

# מטלת סיום בקורס תקשורת ומחשוב:

מגישים: ליאל ברניקר , רבקה סטריליץ

## תוכן עניינים:

### חלק 1:

מכיל תשובות לשאלות 1.1 – 1.4

### חלק 2:

מכיל תשובות לשאלות 2.1 – 2.3

## מקורות אינטרנטיים:

מכיל פירוט על מקורות מידע ששומשו במטלה והיכן שומשו בתשובות במטלה

## הערות נוספות:

הערות לגבי המטלה

## חלק 1:

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

#### Task 1.1: Sniffing Packets:

##### Task 1.1A.:

בשאלה זו אנו צריכים להריץ תוכנית בפיתון אשר מבצעת Sniffing Packets  
נוסיף תצלומי מסך להראות שאכן הצלחנו לרחרח את הפקטות.

הסנפת פקטות עם root privilege

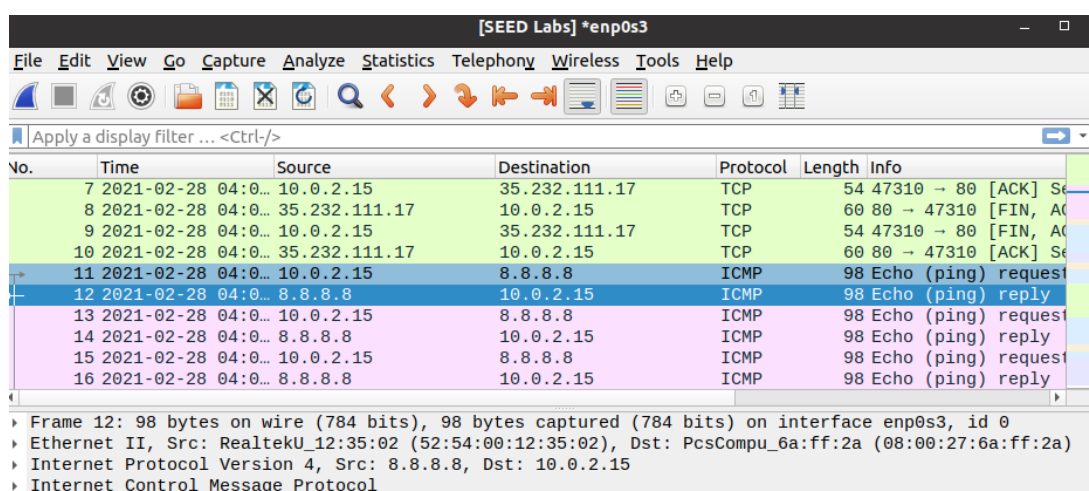
#### icmp echo replay

#### icmp echo request

```
###[ Ethernet ]###
dst      = 08:00:27:6a:ff:2a
src      = 52:54:00:12:35:02
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x20
len      = 84
id       = 7546
flags    =
frag     = 0
ttl      = 111
proto    = icmp
chksum   = 0x11f1
src      = 8.8.8.8
dst      = 10.0.2.15
\options \
###[ ICMP ]###
type     = echo reply
```

```
[02/28/21]seed@VM:~/.../task 1.1$ sudo chmod a+x sniffer1.1a.py
[02/28/21]seed@VM:~/.../task 1.1$ sudo python3 sniffer1.1a.py
###[ Ethernet ]###
dst      = 52:54:00:12:35:02
src      = 08:00:27:6a:ff:2a
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x00
len      = 84
id       = 49087
flags    = DF
frag     = 0
ttl      = 64
proto    = icmp
chksum   = 0x5ecb
src      = 10.0.2.15
dst      = 8.8.8.8
\options \
###[ ICMP ]###
```

## צילום Wireshark:



The screenshot shows the Wireshark interface with a packet capture on interface enp0s3. The packet list shows several ICMP Echo (ping) requests and replies. The selected packet (No. 12) is an ICMP Echo (ping) reply from 10.0.2.15 to 8.8.8.8. The packet details pane shows the frame structure: Ethernet II, Internet Protocol Version 4, and Internet Control Message Protocol.

No.	Time	Source	Destination	Protocol	Length	Info
7	2021-02-28 04:0...	10.0.2.15	35.232.111.17	TCP	54	47310 → 80 [ACK] Seq...
8	2021-02-28 04:0...	35.232.111.17	10.0.2.15	TCP	60	80 → 47310 [FIN, ACK] Seq...
9	2021-02-28 04:0...	10.0.2.15	35.232.111.17	TCP	54	47310 → 80 [FIN, ACK] Seq...
10	2021-02-28 04:0...	35.232.111.17	10.0.2.15	TCP	60	80 → 47310 [ACK] Seq...
11	2021-02-28 04:0...	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
12	2021-02-28 04:0...	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
13	2021-02-28 04:0...	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
14	2021-02-28 04:0...	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
15	2021-02-28 04:0...	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
16	2021-02-28 04:0...	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply

Frame 12: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0  
Ethernet II, Src: RealtekU\_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu\_6a:ff:2a (08:00:27:6a:ff:2a)  
Internet Protocol Version 4, Src: 8.8.8.8, Dst: 10.0.2.15  
Internet Control Message Protocol

## הסגפת פקטות ללא root privilege:

כאשר בצענו ריצה של התוכנית ללא root privilege ניתן לראות שקיבלנו שגיאה של ההרצה , זאת מכיוון שללא root privilege אין לנו את ההרשאות הרצויות כדי לבצע רחרוח של פקטות ולכן התוכנית אינה רצה .

כאשר אנו ניגשים ב root אנו מקבלים הרשאות וגישה לביצוע שינויים נרחבים , ושימוש בחרוח של פאקטות מהרשת דורש הרשאות מסוימות, ללא ההרשאות לא נוכל לבצע פעולות אלו.

```
[02/28/21]seed@VM:~/../task 1.1$ sudo chmod a+x sniffer1.1a.py
[02/28/21]seed@VM:~/../task 1.1$ python3 sniffer1.1a.py
Traceback (most recent call last):
  File "sniffer1.1a.py", line 6, in <module>
    pkt = sniff(iface=[ 'docker0', 'enp0s3', 'lo'], 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=ifname, *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:

### Capture only the ICMP packet

בשאלה זו אנו צריכים להריץ תוכנית בפיתון אשר מבצעת Sniffing Packets ספציפית לפקטות icmp

```
pkt = sniff(iface=['docker0', 'enp0s3', 'lo'], filter='icmp', prn=print_pkt)
```

נוסיף תצלומי מסך להראות שאכן הצלחנו לחרח את הפקטות.הנכונות

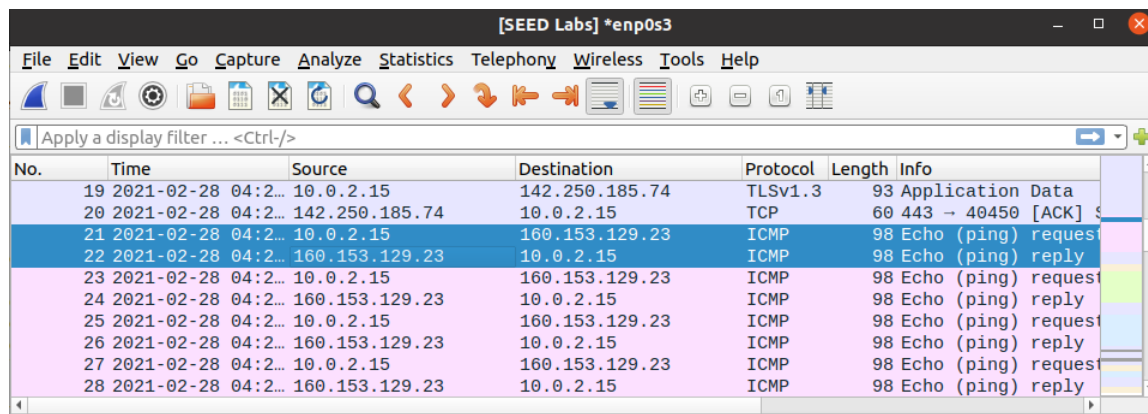
### icmp echo replay

```
###[ Ethernet ]###
dst      = 08:00:27:6a:ff:2a
src      = 52:54:00:12:35:02
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x20
len      = 84
id       = 7621
flags    = 
frag     = 0
ttl      = 45
proto    = icmp
chksum   = 0x4205
src      = 160.153.129.23
dst      = 10.0.2.15
\options \
###[ ICMP ]###
type     = echo-reply
code     = 0
chksum   = 0xdfd1
id       = 0x8
seq      = 0x1
###[ Raw ]###
load     = '\x1ca;\x00\x00'
f !"#$%&\'()*+,-./01234567'
```

### icmp echo request

```
[02/28/21]seed@VM:~/../task 1.1$ sudo chmod a+x sniffer1.1b_icmp.py
[02/28/21]seed@VM:~/../task 1.1$ sudo python3 sniffer1.1b_icmp.py
###[ Ethernet ]###
dst      = 52:54:00:12:35:02
src      = 08:00:27:6a:ff:2a
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x0
len      = 84
id       = 11176
flags    = DF
frag     = 0
ttl      = 64
proto    = icmp
chksum   = 0xe141
src      = 10.0.2.15
dst      = 160.153.129.23
\options \
###[ ICMP ]###
type     = echo-request
code     = 0
chksum   = 0xd7d1
id       = 0x8
seq      = 0x1
###[ Raw ]###
load     = '\x1ca;\x00\x00\x00\x00\xfb\x90\x0e\x00\x00'
f !"#$%&\'()*+,-./01234567'
```

צילום Wireshark:



No.	Time	Source	Destination	Protocol	Length	Info
19	2021-02-28 04:2...	10.0.2.15	142.250.185.74	TLSv1.3	93	Application Data
20	2021-02-28 04:2...	142.250.185.74	10.0.2.15	TCP	60	443 → 40450 [ACK] S...
21	2021-02-28 04:2...	10.0.2.15	160.153.129.23	ICMP	98	Echo (ping) request
22	2021-02-28 04:2...	160.153.129.23	10.0.2.15	ICMP	98	Echo (ping) reply
23	2021-02-28 04:2...	10.0.2.15	160.153.129.23	ICMP	98	Echo (ping) request
24	2021-02-28 04:2...	160.153.129.23	10.0.2.15	ICMP	98	Echo (ping) reply
25	2021-02-28 04:2...	10.0.2.15	160.153.129.23	ICMP	98	Echo (ping) request
26	2021-02-28 04:2...	160.153.129.23	10.0.2.15	ICMP	98	Echo (ping) reply
27	2021-02-28 04:2...	10.0.2.15	160.153.129.23	ICMP	98	Echo (ping) request
28	2021-02-28 04:2...	160.153.129.23	10.0.2.15	ICMP	98	Echo (ping) reply

Capture any TCP packet that comes from a particular IP and with a destination port number 23

בשאלה זו אנו צריכים להריץ תוכנית בפיתון אשר מבצעת Sniffing Packets ספציפית לפקטות tcp  
בport 23 שמגיעה מקו מסוים נגדיר קו זה 10.0.2.15

```
pkt = sniff(iface=['docker0', 'enp0s3', 'lo'], filter='tcp and dst port 23 and src host  
10.0.2.15', prn=print_pkt)
```

נוסיף תצלומי מסך להראות שאכן הצלחנו לרחרח את הפקטות.הנכונות

נציין כי port 23 הינו הרת של telnet

מקור מידע על telnet

[https://en.wikipedia.org/wiki/Telnet#:~:text=Telnet%20is%20a%20client%2Dserver,application.%20\(telnetd\)%20is%20listening](https://en.wikipedia.org/wiki/Telnet#:~:text=Telnet%20is%20a%20client%2Dserver,application.%20(telnetd)%20is%20listening)

tcp packet

```
[02/28/21]seed@VM:~/.../task 1.1$ sudo chmod a+x sniffer1.1b_tcp_port23.py  
[02/28/21]seed@VM:~/.../task 1.1$ sudo python3 sniffer1.1b_tcp_port23.py  
###[ Ethernet ]###  
dst      = 00:00:00:00:00:00  
src      = 00:00:00:00:00:00  
type     = IPv4  
###[ IP ]###  
version  = 4  
ihl      = 5  
tos      = 0x10  
len      = 60  
id       = 4469  
flags    = DF  
frag     = 0  
ttl      = 64  
proto    = tcp  
chksum   = 0x7116  
src      = 10.0.2.15  
dst      = 172.17.0.1  
options  = \n  
###[ TCP ]###  
sport    = 46474  
dport    = telnet  
seq      = 2662740601  
ack      = 0  
dataofs  = 10  
reserved = 0  
flags    = S  
window   = 65495  
chksum   = 0xb84f  
urgptr   = 0  
options  = [('MSS', 65495), ('SackOK', b''), ('Timestamp', (4163135587, 0)), ('NOP', None), ('WScale', 7)]
```

צילום Wireshark:

No.	Time	Source	Destination	Protocol	Length	Info
1	2021-02-28 05:0...	10.0.2.15	172.17.0.1	TCP	68	46450 → 23 [FIN, A
2	2021-02-28 05:0...	172.17.0.1	172.17.0.1	TCP	68	23 → 46450 [FIN, A
3	2021-02-28 05:0...	10.0.2.15	172.17.0.1	TCP	68	46450 → 23 [ACK] S
4	2021-02-28 05:0...	10.0.2.15	35.232.111.17	TCP	76	47430 → 80 [SYN] S
5	2021-02-28 05:0...	35.232.111.17	10.0.2.15	TCP	62	80 → 47430 [SYN, A
6	2021-02-28 05:0...	10.0.2.15	35.232.111.17	TCP	56	47430 → 80 [ACK] S
7	2021-02-28 05:0...	10.0.2.15	35.232.111.17	HTTP	143	GET / HTTP/1.1
8	2021-02-28 05:0...	35.232.111.17	10.0.2.15	TCP	62	80 → 47430 [ACK] S
9	2021-02-28 05:0...	35.232.111.17	10.0.2.15	HTTP	204	HTTP/1.1 204 No Con
10	2021-02-28 05:0...	10.0.2.15	35.232.111.17	TCP	56	47430 → 80 [ACK] S

Frame 1: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0
Linux cooked capture
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 172.17.0.1
Transmission Control Protocol, Src Port: 46450, Dst Port: 23, Seq: 3795425806, Ack: 1877727392, Len: 1000
Source Port: 46450
Destination Port: 23
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 3795425806
[Next sequence number: 3795425807]
Acknowledgment number: 1877727392
1000 .... = Header Length: 32 bytes (8)

∴ Capture packets comes from or to go to a particular subnet. You can pick any subnet  
 בשאלה זו אנו צריכים להריץ תוכנית בפיתון אשר מבצעת Sniffing Packets ספציפית לפקטות ב subnet  
 128.230.0.0/16 לפי הבחירה שלנו של subnet שאינו קשור לvm שלנו

```
pkt = sniff(iface=['docker0', 'enp0s3', 'lo'], filter='dst net 128.230.0.0/16', prn=print_pkt)
```

נוסיף תצלומוי מסך להראות שאכן הצלחנו לחרח את הפקטות.הנכונות

ניתן לראות כי בצענו ping אל כתובת 128.230.0.3 אשר כתובת זו נמצאת ב subnet של

128.230.0.0/16

icmp packet in subnet 128.230.0.0/16:

```
[02/28/21]seed@VM:~/.../task 1.1$ sudo chmod a+x sniffer1.1b_subnet.py
[02/28/21]seed@VM:~/.../task 1.1$ sudo python3 sniffer1.1b_subnet.py
###[ Ethernet ]###
  dst      = 52:54:00:12:35:02
  src      = 08:00:27:6a:ff:2a
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 6097
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0x95e0
  src      = 10.0.2.15
  dst      = 128.230.0.3
  \options \
###[ ICMP ]###
  type     = echo-request
  code     = 0
  chksum   = 0xb7a5
  id       = 0x9
  seq      = 0x1
###[ Raw ]###
  load     = '\x1bn;\x00\x00\x00\x00!\xaf\t\x00\x00\x00\x00\x00\x00'
$%&\'()*+,-./01234567'
```

## ציילום wireshark

No.	Time	Source	Destination	Protocol	Length	Info
1	2021-02-28 05:1...	10.0.2.15	128.230.0.3	ICMP	100	Echo (ping) request
2	2021-02-28 05:1...	10.0.2.15	128.230.0.3	ICMP	100	Echo (ping) request
3	2021-02-28 05:1...	10.0.2.15	128.230.0.3	ICMP	100	Echo (ping) request
4	2021-02-28 05:1...	10.0.2.15	128.230.0.3	ICMP	100	Echo (ping) request
5	2021-02-28 05:1...	128.230.61.99	10.0.2.15	ICMP	72	Destination unreach
6	2021-02-28 05:1...	128.230.61.99	10.0.2.15	ICMP	72	Destination unreach
7	2021-02-28 05:1...	128.230.61.99	10.0.2.15	ICMP	72	Destination unreach
8	2021-02-28 05:1...	10.0.2.15	128.230.0.3	ICMP	100	Echo (ping) request
9	2021-02-28 05:1...	PcsCompu_6a:ff:2a		ARP	44	Who has 10.0.2.2?
10	2021-02-28 05:1...	10.0.2.15	128.230.0.3	ICMP	100	Echo (ping) request

Frame 1: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0  
 Linux cooked capture  
 Internet Protocol Version 4, Src: 10.0.2.15, Dst: 128.230.0.3  
 Internet Control Message Protocol  
 Type: 8 (Echo (ping) request)  
 Code: 0  
 Checksum: 0xb7a5 [correct]  
 [Checksum Status: Good]  
 Identifier (BE): 9 (0x0009)  
 Identifier (LE): 2304 (0x0900)  
 Sequence number (BE): 1 (0x0001)  
 Sequence number (LE): 256 (0x0100)

## :Task 1.2: Spoofing ICMP Packets

ניתן לראות כי ביצענו spoofing לפקטת icmp, מסוג echo request, את כתובת ה ip של הפקטה שיצרנו הגדרנו בתור 1.2.3.4 שזוהי אינה כתובת ip אמיתית. ניתן לראות ב Wireshark כי אכן נשלחה פקטה שה source שלה הינו 1.2.3.4 וה destination שלה הינו 172.17.0.1 שזו היא אחת מכתובות ה host שעל המחשב שלנו. לאחר מכן ניתן לראות כי נשלחה אפילו echo replay מכתובת 172.17.0.1 אל 1.2.3.4

No.	Time	Source	Destination	Protocol	Length	Info
1	2021-02-28 09:1...	52.25.93.75	10.0.2.15	TLSv1.2	87	Application Data
2	2021-02-28 09:1...	10.0.2.15	52.25.93.75	TCP	56	57878 → 443 [ACK] Seq=36
3	2021-02-28 09:1...	10.0.2.15	52.25.93.75	TLSv1.2	91	Application Data
4	2021-02-28 09:1...	52.25.93.75	10.0.2.15	TCP	62	443 → 57878 [ACK] Seq=16
5	2021-02-28 09:1...	1.2.3.4	172.17.0.1	ICMP	44	Echo (ping) request id=
6	2021-02-28 09:1...	172.17.0.1	1.2.3.4	ICMP	44	Echo (ping) reply id=
7	2021-02-28 09:1...	PcsCompu_6a:ff:2a		ARP	44	Who has 10.0.2.2? Tell 1
8	2021-02-28 09:1...	RealtekU_12:35:02		ARP	62	10.0.2.2 is at 52:54:00:

Frame 5: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface any, id 0  
 Linux cooked capture  
 Internet Protocol Version 4, Src: 1.2.3.4, Dst: 172.17.0.1  
 Internet Control Message Protocol  
 Type: 8 (Echo (ping) request)  
 Code: 0  
 Checksum: 0xf7ff [correct]  
 [Checksum Status: Good]  
 Identifier (BE): 0 (0x0000)  
 Identifier (LE): 0 (0x0000)

### :Task 1.3: Traceroute

בחלק זה רשמנו קוד של traceroot על מנת למצוא את המרחק בין ה host שלנו לבין יעד מסויים (המרחק מחושב על פי כמות הראוטרים שצריך לעבור עד שמגיעים ליעד ) על מנת למצוא את ה traceroot נעזר ב ttl

התכנית שכתבנו מחשבת את ה traceroot באופן הבא :

נשתמש בפונקציית sr1, פונקציה זו שולחת את פקטת הבקשה שהכנו ומקבלת חזרה פקטה אחת, אותה נשמור בתור response כעת נבדוק :

\*אם הפקטה שקיבלנו שווה ל none אז ככל הנראה לא איתרנו ראوتر במסלול

\*אחרת אם הפקטה שקיבלנו היא מטיפ 0 כלומר תגובה, אזי הגענו ליעד לכן נצא מהלולאה ונדפיס את ההודעה המתאימה

77	2021-03-03 12:3...	127.0.0.1	127.0.0.1	DNS	91 Standard query response 0x0000 www.amitdvir.com
78	2021-03-03 12:3...	10.0.2.15	160.153.129.23	ICMP	44 Echo (ping) request id=0x0000, seq=0/0, ttl=20
79	2021-03-03 12:3...	160.153.129.23	10.0.2.15	ICMP	62 Echo (ping) reply id=0x0000, seq=0/0, ttl=45
80	2021-03-03 12:3...	10.0.2.15	91.189.89.198	NTP	92 NTP Version 4, client
81	2021-03-03 12:3...	91.189.89.198	10.0.2.15	NTP	92 NTP Version 4, server

Frame 79: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface any, id 0

- Linux cooked capture
- Internet Protocol Version 4, Src: 160.153.129.23, Dst: 10.0.2.15
- Internet Control Message Protocol
  - Type: 0 (Echo (ping) reply)
  - Code: 0
  - Checksum: 0xffff [correct]

\*אם הפקטה היא מטיפ 11 exceeded ( כלומר איתרנו ראوتر במסלול אבל ה ttl לא הספיק על מנת להגיע ליעד ) נעלה את הקאונטר המונה את מספר הראוטרים במסלול וגם את ה ttl באחד .

נוכל לראות כי אכן במהלך הרצת התכנית נתפסה פקטה exceeded ב wireshark

6	2021-03-03 12:3...	127.0.0.53	127.0.0.1	DNS	108 Standard query response 0xfd26 A www.amitdvir.com
7	2021-03-03 12:3...	PcsCompu_91:d6:ff		ARP	44 Who has 10.0.2.2? Tell 10.0.2.15
8	2021-03-03 12:3...	RealtekU_12:35:02		ARP	62 10.0.2.2 is at 52:54:00:12:35:02
9	2021-03-03 12:3...	10.0.2.15	160.153.129.23	ICMP	44 Echo (ping) request id=0x0000, seq=0/0, ttl=1
10	2021-03-03 12:3...	10.0.2.2	10.0.2.15	ICMP	72 Time-to-live exceeded (Time to live exceeded in transit)
11	2021-03-03 12:3...	10.0.2.15	160.153.129.23	ICMP	44 Echo (ping) request id=0x0000, seq=0/0, ttl=2
12	2021-03-03 12:3...	192.168.1.1	10.0.2.15	ICMP	72 Time-to-live exceeded (Time to live exceeded in transit)
13	2021-03-03 12:3...	10.0.2.15	160.153.129.23	ICMP	44 Echo (ping) request id=0x0000, seq=0/0, ttl=3

Frame 10: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface any, id 0

- Linux cooked capture
- Internet Protocol Version 4, Src: 10.0.2.2, Dst: 10.0.2.15
- Internet Control Message Protocol
  - Type: 11 (Time-to-live exceeded)
  - Code: 0 (Time to live exceeded in transit)
  - Checksum: 0xf4ff [correct]

ניתן לראות כי אכן לאחר הרצת הקוד התקבל כי ה traceroot אל האתר של ד"ר עמית דביר היא 10 ראוטרים

```

seed@VM: ~/.../task 1.3
.....
router ip: 195.219.50.21 , router number 7 in trace
.....
no replay sent back
.....
no replay sent back
.....
router ip: 213.244.164.90 , router number 8 in trace
.....
router ip: 188.121.32.5 , router number 9 in trace
.....
router ip: 188.121.32.115 , router number 10 in trace
.....
no replay sent back
.....
no replay sent back
.....
no replay sent back
.....
no replay sent back
.....
the trace route to the ip address www.amitdvir.com contain :10 routers
[03/03/21]seed@VM:~/.../task 1.3$ ^C
[03/03/21]seed@VM:~/.../task 1.3$

```

בנוסף על מנת לבדוק כי אכן צדקנו השונו את התוצאה שלנו ל traceroot ב windows

```

C:\WINDOWS\system32\cmd.exe
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\B000000>tracert www.amitdvir.com

Tracing route to amitdvir.com [160.153.129.23]
over a maximum of 30 hops:
  0  140 ms  3 ms  2 ms  OpenWrt.lan [192.168.1.1]
  1  35 ms  37 ms  19 ms  82.102.128.241
  2  *      *      *      Request timed out.
  3  *      *      *      Request timed out.
  4  *      *      *      Request timed out.
  5  *      *      *      Request timed out.
  6  52 ms  98 ms  96 ms  82.102.132.78
  7  111 ms  97 ms  101 ms  80.179.166.53.static.012.net.il [80.179.166.53]
  8  97 ms  99 ms  200 ms  ix-ae-11-0.tcore1.fr0-frankfurt.as6453.net [195.219.219.105]
  9  102 ms  98 ms  200 ms  if-ae-45-2.tcore2.fr0-frankfurt.as6453.net [195.219.50.21]
 10  *      *      *      Request timed out.
 11  *      165 ms  99 ms  ae-2-2.ear2.Amsterdam1.Level3.net [4.69.203.206]
 12  132 ms  97 ms  96 ms  ae6.ibrsa0105-01.ams3.bb.godaddy.com [213.244.164.90]
 13  96 ms  102 ms  90 ms  ae2.ams3-bbsa0106-01.bb.gdinf.net [188.121.32.5]
 14  91 ms  98 ms  97 ms  188.121.32.115
 15  *      *      *      Request timed out.
 16  *      *      *      Request timed out.
 17  *      *      *      Request timed out.
 18  *      *      *      Request timed out.
 19  104 ms  99 ms  80 ms  ip-160-153-129-23.ip.secureserver.net [160.153.129.23]

Trace complete.

C:\Users\B000000>

```

### Task 1.4: Sniffing and-then Spoofing:

בחלק זה רשמנו sniffer and spoffer התוכנית מזהה בקשת icmp echo request ועונה בicmp echo replay כדי להראות שאכן התוכנית עובדת ואכן עונה עם פאקטת echo replay icmp . כל פעם שהתוכנית קולטת icmp echo request . הרצנו בhost בdocker ping לקו שונים לפי הבקשה בשאלה והפעלנו בהתאם את הsniffer and spoffer בvm שלנו כדי להחזיר תשובה ל ping של ה host בdocker .



### ping 1.2.3.4 # a non-existing host on the Internet

תחילה בצענו ping לקו 1.2.3.4 זהו ip אשר לא קיים ברשת

קריאה ל ping ב: docker:

```
[03/01/21]seed@VM:~/.../Labsetup$ docksh 62
root@62047f7ef997:/# 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=90.6 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=51.2 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=50.1 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=47.9 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=57.4 ms
```

ניתן לראות כי אכן התקבלה תשובה לשליחת הבקשה לקו זה למרות שהוא אינו קיים ברשת, זה מכיוון שהתוכנית sniffer and spoffer אכן פעלה ויצרה פאקטת icmp פיקטיבית ושלחה לקו שביקש את ה echo request.

רחרוח פאקטת icmp echo request ושליחת הפאקטה שנוצרה על ידי תוכנית בהתאם לנתונים של הפקאטה שרחרחה:

```
[03/01/21]seed@VM:~/.../task 1.4$ sudo python3 spoofing.py
spoofer packer information:
original src is: 10.0.2.15
original dst is: 1.2.3.4
.
Sent 1 packets.
send icmp echo replay
new src is: 1.2.3.4
new dst is: 10.0.2.15
.....
spoofer packer information:
original src is: 10.9.0.5
original dst is: 1.2.3.4
.
Sent 1 packets.
send icmp echo replay
new src is: 1.2.3.4
new dst is: 10.9.0.5
.....
spoofer packer information:
original src is: 10.0.2.15
original dst is: 1.2.3.4
```

בתצלום ניתן לראות כי אכן התוכנית רחרחה את הפאקטה יצרה פאקטה חדשה והפכה את ה source ip

עם ה destination ip והגדירה את סוג הפאקטה icmp בצורה הבאה

```
ip = IP(src = pkt[IP].dst, dst = pkt[IP].src, ihl = pkt[IP].ihl)
icmp = ICMP(type = 0, id = pkt[ICMP].id, seq = pkt[ICMP].seq)
```

## צילום Wireshark :

No.	Time	Source	Destination	Protocol	Length	Info
44	2021-03-01 13:4...	10.9.0.5	1.2.3.4	ICMP	100	Echo (ping) request
45	2021-03-01 13:4...	10.9.0.5	1.2.3.4	ICMP	100	Echo (ping) request
46	2021-03-01 13:4...	10.0.2.15	1.2.3.4	ICMP	100	Echo (ping) request
47	2021-03-01 13:4...	1.2.3.4	10.0.2.15	ICMP	100	Echo (ping) reply
48	2021-03-01 13:4...	1.2.3.4	10.9.0.5	ICMP	100	Echo (ping) reply
49	2021-03-01 13:4...	1.2.3.4	10.9.0.5	ICMP	100	Echo (ping) reply
50	2021-03-01 13:4...	10.9.0.5	1.2.3.4	ICMP	100	Echo (ping) request
51	2021-03-01 13:4...	10.9.0.5	1.2.3.4	ICMP	100	Echo (ping) request
52	2021-03-01 13:4...	10.0.2.15	1.2.3.4	ICMP	100	Echo (ping) request
53	2021-03-01 13:4...	1.2.3.4	10.0.2.15	ICMP	100	Echo (ping) reply

▶ Frame 49: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0  
 ▶ Linux cooked capture  
 ▶ Internet Protocol Version 4, Src: 1.2.3.4, Dst: 10.9.0.5  
 ▶ Internet Control Message Protocol  
   Type: 0 (Echo (ping) reply)  
   Code: 0  
   Checksum: 0xcff8 [correct]

בתצלום ניתן לראות כי אכן נשלחה פאקטת icmp echo request ולאחר מכן נשלחה פאקטת icmp echo replay. נראה כי כתובות ה ip אכן תואמות לשליחת ה ping וכי התוכנית sniffer and spoffer

אכן עבדה בצורה תקינה והחזירה ping מ ip שאינו קיים ברשת

## ping 10.9.0.99 # a non-existing host on the LAN

תחילה בצענו ping לקו 10.9.0.99 זהו ip אשר לא קיים ברשת הפנימית

## קריאה ל ping ב: docker:

```

rtt min/avg/max/mdev = 12.773/23.621/152.604/19.820 ms, pipe 4
root@62047f7ef997:/# 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=102 ms
From 10.9.0.1: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.99)
64 bytes from 10.9.0.99: icmp_seq=2 ttl=64 time=28.3 ms
From 10.9.0.1: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.99)
64 bytes from 10.9.0.99: icmp_seq=3 ttl=64 time=22.4 ms
From 10.9.0.1: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.99)
64 bytes from 10.9.0.99: icmp_seq=4 ttl=64 time=15.9 ms
From 10.9.0.1: icmp_seq=5 Redirect Host(New nexthop: 10.9.0.99)
64 bytes from 10.9.0.99: icmp_seq=5 ttl=64 time=30.4 ms
From 10.9.0.1: icmp_seq=6 Redirect Host(New nexthop: 10.9.0.99)
64 bytes from 10.9.0.99: icmp_seq=6 ttl=64 time=24.9 ms
64 bytes from 10.9.0.99: icmp_seq=7 ttl=64 time=26.3 ms
From 10.9.0.1: icmp_seq=8 Redirect Host(New nexthop: 10.9.0.99)
64 bytes from 10.9.0.99: icmp_seq=8 ttl=64 time=21.2 ms

```

ניתן לראות כי אכן התקבלה תשובה לשליחת הבקשה לקו זה למרות שהוא אינו קיים ברשת הפנימית, זה מכיוון שהתוכנית sniffer and spoffer אכן פעלה ויצרה פאקטת arp פיקטיבית ושלחה שאכן מופיע host עם ה ip הנתון ברשת הפנימית שביקש את ה request.

## צילום Wireshark :

No.	Time	Source	Destination	Protocol	Length	Info
7	2021-03-07 06:54...	10.9.0.5	10.9.0.99	ICMP	100	Echo (ping) request id=0x0015, seq=1/256, ttl=64 (no res
8	2021-03-07 06:54...	10.9.0.5	10.9.0.99	ICMP	100	Echo (ping) request id=0x0015, seq=1/256, ttl=64 (reply
9	2021-03-07 06:54...	02:42:76:24:b2:85		ARP	44	Who has 10.9.0.99? Tell 10.9.0.1
10	2021-03-07 06:54...	02:42:76:24:b2:85		ARP	44	Who has 10.9.0.99? Tell 10.9.0.1
11	2021-03-07 06:54...	02:42:76:24:b2:85		ARP	44	Who has 10.9.0.5? Tell 10.9.0.1
12	2021-03-07 06:54...	02:42:76:24:b2:85		ARP	44	Who has 10.9.0.5? Tell 10.9.0.1
13	2021-03-07 06:54...	02:42:0a:09:00:05		ARP	44	10.9.0.5 is at 02:42:0a:09:00:05
14	2021-03-07 06:54...	02:42:0a:09:00:05		ARP	44	10.9.0.5 is at 02:42:0a:09:00:05
15	2021-03-07 06:54...	10.9.0.99	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0015, seq=1/256, ttl=64 (reque
16	2021-03-07 06:54...	10.9.0.99	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0015, seq=1/256, ttl=64
17	2021-03-07 06:54...	00:00:00_00:00:00		ARP	44	10.9.0.99 is at 00:00:00:00:00:00

▶ Frame 13: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface any, id 0  
 ▶ Linux cooked capture  
 ▶ Address Resolution Protocol (reply)  
   Hardware type: Ethernet (1)  
   Protocol type: IPv4 (0x0800)  
   Hardware size: 6  
   Protocol size: 4  
   Opcode: reply (2)  
   Sender MAC address: 02:42:0a:09:00:05 (02:42:0a:09:00:05)  
   Sender IP address: 10.9.0.5  
   Target MAC address: 02:42:76:24:b2:85 (02:42:76:24:b2:85)  
   Target IP address: 10.9.0.1

בתצלום ניתן לראות כי אכן נשלחה פקטת icmp echo request ולאחר מכן נשלחה פקטת icmp echo replay, אך בנוסף נראה כי נשלחה פקטת arp השואלת ברשת הפנימית מי מכיל את הכתובת קו הזו, הבקשה נשלחה ומחכה שמשהו יענה בתוך הרשת ויחזיר את address mac שלו, ניתן לראות כי החזרנו arp replay אשר מכיל את address mac. ומכאן הוחזר לכתובת 10.9.0.5 כי אכן הכתובת שאינה קיימת ברשת, קיימת ומחזירה icmp replay.

## ping 8.8.8.8 # an existing host on the Internet

תחילה בצענו ping לקו 8.8.8.8 זהו ip אשר קיים ברשת

קריאה ping ב: docker

```

root@62047f7ef997:/# 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=111 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=64 time=116 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=70.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=108 time=103 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=93.8 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=108 time=125 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=4 ttl=108 time=96.3 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=116 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=5 ttl=64 time=88.9 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=108 time=98.9 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=6 ttl=64 time=64.9 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=108 time=104 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=7 ttl=64 time=72.6 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=108 time=109 ms (DUP!)

```

ניתן לראות כי אכן התקבלה תשובה לשליחת הבקשה לקו 8.8.8.8, התשובה שהתקבלה הינה תשובה כפולה כמו שניתן להבחין בצילום המסך (!DUP). זאת מכיוון שכתובת ip זו (8.8.8.8) הינה של google.com ומכאן צריך לקבל replay icmp, בנוסף גם התוכנית שלנו תחזיר icmp replay. כפי שניתן לראות בטרמינל אכן מתקבלת תשובה כפולה מכתובת זו.

#### צילום Wireshark :

No.	Time	Source	Destination	Protocol	Length	Info
36	2021-03-07 09:18...	8.8.8.8	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0017, seq=2/512, ttl=108
37	2021-03-07 09:18...	8.8.8.8	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0017, seq=2/512, ttl=108
38	2021-03-07 09:18...	10.9.0.5	8.8.8.8	ICMP	100	Echo (ping) request id=0x0017, seq=3/768, ttl=64 (no res
39	2021-03-07 09:18...	10.9.0.5	8.8.8.8	ICMP	100	Echo (ping) request id=0x0017, seq=3/768, ttl=64 (reply
40	2021-03-07 09:18...	10.0.2.15	8.8.8.8	ICMP	100	Echo (ping) request id=0x0017, seq=3/768, ttl=63 (reply
41	2021-03-07 09:18...	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply id=0x0017, seq=3/768, ttl=64 (reques
42	2021-03-07 09:18...	8.8.8.8	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0017, seq=3/768, ttl=64 (reques
43	2021-03-07 09:18...	8.8.8.8	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0017, seq=3/768, ttl=64
44	2021-03-07 09:18...	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply id=0x0017, seq=3/768, ttl=109
45	2021-03-07 09:18...	8.8.8.8	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0017, seq=3/768, ttl=108
46	2021-03-07 09:18...	8.8.8.8	10.9.0.5	ICMP	100	Echo (ping) reply id=0x0017, seq=3/768, ttl=108

כפי שניתן לראות בצילום מסך זה בו נשלחה הפקטא מה host ב docker ו header של icmp reply שנשלחה על ידי התוכנית שלנו. נראה כי לכל icmp request נשלחו שני icmp replay, זאת מכיוון שאחת מגיעה מ google והשניה מגיעה מהתוכנית שלנו אשר יוצרת פקאטה בשיטת spoofing.

## חלק 2:

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

#### Task 2.1: Writing Packet Sniffing Program:

##### Task 2.1A: Understanding How a Sniffer Works

#### Question 1

1. הגדרת הסביבה -נבחר מאיזה אינטרפייס נרצה להסניף את פקטות

2. אתחול-נאתחל את pcap ונגדיר אותו לרחרח על האינטרפייס שלנו [pcap\_open\_live]

3. סינון תעבורה -נגדיר פילטר שעל פיו נרחרר כלומר אם נרצה לתפוס רק פקטות מסוג מסויים נצטרך להשתמש בפילטר המתאים.[pcap\_compile,pcap\_setfilter]

4.נבצע רחרוח של פקטות בלולאה ונעבד את המידע מהפקטות [pcap\_loop] בתוך פונקציה אשר תבצע פעולות עם המידע שנקלט

## Question 2

אנו צריכים את ה- root privilege על מנת שיהיו לנו את ההרשאות לבצע רחרוח  
ניתן לראות כי כאשר אנו מריצים את התכנית עם root privilege אכן מתקבלות פקטות

The screenshot shows a terminal window on the left and a Wireshark network capture window on the right. The terminal window displays the output of a ping command from 10.0.2.15 to 8.8.8.8. The output shows 4 packets transmitted, 3 received, and a 25% packet loss. The Wireshark window shows a capture of the network traffic. The capture list on the left shows several packets, including ICMP Echo (ping) replies and TCP segments. The packet details pane on the right shows the selected packet (No. 14) as an ICMP Echo (ping) reply from 8.8.8.8 to 10.0.2.15.

```
[03/01/21]seed@VM:~/../task 2.1$ 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=112 time=90.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=112 time=84.3 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=112 time=95.5 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3008ms
rtt min/avg/max/mdev = 84.254/90.061/95.548/4.616 ms
[03/01/21]seed@VM:~/../task 2.1$
```

Protocol: ICMP  
the source ip is: 8.8.8.8  
the destination ip is: 10.0.2.15  
Protocol: ICMP  
the source ip is: 10.0.2.15  
the destination ip is: 8.8.8.8  
Protocol: ICMP  
the source ip is: 8.8.8.8  
the destination ip is: 10.0.2.15  
Protocol: ICMP  
the source ip is: 10.0.2.15  
the destination ip is: 8.8.8.8  
Protocol: ICMP  
the source ip is: 8.8.8.8  
the destination ip is: 10.0.2.15  
Protocol: ICMP  
the source ip is: 10.0.2.15  
the destination ip is: 8.8.8.8  
Protocol: ICMP  
the source ip is: 8.8.8.8  
the destination ip is: 10.0.2.15

No.	Time	Source	Destination	Protocol	Length	Info
14	2021-03-01 14:11:14.111	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply
15	2021-03-01 14:11:14.111	185.60.216.53	10.0.2.15	TLSv1.2	112	Application Data
16	2021-03-01 14:11:14.111	185.60.216.53	185.60.216.53	TCP	56	37002 → 443 [ACK]
17	2021-03-01 14:11:14.111	185.60.216.53	10.0.2.15	TLSv1.2	406	Application Data
18	2021-03-01 14:11:14.111	185.60.216.53	10.0.2.15	TCP	56	37002 → 443 [ACK]
19	2021-03-01 14:11:14.111	185.60.216.53	10.0.2.15	TLSv1.2	1473	Application Data
20	2021-03-01 14:11:14.111	10.0.2.15	185.60.216.53	TCP	56	37002 → 443 [ACK]
21	2021-03-01 14:11:14.111	PcsCompu_91:d6:ff		ARP	44	Who has 10.0.2.2?
22	2021-03-01 14:11:14.111	RealtekU_12:35:02		ARP	62	10.0.2.2 is at 52:

Frame 1: 112 bytes on wire (896 bits), 112 bytes captured (896 bits) on interface any, id 0  
Linux cooked capture  
Internet Protocol Version 4, Src: 185.60.216.53, Dst: 10.0.2.15  
Transmission Control Protocol, Src Port: 443, Dst Port: 37002, Seq: 97479352, Ack: 3215126201, Len: 56  
Transport Layer Security

לעומת זאת ללא שימוש ב root privilege התכנית לא רצה

```
seed@VM: ~/../task 2.1
[03/01/21]seed@VM:~/../task 2.1$ ./sniff_r
Segmentation fault
[03/01/21]seed@VM:~/../task 2.1$ ^C
[03/01/21]seed@VM:~/../task 2.1$
```

### Quesiton 3

באופן כללי כאשר promiscuous mode כבוי לא נוכל להסניף פקטות שלא ממוענות אלינו או נשלחות ממנו אבל עדיין תהיה לנו אפשרות להסניף פקטות אשר נשלחות מהמחשב שלנו או נשלחות למחשב שלנו

במידה ויש לנו כמה מחשבים המחוברים לאותה רשת דרך מכשיר hub אז בכל פעם שפקטה מגיעה לפורט מסויים של ה hub היא מועתקת ונשלחת לכל שאר הפורטים המחוברים למחשב ולכן היא מגיעה לכל המחשבים המחוברים לרשת

אך במידה והם מחוברים דרך סוויץ- אזי הסוויץ משתמש בטבלת mac adress כלומר הסוויץ "לומד" את כתובות ה MAC ושומר אותם בטבלה ולכן פקטות שנשלחות על ידי שימוש בהתקן סוויץ לא יעברו דרך כל המחשבים המחוברים לאותה רשת אלא ישלחו ישירות למחשב הנמען .

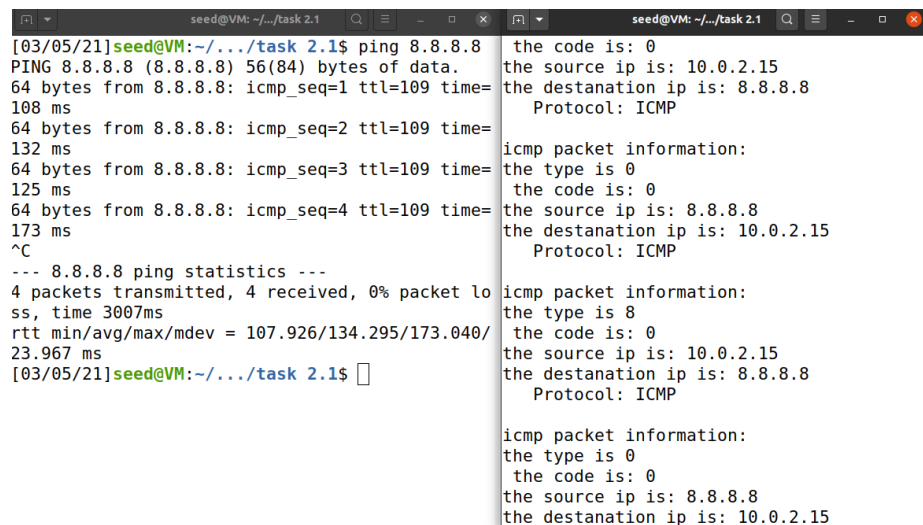
לכן בשאלה זו מאחר והנתב שלנו בעל התקן סוויץ -לא נראה שינוי, בין אם הפרמיסקיוס מוד יופעל ובין אם לא. כי פקטות שלא ממוענות אלינו או לא נשלחות מהמחשב שלנו לא יגיעו אלינו .

### Task 2.1B: Writing Filters

#### Capture the ICMP packets between two specific host

בשאלה זו אנו צריכים לכתוב sniffing program אשר תתפוס פקטות icmp הנשלחות מ host מסויים ל host מסויים אחר.

נריך את קוד ה sniffer לפקטות icmp שכתבנו ונשלח פינג ל ip 8.8.8.8 נרצה לראות כי הפקטות שנתפסו אלו רק פקטות שנשלחו מהמחשב שלנו ל 8.8.8.8 ולהיפך



```
seed@VM: ~/.../task 2.1$ ping 8.8.8.8
[03/05/21]seed@VM:~/.../task 2.1$ 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=109 time=108 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=109 time=132 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=109 time=125 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=109 time=173 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3007ms
rtt min/avg/max/mdev = 107.926/134.295/173.040/23.967 ms
[03/05/21]seed@VM:~/.../task 2.1$
```

```
the code is: 0
the source ip is: 10.0.2.15
the destination ip is: 8.8.8.8
Protocol: ICMP

icmp packet information:
the type is 0
the code is: 0
the source ip is: 8.8.8.8
the destination ip is: 10.0.2.15
Protocol: ICMP

icmp packet information:
the type is 8
the code is: 0
the source ip is: 10.0.2.15
the destination ip is: 8.8.8.8
Protocol: ICMP

icmp packet information:
the type is 0
the code is: 0
the source ip is: 8.8.8.8
the destination ip is: 10.0.2.15
```

ניתן לראות ב wireshark כי אכן רק פקטות שנשלחו בין 2 host מסויים לבחירתנו (המחשב שלנו 10.0.2.15 וגוגל 8.8.8.8 ) נתפסו.



No.	Time	Source	Destination	Protocol	Length	Info
5	2021-03-05 07:5...	10.0.2.15	8.8.8.8	ICMP	100	Echo (ping) request id=0x000a, seq=1/256,
6	2021-03-05 07:5...	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply id=0x000a, seq=1/256,
7	2021-03-05 07:5...	10.0.2.15	8.8.8.8	ICMP	100	Echo (ping) request id=0x000a, seq=2/512,
8	2021-03-05 07:5...	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply id=0x000a, seq=2/512,
9	2021-03-05 07:5...	10.0.2.15	8.8.8.8	ICMP	100	Echo (ping) request id=0x000a, seq=3/768,
10	2021-03-05 07:5...	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply id=0x000a, seq=3/768,
11	2021-03-05 07:5...	10.0.2.15	8.8.8.8	ICMP	100	Echo (ping) request id=0x000a, seq=4/1024,
12	2021-03-05 07:5...	8.8.8.8	10.0.2.15	ICMP	100	Echo (ping) reply id=0x000a, seq=4/1024,

<ul style="list-style-type: none"> <li>Frame 1: 96 bytes on wire (768 bits), 96 bytes captured (768 bits) on interface any, id 0</li> <li>Linux cooked capture</li> <li>Internet Protocol Version 4, Src: 10.0.2.15, Dst: 157.240.214.60</li> <li>Transmission Control Protocol, Src Port: 45878, Dst Port: 443, Seq: 3963457103, Ack: 2938389147, Len: 40</li> <li>Transport Layer Security</li> </ul>
---

## Capture the TCP packets with a destination port number in the range from 10 to 100

בשאלה זו התבקשנו לכתוב sniffer program אשר ירחרח אחר פקטות tcp שה dest port שלהן הוא בטווח שבין 10 ל 100

השתמשנו בפילטר זה על מנת לתפוס רק פקטות tcp שה dest port שלהן הוא בין 10-100

```
char filter_exp[] = "tcp and dst portrange 10-100";
```

על מנת לתפוס את הפקטות הרצויות לנו ביצענו פקודת telnet לקי 10.9.0.1 מהדוקר, ומהטרמינל הרצנו את קוד ה sniffer שכתבנו .

ניתן לראות כי אכן הפקטות שנתפסו הן פקטות tcp אשר ה dest port שלהן הוא 23 (ומקיים כי הוא בין 10-100) נציין כי telnet משתמש בפורט 23

```
seed@VM: ~/task 2.1
Protocol: TCP

tcp packet information:
the source port is 44956
the the destination port is: 23
the source ip is: 10.9.0.5
the destination ip is: 10.9.0.1
Protocol: TCP

tcp packet information:
the source port is 44956
the the destination port is: 23
the source ip is: 10.9.0.5
the destination ip is: 10.9.0.1
Protocol: TCP

tcp packet information:
the source port is 44956
the the destination port is: 23
the source ip is: 10.9.0.5
the destination ip is: 10.9.0.1
Protocol: TCP

tcp packet information:
the source port is 44956
the the destination port is: 23
the source ip is: 10.9.0.5
the destination ip is: 10.9.0.1
Protocol: TCP
```

12	2021-03-05 08:12...	10.9.0.5	10.9.0.1	TCP	68 [TCP Retransmission] 23 → 44
13	2021-03-05 08:12...	10.9.0.5	10.9.0.1	TCP	68 44956 → 23 [ACK] Seq=3421109
14	2021-03-05 08:12...	10.9.0.5	10.9.0.1	TCP	68 [TCP Dup ACK 13#1] 44956 → 2
15	2021-03-05 08:12...	10.9.0.5	10.9.0.1	TELNET	71 Telnet Data ...
16	2021-03-05 08:12...	10.9.0.5	10.9.0.1	TCP	71 [TCP Retransmission] 44956
17	2021-03-05 08:12...	10.9.0.1	10.9.0.5	TCP	68 23 → 44956 [ACK] Seq=1801312
18	2021-03-05 08:12...	10.9.0.1	10.9.0.5	TCP	68 [TCP Dup ACK 17#1] 23 → 4495
19	2021-03-05 08:12...	10.9.0.1	10.9.0.5	TELNET	101 Telnet Data ...
20	2021-03-05 08:12...	10.9.0.1	10.9.0.5	TCP	101 [TCP Retransmission] 23 → 44
21	2021-03-05 08:12...	10.9.0.5	10.9.0.1	TCP	68 44956 → 23 [ACK] Seq=3421109
22	2021-03-05 08:12...	10.9.0.5	10.9.0.1	TCP	68 [TCP Dup ACK 21#1] 44956 → 2
23	2021-03-05 08:12...	10.9.0.5	10.9.0.1	TELNET	111 Telnet Data ...
24	2021-03-05 08:12...	10.9.0.5	10.9.0.1	TCP	111 [TCP Retransmission] 44956
25	2021-03-05 08:12...	10.9.0.1	10.9.0.5	TCP	68 23 → 44956 [ACK] Seq=1801312
26	2021-03-05 08:12...	10.9.0.1	10.9.0.5	TCP	68 [TCP Dup ACK 25#1] 23 → 4495
27	2021-03-05 08:12...	10.9.0.1	10.9.0.5	TELNET	71 Telnet Data ...
28	2021-03-05 08:12...	10.9.0.1	10.9.0.5	TCP	71 [TCP Retransmission] 23 → 44

Frame 16: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface any, id 0  
 Linux cooked capture  
 Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.1  
 Transmission Control Protocol, Src Port: 44956, Dst Port: 23, Seq: 3421109888, Ack: 1801312191, Len: 3  
 Source Port: 44956  
 Destination Port: 23  
 [Stream index: 0]  
 [TCP Segment Len: 3]  
 Sequence number: 3421109888  
 [Next sequence number: 3421109891]

0000 00 00 00 01 00 06 02 42 0a 09 00 05 00 00 08 00 .....B .....

## Task 2.1C: Sniffing Passwords

בחלק זה התבקשנו לכתוב sniffing program על מנת למצוא את הסיסמא שמזינים כאשר משתמשים ב telnet

על מנת לעשות זאת הדפסנו את ה data של פקטות ה tcp שתפסנו, נסמן את הסיסמא ב (dees).data

```

the source ip is: 10.9.0.5
the destination ip is: 10.9.0.1
52the packet data is:
  ff?f_k*

tcp packet information:
the source port is 45168
the the destination port is: 23
the source ip is: 10.9.0.5
the destination ip is: 10.9.0.1
53the packet data is:
  ffD6_k*d      Nbb`_U_B_B      Z_h

tcp packet information:
the source port is 45168
the the destination port is: 23
the source ip is: 10.9.0.5
the destination ip is: 10.9.0.1
53the packet data is:
  ffE_p&e      Nbb`_N#B_B      Z_h

tcp packet information:
the source port is 45168
the the destination port is: 23
the source ip is: 10.9.0.5
the destination ip is: 10.9.0.1
53the packet data is:
  ffFK_q_e      Nbb`_*B_B      Z_h

tcp packet information:
the source port is 45168
the the destination port is: 23
the source ip is: 10.9.0.5
the destination ip is: 10.9.0.1
53the packet data is:
  ffG9_r_s      Nbb`_Z_8B_B      Z_h

tcp packet information:
the source port is 45168
the the destination port is: 23
the source ip is: 10.9.0.5
the destination ip is: 10.9.0.1
54the packet data is:
  ffI_r      0bb`_1_&B_B      Z_h

```



גם ב Wireshark ניתן לראות כי 4 הפקטות המכילות את הסיסמא נתפסו אחת אחרי השניה

97	2021-03-05 11:5...	10.9.0.5	10.9.0.1	TELNET	69 Telnet Data ...
98	2021-03-05 11:5...	10.9.0.5	10.9.0.1	TCP	69 [TCP Keep-Alive] 45168 → 23 [PSH, ACK] Seq=3781775032
99	2021-03-05 11:5...	10.9.0.1	10.9.0.5	TCP	68 23 → 45168 [ACK] Seq=571764231 Ack=3781775033 Win=651
100	2021-03-05 11:5...	10.9.0.1	10.9.0.5	TCP	68 [TCP Keep-Alive ACK] 23 → 45168 [ACK] Seq=571764231 A
101	2021-03-05 11:5...	10.9.0.5	10.9.0.1	TELNET	69 Telnet Data ...
102	2021-03-05 11:5...	10.9.0.5	10.9.0.1	TCP	69 [TCP Keep-Alive] 45168 → 23 [PSH, ACK] Seq=3781775033
103	2021-03-05 11:5...	10.9.0.1	10.9.0.5	TCP	68 23 → 45168 [ACK] Seq=571764231 Ack=3781775033 Win=651
104	2021-03-05 11:5...	10.9.0.1	10.9.0.5	TCP	68 [TCP Keep-Alive ACK] 23 → 45168 [ACK] Seq=571764231 A
105	2021-03-05 11:5...	10.9.0.5	10.9.0.1	TELNET	69 Telnet Data ...
106	2021-03-05 11:5...	10.9.0.5	10.9.0.1	TCP	69 [TCP Keep-Alive] 45168 → 23 [PSH, ACK] Seq=3781775034
107	2021-03-05 11:5...	10.9.0.1	10.9.0.5	TCP	68 23 → 45168 [ACK] Seq=571764231 Ack=3781775035 Win=651
108	2021-03-05 11:5...	10.9.0.1	10.9.0.5	TCP	68 [TCP Keep-Alive ACK] 23 → 45168 [ACK] Seq=571764231 A
109	2021-03-05 11:5...	10.9.0.5	10.9.0.1	TELNET	70 Telnet Data ...

על מנת לראות יותר בבירור נכנסנו למידע על כל פקטה וסימנו את האות הנמצאת ב data של אותה פקטה נקרא את האותיות מימין לשמאל לפי סדר התפיסה של הפקטות (מ93 עד 105)

ניתן לראות כי הסיסמא שמופיעה באיזור הpayload של פאקטות ה tcp הינו "dees" וזו היא אכן הסיסמא למשתמש הseed

## Task 2.2A: Write a spoofing program

בחלק זה נבצע ספויפינג לפקטות icmp או ניצור פקטות icmp של replay שכתובת ה src שלה נערכה על ידינו לכתובת ip שאינה קיימת ברשת

ניתן לראות כי אכן נקלטה ב Wireshark פקטת replay מכתובת 1.2.3.4 אל כתובת 10.9.0.1

No.	Time	Source	Destination	Protocol	Length	Info
4	2021-03-05 06:2...	10.0.2.15	157.240.221.60	TCP	56	34900 → 443 [ACK] Seq=3576410909 Ack=283290
5	2021-03-05 06:2...	157.240.221.60	10.0.2.15	TLSv1.2	321	Application Data
6	2021-03-05 06:2...	10.0.2.15	157.240.221.60	TCP	56	34900 → 443 [ACK] Seq=3576410909 Ack=283290
7	2021-03-05 06:2...	1.2.3.4	10.9.0.1	ICMP	44	Echo (ping) reply id=0x0000, seq=0/0, tt
8	2021-03-05 06:2...	127.0.0.1	127.0.0.53	DNS	99	Standard query 0x9e70 A firefox.settings.se
9	2021-03-05 06:2...	10.0.2.15	192.168.1.1	DNS	110	Standard query 0x3056 A firefox.settings.se

Frame 7: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface any, id 0  
 Linux cooked capture  
 Internet Protocol Version 4, Src: 1.2.3.4, Dst: 10.9.0.1  
 Internet Control Message Protocol  
 Type: 0 (Echo (ping) reply)  
 Code: 0  
 Checksum: 0xffff [correct]  
 [Checksum Status: Good]  
 Identifier (BE): 0 (0x0000)  
 Identifier (LE): 0 (0x0000)  
 Sequence number (BE): 0 (0x0000)  
 Sequence number (LE): 0 (0x0000)

## Task 2.2B: Spoof an ICMP Echo Request

בחלק זה נבצע spoofing לפקטת icmp אנו ניצור פקטת בקשה icmp אשר נערוך את כתובת ה src שלה לכתובת מפוברקת שאינה בהכרח קיימת. נשלח פקטה זו ונרצה לראות כי אכן התקבלה פקטת תגובה לאותה הכתובת.

ניתן לראות כי נקלטו בwireshark 2 פקטות - פקטת request מכתובת 1.2.3.4 אל כתובת 10.9.0.1 ואכן נשלחה פקטת reply מכתובת 10.9.0.1 אל 1.2.3.4

6	2021-03-05 06:3...	44.238.3.246	10.0.2.15	TCP	62	443 → 55876 [ACK] Seq=28081406
7	2021-03-05 06:3...	1.2.3.4	10.9.0.1	ICMP	44	Echo (ping) request id=0x0000
8	2021-03-05 06:3...	10.9.0.1	1.2.3.4	ICMP	44	Echo (ping) reply id=0x0000
9	2021-03-05 06:3...	10.0.2.15	157.240.221.60	TLSv1.2	96	Application Data

Frame 7: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface any, id 0  
 Linux cooked capture  
 Internet Protocol Version 4, Src: 1.2.3.4, Dst: 10.9.0.1  
 Internet Control Message Protocol  
 Type: 8 (Echo (ping) request)  
 Code: 0  
 Checksum: 0xf7ff [correct]  
 [Checksum Status: Good]  
 Identifier (BE): 0 (0x0000)  
 Identifier (LE): 0 (0x0000)  
 Sequence number (BE): 0 (0x0000)

## Question 4

בשאלה זו התבקשנו לבדוק האם אפשר לשנות את שדה האורך של ה ip לאורך שירותי ללא התחשבות בגודל הפקטה במציאות

לא ניתן לעשות זאת

ניתן לראות כי כאשר שינינו שדה זה לאורך 6 למשל, קיבלנו הודעת שגיאה ב wireshark

Malformed packet הודעה זו מצביעה על כך שלא ניתן להמשיך ולנתח את תוכן הפקטה ישנן כמה סיבות שבהן ניתן לקבל הודעת שגיאה זו, אחת מהן היא אכן הבדל בן אורך הפקטה שהוזן לאורך הפקטה במציאות

The screenshot shows a terminal window at the top with the command `sudo ./spoofer_rqt` being executed. Below the terminal is the Wireshark network protocol analyzer interface. The packet list pane shows two captured packets, both of type ICMP. The second packet, at time 2.2021, is highlighted in red and labeled as '[Malformed Packet]'. The packet details pane for this packet shows the following information:

- Frame 1: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface any, id 0
- Linux cooked capture
- Internet Protocol Version 4, Src: 1.2.3.4, Dst: 10.9.0.1
- 0100 ..... = Version: 4
- ... 0110 = Header Length: 24 bytes (6)
- Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
- Total Length: 28
- Identification: 0x0136 (310)
- Flags: 0x0000
- Fragment offset: 0
- Time to live: 99
- Protocol: ICMP (1)
- Header checksum: 0x479c [validation disabled]
- [Header checksum status: Unverified]
- Source: 1.2.3.4
- Destination: 10.9.0.1
- Options: (4 bytes)
- [Malformed Packet: ICMP]
- [Expert Info (Error/Malformed): Malformed Packet (Exception occurred)]

## Question 5

לא חייב לחשב את ה checksum של ה ip מאחר והוא מחושב באופן אוטומטי על ידי מערכת ההפעלה

## Question 6

ללא שימוש ב root privilege לא נוכל לפתוח raw socket פעולה זו מוגנת עי הרשאות root כדי למנוע ממשתמשים להגיע לחמרה הפיזית של המחשב ככלל כאשר אנו מחוברים כרוט ישנה אפשרות לעשות טעויות חמורות כמו למשל למחוק את ה root directory או קבצים חשובים לכן פעולות מסוימות מוגנות על ידי הרשאות רוט כדי למנוע מטעויות שכלול לקרות.

## Task 2.3: Sniff and then Spoof

בשאלה זו התבקשנו לכתוב תוכנית המרחיחה פאקטות (במקרה הנ"ל פקטות מסוג icmp) ולאחר מכן יוצרת פאקטה מזויפת אשר מיצגת תשובה של icmp (icmp replay). כדי ליצור תוכנית כזו, כתבנו תוכנית אשר מכילה סניפר בעל פילטר של פאקטות icmp בלבד. התוכנית מבצעת spoof רק כאשר פאקטות ה icmp הינן מ type 8, פאקטות מסוג icmp request. במידה ואכן מ type 8 ניצור פאקטת icmp בעלת המידע הנחוץ ולאחר מכן נשלח אותה אל כתובת היעד(שהיא הינה כתובת ה source של הפאקטה שרחרחנו).

ביצענו זאת על ידי ה seed vm hosti מן ה docker אשר מדמה מכונה וירטואלית נוספת.

```

seed@VM: ~/.../Labsetup$ dockps
62047f7ef997 host-10.9.0.5
f6df85d61401 seed-attacker
[03/06/21]seed@VM:~/.../Labsetup$ docksh 62
root@62047f7ef997:/# 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=99 time=
507 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=99 time=
531 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=99 time=
555 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=99 time=
577 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=99 time=
601 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=99 time=
624 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=99 time=
648 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=99 time=
675 ms

seed@VM: ~/.../task 2.3$ gcc -Wall -g -o s_and_s sniff_and_spoof.c -
lpcap
sniff_and_spoof.c: In function 'main':
sniff_and_spoof.c:131:3: warning: 'net' is
used uninitialized in this function [-Wunin
itialized]
131 | pcap_compile(handle, &fp, "icmp"
    | ^
    | ^
[03/06/21]seed@VM:~/.../task 2.3$ sudo ./s
_and_s
spoofing packet.....
spoofing packet.....
spoofing packet.....
spoofing packet.....
spoofing packet.....
spoofing packet.....
spoofing packet.....
spoofing packet.....

```

בצילומי המסך הבאים נראה כי הרצנו Ping מהhost ב docker לכתובת 1.2.3.4 , הרצנו לכתובת זו כי  
 אנו יודעים שהיא אינה כתובת אמיתית ( אינה אמורה להחזיר תשובה ) . כפי שניתן לראות בטרמינל אכן  
 מתקבלת תשובה מכתובת זו. בנוסף ניתן לראות בטרמינל של הseed vm כי אכן נשלחת פקאטת  
 .spoofing

### צילום מסך Wireshark:

The image shows a Wireshark packet capture. The top pane displays a list of packets. Packet 4 is an ICMP Echo (ping) request from 10.9.0.5 to 1.2.3.4. Packet 5 is the corresponding reply from 1.2.3.4 to 10.9.0.5. The bottom pane shows the details of packet 4, which is an ICMP Echo (ping) request. The details include: Type: 8 (Echo (ping) request), Code: 0, Checksum: 0xfbf3 [correct], Identifier (BE): 16 (0x0010), Identifier (LE): 4096 (0x1000), Sequence number (BE): 1 (0x0001), Sequence number (LE): 256 (0x0100), and Response frame: 6. The timestamp is Mar 6, 2021 14:34:12.000000000 EST.

כפי שניתן לראות בצילום מסך זה בו נראים הפרטים של הheader של הicmp request שנשלחה  
 מהuser ב docker . נראה כי פרטי הicmp , seq , id, 'היו זהות ל icmp replay בתמונת המסך ההבאה  
 של הפקאטה שביצענו איתה spoof . בנוסף נראה כי אכן התקבלה תשובה לפקאטה זו

## צילום מסך Wireshark:

5	2021-03-06 14:34...	10.0.2.15	1.2.3.4	ICMP	100	Echo (ping) request	id=0x0010, seq=1/256, tt...
6	2021-03-06 14:34...	1.2.3.4	10.9.0.5	ICMP	100	Echo (ping) reply	id=0x0010, seq=1/256, tt...
7	2021-03-06 14:34...	1.2.3.4	10.9.0.5	ICMP	100	Echo (ping) reply	id=0x0010, seq=1/256, tt...
8	2021-03-06 14:34...	10.9.0.5	1.2.3.4	ICMP	100	Echo (ping) request	id=0x0010, seq=2/256, tt...

▶ Frame 6: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0  
 ▶ Linux cooked capture  
 ▶ Internet Protocol Version 4, Src: 1.2.3.4, Dst: 10.9.0.5  
 ▶ Internet Control Message Protocol  
   Type: 0 (Echo (ping) reply)  
   Code: 0  
   Checksum: 0x03f4 [correct]  
   [Checksum Status: Good]  
   Identifier (BE): 16 (0x0010)  
   Identifier (LE): 4096 (0x1000)  
   Sequence number (BE): 1 (0x0001)  
   Sequence number (LE): 256 (0x0100)  
   [Request frame: 4]  
   [Response time: 930.723 ms]  
   Timestamp from icmp data: Mar 6, 2021 14:34:12.000000000 EST  
   [Timestamp from icmp data (relative): 1.450614850 seconds]

כפי שניתן לראות בצילום מסך זה בו נראים הפרטים של header של icmp reply שנשלחה על ידי התוכנית שלנו. נראה כי פרטי icmp, id, seq, זהים לתמונת המסך הקודמת שייצגה את פאקטת icmp request בנוסף נראה כי אכן הchecksum של הפאקטה חושב בצורה נכונה, וכי היא אכן מיוצגת type 0 אשר מייצגת replay.

## Docker terminal and seed vm terminal

```

seed@VM: ~/.../task 2.3
[03/06/21]seed@VM:~/.../task 2.3$ sudo ./s_and_s
spoofing packet.....
spoofing packet.....
spoofing packet.....
spoofing packet.....
spoofing packet.....
spoofing packet.....

seed@VM: ~/.../Labsetup
6 packets transmitted, 6 received, +6 duplicates, 0% packet loss, time 500lms
rtt min/avg/max/mdev = 17.685/155.793/1017.326/260.707 ms, pipe 2
root@62047f7ef997:/# 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=89.9 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=99 time=530 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=108 time=169 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=99 time=540 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=108 time=83.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=99 time=562 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=4 ttl=108 time=88.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=99 time=586 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=5 ttl=108 time=83.9 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=99 time=608 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=6 ttl=108 time=83.4 ms

```

בצילומי המסך הבאים נראה כי הרצנו Ping מהhost ב docker לכתובת 8.8.8.8, הרצנו לכתובת זו כי ברצוננו לראות שאכן נקבל תשובה כפולה, זאת מכיוון שכתובת קו זו הינה של google.com ומכאן צריך לקבל icmp replay, בנוסף גם התוכנית שלנו תחזיר icmp replay. כפי שניתן לראות בטרמינל אכן מתקבלת תשובה כפולה מכתובת זו. בנוסף ניתן לראות בטרמינל של seed vm כי אכן נשלחת פקאטת spoofing.

## צילום מסך Wireshark:

11	2021-03-06 14:47...	10.9.0.5	8.8.8.8	ICMP	100 Echo (ping) request	id=0x0012, seq=1/256, ttl=64 (no res
12	2021-03-06 14:47...	10.9.0.5	8.8.8.8	ICMP	100 Echo (ping) request	id=0x0012, seq=1/256, ttl=64 (reply
13	2021-03-06 14:47...	10.0.2.15	8.8.8.8	ICMP	100 Echo (ping) request	id=0x0012, seq=1/256, ttl=63 (reply
14	2021-03-06 14:47...	8.8.8.8	10.0.2.15	ICMP	100 Echo (ping) reply	id=0x0012, seq=1/256, ttl=109 (requ
15	2021-03-06 14:47...	8.8.8.8	10.9.0.5	ICMP	100 Echo (ping) reply	id=0x0012, seq=1/256, ttl=108 (requ
16	2021-03-06 14:47...	8.8.8.8	10.9.0.5	ICMP	100 Echo (ping) reply	id=0x0012, seq=1/256, ttl=108
17	2021-03-06 14:47...	8.8.8.8	10.9.0.5	ICMP	100 Echo (ping) reply	id=0x0012, seq=1/256, ttl=99
18	2021-03-06 14:47...	8.8.8.8	10.9.0.5	ICMP	100 Echo (ping) reply	id=0x0012, seq=1/256, ttl=99
19	2021-03-06 14:47...	10.9.0.5	8.8.8.8	ICMP	100 Echo (ping) request	id=0x0012, seq=2/512, ttl=64 (no res
20	2021-03-06 14:47...	10.9.0.5	8.8.8.8	ICMP	100 Echo (ping) request	id=0x0012, seq=2/512, ttl=64 (reply
21	2021-03-06 14:47...	10.0.2.15	8.8.8.8	ICMP	100 Echo (ping) request	id=0x0012, seq=2/512, ttl=63 (reply

כפי שניתן לראות בצילום מסך זה בו נראים הפרטים של header של icmp request אשר נשלחה מהדבר host ב docker ו header של icmp reply שנשלחה על ידי התוכנית שלנו . נראה כי לכל icmp request נשלחו שני icmp replay , זאת מכיוון שאחת מגיעה מ google והשניה מגיעה מהתוכנית שלנו אשר יוצרת פקאטה בשיטת spoofing .

## מקורות אינטרנטיים:

- <https://scapy.readthedocs.io/en/latest/usage.html>  
השתמשנו באתר כדי ללמוד את צורת הכתיבה של פילטרים ש scapy כדי לרחרח אחר פטקטות ספציפיות . בשאלה 1.2 התבקשנו לרחרח אחר פאקטות מסוג tcp , icmp .  
השתמשנו במקור מידע זה לשאלות 1.2, 1.3, 1.4
- [https://subscription.packtpub.com/book/networking\\_and\\_servers/9/781784399771/10/ch10lvl1sec59/arp-watcher](https://subscription.packtpub.com/book/networking_and_servers/9/781784399771/10/ch10lvl1sec59/arp-watcher)  
השתמשנו באתר הנלווה כדי להבין את מבנה הפאקטה של arp לשאלה 1.4
- <https://cryptsus.com/blog/icmp-reverse-shell.html>  
נעזרנו באתר זה לקביעת מבנה הפקאטה של ip ו icmp בפיתון , נעזרנו ב 1.3-1.4
- <https://linux.die.net/man/7/pcap-filter>  
באתר זה נעזרנו כדי לקבוע את מבנה כתיבת הפילטרים הדרושים בשאלות 2.1-2.3
- <https://code.woboq.org/qt5/include/netinet/ip.h.html>  
באתר זה נעזרנו לגבי המבנה של ip מתוך /netinet/ip.h 2.1-2.3

\*

<https://unix.superglobalmegacorp.com/BSD4.4/newsrsrc/inet/tcp.h.html>

באתר זה נעזרנו לגבי המבנה של tcpמתוך /netinet/tcp

2.1-2.2

#### הערות נוספות:

- רוב קבצי ה pcap הינם קבצים של הצילומי מסך מן המסמך הנלווה, אבל ישנם קבצי pcap שיצרנו לאחר מכן (קבצים זהים במידע) זאת מכיוון שרק באמצע המטלה גילינו שצריך לצרף קבצי pcap ולא רק תצלומי מסך של wireshark
- קודים שונים מן התשובות שהובאו בחלק של הקבצי c נלקחו ונערכו מן הקודים שפורסמו אצל המתרגל נור שליאון בתרגולים 7, 8. השתמשנו בקוד של pcap לסניפרים השונים, ובנוסף השתמשנו בקודים מן התרגולים כדי ליצור spoof לפאקטה ויצירת row socket.