Destination Host Unreachable  
^C  
--- 10.9.0.99 ping statistics ---  
11 packets transmitted, 0 received, +9 errors, 100% packet loss, time 10250ms  
pipe 3
```

No.	Time	Source	Destination	Protocol	Length	Info
5	2022-01-01 14:3...	PcsCompu_de:a2:20	RealtekU_12:35:00	ARP	60	Who has 10.0.2.1? Tell 10.0.2.6
6	2022-01-01 14:3...	RealtekU_12:35:00	PcsCompu_de:a2:20	ARP	60	10.0.2.1 is at 52:54:00:12:35:00

8.8.8.8: בואן דומה למקרים קודמים השתמשנו באותם מכונות וירטואליות. בניגוד לכתובות הקודמות, **8.8.8.8** באמת קיימת ברשת אנטרנט, לכן לאחר הפעלת ה *spoofing* מהמכונה הראשונה ושליחת ping 8.8.8.8 מהשנייה שנקבל תשובות כפולות (*Duplicated response*) מכיוון שמקבלים תשובה גם מהשרת שלנו וגם ה *spoofing*.

```
seed@VM: ~  
[01/01/22]seed@VM:~$ 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=117 time=43.7 ms  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=64 time=51.4 ms (DUP!)  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=27.6 ms  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=43.9 ms (DUP!)  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=18.1 ms  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=45.1 ms (DUP!)  
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=23.1 ms  
64 bytes from 8.8.8.8: icmp_seq=4 ttl=117 time=44.0 ms (DUP!)  
64 bytes from 8.8.8.8: icmp_seq=5 ttl=64 time=20.0 ms  
64 bytes from 8.8.8.8: icmp_seq=5 ttl=117 time=44.3 ms (DUP!)  
64 bytes from 8.8.8.8: icmp_seq=6 ttl=64 time=18.6 ms  
64 bytes from 8.8.8.8: icmp_seq=6 ttl=117 time=45.5 ms (DUP!)  
64 bytes from 8.8.8.8: icmp_seq=7 ttl=117 time=43.9 ms  
64 bytes from 8.8.8.8: icmp_seq=7 ttl=64 time=55.4 ms (DUP!)  
^C  
--- 8.8.8.8 ping statistics ---  
7 packets transmitted, 7 received, +7 duplicates, 0% packet loss, time 6016ms  
rtt min/avg/max/mdev = 18.147/37.460/55.394/12.488 ms
```

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

No.	Time	Source	Destination	Protocol	Length	Info
10	2022-01-01 14:5...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) reply id=0x0008, seq=1/256, ttl=64
11	2022-01-01 14:5...	10.0.2.7	8.8.8.8	ICMP	98	Echo (ping) request id=0x0008, seq=2/512, ttl=64 (reply in 12)
12	2022-01-01 14:5...	10.0.2.7	10.0.2.7	ICMP	98	Echo (ping) reply id=0x0008, seq=2/512, ttl=64 (request in 11)
13	2022-01-01 14:5...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) reply id=0x0008, seq=2/512, ttl=117
14	2022-01-01 14:5...	10.0.2.7	8.8.8.8	ICMP	98	Echo (ping) request id=0x0008, seq=3/768, ttl=64 (reply in 15)
15	2022-01-01 14:5...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) reply id=0x0008, seq=3/768, ttl=64 (request in 14)
16	2022-01-01 14:5...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) reply id=0x0008, seq=3/768, ttl=117
17	2022-01-01 14:5...	10.0.2.7	8.8.8.8	ICMP	98	Echo (ping) request id=0x0008, seq=4/1024, ttl=64 (reply in 18)
18	2022-01-01 14:5...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) reply id=0x0008, seq=4/1024, ttl=64 (request in 17)
19	2022-01-01 14:5...	10.0.2.7	10.0.2.7	ICMP	98	Echo (ping) reply id=0x0008, seq=4/1024, ttl=117

Frame 4: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_4c:2b:79 (08:00:27:4c:2b:79), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
Address Resolution Protocol (request)

Task 2.1: Writing Packet Sniffing Program

Task 2.1A: Understanding How a Sniffer Works

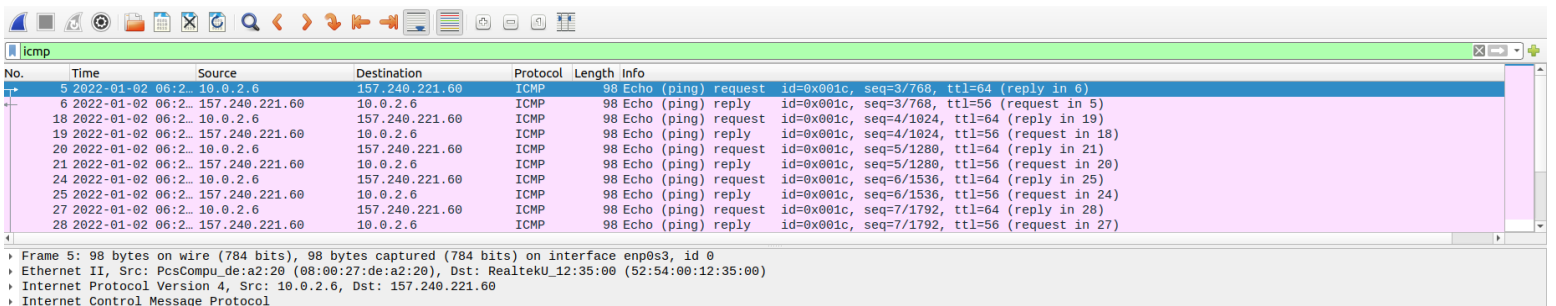
בסעיף הזה, השתמשנו בסניפר שכתבנו במטלה 5 שמסנן פקטות עם פרוטוקול ICMP. כפח שניתן לראות בצילום מסך המכונה וירטואלית מפעילה את הסניפר `task_2_1_A.c`, ושולחים ping לכתובת IP 157.240.221.60 שהיא שייכת ל-WhatsApp מטרמינל אחר, הסניפר קולט את הפקטות הרלוונטיות אליו. בנוסף ניתן לראות אותם פקטות בוויירשארק.

```
seed@VM: ~/sniffer
[01/02/22]seed@VM:~/sniffer$ gcc task_2_1_A.c -lpcap -o test
[01/02/22]seed@VM:~/sniffer$ sudo ./test
***** PACKET SNIFFING *****
>> PROTOCOL: ICMP
>> PACKET #0
>> SRC_IP: 157.240.221.60
>> DST_IP: 10.0.2.6

>> PROTOCOL: ICMP
>> PACKET #1
>> SRC_IP: 10.0.2.6
>> DST_IP: 157.240.221.60

>> PROTOCOL: ICMP
>> PACKET #2
>> SRC_IP: 157.240.221.60
>> DST_IP: 10.0.2.6
```

```
seed@VM: ~/Desktop
[01/02/22]seed@VM:~/Desktop$ ping 157.240.221.60
PING 157.240.221.60 (157.240.221.60) 56(84) bytes of data.
64 bytes from 157.240.221.60: icmp_seq=1 ttl=56 time=64.1 ms
64 bytes from 157.240.221.60: icmp_seq=2 ttl=56 time=64.3 ms
64 bytes from 157.240.221.60: icmp_seq=3 ttl=56 time=63.7 ms
64 bytes from 157.240.221.60: icmp_seq=4 ttl=56 time=73.1 ms
64 bytes from 157.240.221.60: icmp_seq=5 ttl=56 time=66.0 ms
64 bytes from 157.240.221.60: icmp_seq=6 ttl=56 time=63.1 ms
64 bytes from 157.240.221.60: icmp_seq=7 ttl=56 time=63.2 ms
64 bytes from 157.240.221.60: icmp_seq=8 ttl=56 time=64.1 ms
64 bytes from 157.240.221.60: icmp_seq=9 ttl=56 time=63.1 ms
^C
--- 157.240.221.60 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8010ms
rtt min/avg/max/mdev = 63.098/64.960/73.145/3.010 ms
```



No.	Time	Source	Destination	Protocol	Length	Info
5	2022-01-02 06:2...	10.0.2.6	157.240.221.60	ICMP	98	Echo (ping) request id=0x001c, seq=3/768, ttl=64 (reply in 6)
6	2022-01-02 06:2...	157.240.221.60	10.0.2.6	ICMP	98	Echo (ping) reply id=0x001c, seq=3/768, ttl=56 (request in 5)
18	2022-01-02 06:2...	10.0.2.6	157.240.221.60	ICMP	98	Echo (ping) request id=0x001c, seq=4/1024, ttl=64 (reply in 19)
19	2022-01-02 06:2...	157.240.221.60	10.0.2.6	ICMP	98	Echo (ping) reply id=0x001c, seq=4/1024, ttl=56 (request in 18)
20	2022-01-02 06:2...	10.0.2.6	157.240.221.60	ICMP	98	Echo (ping) request id=0x001c, seq=5/1280, ttl=64 (reply in 21)
21	2022-01-02 06:2...	157.240.221.60	10.0.2.6	ICMP	98	Echo (ping) reply id=0x001c, seq=5/1280, ttl=56 (request in 20)
24	2022-01-02 06:2...	10.0.2.6	157.240.221.60	ICMP	98	Echo (ping) request id=0x001c, seq=6/1536, ttl=64 (reply in 25)
25	2022-01-02 06:2...	157.240.221.60	10.0.2.6	ICMP	98	Echo (ping) reply id=0x001c, seq=6/1536, ttl=56 (request in 24)
27	2022-01-02 06:2...	10.0.2.6	157.240.221.60	ICMP	98	Echo (ping) request id=0x001c, seq=7/1792, ttl=64 (reply in 28)
28	2022-01-02 06:2...	157.240.221.60	10.0.2.6	ICMP	98	Echo (ping) reply id=0x001c, seq=7/1792, ttl=56 (request in 27)

Frame 5: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_de:a2:20 (08:00:27:de:a2:20), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
Internet Protocol Version 4, Src: 10.0.2.6, Dst: 157.240.221.60
Internet Control Message Protocol

Question 1: הסנפה על ידי ספריית <pcap> משתמשת בארבע פונקציות עיקריות:

- **pcap open live**: פונקציה הזאת פותחת device כדי לתפוס את הפקטות, היא מקבלת ארבעה ארגומנטים, הראשון הוא השם של הממשק השני הוא המספר המקסימלי של בתים לתפיסת פקטות, השלישי מציין שהמכשיר יעבור למצב מופקרת הרביעי נותן את פסק הזמן לקריאה באלפיות שניות והאחרון מחזיר שגיאה ומוגדר רק כאשר `pcap_open_live` נכשלת.
- **pcap compile**: הפונקציה הזאת תיצור את הפילטר שלנו עבור הסניפר, היא מורכבת מארבעה פרמטרים. הראשון מצביע על תיאור של פקטה שנתפסה שהוחזר מ-`pcap_open_live`. השני מצביע על מבנה `bpf_program` אשר יושלם על ידי `pcap_compile`. השלישי מכיל את פרמטר המסנן, הרביעי שולט על מבצעים אופטימיזציה בקוד המתקבל ואחרון מציין את `netmask` של הרשת.

- **pcap_setfilter**: פונקציה זו מפעילה את המסנן שיצרנו ב-*pcap_compile*. היא קודם מקבלת מצביע על תיאור של פקטה שנתפסה שהוחזר מ-*pcap_open_live* וגם מצביע על מבנה *bpf_program* אשר יושלם על ידי *pcap_compile*.

- **pcap_loop**: פונקציה זו מבצעת את התהליך של תפיסת פקטות. יש לה מצביע על תיאור של פקטה שנתפסה שהוחזר מ-*pcap_open_live*, זה ישמש לאחסון נתונים. הארגומנט השני מציין את המספר המקסימלי של פקטות לעיבוד לפני ההחזרה, השלישי הוא פונקציית *callback* לטיפול בחבילה ואחרון מציין את הארגומנט הראשון שיעבור *callback*.

Question 2: ספריית *pcap* צריכה לגשת לממשק של הרשת, היא משתמשת ב *raw socket*, בנוסף התהליך צריך לרוץ ב-*promiscuous mode* לכן ניתן לבצע את הפעולות האלה אך ורק ב *root privilege*. אם ננסה להריץ את התוכנית בלי ה-*root privilege* התוכנית תזרוק שגיאה.

Question 3: מצב מופקר (*promiscuous mode*) מאפשר לסניפר של הרשת לגשת לממשק ולנתונים להעביר את כל התעבורה מהרשת ולא רק את התעבורה שהרשת הייתה אמורה לקבל. המצב מופקר מוגדר בפרמטר השלישי של הפונקציה *pcap_open_live*.

כאשר מאתחלים את הארגומנט ל-1 יש לסניפר גישה לכל התעבורה ומקבל חבילות. כשאר אנחנו מגדירים את הפרמטר ל-0 ומעבירים את המכונה למצב Deny בהגדרות, הסניפר לא קולט פקטות.

Task 2.1B: Writing Filters

ICMP.

השתמשו בקוד מהסעיף הקודם כדי להסניף פקטות ICMP, שינינו את הפילטר כי אנחנו צריכים להסניף פקטות בין שני host ספציפיים.

Char filter_exp[] = "icmp and src host 10.2.0.6 and dst host 8.8.8.8"

ההרצנו את הקוד *task_2_1_B_ICMP.c* מהמכונה וירטואלית **10.2.0.6** ושלחנו *ping 8.8.8.8* מהמכונה השנייה.

```
seed@VM: ~/Desktop
seed@VM: ~/Desktop
[01/02/22]seed@VM:~/Desktop$ gcc task_2_1_B_ICMP.c -lpcap -o test
[01/02/22]seed@VM:~/Desktop$ sudo ./test
***** PACKET SNIFFING *****
>> PROTOCOL: ICMP
>> PACKET #0
>> SRC_IP: 10.0.2.7
>> DST_IP: 8.8.8.8
>> PROTOCOL: ICMP
>> PACKET #1
>> SRC_IP: 8.8.8.8
>> DST_IP: 10.0.2.7
>> PROTOCOL: ICMP
>> PACKET #2
>> SRC_IP: 10.0.2.7
>> DST_IP: 8.8.8.8
>> PROTOCOL: ICMP
>> PACKET #3
>> SRC_IP: 8.8.8.8
>> DST_IP: 10.0.2.7
```

```
seed@VM: ~
[01/02/22]seed@VM:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=115 time=48.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=67.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=115 time=49.8 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=115 time=43.8 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=115 time=57.9 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=115 time=46.9 ms
```

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-01-02 09:3...	10.0.2.7	8.8.8.8	ICMP	98	Echo (ping) request id=0x000e, seq=1/256, ttl=64 (reply in ...)
2	2022-01-02 09:3...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) reply id=0x000e, seq=1/256, ttl=115 (request ...)
3	2022-01-02 09:3...	10.0.2.7	8.8.8.8	ICMP	98	Echo (ping) request id=0x000e, seq=2/512, ttl=64 (reply in ...)
4	2022-01-02 09:3...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) reply id=0x000e, seq=2/512, ttl=115 (request ...)
5	2022-01-02 09:3...	10.0.2.7	8.8.8.8	ICMP	98	Echo (ping) request id=0x000e, seq=3/768, ttl=64 (reply in ...)
6	2022-01-02 09:3...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) reply id=0x000e, seq=3/768, ttl=115 (request ...)
7	2022-01-02 09:3...	10.0.2.7	8.8.8.8	ICMP	98	Echo (ping) request id=0x000e, seq=4/1024, ttl=64 (reply in ...)
8	2022-01-02 09:3...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) reply id=0x000e, seq=4/1024, ttl=115 (request ...)

▶ Frame 10: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0
 ▶ Ethernet II, Src: RealtekU_12:35:00 (52:54:00:12:35:00), Dst: PcsCompu_4c:2b:79 (08:00:27:4c:2b:79)
 ▶ Internet Protocol Version 4, Src: 8.8.8.8, Dst: 10.0.2.7
 ▶ Internet Control Message Protocol

TCP: השתמשנו בקוד מהסעיף הקודם כדי להסניף פקטות TCP, שינינו את הפילטר כי אנחנו צריכים להסניף פקטות TCP עם פורט יעד בטווח 10-100

`Char filter_exp[] = "tcp and portrange 10-100"`

ההרצנו את הקוד `task_2_1_B_TCP.c` מהמכונה וירטואלית **10.2.0.6** ושלחנו `telnet 8.8.8.8` פעם ראשונה עבור פורט 50 (יש תעבורה) ופעם שנייה עבור פורט 110 (אין תעבורה).

```

seed@VM: ~/Desktop
[01/02/22]seed@VM:~/Desktop$ gcc task_2_1_B_TCP.c -lpcap -o test2
[01/02/22]seed@VM:~/Desktop$ sudo ./test2
***** PACKET SNIFFING *****
>> PROTOCOL: TCP
>> PACKET #0
>> SRC_IP: 10.0.2.6
>> DST_IP: 8.8.8.8

>> PROTOCOL: TCP
>> PACKET #1
>> SRC_IP: 10.0.2.6
>> DST_IP: 8.8.8.8

>> PROTOCOL: TCP
>> PACKET #2
>> SRC_IP: 10.0.2.6
>> DST_IP: 8.8.8.8

^Z
[22]+  Stopped                  sudo ./test2
[01/02/22]seed@VM:~/Desktop$ gcc task_2_1_B_TCP.c -lpcap -o test2
[01/02/22]seed@VM:~/Desktop$ sudo ./test2
***** PACKET SNIFFING *****
^Z
[23]+  Stopped                  sudo ./test2
[01/02/22]seed@VM:~/Desktop$

```

```

seed@VM: ~/Desktop
[01/02/22]seed@VM:~/Desktop$ telnet 8.8.8.8 50
Trying 8.8.8.8...
^Z^C
[01/02/22]seed@VM:~/Desktop$ telnet 8.8.8.8 110
Trying 8.8.8.8...
^C

```

Task 2.1C: Sniffing Passwords

השתמשנו בקוד הקודם להסנפת פקטות TCP אך ביצענו כמה שינויים והספוד על מנת לתפוס את הסיסמה (*dees*) של מכונה וירטואלית שאנחנו עובדים בה. קודם כל בפילטר השארנו את:

`char filter_exp[] = "proto TCP and dst portrange 10-100"`

בנוסף הוספנו לתוכנית `task_2_C.c` את ה-`TCP header` עם כל הפרמטרים שלו ורצנו על לולאה על ה-`payload` כדי להסניף את התעבורה של `telnet` והנתונים שנשלחו מהמכונה `Telnet.10.0.2.7` בעצם מפצל את הסיסמה שמוכלת בנתונים ושולח אותו בספר פקטות אחד לכל תו מהסיסמה.

```
seed@VM: ~/Desktop
```

```
.....4.(...
>> PROTOCOL: TCP
>> IP SRC: 10.0.2.6
>> SRC PORT: 53952
>> IP DST: 10.0.2.7
>> DST PORT: 23
.....6;(...d
>> PROTOCOL: TCP
>> IP SRC: 10.0.2.6
>> SRC PORT: 53952
>> IP DST: 10.0.2.7
>> DST PORT: 23
.....7.(...e
>> PROTOCOL: TCP
>> IP SRC: 10.0.2.6
>> SRC PORT: 53952
>> IP DST: 10.0.2.7
>> DST PORT: 23
.....7.(...e
>> PROTOCOL: TCP
>> IP SRC: 10.0.2.6
>> SRC PORT: 53952
>> IP DST: 10.0.2.7
>> DST PORT: 23
.....8.(...s
>> PROTOCOL: TCP
>> IP SRC: 10.0.2.6
>> SRC PORT: 53952
>> IP DST: 10.0.2.7
```

```
seed@VM: ~
```

```
[01/02/22]seed@VM:~$ telnet 10.0.2.7
Trying 10.0.2.7...
Connected to 10.0.2.7.
Escape character is '^]'.
Jbuntu 20.04.1 LTS
VM login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

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

3 updates can be installed immediately.
3 of these updates are security updates.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
```

Task 2.2: Spoofing

Task 2.2.A: Write a spoofing program

כפי שניתן לראות בצילומי מסך, ההרצנו את הקוד `task_2_2_A.c` שעושה *spoofing* מהמכונה וירטואלית שלנו 10.0.2.6 לשרת של גוגל 8.8.8.8, ניתן לראות בוירשארק שהתוכנית שלחה את הפקטה *spoofed*.

```
void send_raw_ip_packet(struct ipheader* ip) {
    struct sockaddr_in dest_info;
    int enable = 1;
    //Step1: Create a raw network socket
    int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);

    //Step2: Set Socket option
    setsockopt(sock, IPPROTO_IP, IP_HDRINCL, &enable, sizeof(enable));

    //Step3: Provide destination information
    dest_info.sin_family = AF_INET;
    dest_info.sin_addr = ip->iph_destip;

    //Step4: Send the packet out
    printf("***** SENDING SPOOFED PACKET *****\n");
    if (sendto(sock, ip, ntohs(ip->iph_len), 0, (struct sockaddr *)&dest_info, sizeof(dest_info)) < 0) {
        printf("sError: failed sending message !", "");
    }
    else{
        printf("\t>> IP SOURCE: %s\n", inet_ntoa(ip->iph_sourceip));
        printf("\t>> IP DEST: %s\n", inet_ntoa(ip->iph_destip));
        printf("\n");
    }
    close(sock);
}
```

```

int main() {
    char buffer[PACKET_LEN];
    memset(buffer, 0, PACKET_LEN);

    // Fill in the ICMP header
    struct icmpheader *icmp = (struct icmpheader *) (buffer + sizeof(struct ipheader));

    //ICMP type 8 for request and 0 for replay
    icmp->icmp_type = 8;

    // Calculate checksum
    icmp->icmp_chksum = 0;
    icmp-> icmp_chksum = in_cksum((unsigned short *)icmp, sizeof(struct icmpheader));

    //Fill in the IP header
    struct ipheader *ip = (struct ipheader *) buffer;
    ip->iph_ver = 4;
    ip->iph_ihl = 5;
    ip->iph_tos = 16;
    ip->iph_ttl = 128;
    ip->iph_sourceip.s_addr = inet_addr("8.8.8.8");
    ip->iph_destip.s_addr = inet_addr("10.0.2.6");
    ip->iph_protocol = IPPROTO_ICMP;
    ip->iph_len = htons(sizeof(struct ipheader) + sizeof(struct icmpheader));

    send_raw_ip_packet (ip);

    return 0;
}

```

```

[01/04/22] seed@VM:~/Desktop$ sudo ./test
***** SENDING SPOOFED PACKET *****
>> IP SOURCE: 8.8.8.8
>> IP DEST: 10.0.2.6

```

No.	Time	Source	Destination	Protocol	Length	Info
7	2022-01-04 14:4...	8.8.8.8	10.0.2.6	ICMP	98	Echo (ping) reply id=0x000e, seq=1/256, ttl=117 (request in 6)
8	2022-01-04 14:4...	10.0.2.6	8.8.8.8	ICMP	98	Echo (ping) request id=0x000e, seq=2/512, ttl=64 (reply in 9)

Task 2.2B: Spoof an ICMP Echo Request.

הקוד שכתבנו בסעיף הקודם הוא spoof של Echo Request עם הצילומי מסך המצורפים.

Question 4

כן, ניתן להגדיר את אורך של ה-IP header לערך שרירותי ללא קשר לגודל החבילה האמיתית. בסופו של דבר גודל חבילה משתנה לגודל האמיתי לא משנה מה שינה המתכנת.

Question 5

לא, אין צורך, מערכת ההפעלה מחשבת לבד שדה זה.

Question 6

על מנת שכל תוכנית תקבל גישה לחומרה ותוכל לעבור למצב מופקר חייבים להשתמש ב- *root privilege* אם נרצה להריץ תוכניות ב- *raw socket*. בנוסף ברגע שיוצרים *raw socket* אפשר לבחור איזה פורט שרוצים אבל יש חוק ברשתות שאי אפשר לעשות *bind* לפורט שהוא פחות מ-1024 בלי הרשאת *root* ולכן במקרה כזה בגלל הסיכוי לבחור פורט נמוך יותר נבקש הרשאת *root*. כלומר במקרה שאין הרשאה תחסם הגישה ותיזרק גישה ב- *bind*.

4.3 Task 2.3: Sniff and then Spoof

בסעיף הזה, כתבנו תוכנית בשם *task_2_3.c* שמבצעת גם sniffing וגם spoofing על הפקטות. השתמשנו בני מכוונות וירטואליות הראשונה עם הכתובת 10.0.2.6 שממנה מריצים את התוכנית, כפי שניתן לראות על הצילום מסך, מופיעות על הטרמינל פקטות אחרי sniffing ו- spoofing עם כתובת מקור IP, כתובת יעד IP ופרוטוקול. עם המכונה השנייה אנחנו שולחים ping ל-שרת DNS של גוגל 8.8.8.8. על ידי הווירשארק נוכל לראות את כל הפקטות ICMP שנשלחו והתקבלו.

seed@VM: ~/Desktop

[01/05/22]seed@VM:~/Desktop\$ gcc task_2_3.c -lpcap -o test

[01/05/22]seed@VM:~/Desktop\$ sudo ./test

***** SNIFFING PACKET *****

>> IP SRC:10.0.2.7
>> IP DST:8.8.8.8
>> PROTOCOL: ICMP

seed@VM: ~

[01/05/22]seed@VM:~\$ ping 8.8.8.8

PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.

64 bytes from 8.8.8.8: icmp_seq=1 ttl=115 time=48.6 ms

64 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=59.6 ms

64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=1191 ms (DUP!)

64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=1192 ms (DUP!)

64 bytes from 8.8.8.8: icmp_seq=1 ttl=115 time=1240 ms (DUP!)

***** SENDING SPOOFED PACKET *****

>> IP SRC:8.8.8.8
>> IP DST:10.0.2.7

***** SNIFFING PACKET *****

>> IP SRC:8.8.8.8
>> IP DST:10.0.2.7
>> PROTOCOL: ICMP

***** SENDING SPOOFED PACKET *****

>> IP SRC:10.0.2.7
>> IP DST:8.8.8.8

***** SNIFFING PACKET *****

>> IP SRC:8.8.8.8
>> IP DST:10.0.2.7
>> PROTOCOL: ICMP

No.	Time	Source	Destination	Protocol	Length	Info
70	2022-01-05 05:2...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) reply id=0x000d, seq=4/1024, ttl=115
71	2022-01-05 05:2...	PcsCompu_4c:2b:79	RealtekU_12:35:00	ARP	42	Who has 10.0.2.1? Tell 10.0.2.7
72	2022-01-05 05:2...	RealtekU_12:35:00	PcsCompu_4c:2b:79	ARP	60	10.0.2.1 is at 52:54:00:12:35:00
73	2022-01-05 05:2...	10.0.2.7	8.8.8.8	ICMP	98	Echo (ping) request id=0x000d, seq=1/256, ttl=128 (reply
74	2022-01-05 05:2...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) reply id=0x000d, seq=1/256, ttl=128 (request
75	2022-01-05 05:2...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) reply id=0x000d, seq=1/256, ttl=128
76	2022-01-05 05:2...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) reply id=0x000d, seq=1/256, ttl=128
77	2022-01-05 05:2...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) reply id=0x000d, seq=1/256, ttl=128
78	2022-01-05 05:2...	8.8.8.8	10.0.2.7	ICMP	98	Echo (ping) request id=0x000d, seq=2/512, ttl=128 (reply
79	2022-01-05 05:2...	10.0.2.7	8.8.8.8	ICMP	98	Echo (ping) reply id=0x000d, seq=2/512, ttl=128 (request

Frame 71: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_4c:2b:79 (08:00:27:4c:2b:79), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
Address Resolution Protocol (request)