

פרויקט סיום – קורס תקשורת –

:Sniffing & Spoofing

Environment Setup using Container

להלן זה מדובר על סביבת העבודה שבה הותכו המשימות בהמשך. קיימות שני אפשרויות או להשתמש בשני Containers או להשתמש בשני מוכנות VMs

אנחנו בחרנו להשתמש בשני מוכנות. בהמשך המטלה תהיה משימה אשר מתיחסת ל-LAN של docker אשר נראה עבור המשימה הזאת אשר לא רלוונטי אלינו בגלל בחירת העבודה עם שני מוכנות ונראה מה אנחנו צריכים לבצע כדי שתהיה משימה.

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

להלן ראשוני במטלה נשתמש בספרית Scapy שבספיישן ו"נרחח" פקודות וגם נציג פקודות.

- **Task 1.1: Sniffing Packets**

Iface	MTU
docker0	1500
enp0s3	1500
lo	65536

We are going to work on the enp0s3 interface

אנחנו נבצע הסנהפה של פקודות באינטראפיס הנ"ל בעזרתו הקוד הבא:

```
from scapy.all import *

def print_pkt(pkt):
    pkt.show()

print("~~~ Sniffing... ~~~")

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

Task 1.1A.

באופן כללי, צריך את הרשות מנהל על מנת להסניף פקודות במכשיר. זה אמצעי אבטחה מכיוון שהטהlixir שבו אנו משתמשים על מנת לקבל פקודות מקבל גישה לכל התהליכי האחרים והמשתמשים האחרים אשר נמצאים במכשיר זה.

תמונה להמחשה:

```
[03/04/21]seed@VM:~/.../Task1.1A$ chmod +x ScapySniff.py
[03/04/21]seed@VM:~/.../Task1.1A$ python3 ScapySniff.py
---Sniffing...
Traceback (most recent call last):
  File "ScapySniff.py", line 7, in <module>
    pkts = sniff(iface='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 906, in _run
    sniff_socket(type=ETH_P_ALL, iface=iface,
  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
[03/04/21]seed@VM:~/.../Task1.1A$ sudo python3 ScapySniff.py
---Sniffing...
###[ Ethernet ]###
dst      = 52:54:00:12:35:00
src      = 08:00:27:76:50:1a
type     = IPv4
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x0
len      = 84
id       = 64486
flags    = DF
frag     = 0
ttl      = 64
proto    = icmp
chksum   = 0x22ad
src      = 10.0.2.6
dst      = 8.8.8.8
[03/04/21]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=112 t
ime=91.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=112 t
ime=91.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=112 t
ime=90.9 ms
^C
-- 8.8.8.8 ping statistics --
3 packets transmitted, 3 received, 0% packe
t loss, time 2006ms
rtt min/avg/max/mdev = 90.872/91.472/91.891
/0.435 ms
[03/04/21]seed@VM:~$
```

Task 1.1B.

בבעוף זה נראה כיצד אנו משתמשים בפילטר BPF אשר מבצע את הסינון כבר מיד אחרי השכבה הראשונה ובכך אנו חוזים במסרים רבים של המעבד.

- Capture only the ICMP packet

```
[02/28/21]seed@VM:~/.../Task1.1A$ chmod a+x ScapySniff.py
[02/28/21]seed@VM:~/.../Task1.1A$ sudo python3 ScapySniff.py
--- Sniffing... ICMP only ---
###[ Ethernet ]##
    dst      = 54:50:00:12:35:00
    src      = 08:00:27:76:50:1a
    type     = IPv4
###[ IP ]##
    version   = 4
    ihl       = 5
    tos       = 0x0
    len       = 84
    id        = 56884
    flags     = DF
    frag      = 0
    ttl       = 64
    proto     = icmp
    chksum   = 0x405f
    src       = 10.0.2.6
    dst       = 8.8.8.8
    \options \
###[ ICMP ]##
    type      = echo-request
    code      = 0
    checksum  = 0xdalc
    id        = 0x3
    seq       = 0x1
###[ Raw ]##
    load      = '\xfel\x8c`\x00\x00\x00\x00#\x1f\x0f\x00\x00\x00\x00\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !%$&(`)+.../01234567'
###[ Ethernet ]##
    dst      = 08:00:27:76:50:1a
[02/28/21]seed@VM:~$ 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=113 time=78.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=78.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=113 time=78.4 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=113 time=78.0 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=113 time=78.7 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=113 time=78.9 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=113 time=77.9 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=113 time=78.3 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=113 time=80.6 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=113 time=78.1 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=113 time=78.4 ms
64 bytes from 8.8.8.8: icmp_seq=12 ttl=113 time=78.1 ms
64 bytes from 8.8.8.8: icmp_seq=13 ttl=113 time=78.5 ms
64 bytes from 8.8.8.8: icmp_seq=14 ttl=113 time=78.1 ms
64 bytes from 8.8.8.8: icmp_seq=15 ttl=113 time=78.1 ms
64 bytes from 8.8.8.8: icmp_seq=16 ttl=113 time=78.4 ms
64 bytes from 8.8.8.8: icmp_seq=17 ttl=113 time=78.6 ms
64 bytes from 8.8.8.8: icmp_seq=18 ttl=113 time=78.8 ms
64 bytes from 8.8.8.8: icmp_seq=19 ttl=113 time=78.6 ms
64 bytes from 8.8.8.8: icmp_seq=20 ttl=113 time=78.4 ms
64 bytes from 8.8.8.8: icmp_seq=21 ttl=113 time=79.0 ms
64 bytes from 8.8.8.8: icmp_seq=22 ttl=113 time=77.5 ms
64 bytes from 8.8.8.8: icmp_seq=23 ttl=113 time=90.0 ms
64 bytes from 8.8.8.8: icmp_seq=24 ttl=113 time=78.4 ms
^Z
[1]+  Stopped                  ping 8.8.8.8
[02/28/21]seed@VM:~$ █
```

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

telnet using port 23 on default and TCP protocol

```

seed@VM:~/Task1.1A$ chmod a+x ScapySniff.py
[02/28/21]seed@VM:~/Task1.1A$ sudo python3 ScapySniff.py
--- Sniffing.. TCP with condition only ---
###[ Ethernet ]##
dst      = 52:54:00:12:35:00
src      = 08:00:27:76:50:1a
type     = IPv4
###[ IP ]##
version  = 4
ihl      = 5
tos      = 0x10
len      = 60
id       = 336
flags    = DF
frag     = 0
ttl      = 64
proto    = tcp
chksum   = 0xd47
src      = 10.0.2.6
dst      = 8.8.8.8
\options  \
###[ TCP ]##
sport    = 60482
dport    = telnet
seq      = 3733640012
ack      = 0
dataofs  = 10
reserved = 0
flags    = S
window   = 64240
chksum   = 0xc44
urgptr   = 0
options  = [('MSS', 1460), ('SACKOK', b''), ('Timestamp', (2443749687, 0)), ('NOP', None), ('WScale', 7)]
[02/28/21]seed@VM:~$ telnet 8.8.8.8
Trying 8.8.8.8...

```

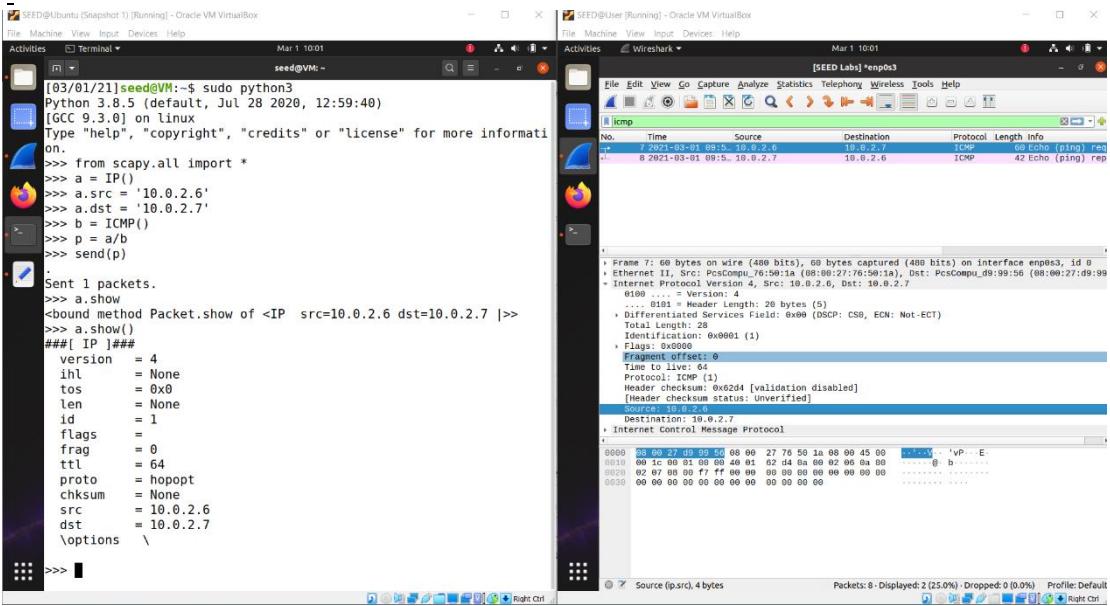
- Capture packets comes from or to go to a particular subnet. You can pick any subnet, such as 128.230.0.0/16; you should not pick the subnet that your VM is attached to.

```

###[ IP ]##
version  = 4
ihl      = 5
tos      = 0x0
len      = 64
id       = 31941
flags    = DF
frag     = 0
ttl      = 64
proto    = icmp
chksum   = 0x30f8
src      = 10.0.2.6
dst      = 128.230.0.0
\options  \
###[ ICMP ]##
type     = echo-request
code    = 0
chksum  = 0xb1a
id       = 0x5
seq      = 0x9
###[ Raw ]##
load    = '\x9f\x93;\x00\x00\x00\x00\x00\x11\x00\x00\x00\x00\x00\x00\x00\x10\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f "#$%`!*)**..01234567'
###[ Ethernet ]##
dst      = 52:54:00:12:35:00
src      = 08:00:27:76:50:1a
type     = IPv4
###[ IP ]##
version  = 4
ihl      = 5
tos      = 0x10
len      = 60
id       = 52175
flags    = DF
frag     = 0
ttl      = 64
proto    = tcp
chksum   = 0xeff0
src      = 10.0.2.6
dst      = 128.230.0.0
\options  \
###[ TCP ]##
sport    = 55224
dport    = telnet
seq      = 3956532276
ack      = 0
dataofs  = 10
reserved = 0
[02/28/21]seed@VM:~$ ping 128.230.0.0
PING 128.230.0.0 (128.230.0.0) 56(84) bytes of data.
^C
--- 128.230.0.0 ping statistics ---
9 packets transmitted, 0 received, 100% packet loss, time 8183ms
[02/28/21]seed@VM:~$ telnet 128.230.0.0
Trying 128.230.0.0...
^C
[02/28/21]seed@VM:~$ 

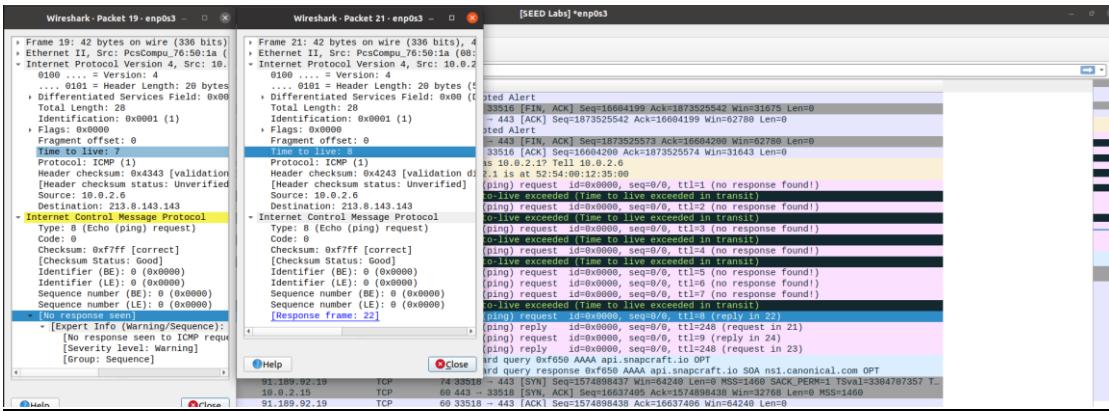
```

Task 1.2: Spoofing ICMP Packets



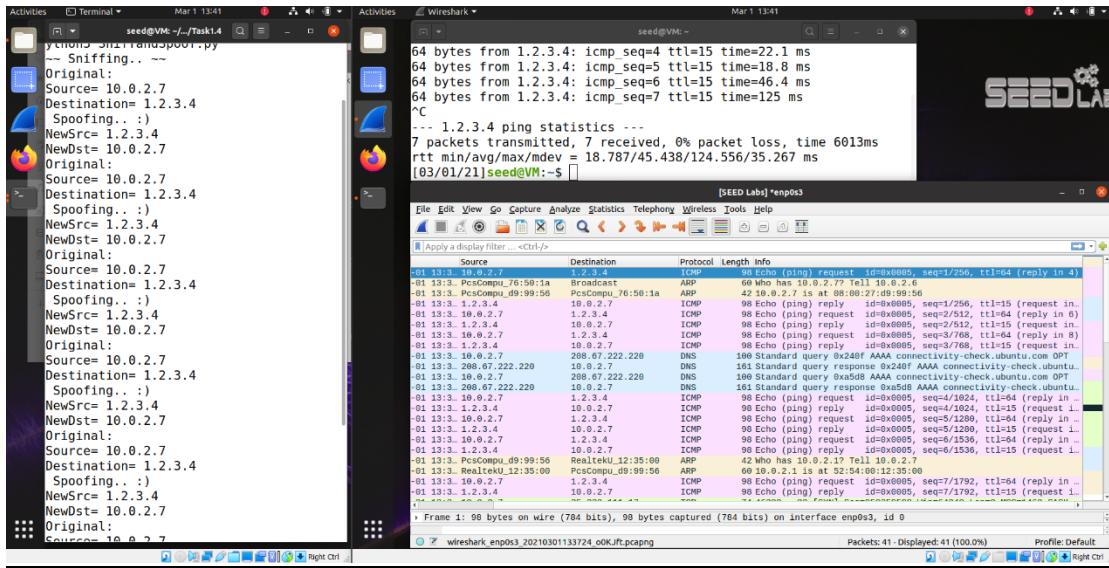
Task 1.3: Traceroute

אנחנו יכולים לראות כאשר אנחנו פותחים את הפקודות שנטפסו בWireshark
הקוד שלוח בולולאת פור בכל פעם עם Time to Live שוגד מלמטה מ1 ומוסים ב10
ייתן לראות שעד 7 אין תגובה מהצד השני והבקשה נגמר לה הח'יט
בTTL = 8 ייתן לראות שכבר קיבלנו תגובה וכן זה המינימום !



Task 1.4: Sniffing and-then Spoofing

בכותרת 1.2.3.4 שלא נמצא באינטראקט
בבקשת ARP הראשונה היא יצאת מהרשות LAN שלנו המקומית וועברת להלאה והלאה
המכונה השנייה תופסת את הפקודות האלה ומזיפת תגובה.



עבור כתובות הIp השניה המבוקשת 10.9.0.99

השליחה שלא תבוצע כמו הכתובת הקודמת אם היינו מבצעים את המטלה בשני Containers
הדבר היה פועל על הרשות המקומית 10.9.0.0 וכאן מכיוון הרשות המקומית שלנו היא
10.0.2.99 לעומת 10.0.2.0

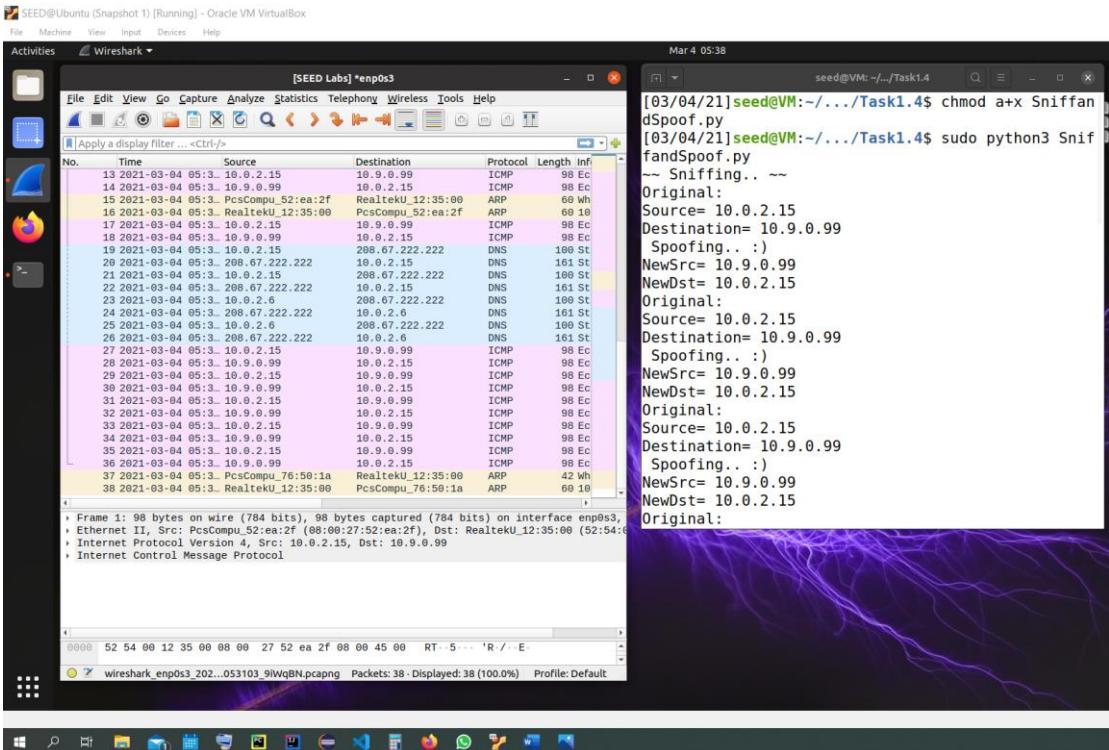
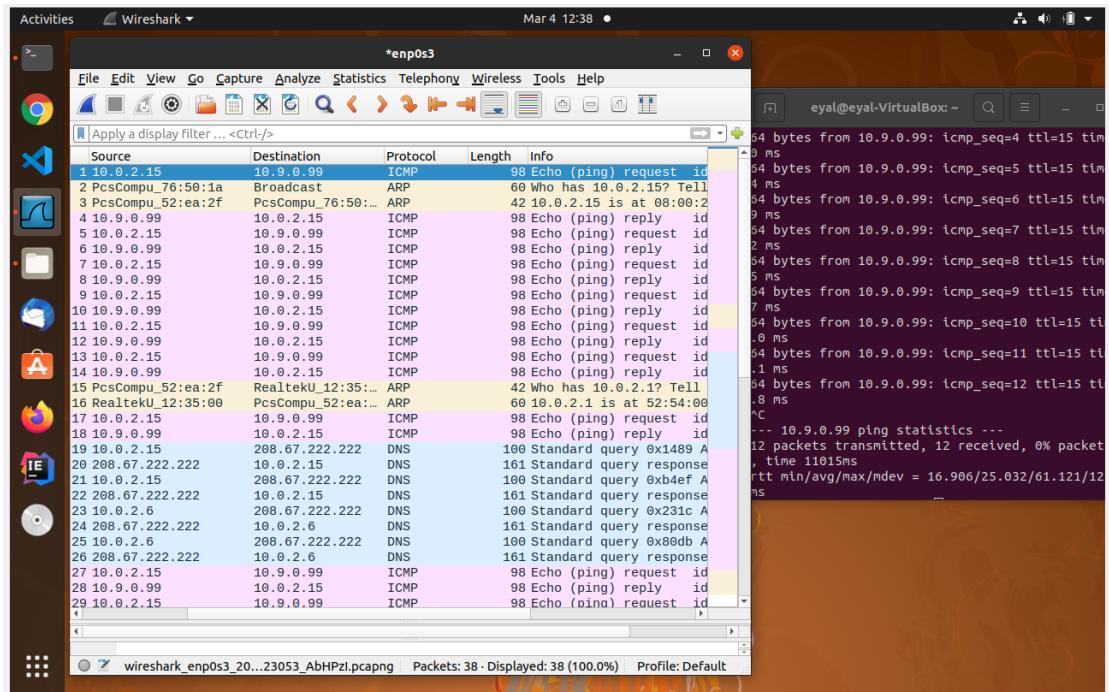
```
[03/04/21]seed@VM:~$ nmap 10.0.2.0/24
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-04 05:15 EST
Nmap scan report for _gateway (10.0.2.1)
Host is up (0.00055s latency).
Not shown: 999 closed ports
PORT      STATE      SERVICE
53/tcp    filtered domain

Nmap scan report for VM (10.0.2.6)
Host is up (0.00092s latency).
Not shown: 997 closed ports
PORT      STATE      SERVICE
21/tcp    open       ftp
22/tcp    open       ssh
23/tcp    open       telnet

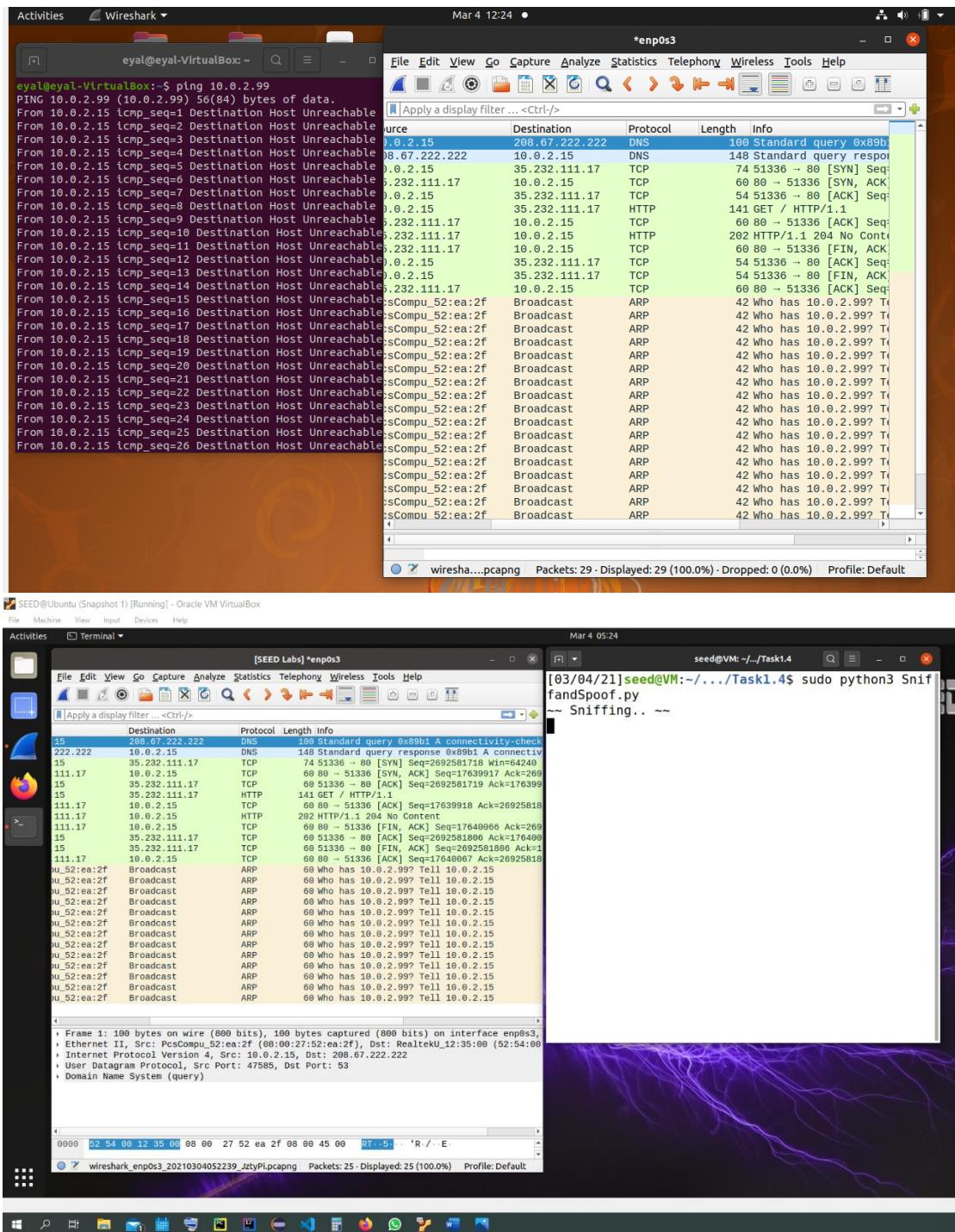
Nmap scan report for 10.0.2.15
Host is up (0.0011s latency).
All 1000 scanned ports on 10.0.2.15 are closed

Nmap done: 256 IP addresses (3 hosts up) scanned in 4.48 seconds
```

עבור הכתובת 10.9.0.99



über הכתובת 10.0.2.99



מעבר : 8.8.8.8

כיוון שהכתובת קיימת היא תחזר

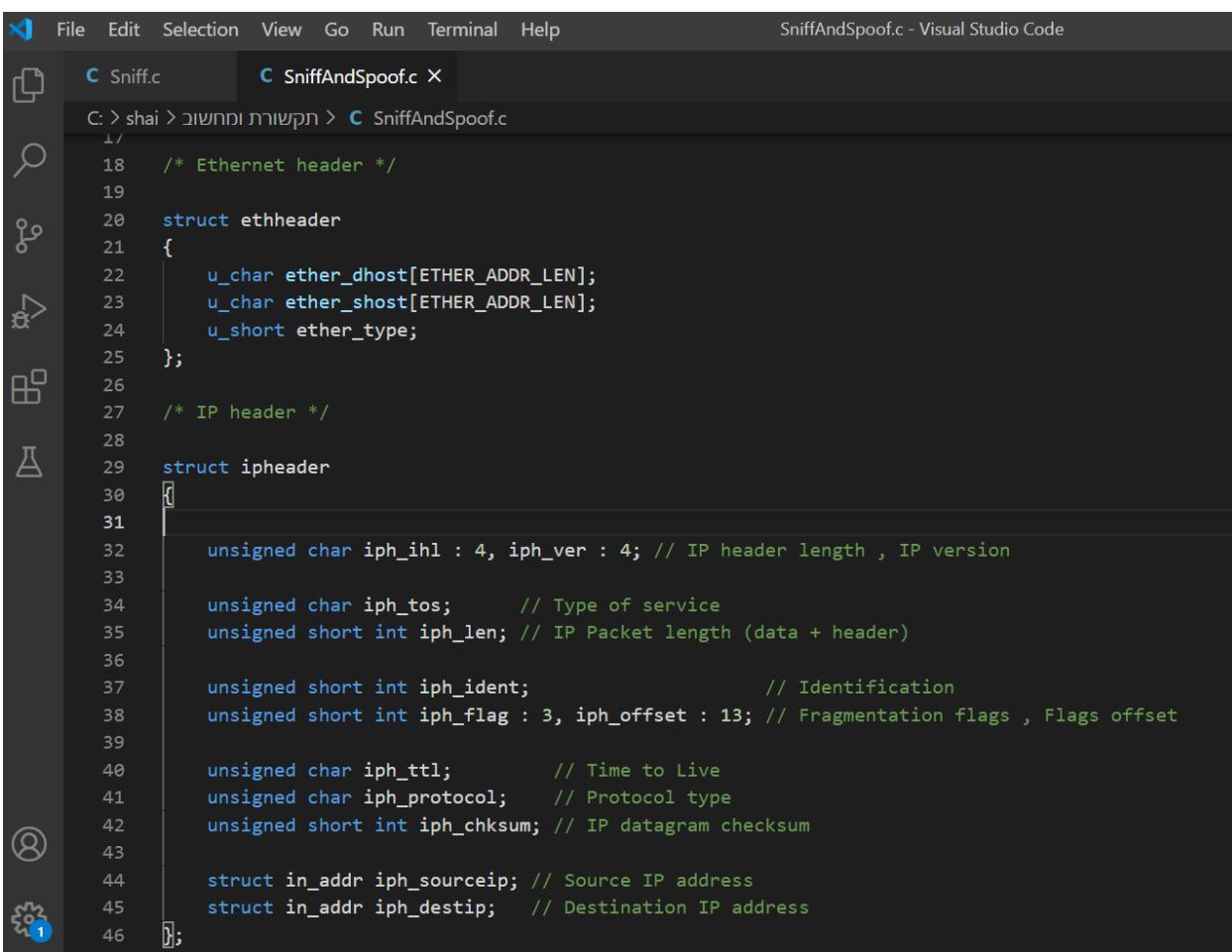
ולכן קיבלנו DUP זה העתקים של הפקות זה אומר שימושו ביצע Spoof אוחת מהפקות היא מזויפת
וכדי לדעת מה המזויף ניתן לראות לפי TTL כמה אנחנו יודעים מה הזמן ומה המיקור

The figure shows two terminal windows and a Wireshark capture window. The left terminal window displays a ping session between the local machine (10.0.2.7) and 192.168.8.8. The right terminal window shows the raw ICMP traffic captured by Wireshark, which includes 15 ICMP echo requests and 15 ICMP echo replies. The Wireshark interface also shows the packet details, bytes, and summary panes.

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

:א: 2.1

דבר ראשון ניצר struct-ים של ethernet-header, ip-header כדי שנוכל לשולט בפונט שלנו ולגשת לשודת שאחכנו מעוניינים בהם. דרך הפונקציה `got_packet` נdfs את SourceIP & DestinationIP ובנוסף גם את סוג ה프וטוקול.



```
File Edit Selection View Go Run Terminal Help SniffAndSpoof.c - Visual Studio Code

C Sniff.c C SniffAndSpoof.c X
C: > shai > תקשורת ומוחשב > C SniffAndSpoof.c
18  /* Ethernet header */
19
20  struct ethheader
21  {
22      u_char ether_dhost[ETHER_ADDR_LEN];
23      u_char ether_shost[ETHER_ADDR_LEN];
24      u_short ether_type;
25  };
26
27  /* IP header */
28
29  struct ipheader
30  {
31
32      unsigned char iph_ihl : 4, iph_ver : 4; // IP header length , IP version
33
34      unsigned char iph_tos;      // Type of service
35      unsigned short int iph_len; // IP Packet length (data + header)
36
37      unsigned short int iph_ident;           // Identification
38      unsigned short int iph_flag : 3, iph_offset : 13; // Fragmentation flags , Flags offset
39
40      unsigned char iph_ttl;      // Time to Live
41      unsigned char iph_protocol; // Protocol type
42      unsigned short int iph_chksum; // IP datagram checksum
43
44      struct in_addr iph_sourceip; // Source IP address
45      struct in_addr iph_destip;   // Destination IP address
46  };
```

```

108
109 void got_packet(u_char *args, const struct pcap_pkthdr *header, const u_char *packet)
110 {
111     printf("\n/*-----*/\n/*-----*/\n/*-----*/\n");
112     printf("Got a packet \n");
113
114     struct ethheader *eth = (struct ethheader *)packet;
115
116     if ( ntohs(eth->ether_type) == 0x0800)
117     {
118         struct ipheader *ip = (struct ipheader *)(packet + sizeof(struct ethheader));
119         struct tcphdr *tcp = (struct tcphdr *)((u_char *)ip + sizeof(struct ipheader));
120         unsigned short pktlen = ntohs(ip->iph_len);
121
122         printf("  Source : %s\n", inet_ntoa(ip->iph_sourceip));
123         printf("  Destination : %s\n", inet_ntoa(ip->iph_destip));
124
125         switch (ip->iph_protocol)
126     {

```

1) יש שלושה דברים הכרחיים ל לעשות:

דבר ראשון, נפתח סשן של pcap שיצר לנו raw-socket דרכו נסניף. בפתיחה הsson הזה נגדיר את הממשק אותו נרצה להסניף. נגדיר כמה DATAה נקבל בכל פאקטה. נגדיר שאנו נכנסים במצב מופקר כדי שנוכל לחרחח את כל התעבורה ולא רק מה שברשת שלנו. נגדיר כל כמה זמן נרצה לקבל את הפאקטות אליו, ובינתיים הם יכנסו לבאר. דבר אחרון נגדיר מבנה נתונים שיקבל שגיאות.

דבר שני, נגדיר את הפילטר שאנו רוצים, למשל "ip proto \icmp", בשפה שמובנת לאדם, וה pcap יתרגם זאת לשפת מכונה ויכניס את זה לתוך כתובות זיכרון של BPF. לאחר מכן נשתמש במצביע ל BPF ובסלרים handlers מה חלק הראשון, בפונקציה .setfilter.

דבר שלישי נפעיל בלולאה אינסופית (כל עוד נרצה לחרחח) את הפונקצייה got_packet שתכלוד לנו את מה שנגדיר לה.

```

C: > shai > תקשורת ומחשוב > C SniffAndSpoof.c
164 int main()
165 {
166
167     pcap_t *handle;
168     char errbuf[PCAP_ERRBUF_SIZE];
169     struct bpf_program fp;
170     //char filter_exp[] = "ip proto \icmp";
171     //char filter_exp[] = "icmp";
172     char filter_exp[] = "tcp port telnet";
173     bpf_u_int32 net;
174
175     // Open live pcap session
176     handle = pcap_open_live("enp0s3", BUFSIZ, 1, 1000, errbuf);
177     // Compile Filter into the Berkeley Packet Filter (BPF)
178     pcap_compile(handle, &fp, filter_exp, 0, net);
179
180     if (pcap_setfilter(handle, &fp) == -1)
181     {
182         pcap_perror(handle, "ERROR");
183         exit(EXIT_FAILURE);
184     }
185
186     // Sniffing..
187
188     pcap_loop(handle, -1, got_packet, NULL);
189
190     pcap_close(handle);
191
192     return 0;
193

```

(2) כמו שתבנו למעלה ה Kapoor מתחל pcap raw-socket, וכך שנו כל להמשיך להיות במצב מופקד נציג root privileges. אם מנסים להפעיל את התוכנית בלי root privileges, נקבל permission denied כשנרצה לקרוא לפונקציה bind ב- raw-socket . הדבר זהה יקרה בתוך החלק הראשון בתשובה לשאלה 1 שתבנו.

(3) בחילוק הראשון בתשובה של שאלה 1 כשאנו רוצים להגדיר מצב מופקד הפרמטר שצרייך להכנסו הוא 1, ואילו הפרמטר 0 זה למצב רגיל. כמו שתכתב לעיל, במצב מופקד נוכל לרחרח את כל התעבורה ולא רק מה

שברשת שלנו, ואילו במצב רגיל אפשר ללחוץ רק פאותות שנשלחות לכתובת ה-Mac שלנו.

```

Activities Terminal Mar 4 14:39 •
eyal@eyal-VirtualBox: ~/Desktop/Codes
/*-----*/
Got a packet
Source : 10.0.2.6
Destination : 10.0.2.15
Protocol: ICMP

/*-----*/
/*-----*/
Got a packet
Source : 10.0.2.15
Destination : 10.0.2.6
Protocol: ICMP

/*-----*/
/*-----*/
Got a packet
Source : 10.0.2.6
Destination : 10.0.2.15
Protocol: ICMP

/*-----*/
/*-----*/
Got a packet
Source : 10.0.2.15
Destination : 10.0.2.6
Protocol: ICMP

/*-----*/
/*-----*/
Got a packet
Source : 10.0.2.6
Destination : 10.0.2.15
Protocol: ICMP

/*-----*/
/*-----*/
Got a packet
Source : 10.0.2.15
Destination : 10.0.2.6
Protocol: ICMP

/*-----*/
/*-----*/
Got a packet
Source : 10.0.2.6
Destination : 10.0.2.15
Protocol: ICMP

/*-----*/
/*-----*/
Got a packet
Source : 10.0.2.15
Destination : 10.0.2.6
Protocol: ICMP

```

OFF

```

Activities Terminal Mar 4 14:39 •
eyal@eyal-VirtualBox: ~/Desktop/Codes
/*-----*/
Got a packet
Source : 10.0.2.6
Destination : 213.8.212.5
Protocol: ICMP

/*-----*/
/*-----*/
Got a packet
Source : 10.0.2.6
Destination : 82.80.239.180
Protocol: ICMP

/*-----*/
/*-----*/
Got a packet
Source : 82.80.239.180
Destination : 10.0.2.6
Protocol: ICMP

/*-----*/
/*-----*/
Got a packet
Source : 10.0.2.6
Destination : 82.80.239.180
Protocol: ICMP

/*-----*/
/*-----*/
Got a packet
Source : 82.80.239.180
Destination : 10.0.2.6
Protocol: ICMP

/*-----*/
/*-----*/
Got a packet
Source : 10.0.2.6
Destination : 10.0.2.15
Protocol: ICMP

```

ON

```

Activities Terminal Mar 4 07:39 •
seed@VM: ~
[03/04/21]seed@VM:~$ ping www.on.co.il
PING www.on.co.il (213.8.212.5) 56(84) bytes of data.
^C
--- www.on.co.il ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

[03/04/21]seed@VM:~$ ping www.one.co.il
PING one.co.il (82.80.239.180) 56(84) bytes of data.

64 bytes from bzq-82-80-239-180.cablep.bezeqint.net (82.80.239.180): i
cmp_seq=1 ttl=245 time=14.1 ms
64 bytes from bzq-82-80-239-180.cablep.bezeqint.net (82.80.239.180): i
cmp_seq=2 ttl=245 time=13.1 ms
^C
--- one.co.il ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 13.147/13.624/14.102/0.477 ms
[03/04/21]seed@VM:~$ ping www.one.co.il
PING one.co.il (82.80.239.180) 56(84) bytes of data.
64 bytes from bzq-82-80-239-180.cablep.bezeqint.net (82.80.239.180): i
cmp_seq=1 ttl=245 time=13.2 ms
64 bytes from bzq-82-80-239-180.cablep.bezeqint.net (82.80.239.180): i
cmp_seq=2 ttl=245 time=14.0 ms
64 bytes from bzq-82-80-239-180.cablep.bezeqint.net (82.80.239.180): i
cmp_seq=3 ttl=245 time=13.2 ms
^C
--- one.co.il ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 13.177/13.478/14.011/0.377 ms
[03/04/21]seed@VM:~$ 

```

```

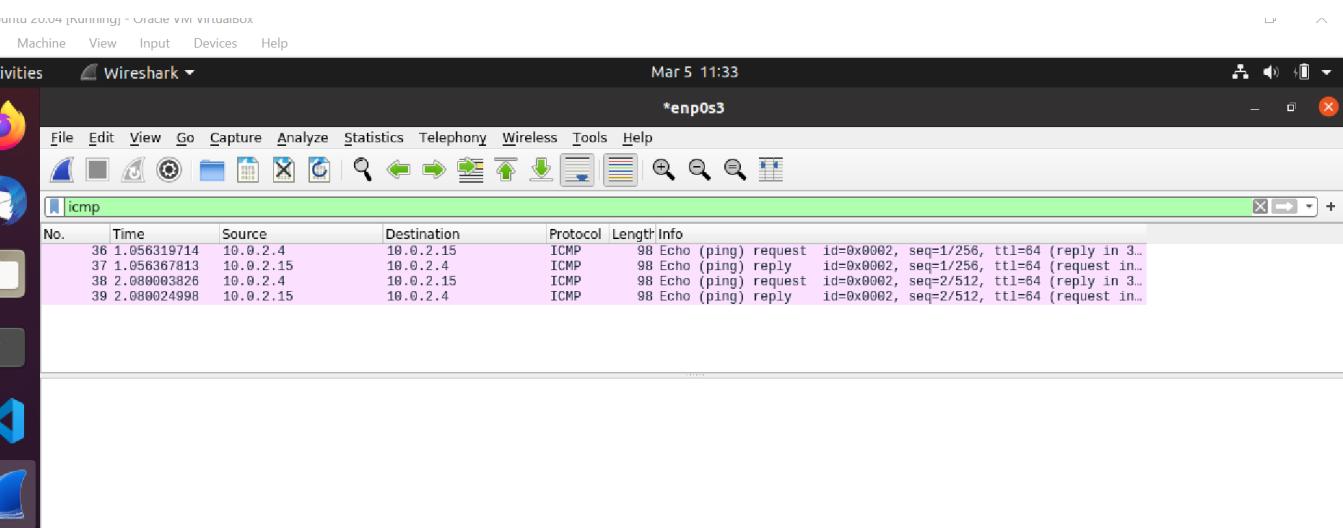
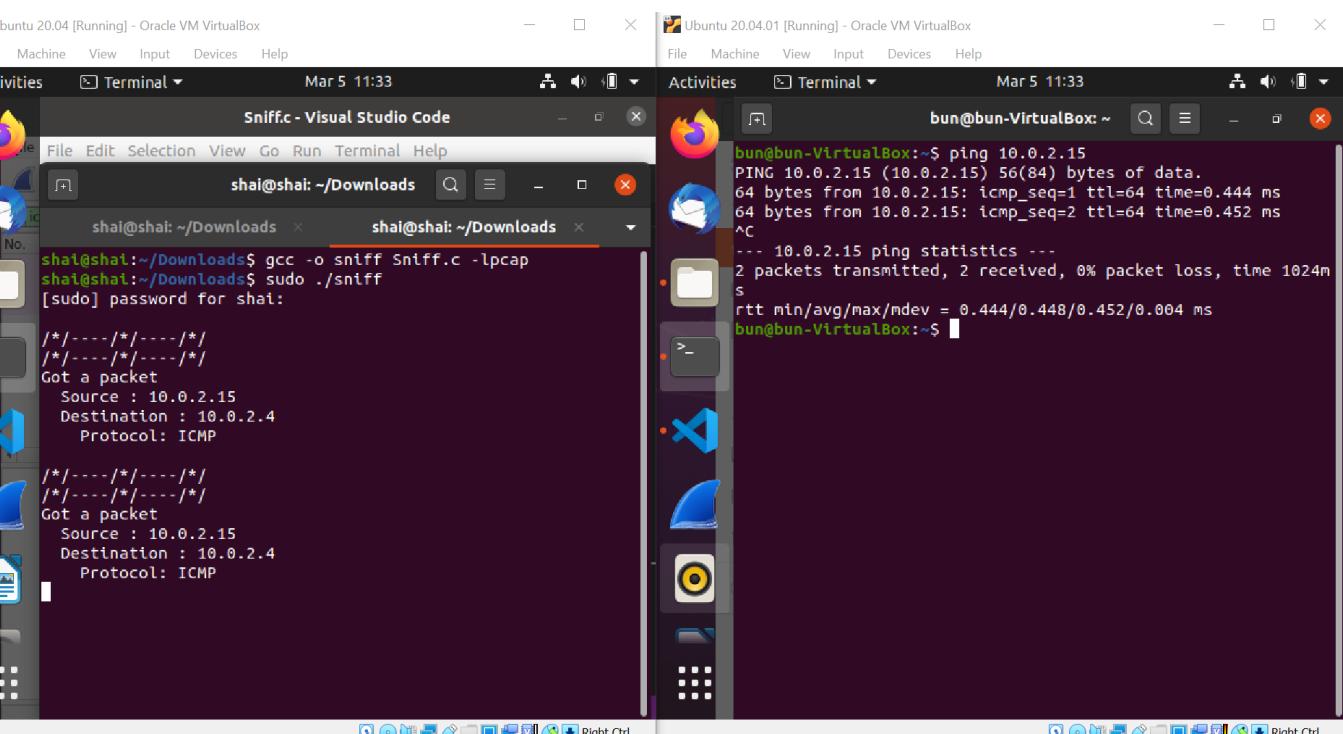
Activities Terminal Mar 4 07:39 •
seed@VM: ~
[03/04/21]seed@VM:~$ ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=0.345 ms
^C
--- 10.0.2.15 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.345/0.345/0.345/0.000 ms
[03/04/21]seed@VM:~$ ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=0.394 ms
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=1.10 ms
^C
--- 10.0.2.15 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1010ms
rtt min/avg/max/mdev = 0.394/0.749/1.104/0.355 ms
[03/04/21]seed@VM:~$ 

```

ב 2.1

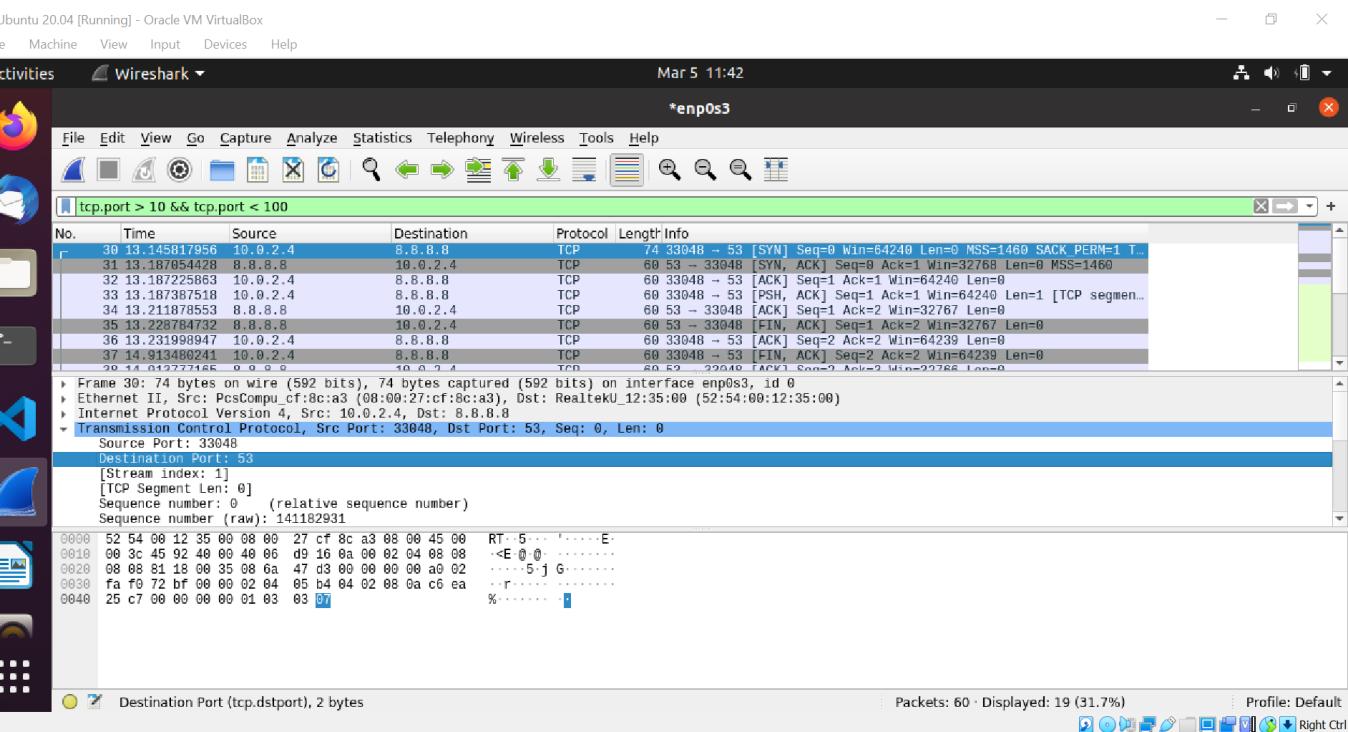
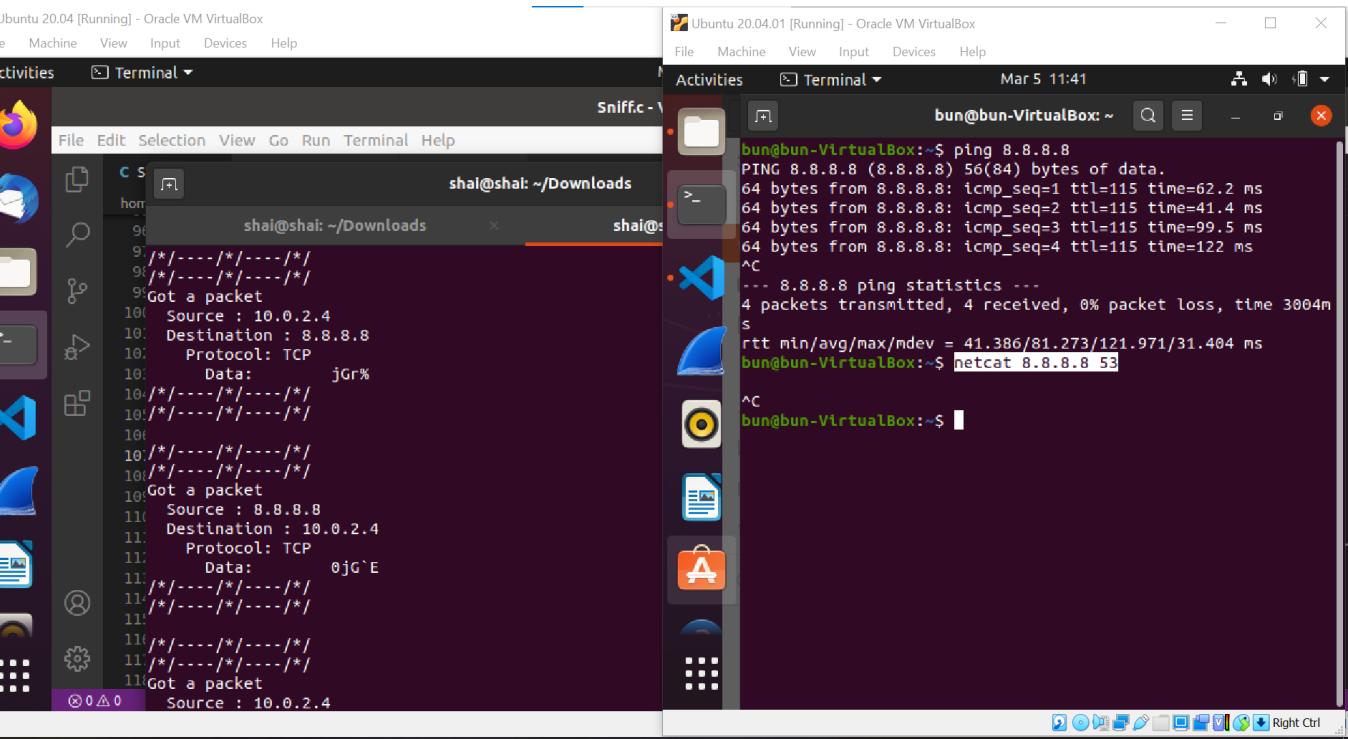
שני host ספציפיים:

```
char filter_exp[] = "icmp and src host 10.0.2.15 and dst host 10.0.2.4";
```



טוווח בין 10-100

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



2.1 ג:

באמצעות pcap ופילטר מתאים לטלנט, הצלחנו לתפוא את הפאקטות הרלוונטיות. משומך כר כשהדפסנו את הפאקטות, יכולנו לראות את הסיסמה שהמשתמש הכניס.

דבר ראשון נגדיר את הפילטר ל- "tcp port telnet"
char filter_exp[] = "tcp port telnet";

דבר נוסף נשנה קצת את הפונקציה `got_packet` כך שהיא תוכל לאתר את הדאטה שאנו רוצים למצוא. השינוי קרה בקטע קוד שמייחס ל- TCP מכיוון שטלנט משתמש ב프וטוקול זה.

```
3     break;
4 case IPPROTO_TCP:
5     printf("    Protocol: TCP\n");
6
7     u_char dos = 4; // data offset
8     int count = 0;
9     if ((pktlen - sizeof(struct ipheader)) > dos)
10    {
11        printf("    Data:      ");
12        u_char *data = (u_char *)tcp + dos;
13        unsigned short password = pktlen - (sizeof(struct ipheader) + dos);
14        for (unsigned short i = 0; i < password; i++)
15        {
16            if(isprint(*data) != 0){
17                printf("%c",*data);
18            }
19            data++;
20        }
21        printf("\n/*-----*/\n/*-----*/\n/*-----*/\n");
22    }
23    break;
24 case IPPROTO_UDP:
25     printf("    Protocol: UDP\n");
```

המשתמש במכונה 10.0.2.6 הכניס את הסיסמה שלו "dees", וזה ישר הודפס אצל המcona 10.0.2.15 שمرחרחת בעזרת טלנט.

Activities Terminal Mar 5 03:16 ●

seed@VM: ~ eyal@eyal-VirtualBox: ~/Desktop/Codes

```
val@eyal-VirtualBox:~$ telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
Last 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

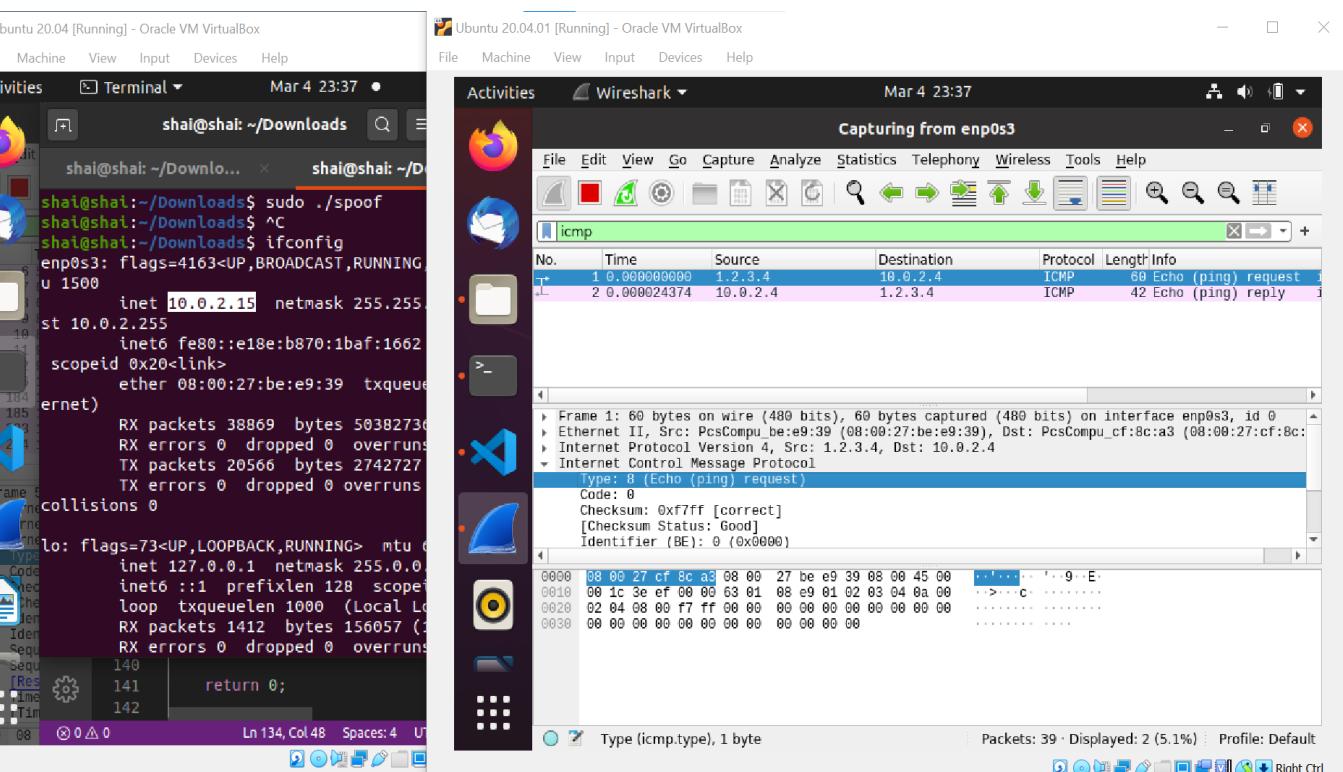
Updates can be installed immediately.
of these updates are security updates.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Thu Mar  4 20:09:14 EST 2021 from 10.0.2.15 on pts/0
[03/04/21]seed@VM:~$ 
```

```
Source : 10.0.2.6 Destination : 10.0.2.15 Protocol: TCP
Data: ➔ GWMLnwpPassword;
/*/----/*/----/*//*/----/*/----*/
Packet:
Source : 10.0.2.15 Destination : 10.0.2.6 Protocol: TCP
Data: WMLH;rnW
/*/----/*/----/*//*/----/*/
Packet:
Source : 10.0.2.15 Destination : 10.0.2.6 Protocol: TCP
Data: ➔ WMLH;nld
/*/----/*/----/*//*/----/*/
Packet:
Source : 10.0.2.6 Destination : 10.0.2.15 Protocol: TCP
Data: HWMM|nK;
/*/----/*/----/*//*/----/*/
Packet:
Source : 10.0.2.15 Destination : 10.0.2.6 Protocol: TCP
Data: ➔ WMHnKe
/*/----/*/----/*//*/----/*/
Packet:
Source : 10.0.2.6 Destination : 10.0.2.15 Protocol: TCP
Data: HWMNzn
/*/----/*/----/*//*/----/*/
Packet:
Source : 10.0.2.15 Destination : 10.0.2.6 Protocol: TCP
Data: ➔ WMNH<cne
/*/----/*/----/*//*/----/*/
Packet:
Source : 10.0.2.6 Destination : 10.0.2.15 Protocol: TCP
Data: HWMOxnHc
/*/----/*/----/*//*/----/*/
Packet:
Source : 10.0.2.15 Destination : 10.0.2.6 Protocol: TCP
Data: ➔ WMOH<rnHs
/*/----/*/----/*//*/----/*/
Packet:
Source : 10.0.2.6 Destination : 10.0.2.15 Protocol: TCP
Data: HWMPvnWr
/*/----/*/----/*//*/----/*/
Packet:
```

:א 2.2

שלחנו פאקטת ICMP מכתובת 10.0.2.15 לכתובת 10.0.2.4, אך השתמשנו
בכתובת מזויפת 1.2.3.4



בכך שהשתמשנו ב- raw-socket יכולנו לזייף כתובת IP שיראה כאילו הפאקטת התקבלה מכתובת שונה משלנו. יכולת זו נובעת מההובדה ש- raw-socket מאפשר לנו למלא בעצמנו את ה- header-ים, ולא יכולנו לשנות את כתובת המקור.

2.2 ב:

נשתמש בקוד מסעיף א', ואפשר לראות בתמונות שצורפו לעיל שהצלחנו.

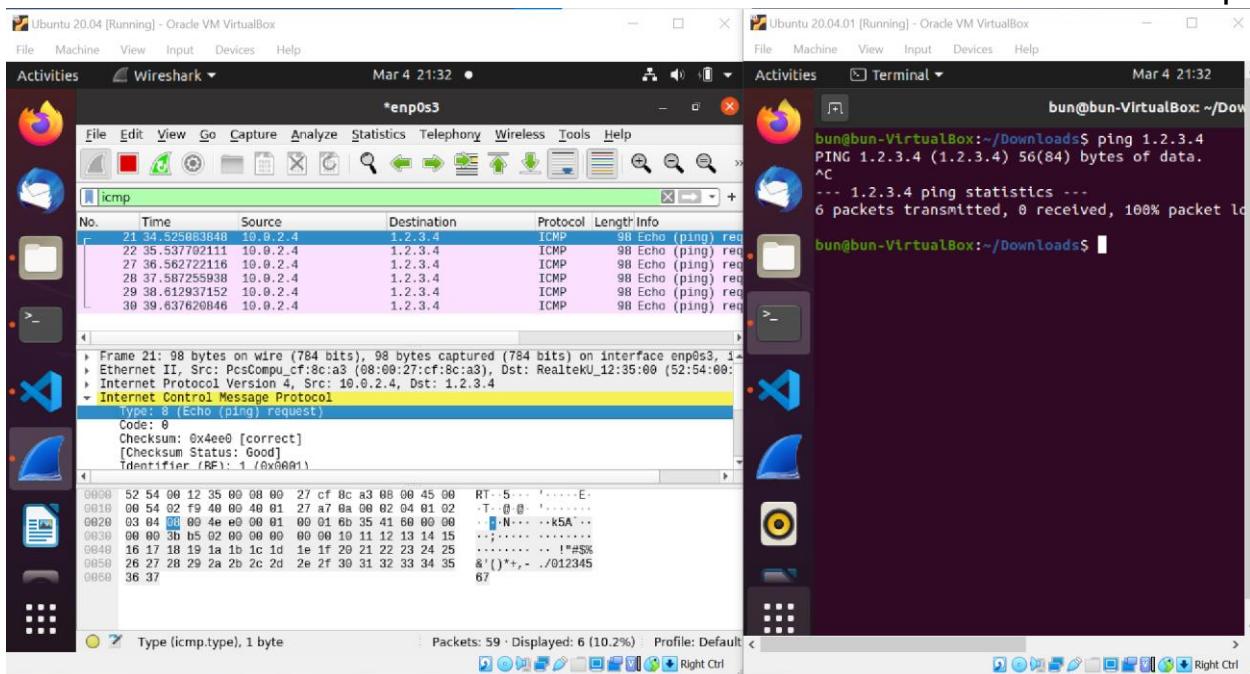
4) כן, מכיוון שאין לנו דרך לאמת שהשدةזה מולא באופן תקין.

5) לא, מכיוון שה- OS תמלא את זה בשבילנו.

6) כדי שנוכל להמשיך להיות במצב מופקר נוצרת root privileges. אם מנסים להפעיל את התוכנית בל' root privileges, נקבל permission denied כאשר כשרצה לקרוא לפקציה bind ב- raw-socket. הדבר הזה יקרה ב- spoof_RawSocket וגם ב- pcap_open_live

:2.3

כאשר נשלח ping 1.2.3.4 (כתובת לא קיימת) ללא תוכנית ששולחת spoof, נקבל את הדבר הבא:



אך כאשר נריץ את התוכנית ממכונה אחרת נוכל לרשוח ו גם לשלח תשובה מזויפת כביכול מכתובת קיימת, כמו שמצוף:

Ubuntu 20.04 [Running] - Oracle VM VirtualBox

Machine View Input Devices Help

Activities Terminal Mar 4 22:41

shai@shai: ~/Downloads bun@bun-VirtualBox: ~/Downloads

```
shai@shai:~/Downloads$ sudo ./sniffspoof
/*-----*/
/*-----*/
Got a packet
  Source : 10.0.2.4
  Destination : 1.2.3.4
  Protocol: ICMP
10.0.2.41.2.3.4
/*-----*/
/*-----*/
/*-----*/
Got a packet
  Source : 1.2.3.4
  Destination : 10.0.2.4
  Protocol: ICMP
1.2.3.410.0.2.4
/*-----*/
/*-----*/
/*-----*/
Got a packet
  Source : 10.0.2.4
  Destination : 1.2.3.4
  Protocol: ICMP
Ident
Seqn
/*-----*/
/*-----*/
/*-----*/
Got a packet
  Source : 10.0.2.4
```

bun@bun-VirtualBox:~/Downloads\$ 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=1123 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=99 time=1125 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=99 time=3171 ms (DUP!)
64 bytes from 1.2.3.4: icmp_seq=1 ttl=99 time=3171 ms (DUP!)
64 bytes from 1.2.3.4: icmp_seq=3 ttl=99 time=1147 ms
^C
--- 1.2.3.4 ping statistics ---
5 packets transmitted, 3 received, +2 duplicates, 40% packet
rtt min/avg/max/mdev = 1123.065/1947.394/3170.943/998.945 ms
bun@bun-VirtualBox:~/Downloads\$

Machine View Input Devices Help

Wireshark Mar 4 22:43

Capturing from enp0s3

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

icmp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.4	1.2.3.4	ICMP	98	Echo (ping) request id=0x000b, seq=1/256, ttl=64 (no respons...
2	0.098333810	1.2.3.4	10.0.2.4	ICMP	98	Echo (ping) request id=0x000b, seq=1/256, ttl=99 (reply in 3)
3	0.098858622	10.0.2.4	1.2.3.4	ICMP	98	Echo (ping) reply id=0x000b, seq=1/256, ttl=64 (request in...
4	1.021969397	10.0.2.4	1.2.3.4	ICMP	98	Echo (ping) request id=0x000b, seq=2/512, ttl=64 (no respons...
5	1.122317627	10.0.2.4	1.2.3.4	ICMP	98	Echo (ping) request id=0x000b, seq=1/256, ttl=99 (reply in 6)
6	1.122401407	1.2.3.4	10.0.2.4	ICMP	98	Echo (ping) reply id=0x000b, seq=1/256, ttl=99 (request in...
7	1.122440743	1.2.3.4	10.0.2.4	ICMP	98	Echo (ping) request id=0x000b, seq=2/512, ttl=99 (reply in 8)
8	1.123314751	10.0.2.4	1.2.3.4	ICMP	98	Echo (ping) reply id=0x000b, seq=2/512, ttl=64 (request in...
9	2.022904244	10.0.2.4	1.2.3.4	ICMP	98	Echo (ping) request id=0x000b, seq=2/768, ttl=64 (no respons...

Frame 6: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0
 Ethernet II, Src: PcsCompu_be:e9:39 (08:00:27:be:e9:39), Dst: PcsCompu_cf:8c:a3 (08:00:27:cf:8c:a3)
 Internet Protocol Version 4, Src: 1.2.3.4, Dst: 10.0.2.4
 Internet Control Message Protocol
 Type: 8 (Echo (ping) reply)
 Code: 0
 Checksum: 0xf4b1 [correct]
 [Checksum Status: Good]
 Identifier (BE): 11 (0x000b)
 Identifier (LE): 2816 (0xb000)

0000 08 00 27 cf 8c a3 08 00 27 be e9 39 08 00 45 00 .T...c GA...
0010 00 54 00 5f 00 00 63 81 47 01 02 03 04 0a 00 ...
0020 02 04 00 00 f4 b1 00 0b 00 01 18 44 41 60 00 00 ...DA...
0030 00 00 eb ca 07 00 00 00 00 10 11 12 13 14 15 ...
0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 ...
0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 &'()**,- ./012345
0060 36 37 67

Packets: 625 · Displayed: 497 (79.5%) Profile: Default