תקשורת

מטלה 1

WIRESHARK INTRO

1) List 3 different protocols that appear in the protocol column in the unfiltered packet-listing window in step 7 above.

2)How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received? (By default, the value of the Time column in the packet-listing window is the amount of time, in seconds, since Wireshark tracing began. To display the Time field in time-of-day format, select the Wireshark View pull down menu, then select Time Display Format, then select Time-of-day.)

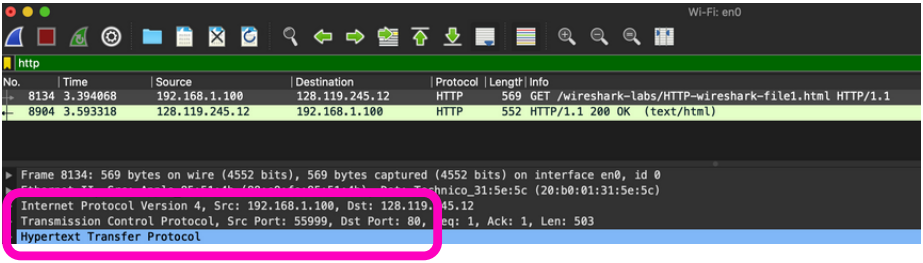
3). What is the Internet address of the gaia.cs.umass.edu (also known as www-net.cs.umass.edu)? What is the Internet address of your computer?)

4). Print the two HTTP messages (GET and OK) referred to in question 2 above. To do so, select Print from the Wireshark File command menu, and select the "Selected Packet Only" and "Print as displayed" radial buttons, and then click OK.

Question 1

List 3 different protocols that appear in the protocol column in the unfiltered packet-listing window in step 7 above.

Answer 1



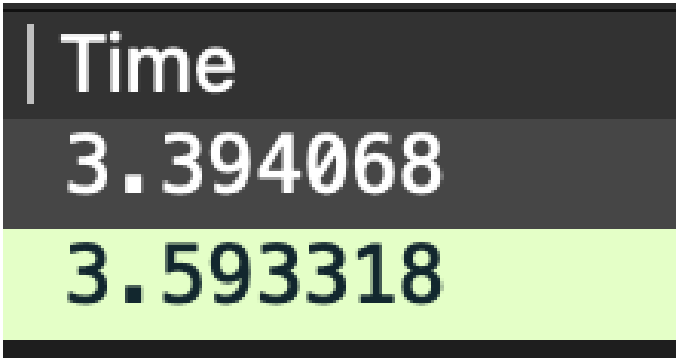
Here we can clearly see 3 different protocoles

- Internet Protocol Version 4. "**IPV4**"
- Transmission Control Protocol "**TCP**"
- Hypertext Transfer Protocol "**HTTP**"

Question 2

How long did it take from when the HTTP GET message was sent until the HTTP
OK reply was received? (By default, the value of the Time column in the packet-
listing window is the amount of time, in seconds, since Wireshark tracing began.
To display the Time field in time-of-day format, select the Wireshark View pull
down menu, then select Time Display Format, then select Time-of-day.)

Answer 2

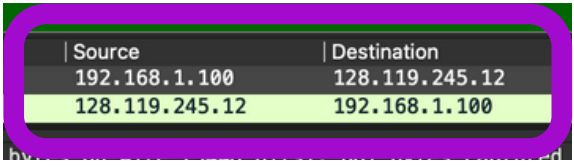


Time To execute 3.394068

Question 3

3). What is the Internet address of the gaia.cs.umass.edu (also known as www-net.cs.umass.edu)?
What is the Internet address of your computer?

Answer 3



Computer Internet address :

- Source Address "**192.168.1.100**"

gaia.cs.umass.edu Internet address :

- Destination Address "**128.119.245.12**"

Question 4

**Print the two HTTP messages (GET and OK) referred to in question 2 above. To
do so, select Print from the Wireshark File command menu, and select the "Selected Packet Only" and "Print as displayed" radial buttons, and then click
OK.**

Answer 4

/var/folders/2h/_cqkmpms0k91t9rl27l22nph0000gn/T/wireshark_Wi-Fi2QIJU0.pcapng 14279 total packets, 2 shown

```
No.     Time           Source                Destination           Protocol Length Info
   8134 3.394068       192.168.1.100         128.119.245.12        HTTP     569    GET /
wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1
Frame 8134: 569 bytes on wire (4552 bits), 569 bytes captured (4552 bits) on interface en0, id
0
Ethernet II, Src: Apple_85:51:4b (88:e9:fe:85:51:4b), Dst: Technico_31:5e:5c (20:b0:01:31:5e:
5c)
Internet Protocol Version 4, Src: 192.168.1.100, Dst: 128.119.245.12
Transmission Control Protocol, Src Port: 55999, Dst Port: 80, Seq: 1, Ack: 1, Len: 503
Hypertext Transfer Protocol
```

WIRESHARK
HTTP

1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?
2. What languages (if any) does your browser indicate that it can accept to the server?
3. What is the IP address of your computer? Of the gaia.cs.umass.edu server?
4. What is the status code returned from the server to your browser?
5. When was the HTML file that you are retrieving last modified at the server?
6. How many bytes of content are being returned to your browser?
7. By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one.
8. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET?
9. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?
10. Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE:" line in the HTTP GET? If so, what information follows the "IF-MODIFIED-SINCE:" header?
11. What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.
12. How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill or Rights?
13. Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?
14. What is the status code and phrase in the response?
15. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?
16. How many HTTP GET request messages did your browser send? To which Internet addresses were these GET requests sent?
17. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.
18. What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser?
19. When your browser's sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?

Question 1

Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?

Answer 1

```
552 HTTP/1.1 200 OK  (text/html)
```

My browser running HTTP version 1.1

Question 2

What languages (if any) does your browser indicate that it can accept to the server?

Answer 2

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
Accept-Encoding: gzip, deflate\r\n
Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7\r\n
```

French & English (US)

Question 3

What is the IP address of your computer? Of the gaia.cs.umass.edu server?

Answer 3

Source	Destination
192.168.1.100	128.119.245.12

Computer Internet address :

- Source Address "**192.168.1.100**"

gaia.cs.umass.edu Internet address :

- Destination Address "**128.119.245.12**"

Question 4

What is the status code returned from the server to your browser?

Answer 4

```
552 HTTP/1.1 200 OK  (text/html)
```

200 OK

Question 5

When was the HTML file that you are retrieving last modified at the server?

Answer 5

```
Last-Modified: Sat, 14 Nov 2020 06:59:02 GMT\r\
```

Saturday 14 Novembre 2020 at 06:59:02

Question 6

How many bytes of content are being returned to your browser?

Answer 6

Content-Length 128 Bytes

Question 7

By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one. No. The raw data appears to match up exactly with what is shown in the packet-listing window.

Answer 7

No, i don't see any header

Question 8

Inspect the contents of the first HTTP GET request from your browser to the
server. Do you see an “IF-MODIFIED-SINCE” line in the HTTP GET?

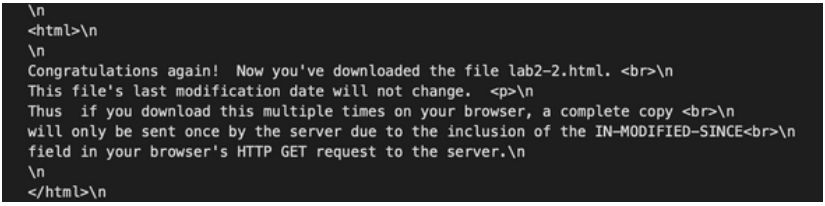
Answer 8

No, i don't see any "IF-MODIFIED-SINCE

Question 9

Inspect the contents of the server response. Did the server explicitly return the
contents of the file? How can you tell?

Answer 9

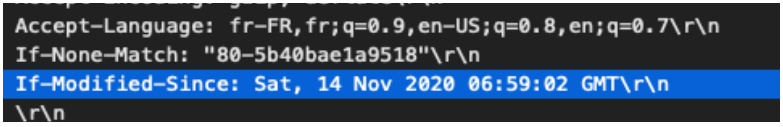


Yes i can clearly see the contain html :
"Congratulation. Again! ..."

Question 10

Now inspect the contents of the second HTTP GET request from your browser to
the server. Do you see an “IF-MODIFIED-SINCE:” line in the HTTP GET? If
so, what information follows the “IF-MODIFIED-SINCE:” header?

Answer 10

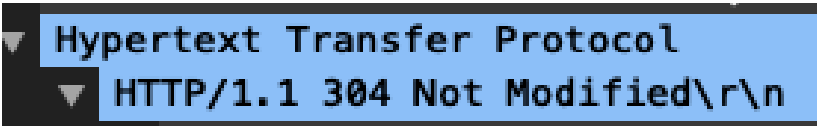


Sat, 14 Nov 2020 06:59:02

Question 11

What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.

Answer 11



The second HTTP GET returned 304 Not Modified because the request don't have to be renew the cash well know this adress so it can be signified that nothing was modified since my last request .

Question 12

How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill or Rights?

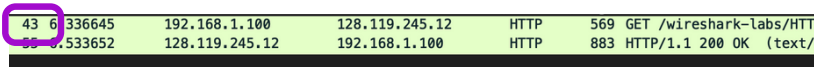
Answer 12

My browser send 1 HTTP GET request message

Question 13

Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?

Answer 13

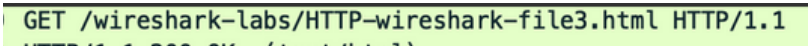


No. 43

Question 14

What is the status code and phrase in the response?

Answer 14

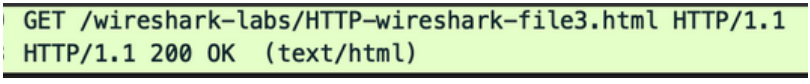


GET /wireshark-labs/HTPP-wireshark-file3.html HTTP/1.1

Question 15

How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

Answer 15

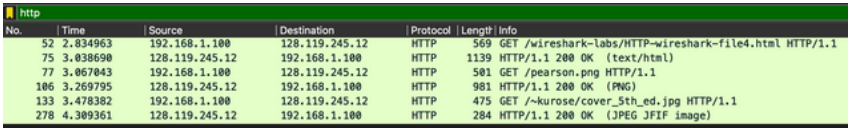


2 TCP segments (1GET)(1 OK).

Question 16

How many HTTP GET request messages did your browser send? To which Internet addresses were these GET requests sent?

Answer 16



http

No.	Time	Source	Destination	Protocol	Length	Info
52	2.834963	192.168.1.100	128.119.245.12	HTTP	569	GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1
75	3.038690	128.119.245.12	192.168.1.100	HTTP	1139	HTTP/1.1 200 OK (text/html)
77	3.067043	192.168.1.100	128.119.245.12	HTTP	501	GET /pearson.png HTTP/1.1
106	3.269795	128.119.245.12	192.168.1.100	HTTP	981	HTTP/1.1 200 OK (PNG)
133	3.478382	192.168.1.100	128.119.245.12	HTTP	475	GET /~kurose/cover_5th_ed.jpg HTTP/1.1
278	4.309361	128.119.245.12	192.168.1.100	HTTP	284	HTTP/1.1 200 OK (JPEG JFIF image)

1 destinations :

- *128.119.245.12*

Question 17

Can you tell whether your browser downloaded the two images serially, orwhether they were downloaded from the two web sites in parallel? Explain.

Answer 17

Picture 1

[3 Reassembled TCP Segments (3611 bytes): #103(1348), #104(1348), #106(915)]

Picture 2

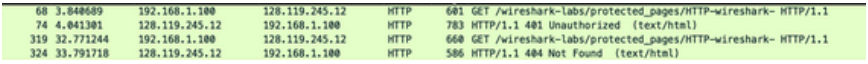
▶ [77 Reassembled TCP Segments (101318 bytes): #137(1348), #138(1348

both images have been downloaded in series

Question 18

What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser?

Answer 18



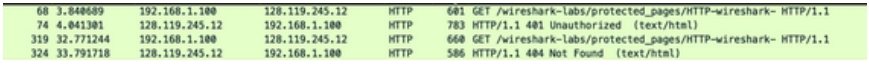
68	3.840689	192.168.1.100	128.119.245.12	HTTP	601 GET /wireshark-labs/protected_pages/HTTP-wireshark- HTTP/1.1
74	4.041301	128.119.245.12	192.168.1.100	HTTP	783 HTTP/1.1 401 Unauthorized (text/html)
319	32.771244	192.168.1.100	128.119.245.12	HTTP	660 GET /wireshark-labs/protected_pages/HTTP-wireshark- HTTP/1.1
324	33.791718	128.119.245.12	192.168.1.100	HTTP	586 HTTP/1.1 404 Not Found (text/html)

The server's response in the initial HTTP GET is the N°68 GET/wireshark-labs/protected_pages/HTTP-wireshark-HTTP/1.1

Question 19

When your browser's sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?

Answer 19



68	3.840689	192.168.1.100	128.119.245.12	HTTP	601 GET /wireshark-labs/protected_pages/HTTP-wireshark- HTTP/1.1
74	4.041301	128.119.245.12	192.168.1.100	HTTP	783 HTTP/1.1 401 Unauthorized (text/html)
319	32.771244	192.168.1.100	128.119.245.12	HTTP	660 GET /wireshark-labs/protected_pages/HTTP-wireshark- HTTP/1.1
324	33.791718	128.119.245.12	192.168.1.100	HTTP	586 HTTP/1.1 404 Not Found (text/html)

Unauthorized

תקשורת

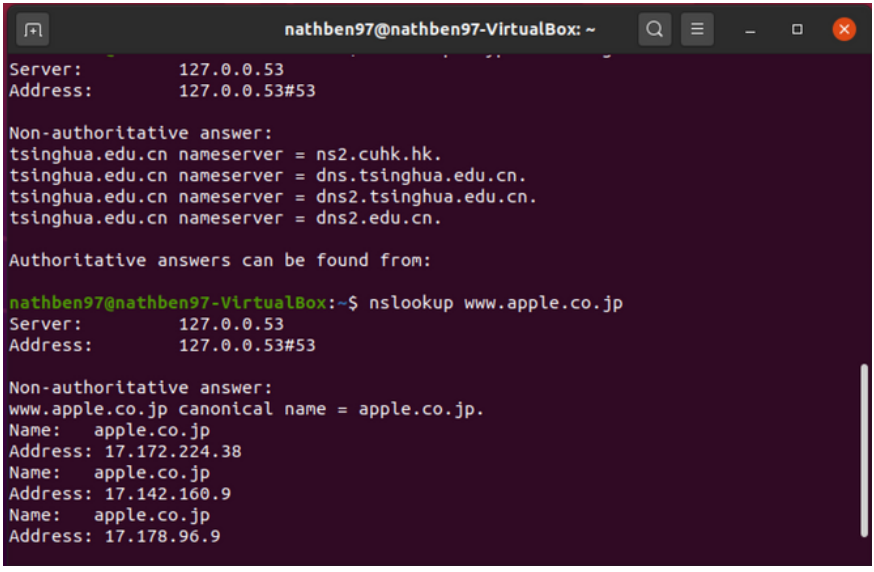
מטלה 1

Nathanael Benichou
342769130

WIRESHARK

DNS

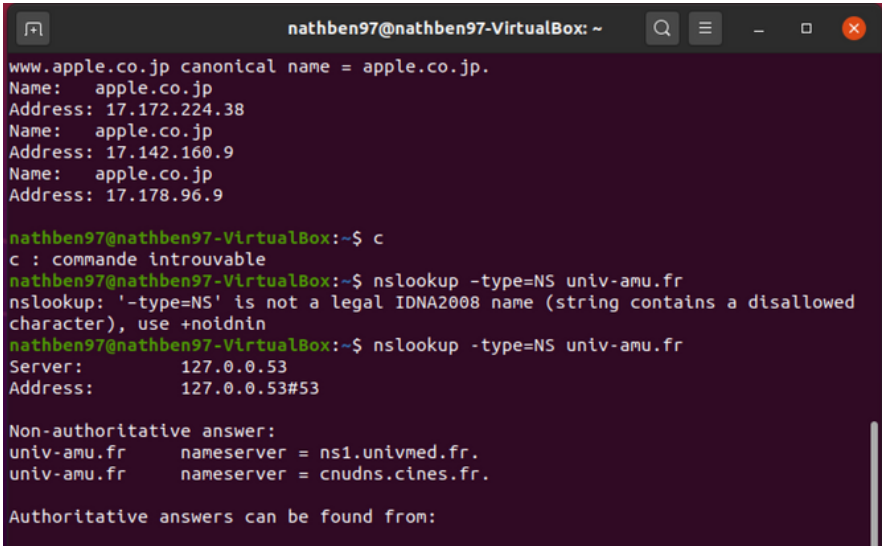
1. Run nslookup to obtain the IP address of a Web server in Asia. What is the IP address of that server?



Adresses :

- 17.172.224.38
- 17.142.160.9
- 17.178.96.9

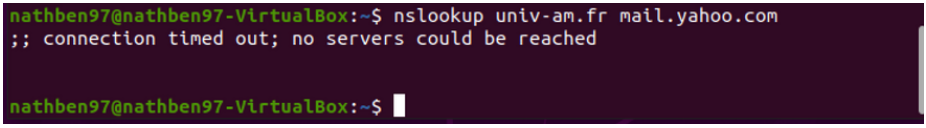
2. Run nslookup to determine the authoritative DNS servers for a university in Europe.



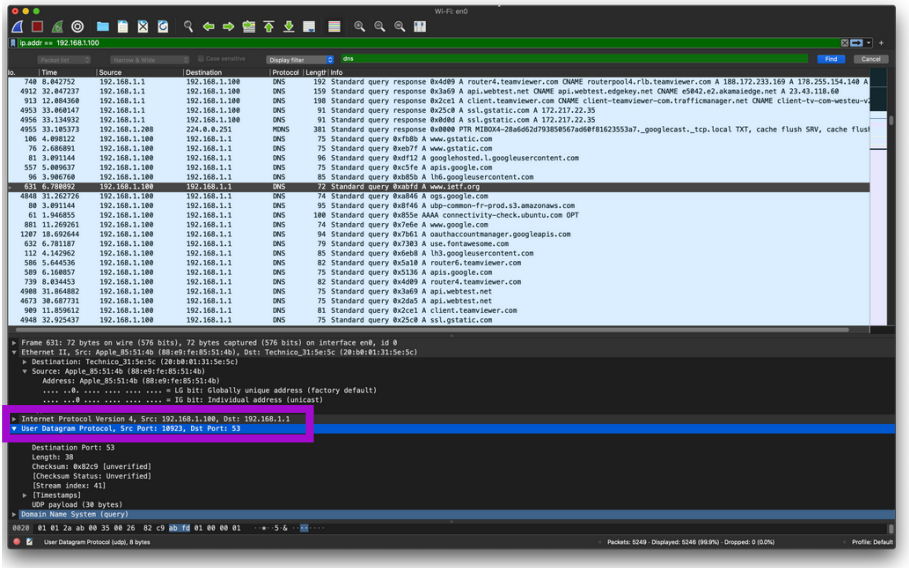
Servers :

- ns1.univmed.fr
- cnudns.cines.fr

3.Run nslookup so that one of the DNS servers obtained in Question 2 is queried for the mail servers for Yahoo! mail. What is its IP address?



Connection timed out no servers could be reached



4.Locate the DNS query and response messages. Are then sent over UDP or TCP?

UDP

5.What is the destination port for the DNS query message? What is the source port of DNS response message?

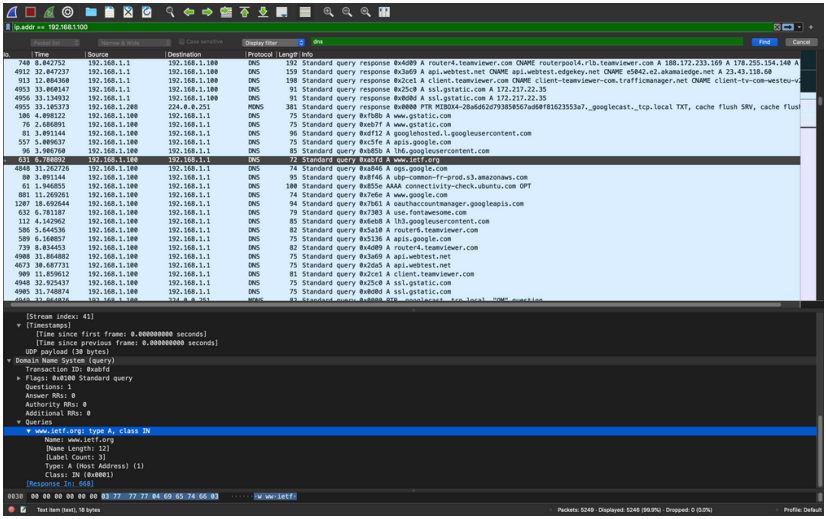
- Source port :10923
- Destination port :53

6 .To what IP address is the DNS query message sent? Use ipconfig to determine the
IP address of your local DNS server. Are these two IP addresses the same?

- Source :192.168.1.100
- Destination :192.168.1.1

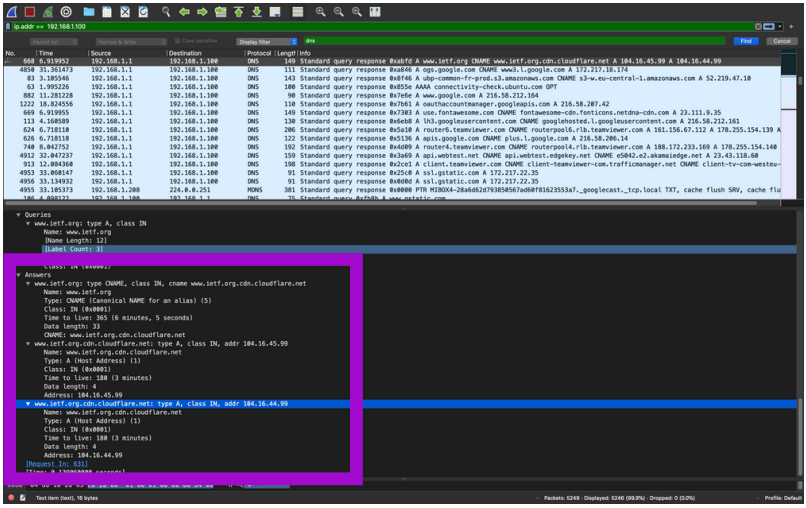
The MacOs (ipconfig /all) is "ifconfig" the inet show us that the. destination feet with our DNS answer 192.168.100

7.Examine the DNS query message. What “Type” of DNS query is it? Does the query message contain any “answers”?



Standard Query ;
Type of DNS Query : A;
no answer.

8. Examine the DNS response message. How many “answers” are provided? What do each of these answers contain?



Answers :

3 answers contains :

(name, type, class, time to live, data ,length ,adress)

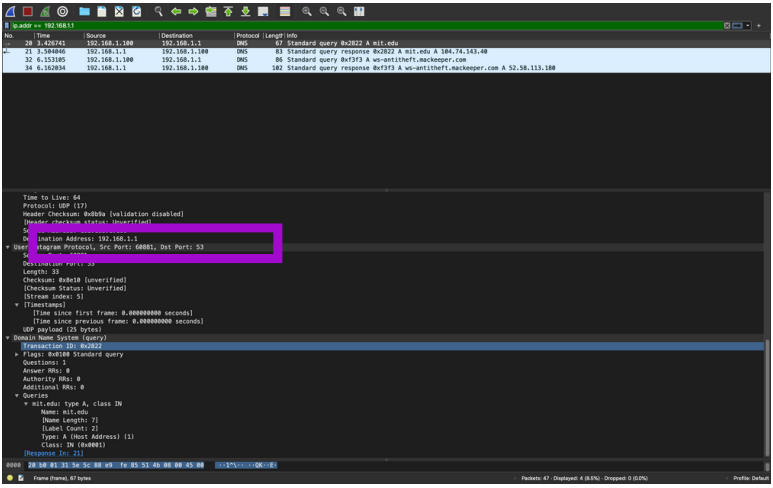
9. Consider the subsequent TCP SYN packet sent by your host. Does the destination IP address of the SYN packet correspond to any of the IP addresses provided in the DNS response message?

Answer : adresse of provided by the DNS server of ietf.org is 104.16.44.99 as seen in the last screenshot

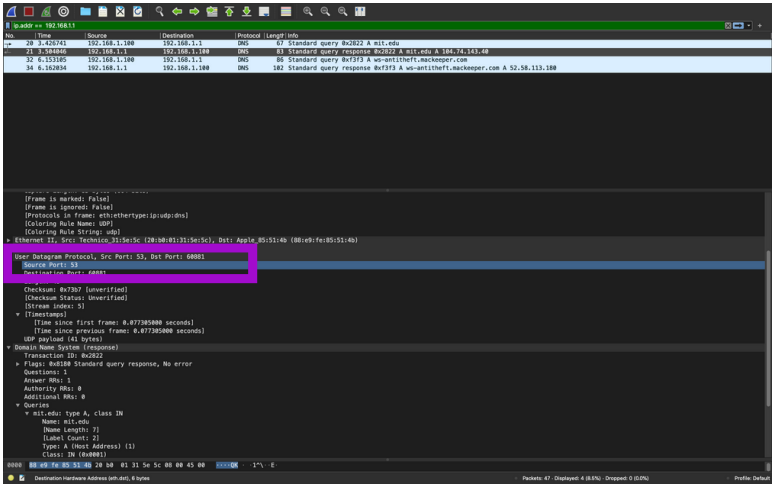
10. This web page contains images. Before retrieving each image, does your host issue new DNS queries?

All the images coming from ietf.org so no need to use a new DNS queries

11. What is the destination port for the DNS query message? What is the source port of DNS response message?

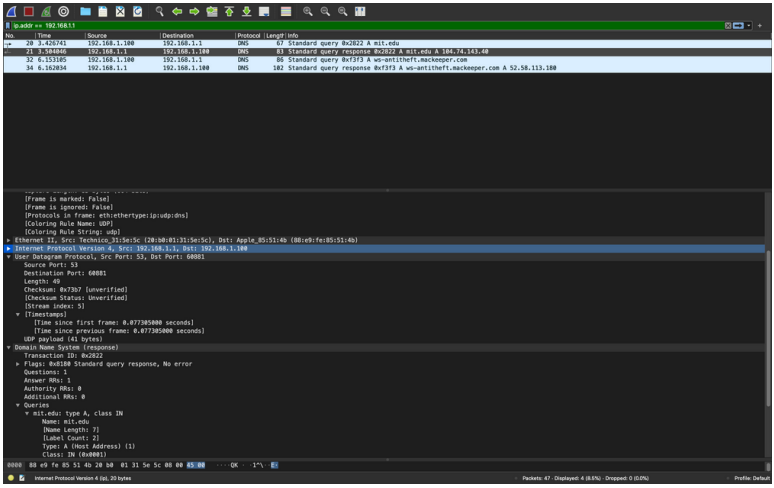


Request port 53

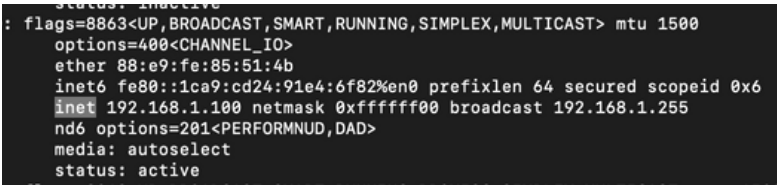


Answer port 53

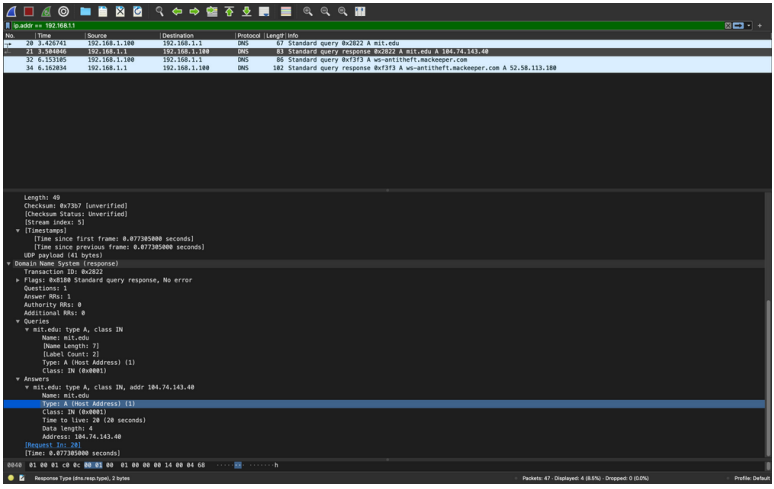
12. To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?



Destination is 192.168.1.100 its my Dns default ip as seen in the ifconfig screenshot



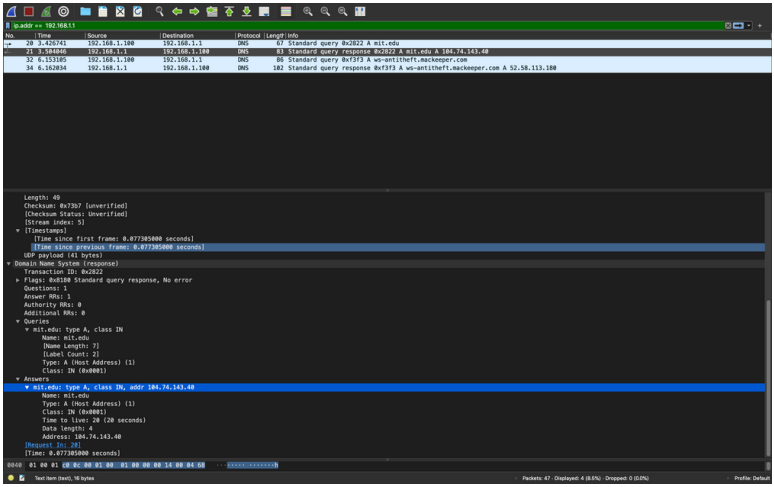
13. Examine the DNS query message. What “Type” of DNS query is it? Does the query message contain any “answers”?



No answer /DNS type :A

14. Examine the DNS response message. How many “answers” are provided? What do each of these answers contain?

15.ScreenShot

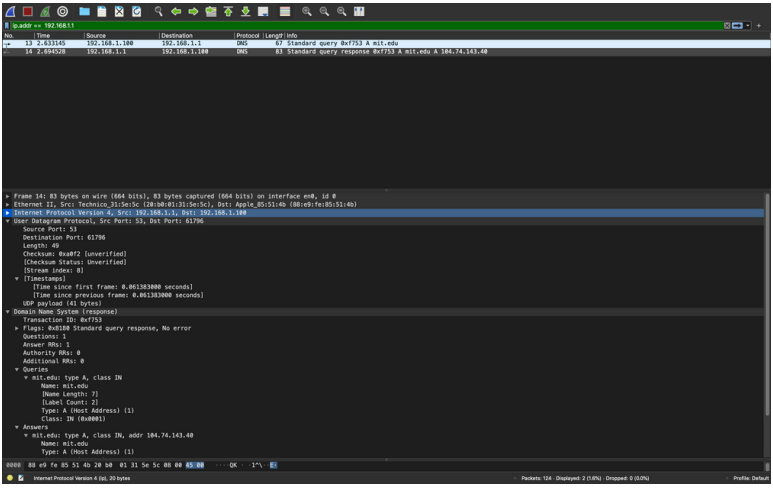


Answers :

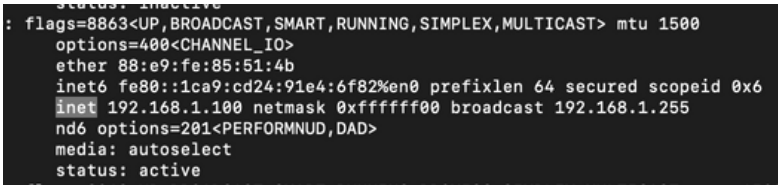
1 answers contains :

(name, type, class, time to live, data ,length ,adress)

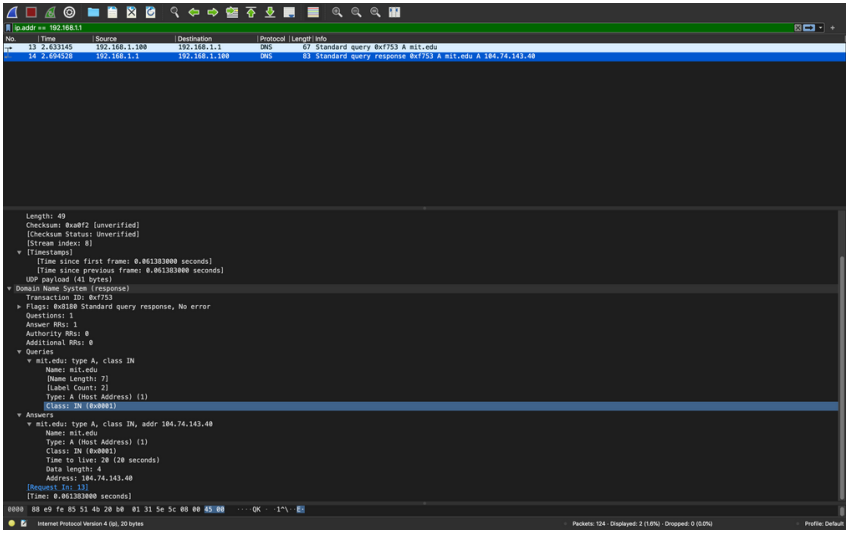
16.To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?



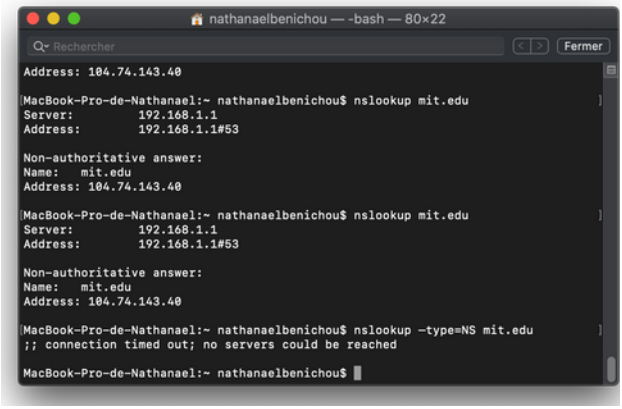
Destination is 192.168.1.100 its my Dns default ip as seen in the ifconfig screenshot



17. Examine the DNS query message. What “Type” of DNS query is it? Does the query message contain any “answers”?

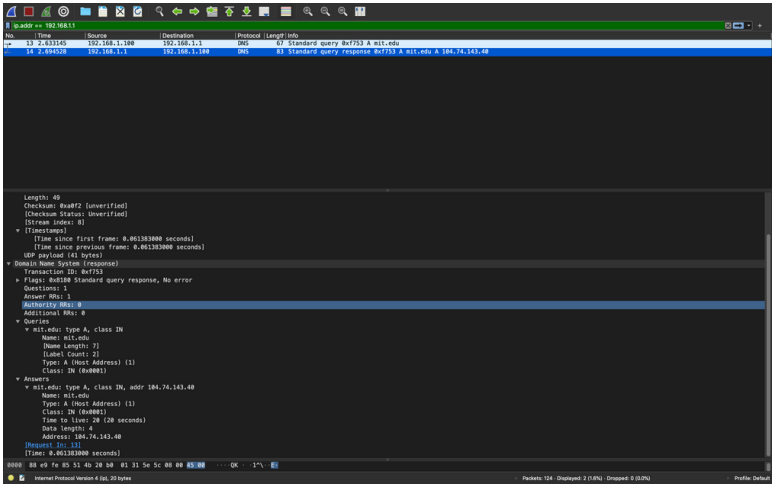


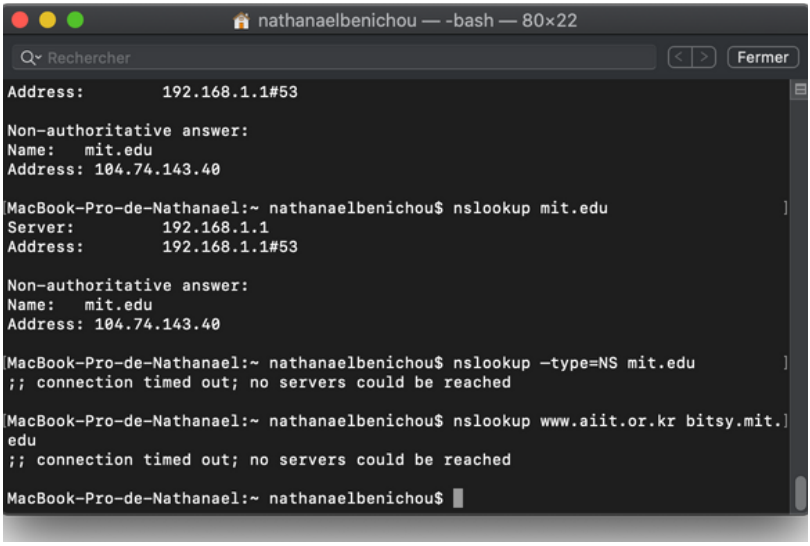
Answer: Type A query, and no answer



18. Examine the DNS response message. What MIT nameservers does the response message provide? Does this response message also provide the IP addresses of the MIT namesers?

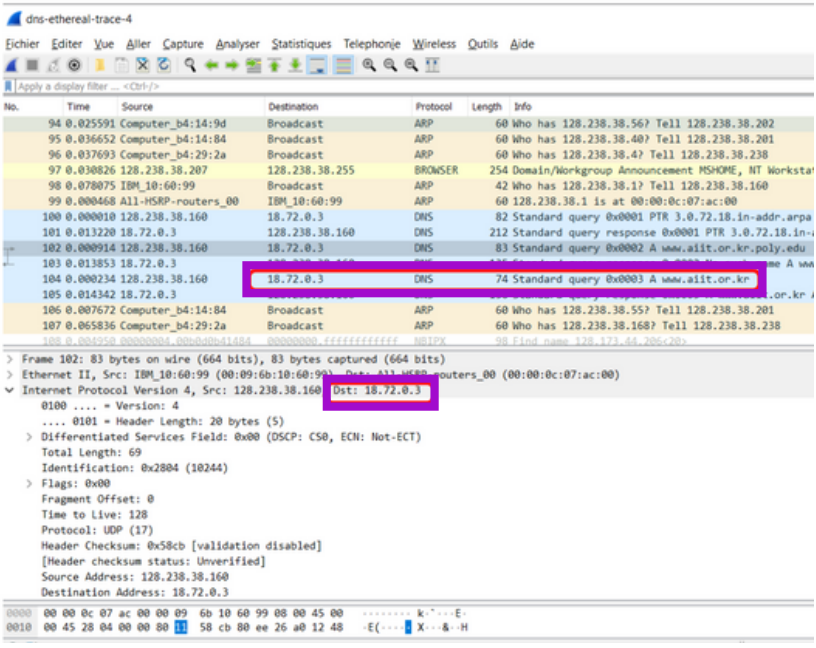
19. Provide a screenshot.





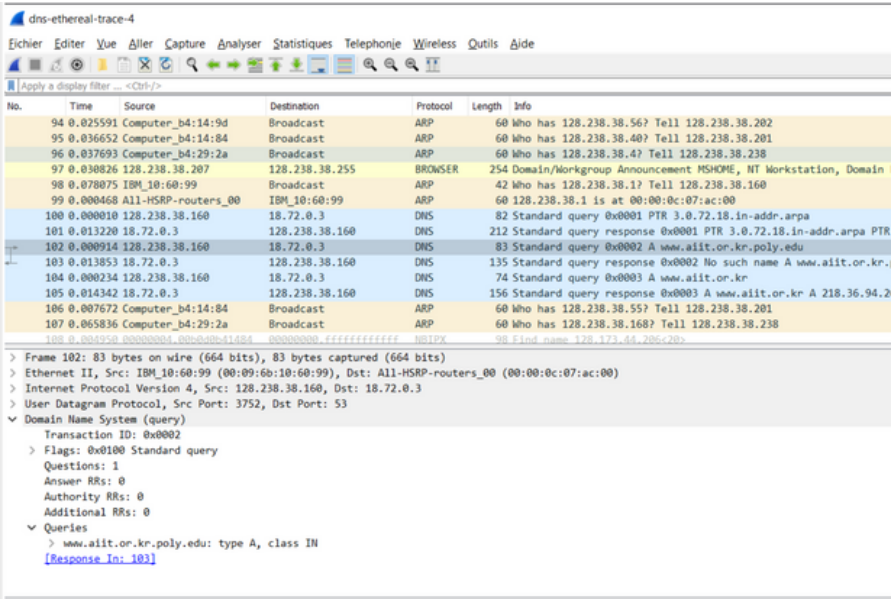
Doesn't work so i used the given pdf in this case

20. To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server? If not, what does the IP address correspond to?



Answer: DNS send to 18.72.0.3
the IP address of the wait response sender.

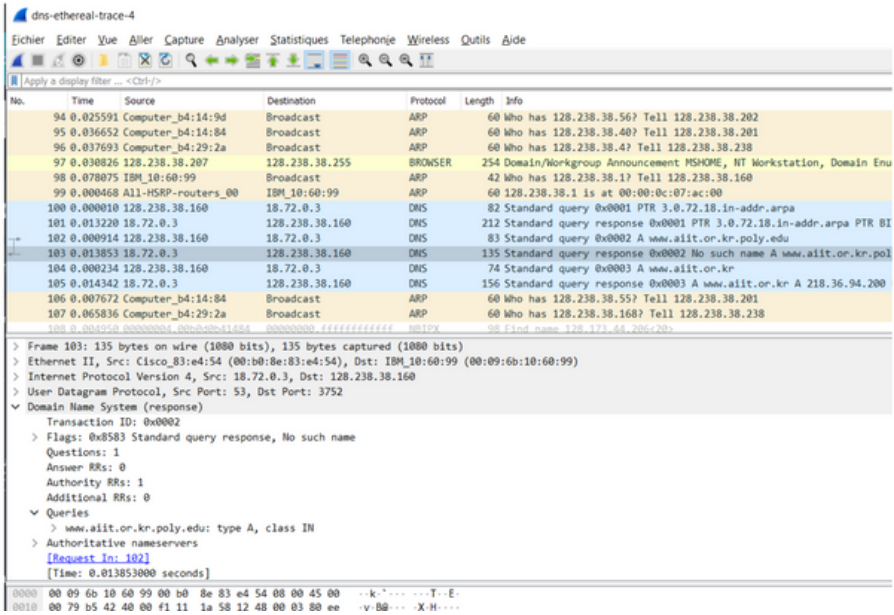
21. Examine the DNS query message. What “Type” of DNS query is it? Does the query message contain any “answers”?



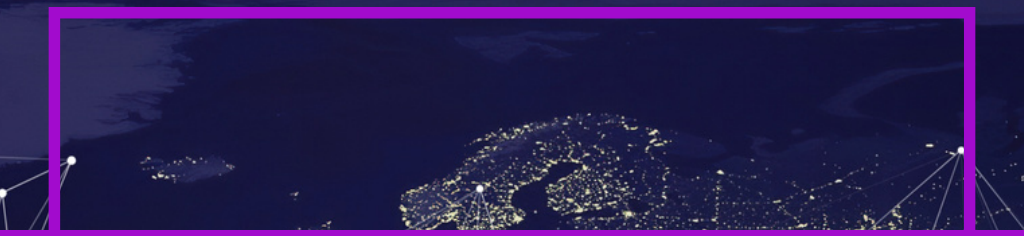
Answer: Type of DNS : A
Standard query
no answer

22. Examine the DNS response message. How many "answers" are provided? What does each of these answers contain?

23.Provide a Screenshot:

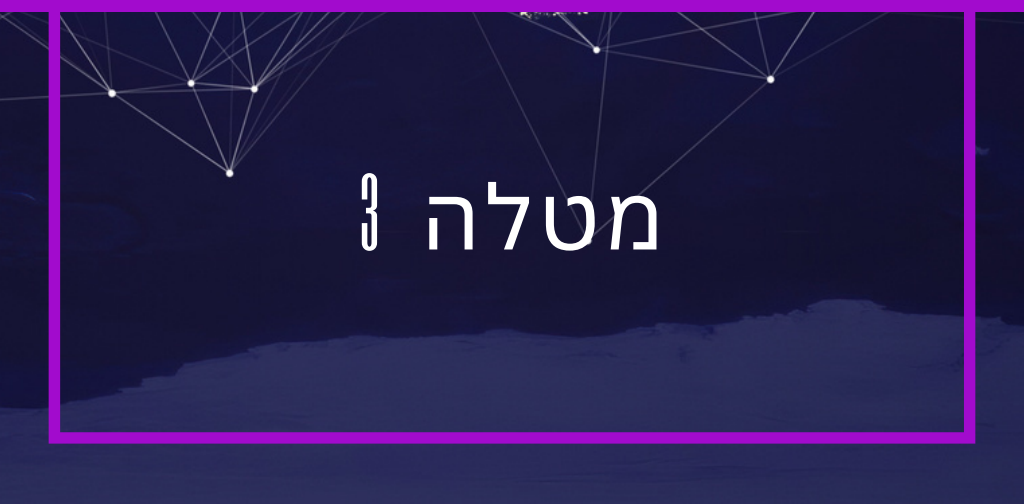


Answer: Type of DNS : A
Standard query
no answer



תקשורת

Nathanael Benichou
342769130
-
JORDAN PEREZ
336165733



מטלה 3

PART I

Question 1

הציגו יתרון אחד לשימוש ב-DoH והסבירו אותו
(כמובן, מעבר לעובדה שהוא מאובטח ומוצפן)

Answer 1

מעבר להצפנת הנתונים, ל- DOH יש יתרון
בהגדלת הביצועים, גם במקרה של אובדן חבילות,
הוא מהיר יותר בשליחת מידע.

Question 2

הציגו והסבירו על שני חסרונות לשימוש בשיטת DoH לעומת DNS הרגיל

Answer 2

א)

DoH מונע מעקב אחר משתמשים.
בחינת שאילתות DNS יכולה להיות חושפנית מאוד וניתן להשתמש בה לטובה (גילוי פעילות של תוכנות זדוניות) או לרוע (לאתר מי מבקר באיזה אתר). שוב, זו המטרה המפורשת של DoH להקשות על הניטור.

לפיכך, מפעילי רשתות אינם יכולים לסנן את התוכן של אתרים לא רצויים מכיוון שמי שמשתמש ב- DOH עוקף את רשת האבטחה הנ"ל, ומסכן את המשתמש.

ב)

השימוש ב- DOH הופך את הרשת לאיטית ופחות יעילה מכיוון שהמידע עובר ברשת מרוחקת

Question 3

בחרו אחד מהחסרונות משאלה (2), הציעו דרך למתן\לעקוף\לפתור חיסרון זה והסבירו אותה.

Answer 3

השימוש ב- GEODNS יכול להפוך את השימוש ב- DOH לאיטי פחות, GeoDns משפר את חיפוש שמות הדומיינים על ידי רזולוציית כתובת בהתבסס על המיקום הגיאוגרפי של הלקוח.

Answer 4

א) יישום ה- DoH ברמת היישום

יתרונות: ניתן ליישם את פרוטוקול ה- DNS ביישומים הנתמכים.

חסרון - אם המשתמש מתעלם מבקשות, היישום לא יוכל להציג את המשתמש שהתעלם ממנו.

ב) יישום DoH ברמת שרת ה- Proxy * ברשת -

יתרון (אנונימיות): ה- proxy הוא נתיב המאפשר להפוך לאנונימי. שימוש זה נותן אפשרות למשתמש כי ה- IP שלהם, ולכן זהותו, נותר לא ידוע.

חסרון - למשתמשים שכנים המזוהים עם אותה רשת יש גישה למידע.

ג) יישום DoH של שרת ה- proxy המקומי -

יתרון
גם אם המשתמשים מחוברים לאותה רשת הבקשות יוצפנו ולכן ההתקפה תהיה קשה יותר.

חסרון אייקון
התקנה כללית, באופן מקומי, עלולה ליצור עומס.

ד) התקן תוסף המיישם את DoH:

יתרונות:
בניגוד ל- Proxy, בהיותו מודול תוסף, אין צורך להתקין אותו בכל המכונות בנפרד.

חִסָרוֹן:
לא כל מערכות ההפעלה תומכות בתוסף

מסקנות

נראה שהכי טוב הוא לחסוך זמן ולהימנע מכל סוג של עומס, הפתרון המועיל ביותר נראה לי הפלאגין.

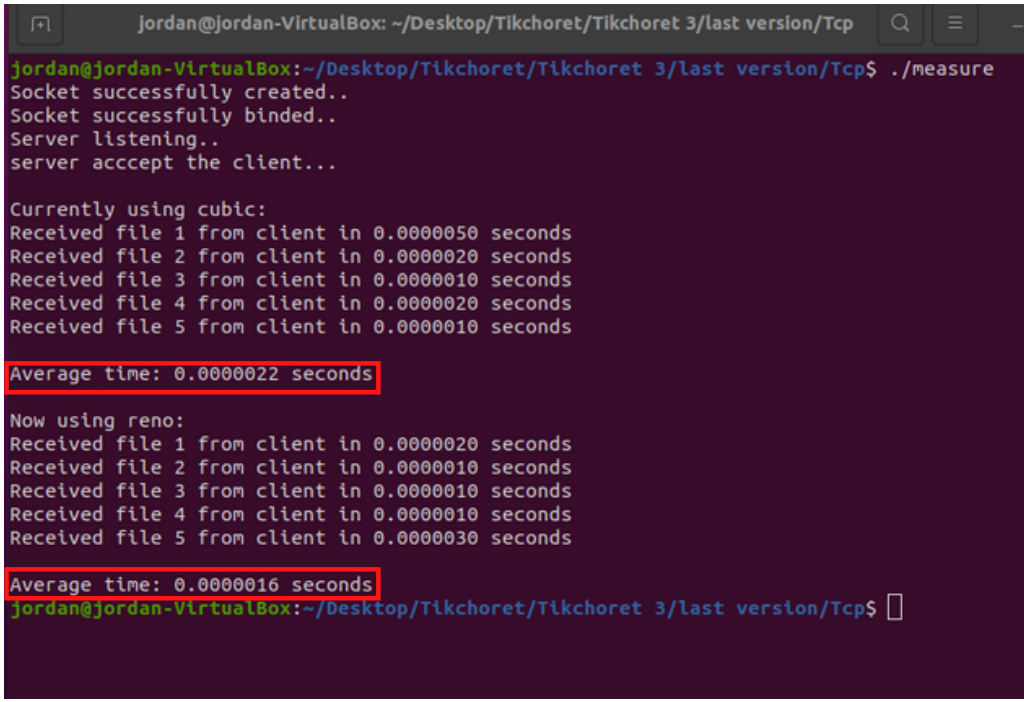
Question 5

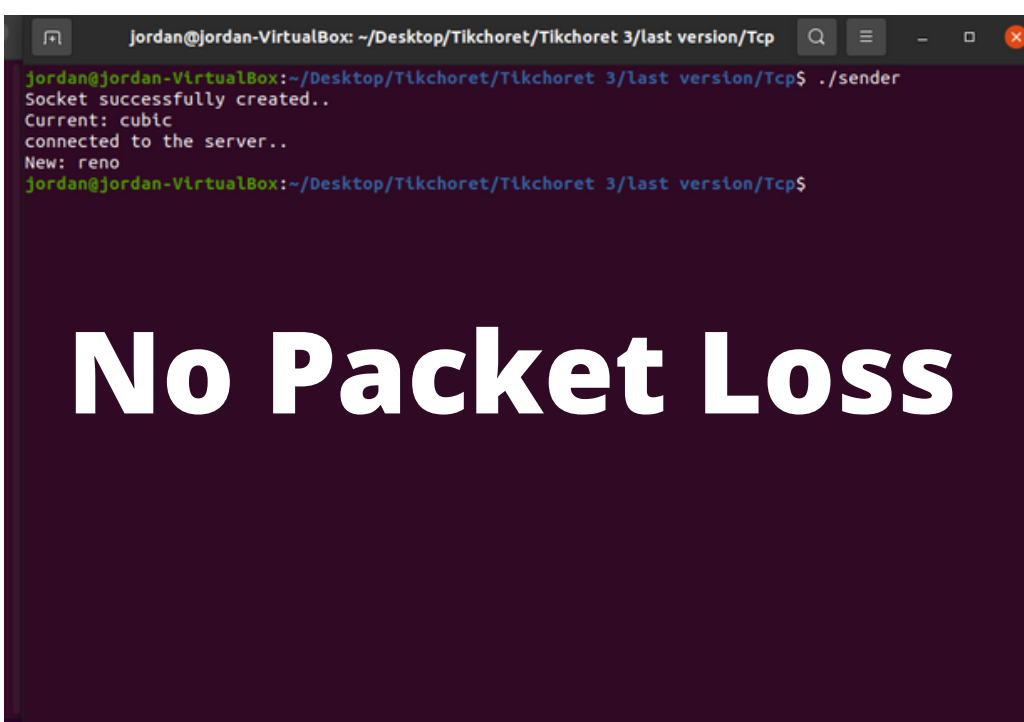
נניח שאנו ברשת שקיים בה איבוד פקטות (packet loss) באחוז לא ידוע ואנו רוצים לטעון דף שצריך 25 שאילתות כדי לבקש את כל המשאבים שבו. הציגו יתרון ברור שיש ל-DoH לעומת Do53. (רמז: מנגנון הקיים ב-TCP)

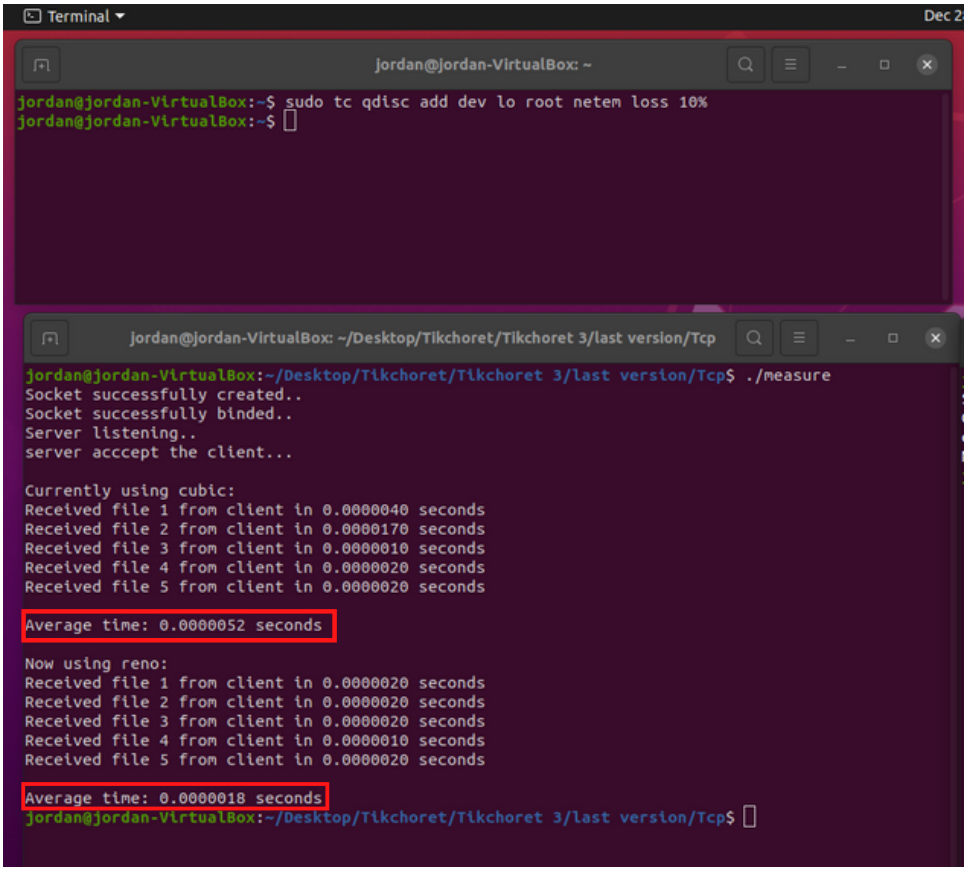
Answer 5

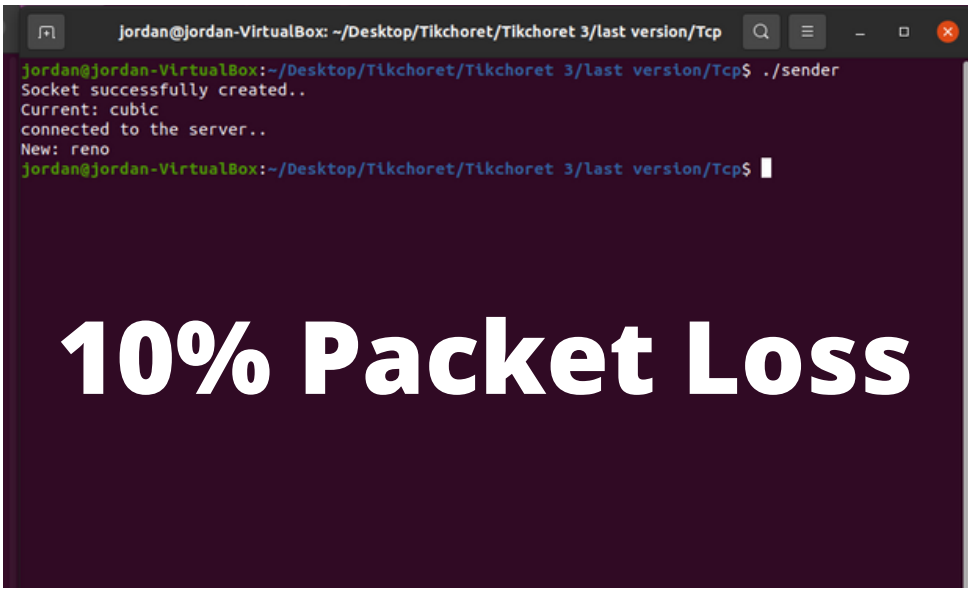
בתהליך Do53 משתמשים בפרוטוקול UDP.
אין לו את אותם המאפיינים.
כאשר משתמשים ב- DoH במהלך אובדן מנות, מנגנון שולח signal, מים לחבילה לפני זה שאבד, ולכן ניתן יהיה לשלוח אותו שוב ללא בעיה.

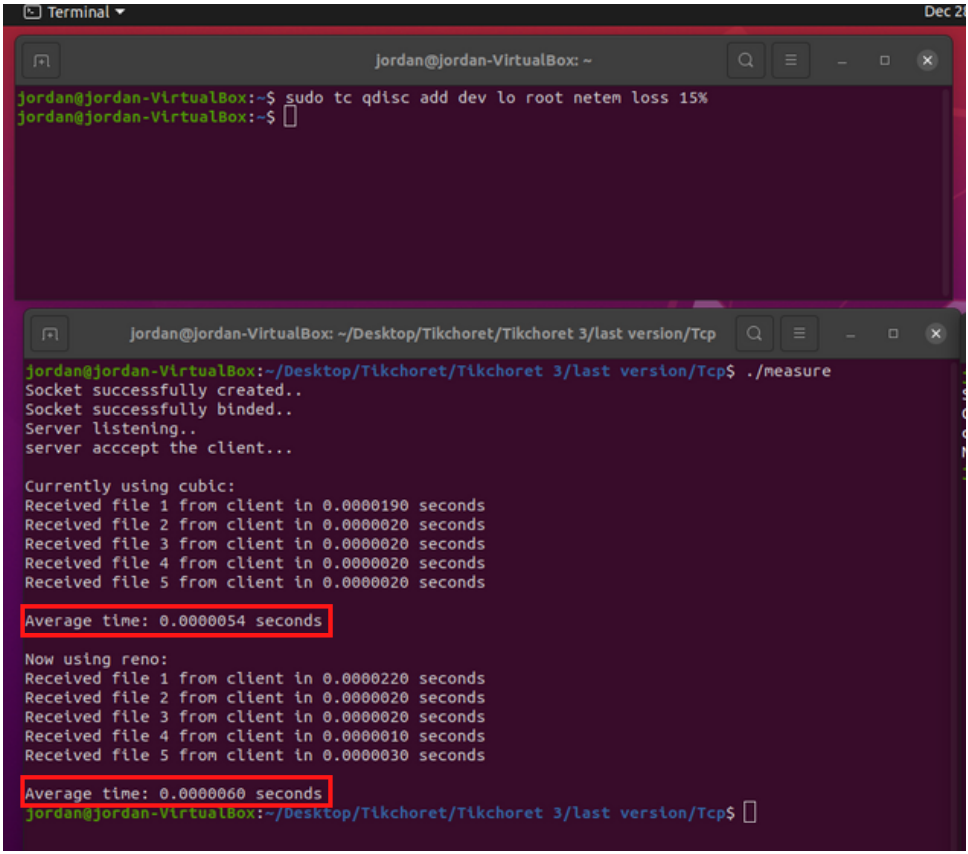
PART II

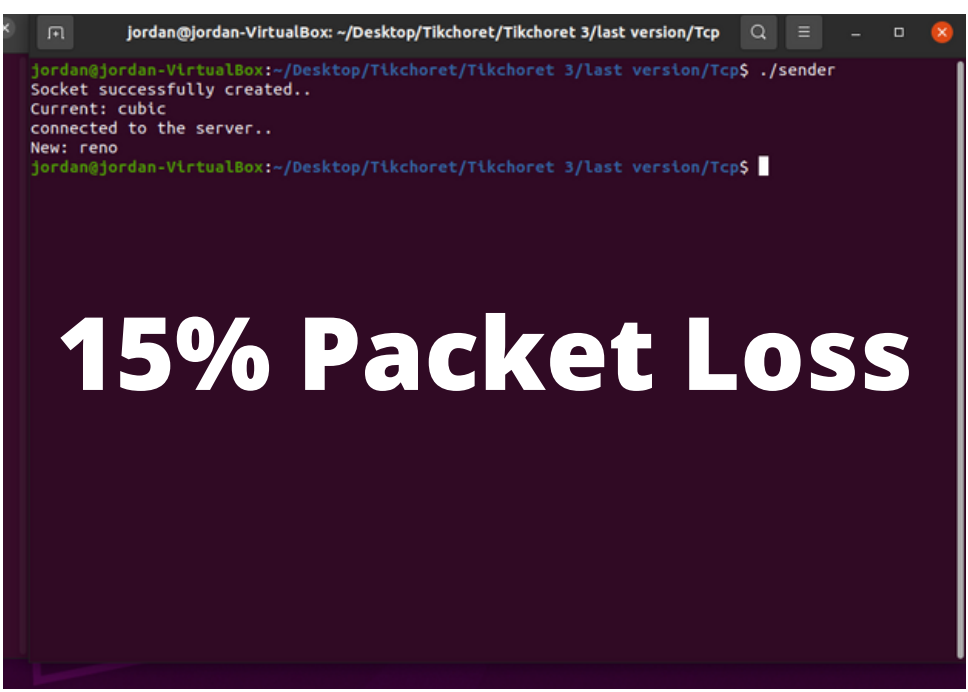

jordan@jordan-VirtualBox: ~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$ ./measure
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
Currently using cubic:
Received file 1 from client in 0.0000050 seconds
Received file 2 from client in 0.0000020 seconds
Received file 3 from client in 0.0000010 seconds
Received file 4 from client in 0.0000020 seconds
Received file 5 from client in 0.0000010 seconds
Average time: 0.0000022 seconds
Now using reno:
Received file 1 from client in 0.0000020 seconds
Received file 2 from client in 0.0000010 seconds
Received file 3 from client in 0.0000010 seconds
Received file 4 from client in 0.0000010 seconds
Received file 5 from client in 0.0000030 seconds
Average time: 0.0000016 seconds
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$

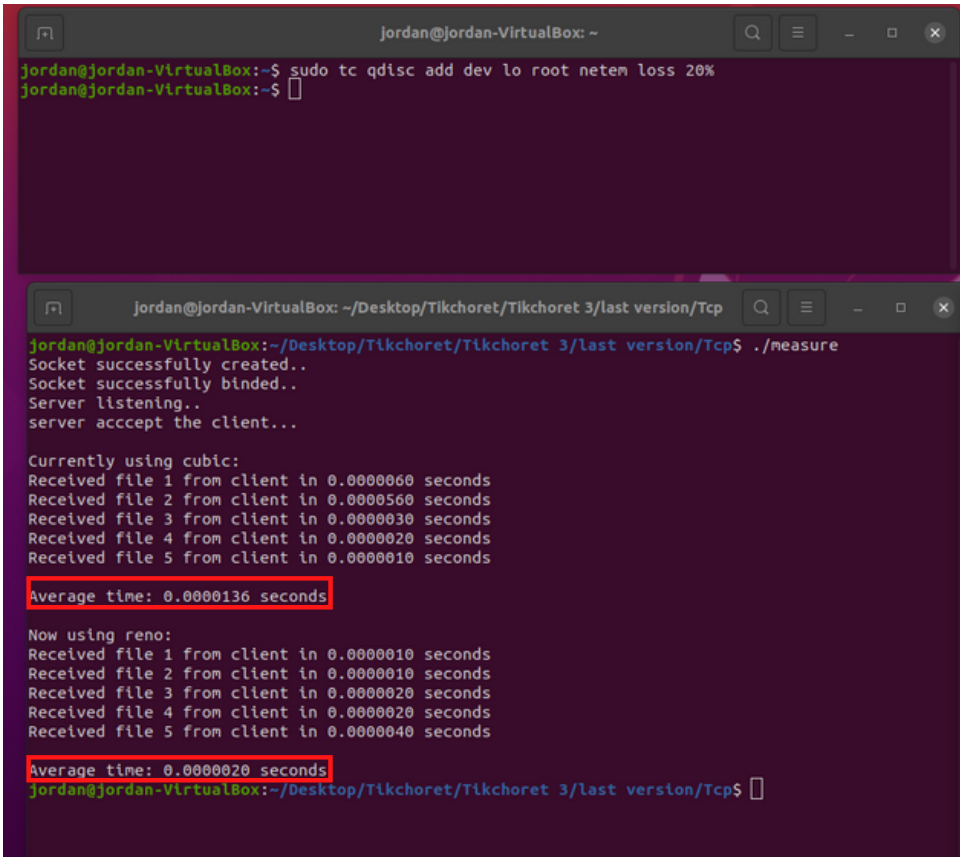

jordan@jordan-VirtualBox: ~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$ ./sender
Socket successfully created..
Current: cubic
connected to the server..
New: reno
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$
No Packet Loss

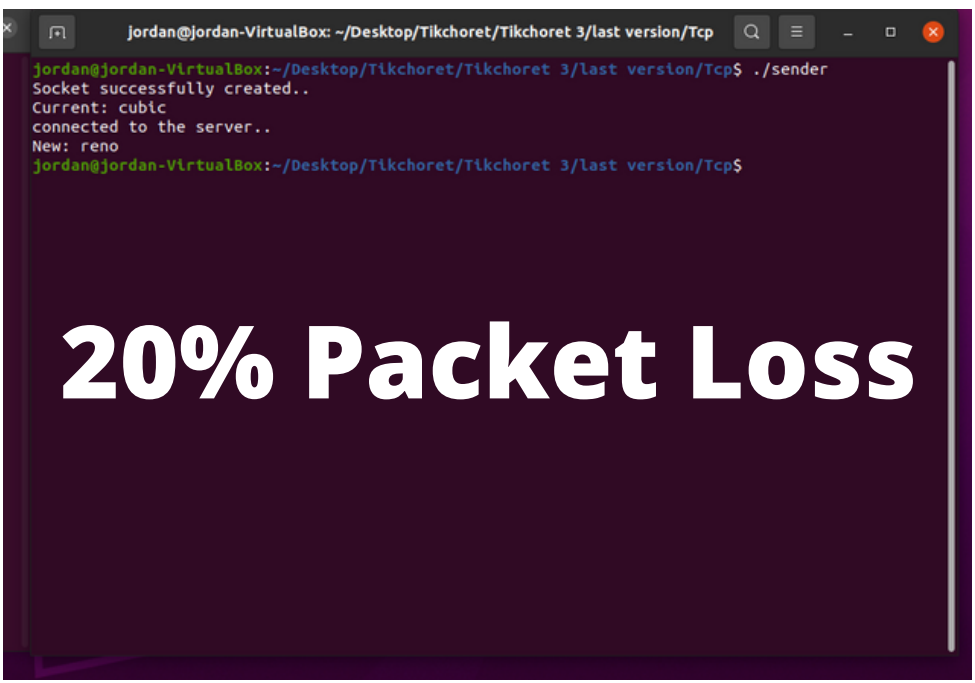

Terminal
Dec 2
jordan@jordan-VirtualBox: ~
jordan@jordan-VirtualBox:~$ sudo tc qdisc add dev lo root netem loss 10%
jordan@jordan-VirtualBox:~$
jordan@jordan-VirtualBox: ~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$ ./measure
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
Currently using cubic:
Received file 1 from client in 0.0000040 seconds
Received file 2 from client in 0.0000170 seconds
Received file 3 from client in 0.0000010 seconds
Received file 4 from client in 0.0000020 seconds
Received file 5 from client in 0.0000020 seconds
Average time: 0.0000052 seconds
Now using reno:
Received file 1 from client in 0.0000020 seconds
Received file 2 from client in 0.0000020 seconds
Received file 3 from client in 0.0000020 seconds
Received file 4 from client in 0.0000010 seconds
Received file 5 from client in 0.0000020 seconds
Average time: 0.0000018 seconds
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$

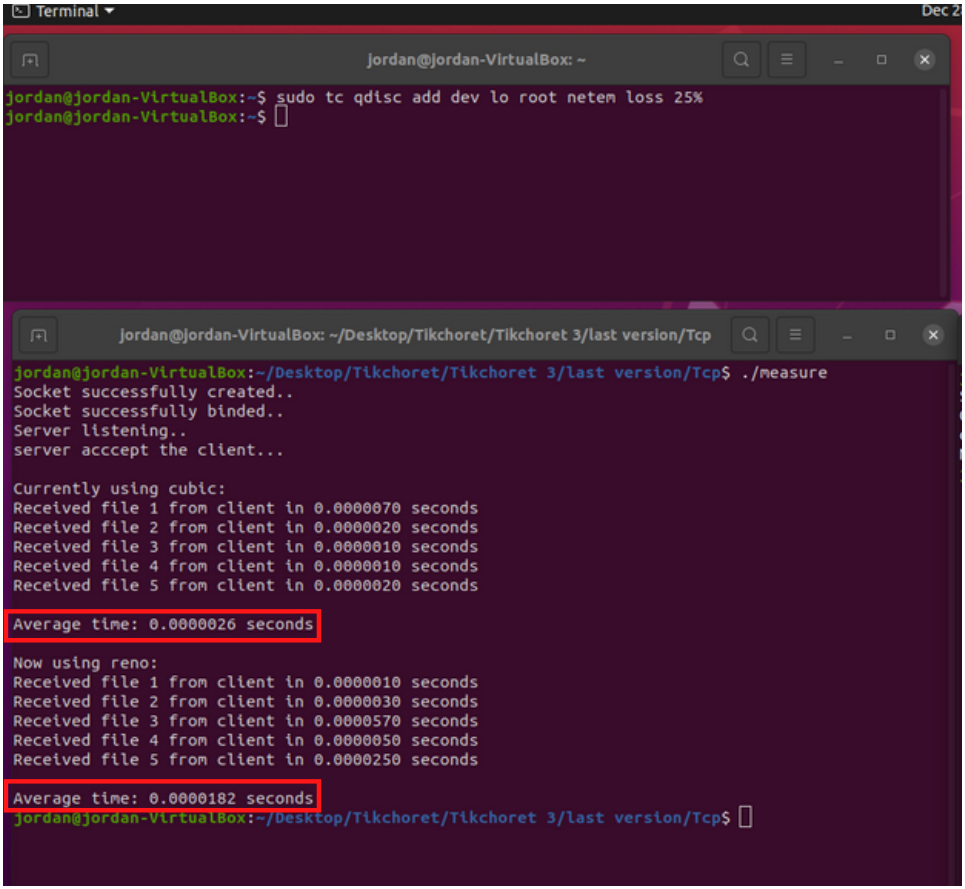

jordan@jordan-VirtualBox: ~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$ ./sender
Socket successfully created..
Current: cubic
connected to the server..
New: reno
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$
10% Packet Loss

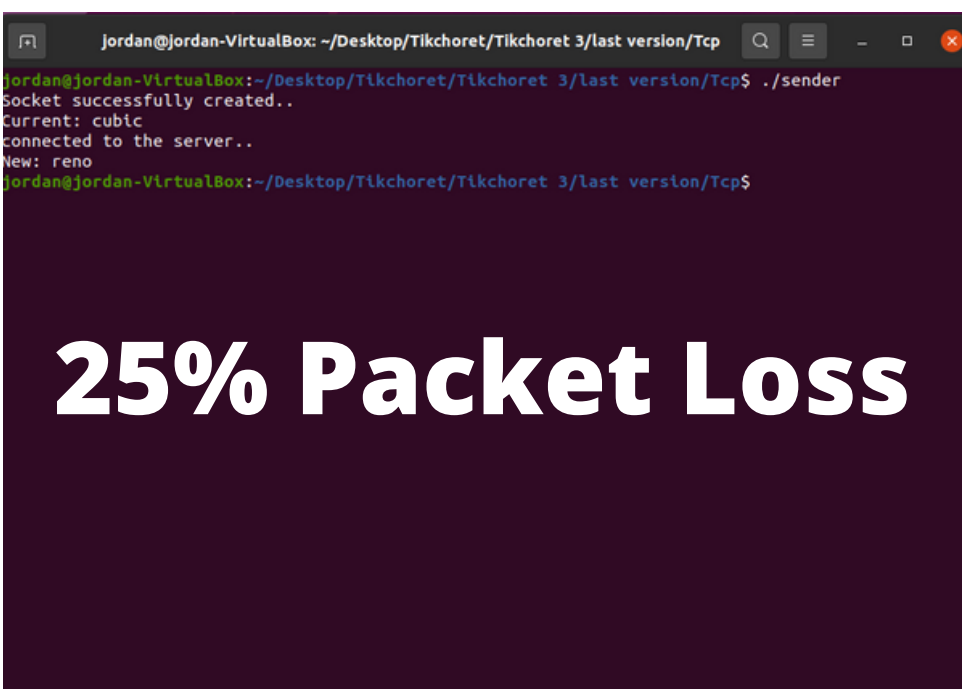

Terminal
jordan@jordan-VirtualBox: ~
jordan@jordan-VirtualBox:~$ sudo tc qdisc add dev lo root netem loss 15%
jordan@jordan-VirtualBox:~$
jordan@jordan-VirtualBox: ~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$ ./measure
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
Currently using cubic:
Received file 1 from client in 0.0000190 seconds
Received file 2 from client in 0.0000020 seconds
Received file 3 from client in 0.0000020 seconds
Received file 4 from client in 0.0000020 seconds
Received file 5 from client in 0.0000020 seconds
Average time: 0.0000054 seconds
Now using reno:
Received file 1 from client in 0.0000220 seconds
Received file 2 from client in 0.0000020 seconds
Received file 3 from client in 0.0000020 seconds
Received file 4 from client in 0.0000010 seconds
Received file 5 from client in 0.0000030 seconds
Average time: 0.0000060 seconds
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$


jordan@jordan-VirtualBox: ~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$ ./sender
Socket successfully created..
Current: cubic
connected to the server..
New: reno
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$
15% Packet Loss

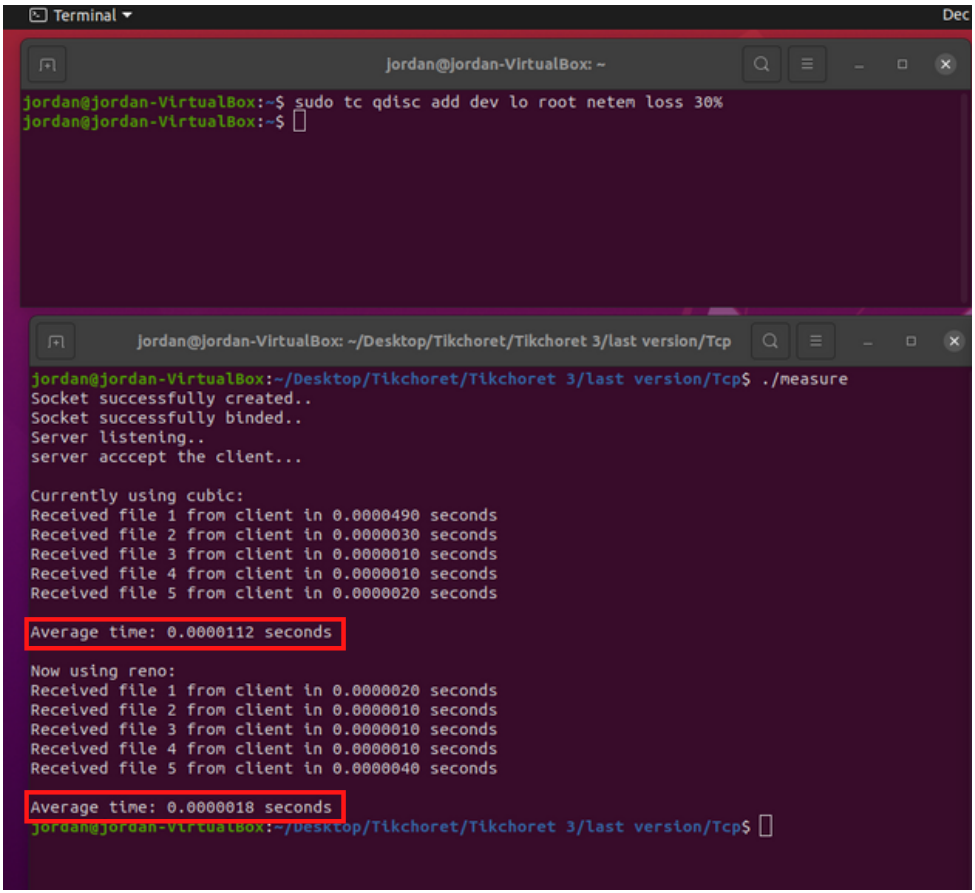

jordan@jordan-VirtualBox: ~
jordan@jordan-VirtualBox:~$ sudo tc qdisc add dev lo root netem loss 20%
jordan@jordan-VirtualBox:~$
jordan@jordan-VirtualBox: ~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$ ./measure
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
Currently using cubic:
Received file 1 from client in 0.0000060 seconds
Received file 2 from client in 0.0000560 seconds
Received file 3 from client in 0.0000030 seconds
Received file 4 from client in 0.0000020 seconds
Received file 5 from client in 0.0000010 seconds
Average time: 0.0000136 seconds
Now using reno:
Received file 1 from client in 0.0000010 seconds
Received file 2 from client in 0.0000010 seconds
Received file 3 from client in 0.0000020 seconds
Received file 4 from client in 0.0000020 seconds
Received file 5 from client in 0.0000040 seconds
Average time: 0.0000020 seconds
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$

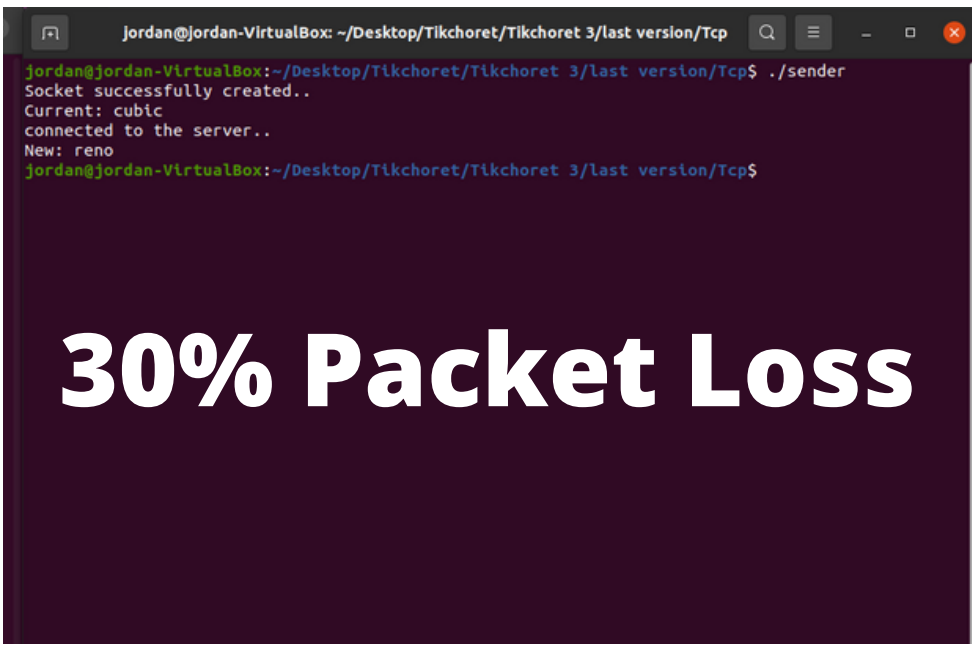

jordan@jordan-VirtualBox: ~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$ ./sender
Socket successfully created..
Current: cubic
connected to the server..
New: reno
jordan@jordan-VirtualBox:~/Desktop/Tikchoret/Tikchoret 3/last version/Tcp$
20% Packet Loss





25% Packet Loss





30% Packet Loss

תקשורת

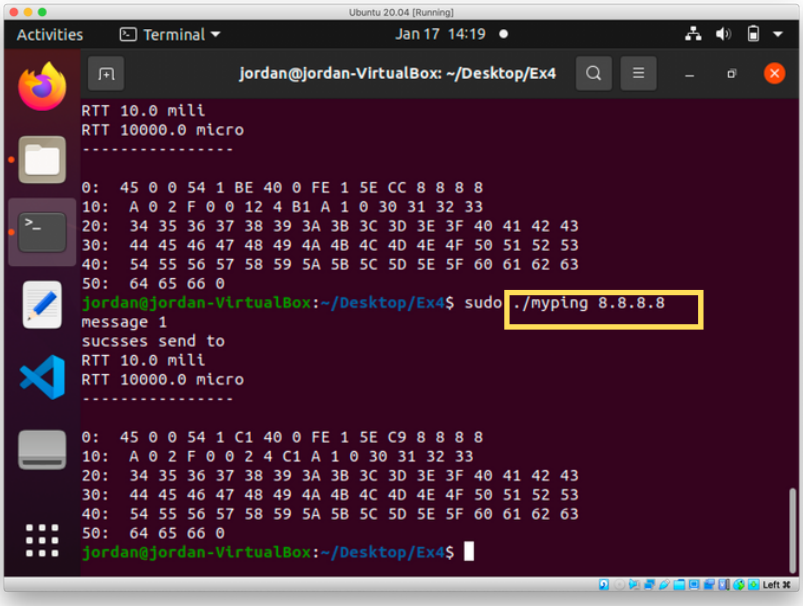
מטלה 4

Nathanael Benichou-Jordan Perez
342769130 - 336165733

PARTIE 1

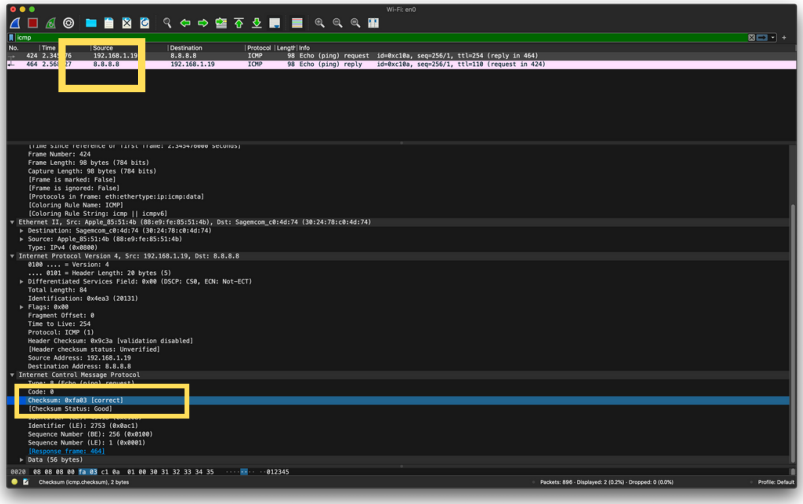
חלק א' - *myping*

Send Ping with myping.c



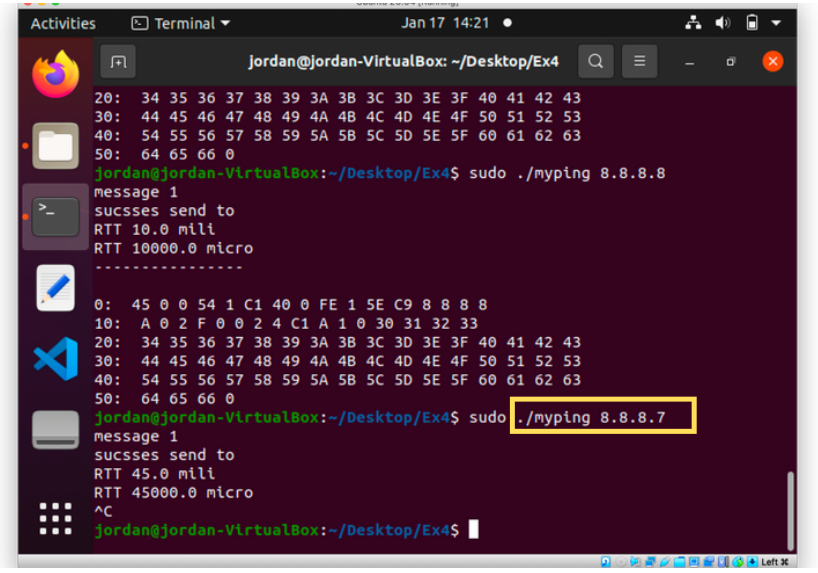
Send ping to 8.8.8.8

Capture pacquets with WireShark

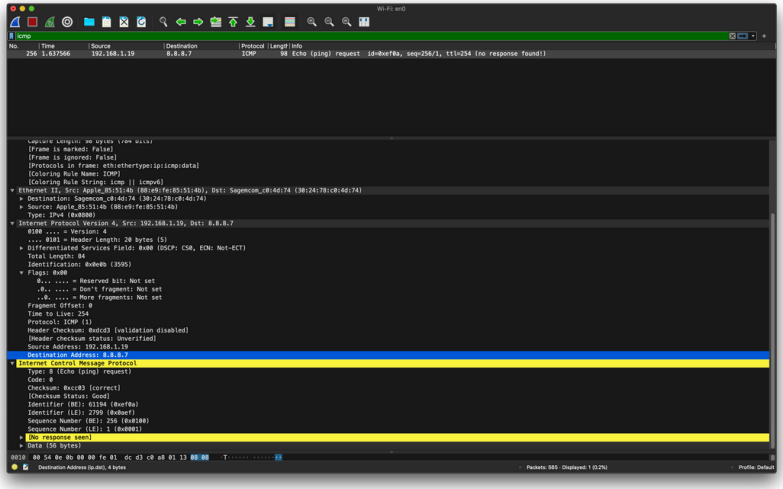


Paquets ICMP well sniffed , CheckSum Correct ,Destination is 8.8.8.8

Send Ping (Another try)



Send ping to 8.8.8.7

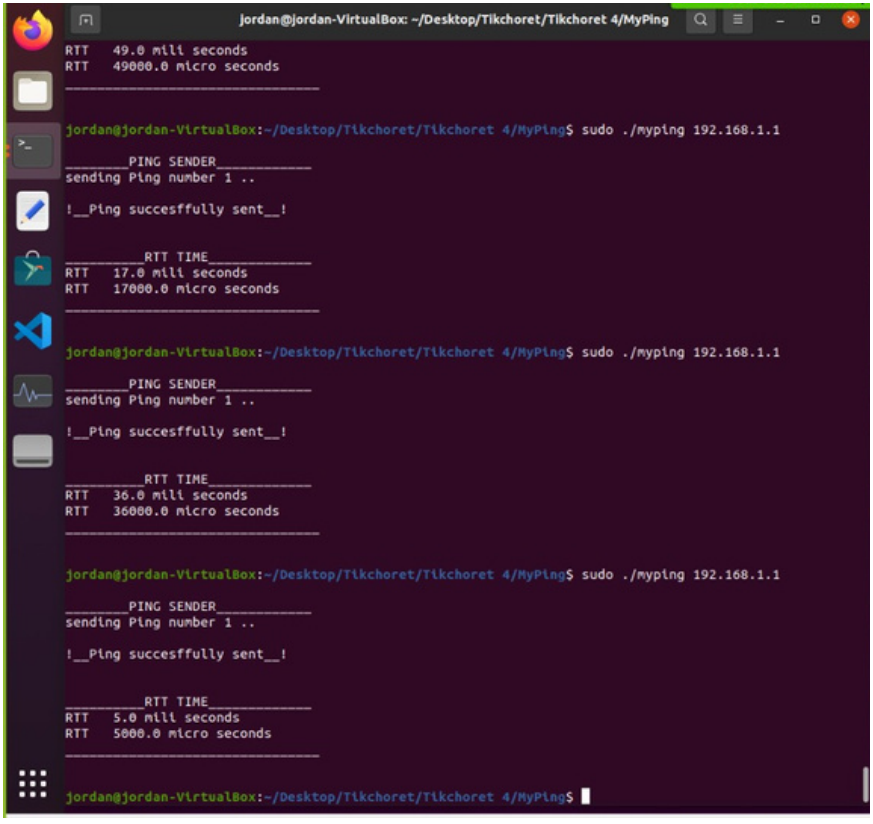


Well Recieved

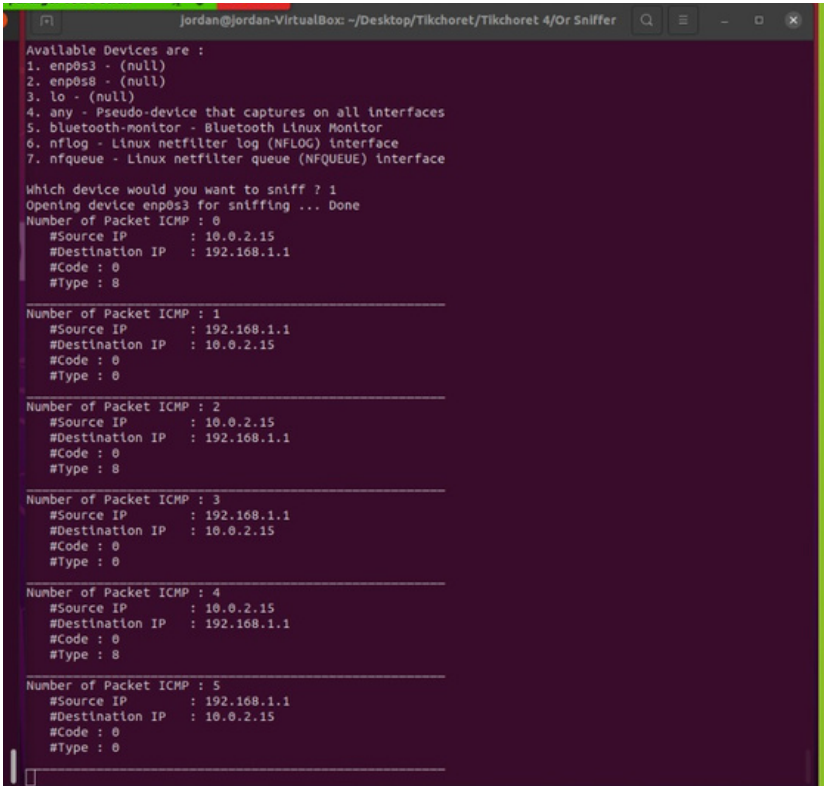
PARTIE 2

חלקן ב' – Sniffing

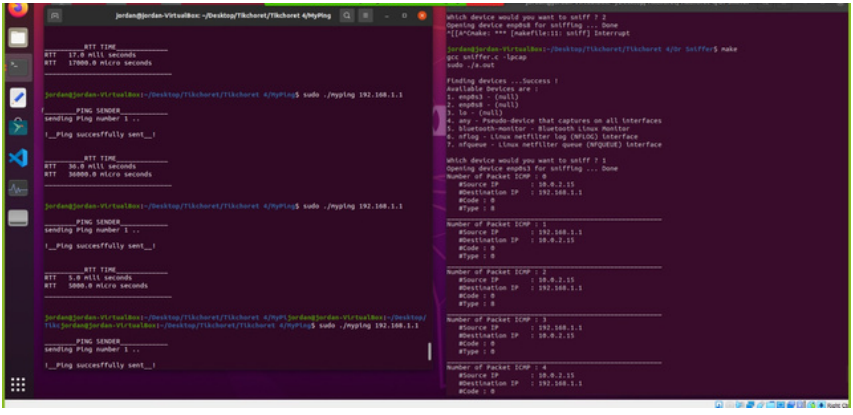
Send 7 Ping to the Same Adress



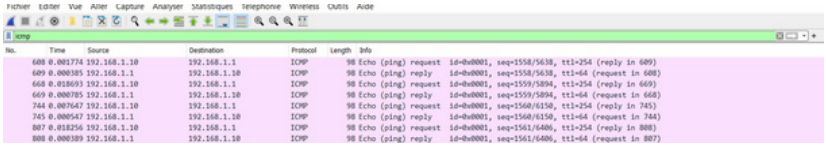
Sniff The Ping Sent with Sniffer.c



Compare My Sniffer And WireShark



Send 4 Ping to the Same Adresse Sniffe With sniffer.c & WireShark



תקשורת

FINAL PROJECT

Packet Sniffing and Spoofing Lab

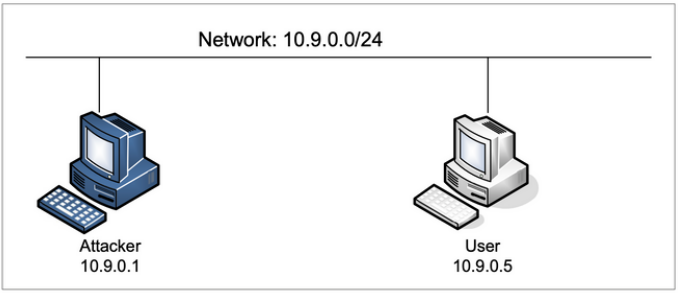
Nathanael Benichou-Jordan Perez
342769130 - 336165733

Packet Sniffing and Spoofing Lab

Packet sniffing and spoofing are the two important concepts in network security; they are two major threats in network communication. Being able to understand these two threats is essential for understanding security measures in networking. There are many packet sniffing and spoofing tools, such as Wireshark, Tcpdump, Netwox, etc. Some of these tools are widely used by security experts, as well as by attackers. Being able to use these tools is important for students, but what is more important for students in a network security course is to understand how these tools work, i.e., how packet sniffing and spoofing are implemented in software.

The objective of this lab is to master the technologies underlying most of the sniffing and spoofing tools. Students will play with some simple sniffer and spoofing programs, read their source code, modify them, perform a man in the middle attack, and eventually gain an in-depth understanding on the technical aspects of these programs. At the end of this lab, students should be able to write their own sniffing and spoofing programs.

Environment Setup using Container

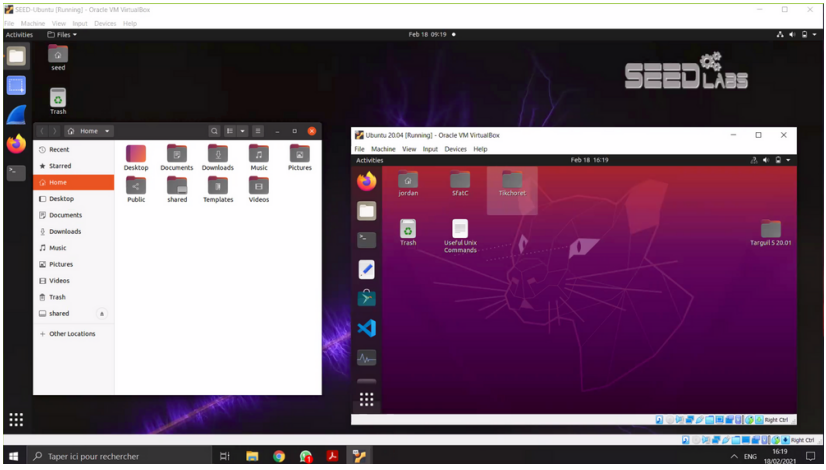


Lab Environment Setup:
Three Virtual Machines emulating Linux Ubuntu 20.04 (SEED Version)

1.VM (Seed attacker) - IP 10.0.2.7

2.VM SEED Ubuntu (Victim 1) - IP 10.0.2.8

3.VM SEED Ubuntu (Victim2) - IP 10.0.2.9



First of all we configured a shared folder for more convenience

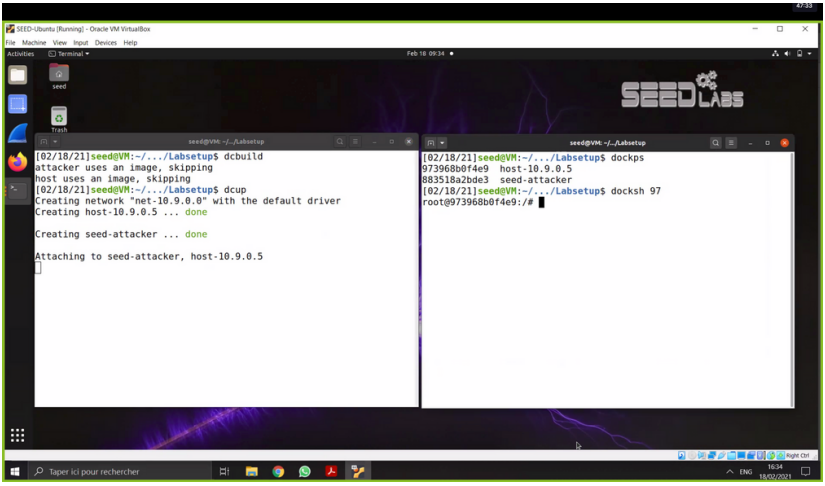
First contact with Dockers

Instead of using multiple VMs we could also use Dockers to emulate containers like this:

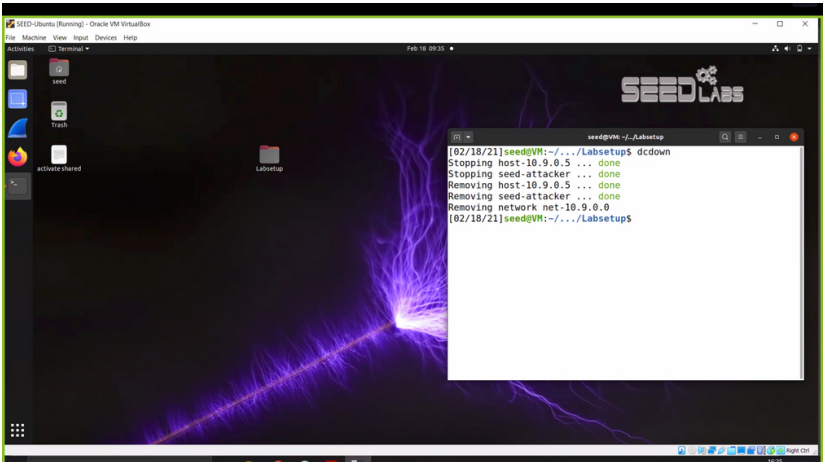
Commands used :

$ dockps

$ docksh <id>



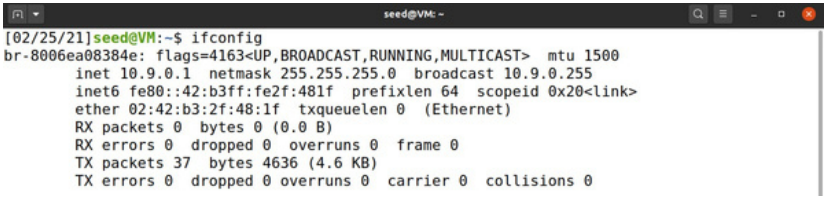
$ dcdown



3 Using Scapy to Sniff and Spoof Packets

3.1 Task 1.1. Sniffing packets

$ ifconfig



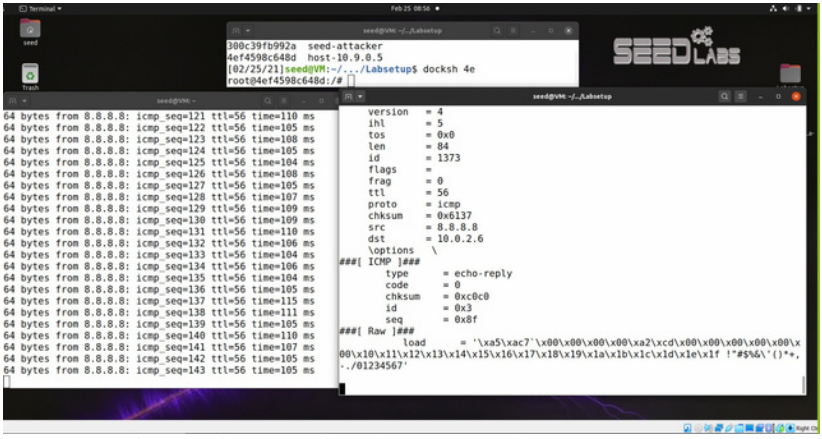
So the name of the Network is :

br-8006ea08384e

Python code to sniff icmp packets

Task 1.1.A

Running Scapy with sudo

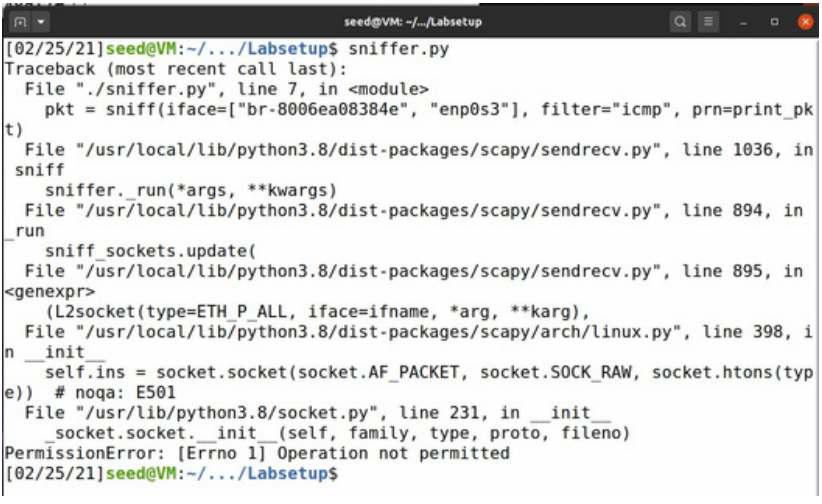


Running the python code of ICMP sniffing (using ID) in root

.

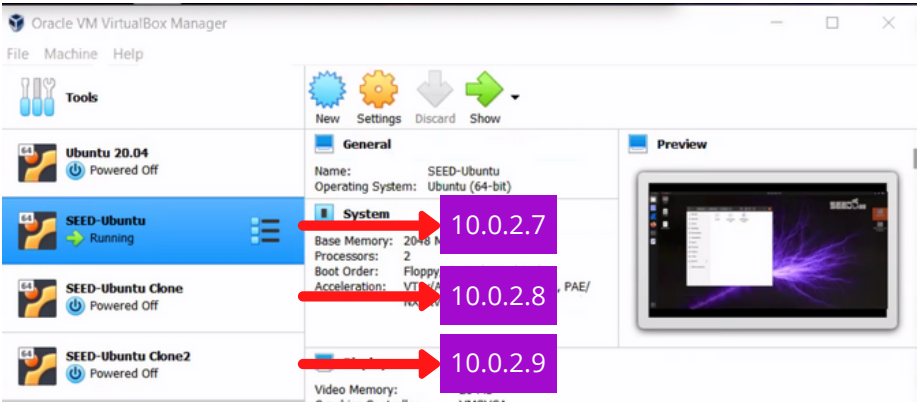
All the ICMP packets sent with the command ping 8.8.8.8 (Google DNS) are sniffed successfully

Running Scapy without sudo

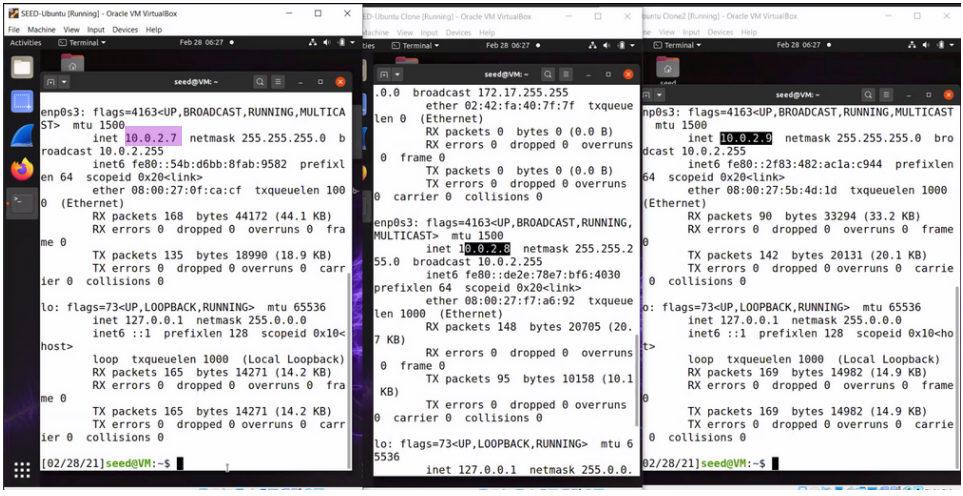


Core Dump/Segmentation fault is a specific kind of error caused by accessing memory that “does not belong to you.”

Using 3 VM (Seed) with different MAC Adresse Attached to NAT network



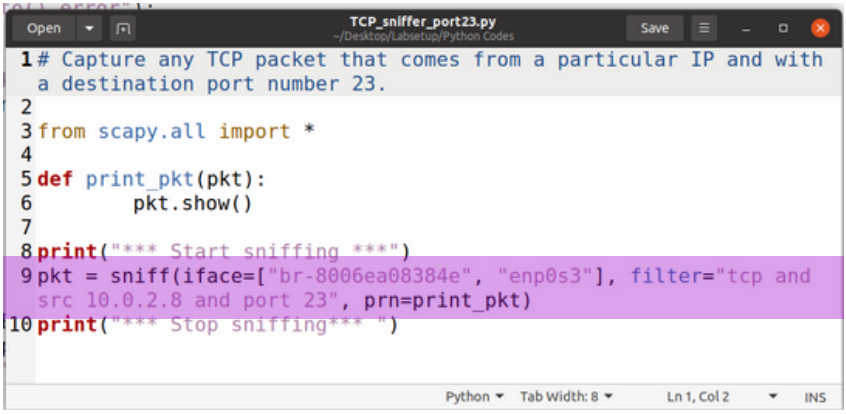
$ifconfig on the three VM



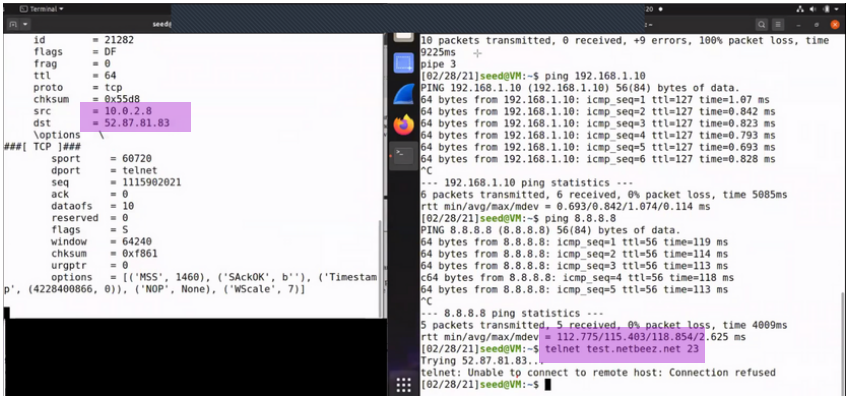
Task 1.1.B

- **Capturing only the ICMP packet: See 1.1.A**
- **Sniffing TCP from 10.0.2.8 with filter on port 23**
  (Using Telnet cmds to check it's working)

Code :



Result :



- **Capturing packets that comes from or that goes to a particular subnet.**
  You should not pick the subnet that your VM is attached to.

pcap filter host command:

Allowable primitives are:

dst host *host*
True if the IPv4/v6 destination field of the packet is *host*, which may be either an address or a name.
src host *host*
True if the IPv4/v6 source field of the packet is *host*.
host *host*
True if either the IPv4/v6 source or destination of the packet is *host*.
Any of the above host expressions can be prepended with the keywords, **ip**, **arp**, **rarp**, or **ip6** as in:

```
ip host host
```

which is equivalent to:

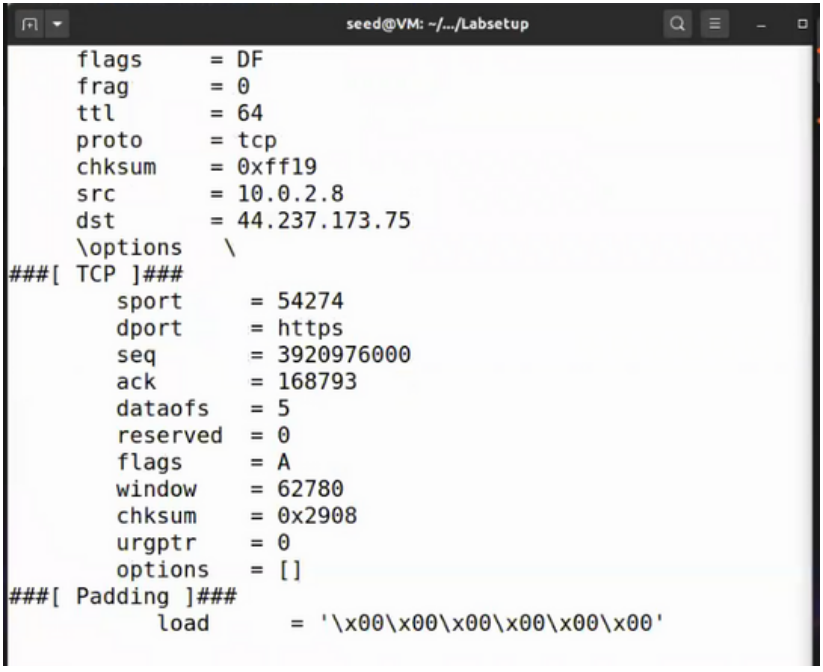
```
ether proto \ip and host host
```

If *host* is a name with multiple IPv4 addresses, each address will be checked for a match.

sudo python3 TCP_Sniffer_subnet.py

```
[02/28/21]seed@VM:~/.../Labsetup$ sudo python3 TCP_sniffer_subn
et.py
*** Start sniffing ***
```

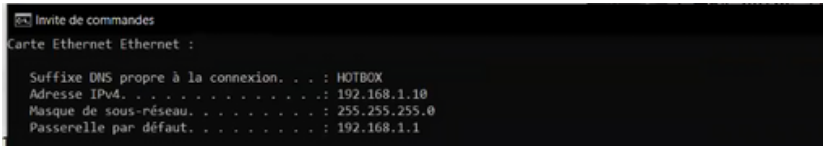
On VM 10.0.2.8 we open Firefox window and we can see that the packets were well sniffed



PS: *When we open a firefox window from VM 10.0.2.7 nothing happens as expected*

3.2 ICMP SPOOFER

Running Ipconfig on windows *(192.168.1.10)*



Task 1.2

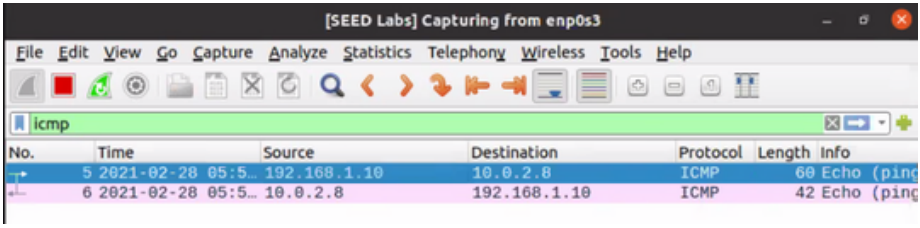
For this part we will need 3 VM.
We will run the following code on VM 10.0.2.7 which will be our man in the middle
The purpose is to spoof packets sent from one VM to another.
The ping will be issued by VM 10.0.2.8 to 10.0.2.9 using the command ping -c 1 10.0.2.9 which only issues a single ping request:

Open *ICMP_spoofer.py ~/Desktop/Labsetup Save

```
1 # ICMP Spoofer
2 from scapy.all import *
3
4 # Creates an IP object from the IP class
5 a = IP()
6
7 a.src = "10.0.2.8" # sets src address
8 a.dst = "192.168.1.10" # sets dst address
9 a.ttl = 10 # sets TTL
10
11 # Creates an ICMP object from the ICMP class
12 b = ICMP()
13
14 # The / operator is overloaded by the IP class, so
   it no longer represents division;instead, it means
   adding b as the payload field of a and modifying
   the fields of a accordingly. As a result, we get a
   new object that represent an ICMP packet
15 p = a/b
16
17 send(p)
18 ls(a)
```

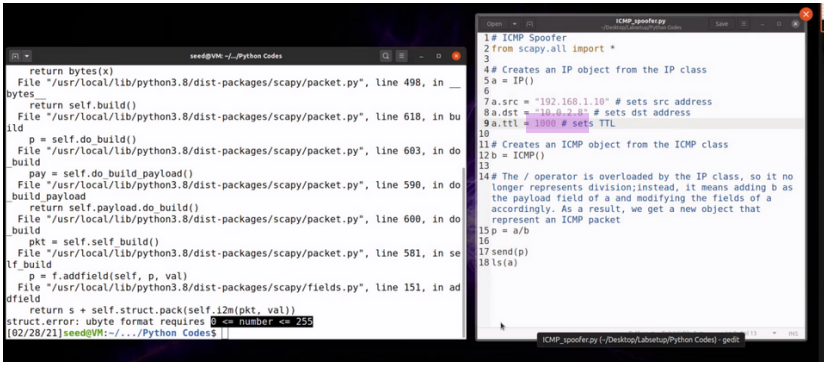
3.2 Task 1.2 Spoofing ICMP Packets

Using Wireshark on VM 10.0.2.9, we can see the ICMP request is successfully spoofed:
Although the request was sent by VM 10.0.2.7 it seems that it has been sent by 192.168.1.10 *(which is not true)*



Task 1.3: Traceroute

The objective of this task is to use Scapy to estimate the distance, in terms of number of routers, between your VM and a selected destination



We can see that with TTL=1000 we got an error message indicating that max TTL is 255

Task 1.4: Sniffing and-then Spoofing

For this part we will also need 3 VM.
We will run the following code on VM 10.0.2.7 which will be our man in the middle.
The purpose is to sniff and spoof packets sent from one VM to another.
The ping will be issued by VM 10.0.2.8 to 10.0.2.9 using the command
ping -c 1 10.0.2.9 which only issues a single ping request:

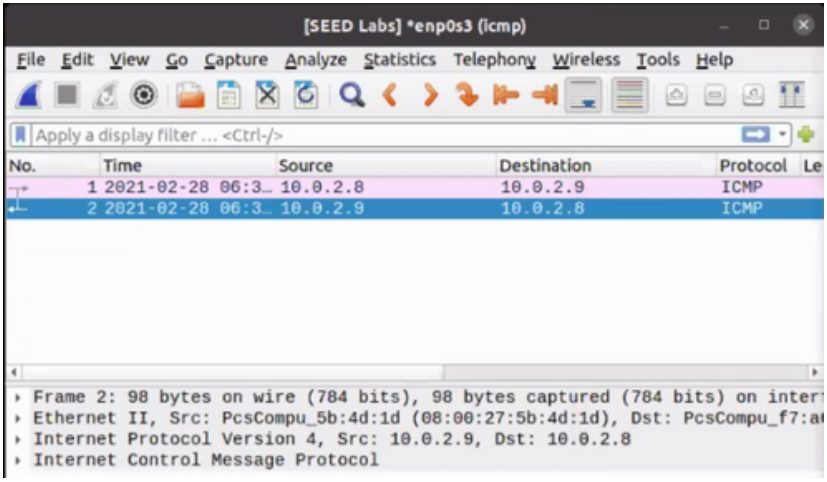
The following code is run on VM 10.0.2.7.
It sniffs the packet then spoofs it by swaping the source IP address with the dest IP address

Open ▾ sniffandspoof.py ~/Desktop/Labsetup/Python Codes Save

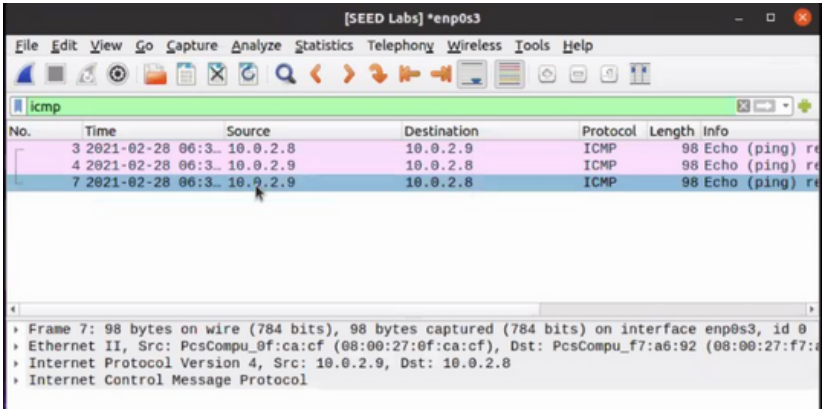
```
1 #!/usr/bin/python3
2 from scapy.all import *
3
4 def spoof_pkt(pkt):
5   if ICMP in pkt and pkt[ICMP].type == 8:
6       print("Original Packet.........")
7       print("Source IP : ", pkt[IP].src)
8       print("Destination IP :", pkt[IP].dst)
9
10      ip = IP(src=pkt[IP].dst, dst=pkt[IP].src,
    ihl=pkt[IP].ihl)
11      icmp = ICMP(type=0, id=pkt[ICMP].id,
    seq=pkt[ICMP].seq)
12      data = pkt[Raw].load
13      newpkt = ip/icmp/data
14
15      print("Spoofed Packet.........")
16      print("Source IP : ", newpkt[IP].src)
17      print("Destination IP :", newpkt[IP].dst)
18
19      send(newpkt,verbose=0)
20
21 pkt = sniff(filter='icmp and src host
    10.0.2.8',prn=spoof_pkt)
```

We are going to see the difference when we run a single ping request with and without spoofing ICMP:
*Using WireShark on VM **10.0.2.9***

Without spoofing



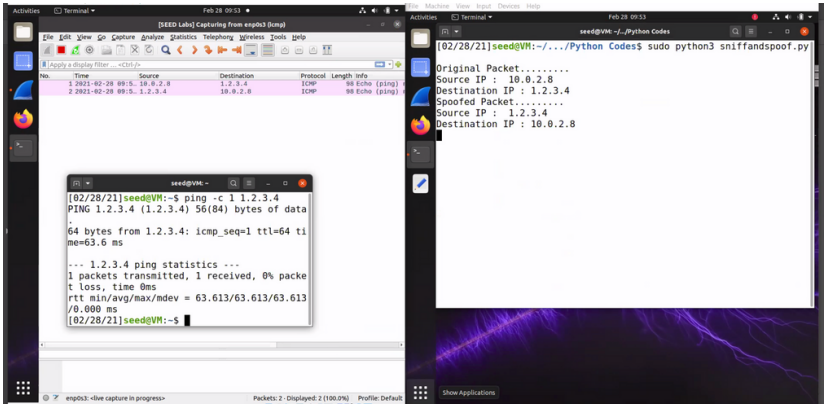
With spoofing



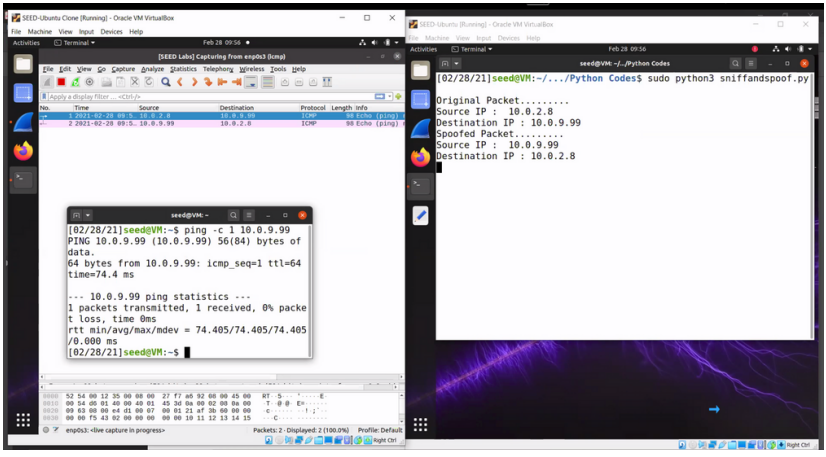
We can see that the sniff and spoof was successful: there is a third ICMP Echo Request from 10.0.2.9 to 10.0.2.8 as expected

```
ping 1.2.3.4      # a non-existing host on the Internet
ping 10.9.0.99    # a non-existing host on the LAN
ping 8.8.8.8      # an existing host on the Internet
```

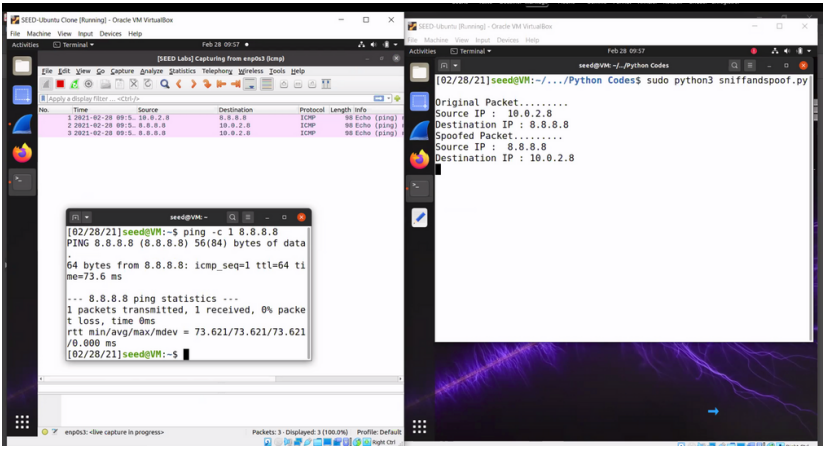
ping 1.2.3.4 *(non-existing host on the Internet)*



ping 10.0.9.99 *(non-existing host on the LAN)*



ping 8.8.8.8 *(an-existing host on the Internet)*

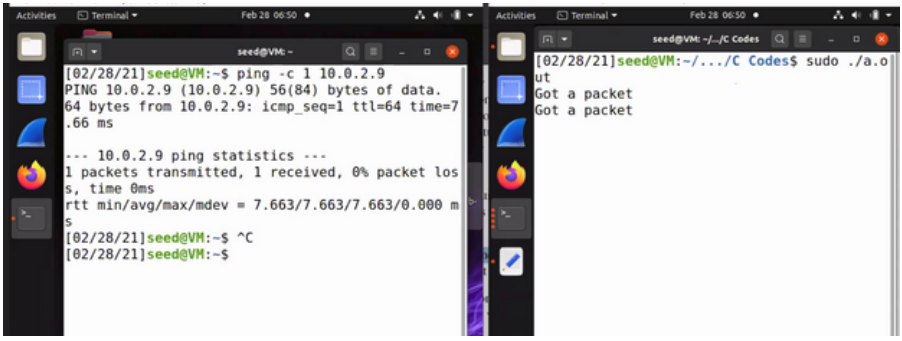


Conclusion: Even though ping requests were issued towards non-existing hosts, the victim still got answers (packets) from them.
It means that the spoof was successful: the attacker successfully altered the communications between the two parties and the victim sees a response from a host that truly doesn't exist.
And when the ping is issued towards an existing host on the Internet (here Google) the communication is also altered successfully (IP source and IP destination swapped)

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

For this set up of tasks we will compile the C code inside the host VM

4.1 Task 2.1 Writing Packet Sniffing Program



Using the code of the pdf (sniff 4.1)

In this task, we need to write a sniffer program in C to print out the source and destination IP addresses of each captured packet

Task 2.1A: Understanding How a Sniffer Works

Question 1. Please use your own words to describe the sequence of the library calls that are essential for sniffer programs. This is meant to be a summary, not detailed explanation like the one in the tutorial or book.

The following calls are essential for this sniffer program:

- pcap_lookupdev : Find the default device on which to capture
- pcap_lookupnet : Is used to determine the IPv4 network number and mask associated with the network device.
- **pcap_open_live** : Open a device to start sniffing
- pcap_datalink : Get the link-layer header type to know what type
- **pcap_compile** : Compile a filter expression
- **pcap_setfilter** : Sets the compiled filter
- **pcap_loop** : Process packets from the capture
- pcap_freecode : Frees up allocated memory generated
- **pcap_close** : Closes the sniffing session

Question 2. Why do you need the root privilege to run a sniffer program? Where does the program fail if it is executed without the root privilege?

We need root privilege to access to the Network Interface Card which needs admin authorization

For the following questions we are going to use the given main. We only changed eth3 to enp0s3:

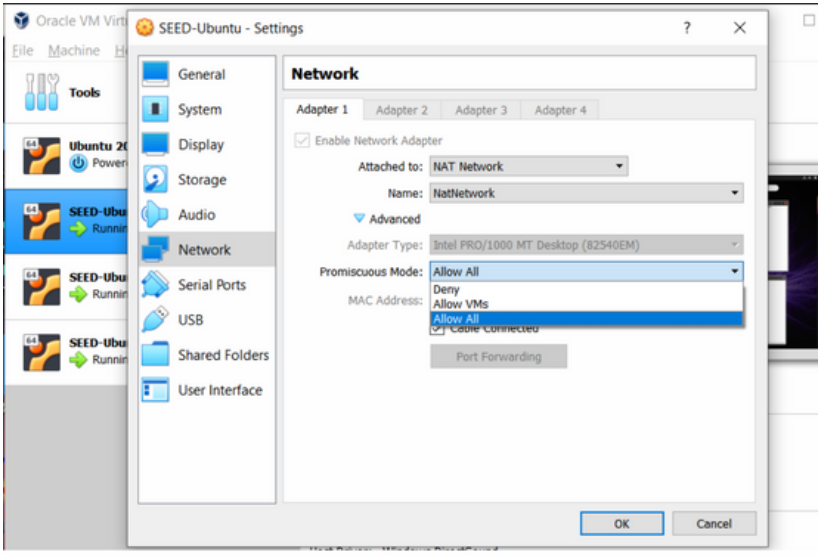
```c
int main()
        {
        pcap_t *handle;
        char errbuf[PCAP_ERRBUF_SIZE];
        struct bpf_program fp;
        char filter_exp[] = "ip proto icmp";
        bpf_u_int32 net;
        // Step 1: Open live pcap session on NIC with name eth3
        //Students needs to change "eth3" to the name
        //found on their own machines (using ifconfig).
        handle = pcap_open_live("enp0s3", BUFSIZ, 1, 1000, errbuf);
        // Step 2: Compile filter_exp into BPF psuedo-code
        pcap_compile(handle, &fp, filter_exp, 0, net);
        pcap_setfilter(handle, &fp);
        // Step 3: Capture packets
        pcap_loop(handle, -1, got_packet, NULL);
        pcap_close(handle);
        return 0;
        //Close the handle
        }
```

<u>Quesiton 3. Please turn on and turn off the promiscuous mode in your sniffer program. Can you demonstrate the difference when this mode is on and off? Please describe how you can demonstrate this.</u>

For this part we will also need 3 VMs.

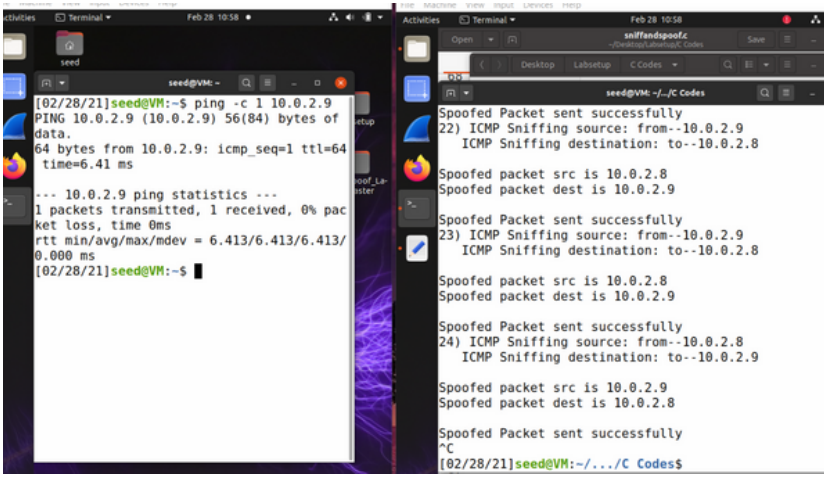
We will run the sniffandspoof.c code on VM 10.0.2.7 which will be our man in the middle but we will test the 3 different options we have for the promiscuous mode in VirtualBox (changes only for the VM being the man in the middle)

Ping requests are sent from VM 10.0.2.8 to VM 10.0.2.9):

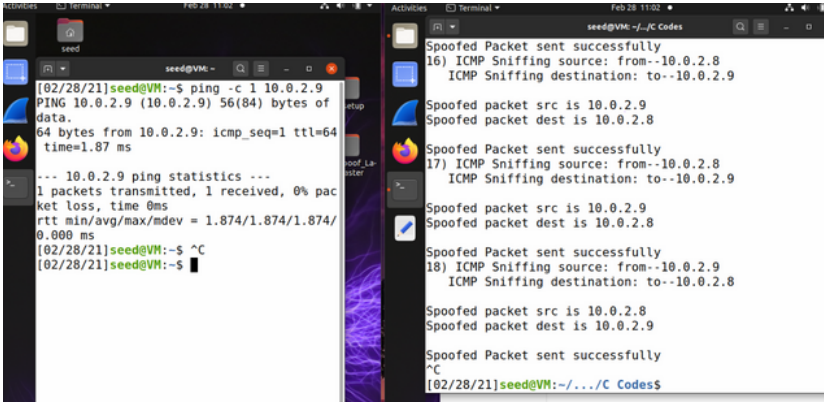


The available modes are: Deny, Allow VMs and Allow All

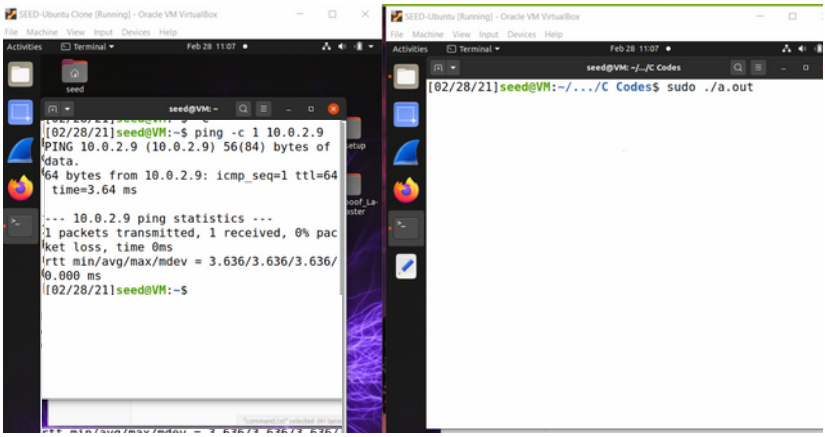
Running sniffandspoof on VM 10.0.2.7 with Promiscous mode Allow All



Running sniffandspoof on VM 10.0.2.7 with Promiscous mode Allow VMs



Running sniffandspoof on VM 10.0.2.7 with Promiscous mode Deny



We can see that in Deny mode absolutely no packets are sniffed nor spoofed.
The reason is because Promiscuous mode allows a network sniffer to pass all the traffic from a network controller and not just the traffic that the network controller was intended to receive.

PS: It's possible to manually switch between promiscuous modes directly in the code:

```
/* promisc mode on */
handle = pcap_open_live(dev, SNAP_LEN, 1, 1000, errbuf);

/* promisc mode off */
handle = pcap_open_live(dev, SNAP_LEN, 0, 1000, errbuf);
```

Task 2.1B: Writing Filters

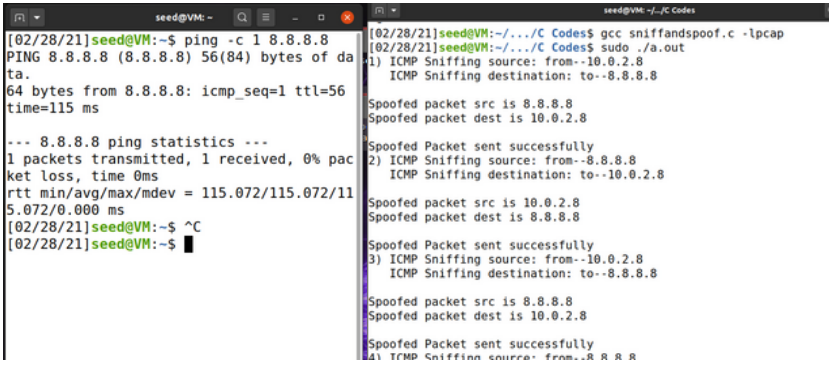
Writing filter expressions for the sniffer program to capture each of the followings:

- **Capture the ICMP packets between two specific hosts**

We can filter only the ICMP packets between two given hosts by modifying the filter_exp[] string this way:

```
int main()
        {
        pcap_t *handle;
        char errbuf[PCAP_ERRBUF_SIZE];
        struct bpf_program fp;
        // char filter_exp[] = "ip proto icmp";

        // ICMP packets between this host and 8.8.8.8
        char filter_exp[] = "icmp and (src host 10.0.2.8 and dst host 8.8.8.8) or (src host 8.8.8.8 and dst host 10.0.2.8)";

        bpf_u_int32 net;
        // Step 1: Open live pcap session on NIC with name eth3
        //Students needs to change "eth3" to the name
        //found on their own machines (using ifconfig).
        handle = pcap_open_live("enp0s3", BUFSIZ, 1, 1000, errbuf);
        // Step 2: Compile filter_exp into BPF psuedo-code
        pcap_compile(handle, &fp, filter_exp, 0, net);
        pcap_setfilter(handle, &fp);
        // Step 3: Capture packets
        pcap_loop(handle, -1, got_packet, NULL);
        pcap_close(handle);
        return 0;
        //Close the handle
        }
```

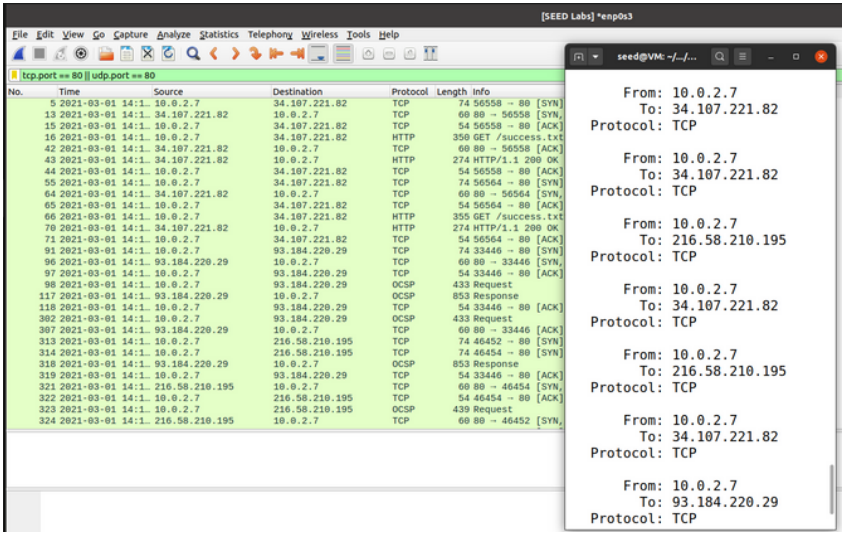
Spoofed succesfully with the filter



- **Capture the TCP packets with a destination port number in the range from 10 to 100.**

We can capture only the TCP packets with a destination port number in the given range from 10 to 100 by modifying the filter_exp[] string this way:

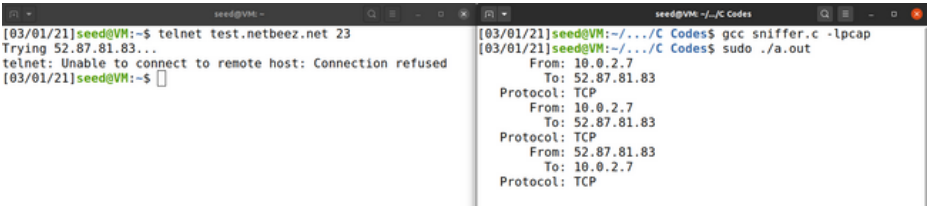
```
int main()
        {
        pcap_t *handle;
        char errbuf[PCAP_ERRBUF_SIZE];
        struct bpf_program fp;

        // TCP packets with dest port 10-100
        char filter_exp[] = "tcp dst portrange 10-100";


        bpf_u_int32 net;
        // Step 1: Open live pcap session on NIC with name eth3
        //Students needs to change "eth3" to the name
        //found on their own machines (using ifconfig).
        handle = pcap_open_live("enp0s3", BUFSIZ, 1, 1000, errbuf);
        // Step 2: Compile filter_exp into BPF psuedo-code
        pcap_compile(handle, &fp, filter_exp, 0, net);
        pcap_setfilter(handle, &fp);
        // Step 3: Capture packets
        pcap_loop(handle, -1, got_packet, NULL);
        pcap_close(handle);
        return 0;
        //Close the handle
        }
```

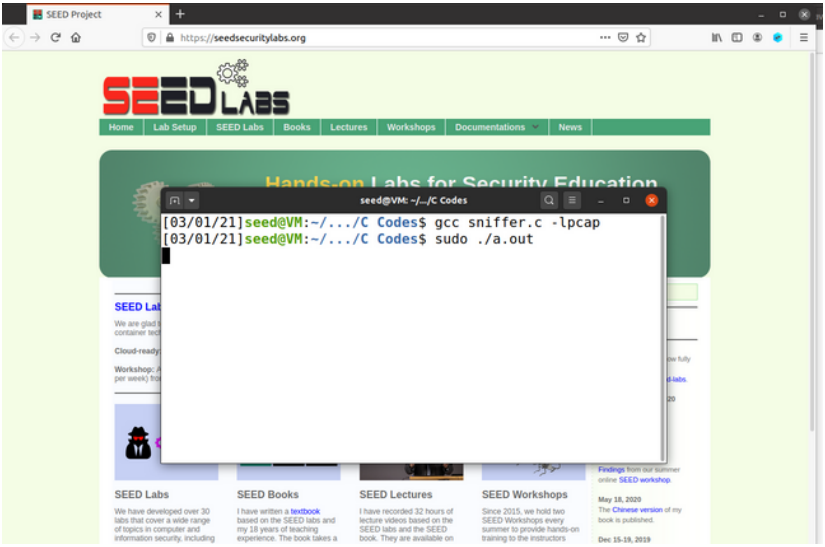


Task 2.1C: Sniffing Passwords

```c
int main()
{
    pcap_t *handle;
    char errbuf[PCAP_ERRBUF_SIZE];
    struct bpf_program fp;


    // TCP packets port 23
    char filter_exp[] = "tcp port 23";

    bpf_u_int32 net;

    // Step 1: Open live pcap session on NIC with name enp0s3
    handle = pcap_open_live("enp0s3", BUFSIZ, 1, 1000, errbuf);

    // Step 2: Compile filter_exp into BPF pseudo-code
    pcap_compile(handle, &fp, filter_exp, 0, net);
    pcap_setfilter(handle, &fp);

    // Step 3: Capture packets
    pcap_loop(handle, -1, got_packet, NULL);

    pcap_close(handle); //Close the handle
    return 0;
}
```

Packets are well sniffed when using telnet (port 23) as expected





Nothing is captured when opening Firefox (since port 23 isn't used) as expected (Indeed, HTTP use port 80)

4.2 Task 2.2: Spoofing

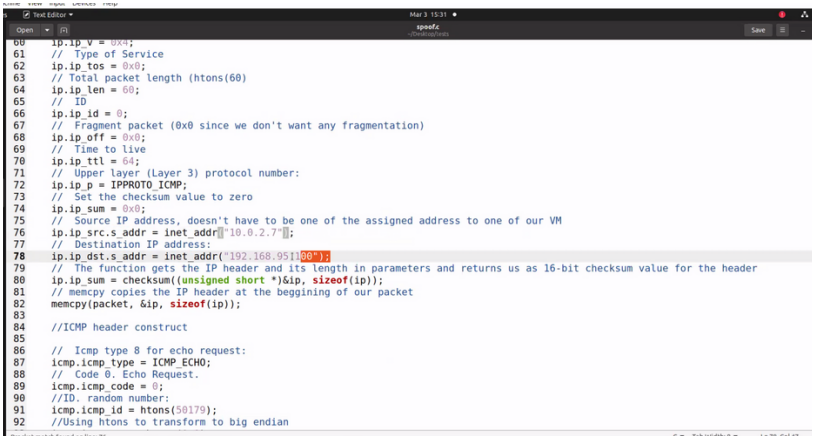
When a normal user sends out a packet, operating systems usually do not allow the user to set all the fields in the protocol headers (such as TCP, UDP, and IP headers). OSes will set most of the fields, while only allowing users to set a few fields, such as the destination IP address, the destination port number, etc.

However, if users have the root privilege, they can set any arbitrary field in the packet headers. This is called packet spoofing, and it can be done through raw sockets.

Raw sockets give programmers the absolute control over the packet construction, allowing programmers to construct any arbitrary packet, including setting the header fields and the payload. Using raw sockets is quite straightforward; it involves four steps: (1) create a raw socket, (2) set socket option, (3) construct the packet, and (4) send out the packet through the raw socket.

Task 2.2.A: Write a spoofing program.

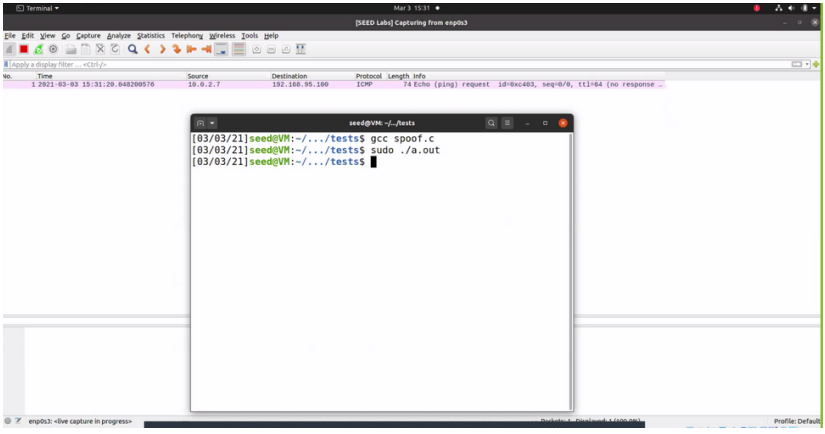
Spoof Code



```
60    ip.ip_v = 0x4;
61    //  Type of Service
62    ip.ip_tos = 0x0;
63    // Total packet length (htons(60)
64    ip.ip_len = 60;
65    //  ID
66    ip.ip_id = 0;
67    //  Fragment packet (0x0 since we don't want any fragmentation)
68    ip.ip_off = 0x0;
69    //  Time to live
70    ip.ip_ttl = 64;
71    //  Upper layer (Layer 3) protocol number:
72    ip.ip_p = IPPROTO_ICMP;
73    //  Set the checksum value to zero
74    ip.ip_sum = 0x0;
75    //  Source IP address, doesn't have to be one of the assigned address to one of our VM
76    ip.ip_src.s_addr = inet_addr("10.0.2.7");
77    //  Destination IP address:
78    ip.ip_dst.s_addr = inet_addr("192.168.95.100");
79    //  The function gets the IP header and its length in parameters and returns us as 16-bit checksum value for the header
80    ip.ip_sum = checksum((unsigned short *)&ip, sizeof(ip));
81    // memcpy copies the IP header at the beggining of our packet
82    memcpy(packet, &ip, sizeof(ip));
83
84    //ICMP header construct
85
86    //  Icmp type 8 for echo request:
87    icmp.icmp_type = ICMP_ECHO;
88    //  Code 0. Echo Request.
89    icmp.icmp_code = 0;
90    //ID. random number:
91    icmp.icmp_id = htons(50179);
92    //Using htons to transform to big endian
```

Task 2.2.B: Spoof an ICMP Echo Request.

Packet successfully spoofed:



• **Question 4.** Can you set the IP packet length field to an arbitrary value, regardless of how big the actual packet is?

The length field in this case should be the length of the IP packet, if not the sendto method (or function) will show an error 'Invalid Argument'. So yes, it is important that it is the length of the IP packet.

• **Question 5.** Using the raw socket programming, do you have to calculate the checksum for the IP header?

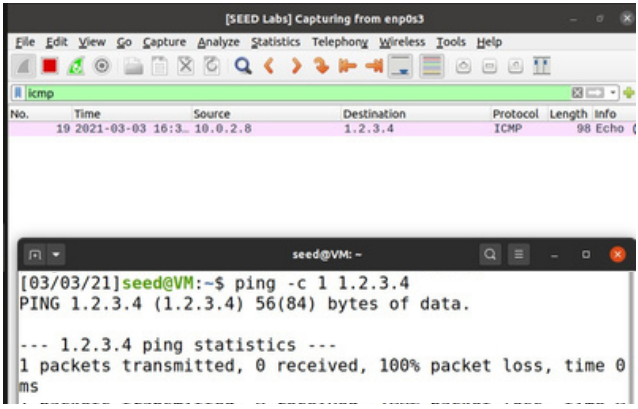
No you don't need to calculate the checksum, it will be filled by the system.

• **Question 6.** Why do you need the root privilege to run the programs that use raw sockets? Where does the program fail if executed without the root privilege?

It's restricted to root because it would break some rules for networking that are in place, without root you can't bind a port lower than 1024. With raw sockets you can simulate a server on any port and spoof custom packets which can interfere with inbound traffic

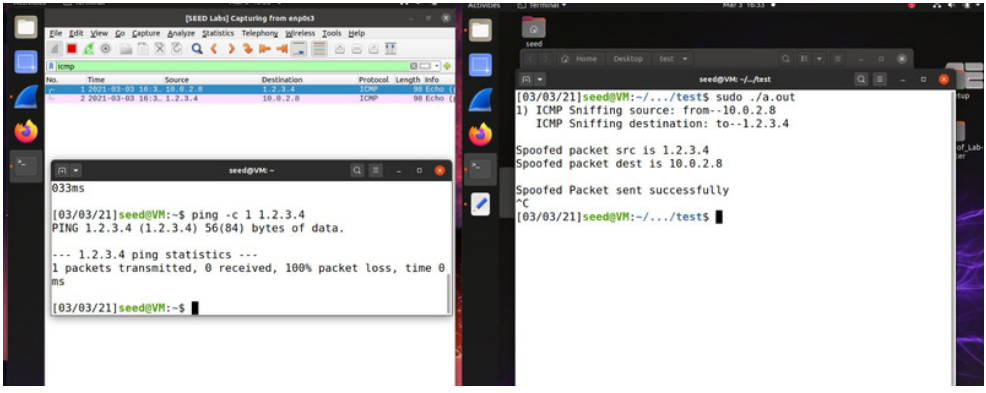
4.3 Task 2.3: Sniff and then Spoof

Sending a ping request to a non-existing host (1.2.3.4) before running sniffandspoof.c



no answer

Sending a ping request to a non-existing host (1.2.3.4) after running sniffandspoof.c



We can see that Wireshark displays a successful ping request even though no packet were really sent (100% packet loss), and in the attacker terminal we can also see that the source IP and the destination IP have been successfully swapped, meaning that the spoofing was successful.