

תרגיל 1 – מבוא לרשתות תקשורת

חלק א'

לפני סינון

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	UDP	45	56776 → 1234 Len=3
2	0.000055	127.0.0.1	127.0.0.1	UDP	45	1234 → 56776 Len=3
3	4.893164	127.0.0.1	224.0.0.251	MDNS	82	Standard query 0x0000 PTR_pgkey-hkp_tcp.local, "qm" question
4	6.264230	127.0.0.1	127.0.0.1	UDP	52	56776 → 1234 Len=10
5	6.264323	127.0.0.1	127.0.0.1	UDP	52	1234 → 56776 Len=10

2. ביצענו סינון בעזרת סנן ה-bookmarks בתוכנת Wireshark. כיוון שאנו יודעים שהתעבורה שאנו מחפשים (התעבורה שלנו) עושה שימוש בפרוטוקול UDP החלטנו לסנן לפיו וקיבלנו את התוצאה הרצויה.

אחרי סינון

Wireshark interface showing packet capture results after filtering for UDP. The packet list shows four packets, all UDP, with the first packet selected. The packet details pane shows the structure of the first packet: Ethernet II, Internet Protocol Version 4, and User Datagram Protocol (Source Port: 56776, Destination Port: 1234). The packet bytes pane shows the raw data in hexadecimal and ASCII.

3. כפי שנאמר בקוד שכתבנו יש שימוש במספרי פורט הן בקוד הלקוח והן בקוד השרת (שורה 5 בשניהם).

- 3.1. בקוד הלקוח – אנו מגדירים לאיזה פורט (ואיזה IP) יש לשלוח את המידע המבוקש.
- 3.2. בקוד השרת – אנו מגדירים פורט ספציפי ועושים קישור בינו לבין הסוקט על מנת לקבל את המידע. פעולת ה-bind קושרת בין הפורט שלנו לסוקט כלשהו, כעת כל תעבורה שתשלח לפורט 1234 (לפי הקוד שלנו) תגיע דרך הסוקט שהגדרנו אלינו.

הדבר בא לידי ביטוי בחבילה בשכבת התעבורה, ניתן לראות את הסגמנט שבו מצוי מספר הפורט צבוע בכחול בעזרת תוכנת Wireshark.

Wireshark interface showing packet capture results after filtering for UDP. The packet list shows four packets, all UDP, with the first packet selected. The packet details pane shows the structure of the first packet: Ethernet II, Internet Protocol Version 4, and User Datagram Protocol (Source Port: 56776, Destination Port: 1234). The packet bytes pane shows the raw data in hexadecimal and ASCII.

Two code editors showing Python code for a client and a server. The client code (Client.py) sends a message to the server. The server code (Server.py) receives the message and prints it.

The image shows a Wireshark packet capture analysis of an ICMP Echo (ping) request. The packet list shows a packet from 10.0.0.0 to 127.0.0.1. The packet details pane shows the Ethernet II, Internet Protocol Version 4, and User Datagram Protocol (UDP) fields. The packet bytes pane shows the raw data in hexadecimal and ASCII.

Packet List:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	UDP	45	56776 → 1234 Len=3
2	0.000055...	127.0.0.1	127.0.0.1	UDP	45	1234 → 56776 Len=3
4	6.264230...	127.0.0.1	127.0.0.1	UDP	52	56776 → 1234 Len=10
5	6.264423...	127.0.0.1	127.0.0.1	UDP	52	1234 → 56776 Len=10

Packet Details:

- Frame 1: 45 bytes on wire (360 bits), 45 bytes captured (360 bits) on interface 0
- Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
 - 0100 = Version: 4
 - 0101 = Header Length: 20 bytes (5)
 - Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 - Total Length: 31
 - Identification: 0x2e6c (11884)
 - Flags: 0x4000, Don't fragment
 - Time to live: 64
 - Protocol: UDP (17)
 - Header checksum: 0x0e00 [validation disabled]
 - [Header checksum status: Unverified]
 - Source: 127.0.0.1
 - Destination: 127.0.0.1
- User Datagram Protocol, Src Port: 56776, Dst Port: 1234
- Data (3 bytes)

Packet Bytes:

Offset	Hex	ASCII
0000	00 00 00 00 00 00 00 00E..
0010	00 1f 2e 6c 40 00 00 11	...l@... ..
0020	00 01 dd c8 04 d2 00 0b	...hey

```
File Edit View Search Terminal Help
matan@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.74.129 netmask 255.255.255.0  broadcast 192.168.74.255
    inet6 fe80::39f0:6b87:da48:8a50 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:4c:24:03 txqueuelen 1000 (Ethernet)
    RX packets 20681 bytes 22059257 (22.0 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 11811 bytes 1516768 (1.5 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 590 bytes 45949 (45.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 590 bytes 45949 (45.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

חלק ב'

1. הרצנו את השרת 3-ו לקוחות: ספיר, מתן וחמי. שלחנו הודעות מגוונות ביניהם בין הלקוחות לשרת. במקביל פתחנו Wireshark והסנפנו את התעבורה כמו כן, ביצענו סינון בעזרת סנן ה- bookmarks בתוכנת Wireshark. כיוון שאנו יודעים שהתעבורה שאנו מחפשים עושה שימוש בפרוטוקול UDP וה- port הוא 1234 - החלטנו לסנן לפיו וקיבלנו את התוצאה הרצויה:

The image shows a Kali Linux desktop environment with three windows open:

- Terminal 1 (Left):** A chat conversation between Sapir, Matan, and Hemi. Sapir is the host, and Matan and Hemi are clients. The chat includes greetings and a discussion about a model.
- Terminal 2 (Right):** A terminal window showing the execution of a python3 script. The script attempts to connect to a server, but it fails with an error: "can't open file 'cl': [Errno 2] No such file or directory".
- Wireshark (Right):** A network packet capture tool showing a list of captured packets. The selected packet (123) is a User Datagram Protocol (UDP) packet from 127.0.0.1 to 127.0.0.1, port 1234. The packet details show the source and destination IP addresses and ports, and the protocol used (UDP).

2. לאחר הסינון ועצירת הwireshark, נראה כיצד הפעולות שעשינו יצרו חבילות

הרשמה של חמי:

הלקוח שלח לשרת את החבילה עם ה-source_port שמערכת ההפעלה הקצתה לו (השתמשנו ב-source port של הלקוחות כדי להבדיל ביניהם)

החבילה המפורטת נמצאת כאן ובפרט גם ה-data

אפשר לראות את הביטים שהוקצו ל-data ואת ההודעה עצמה שהשרת קיבל

No.	Time	Source	Destination	Protocol	Length	Info
123	203.6191...	127.0.0.1	127.0.0.1	UDP	51	56948 → 1234 Len=7
124	203.6192...	127.0.0.1	127.0.0.1	UDP	44	1234 → 56948 Len=0
125	216.8076...	127.0.0.1	127.0.0.1	UDP	51	44067 → 1234 Len=7
126	216.8077...	127.0.0.1	127.0.0.1	UDP	49	1234 → 44067 Len=5
135	237.9392...	127.0.0.1	127.0.0.1	UDP	54	44067 → 1234 Len=10
136	237.9396...	127.0.0.1	127.0.0.1	UDP	44	1234 → 44067 Len=0
138	245.2098...	127.0.0.1	127.0.0.1	UDP	54	44067 → 1234 Len=10
139	245.2099...	127.0.0.1	127.0.0.1	UDP	44	1234 → 44067 Len=0
140	254.8207...	127.0.0.1	127.0.0.1	UDP	50	41132 → 1234 Len=6
141	254.8208...	127.0.0.1	127.0.0.1	UDP	50	1234 → 41132 Len=12
142	274.2763...	127.0.0.1	127.0.0.1	UDP	71	41132 → 1234 Len=27
143	274.2764...	127.0.0.1	127.0.0.1	UDP	44	1234 → 41132 Len=0
144	283.0328...	127.0.0.1	127.0.0.1	UDP	45	56948 → 1234 Len=1
145	283.0329...	127.0.0.1	127.0.0.1	UDP	44	1234 → 56948 Len=0

Frame 140: 50 bytes on wire (400 bits), 50 bytes captured (400 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
User Datagram Protocol, Src Port: 41132, Dst Port: 1234
Source Port: 41132
Destination Port: 1234
Length: 14
Checksum: 0xfe21 [unverified]
[Checksum Status: Unverified]
[Stream index: 16]
Data (6 bytes)
Data: 312048656d69
[Length: 6]

0000 00 00 03 04 00 06 00 00 00 00 00 00 53 65 08 00Se..
0010 45 00 00 03 00 06 00 00 40 11 3c 35 7f 00 00 01 E..(..@. @.<5..
0020 7f 00 00 01 a0 ac 04 d2 00 0e fe 21 31 20 48 65!..He..
0030 6d 69m..

קבלת שמות המשתמשים:

אחרי ההרשמה הלקוח חמי מקבל מהשרת את שמות המשתמשים. אפשר לראות שהשרת שולח ללקוח הרלוונטי עם ה-port שאנחנו הקצנו לו - 1234

No.	Time	Source	Destination	Protocol	Length	Info
123	203.6191...	127.0.0.1	127.0.0.1	UDP	51	56948 → 1234 Len=7
124	203.6192...	127.0.0.1	127.0.0.1	UDP	44	1234 → 56948 Len=0
125	216.8076...	127.0.0.1	127.0.0.1	UDP	51	44067 → 1234 Len=7
126	216.8077...	127.0.0.1	127.0.0.1	UDP	49	1234 → 44067 Len=5
135	237.9392...	127.0.0.1	127.0.0.1	UDP	54	44067 → 1234 Len=10
136	237.9396...	127.0.0.1	127.0.0.1	UDP	44	1234 → 44067 Len=0
138	245.2098...	127.0.0.1	127.0.0.1	UDP	54	44067 → 1234 Len=10
139	245.2099...	127.0.0.1	127.0.0.1	UDP	44	1234 → 44067 Len=0
140	254.8207...	127.0.0.1	127.0.0.1	UDP	50	41132 → 1234 Len=6
141	254.8208...	127.0.0.1	127.0.0.1	UDP	56	1234 → 41132 Len=12
142	274.2763...	127.0.0.1	127.0.0.1	UDP	71	41132 → 1234 Len=27
143	274.2764...	127.0.0.1	127.0.0.1	UDP	44	1234 → 41132 Len=0
144	283.0328...	127.0.0.1	127.0.0.1	UDP	45	56948 → 1234 Len=1
145	283.0329...	127.0.0.1	127.0.0.1	UDP	44	1234 → 56948 Len=0

Frame 141: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
User Datagram Protocol, Src Port: 1234, Dst Port: 41132
Source Port: 1234
Destination Port: 41132
Length: 20
Checksum: 0xfe27 [unverified]
[Checksum Status: Unverified]
[Stream index: 16]
Data (12 bytes)
Data: 53617069722c204d6174616e
[Length: 12]

0000 00 00 03 04 00 06 00 00 00 00 00 00 08 00 08 00
0010 45 00 00 28 00 0e 40 00 40 11 3c 35 7f 00 00 01 E..(..@. @.<5..
0020 7f 00 00 01 04 d2 a0 ac 00 14 fe 27 53 61 70 69Sap1
0030 72 2c 20 4d 61 74 61 6er, Matan