Internet Of Things - Second Challenge

Authors:
Kevin Abate: 10812892
Lorenzo Pigato: 10766953 [Team Leader]

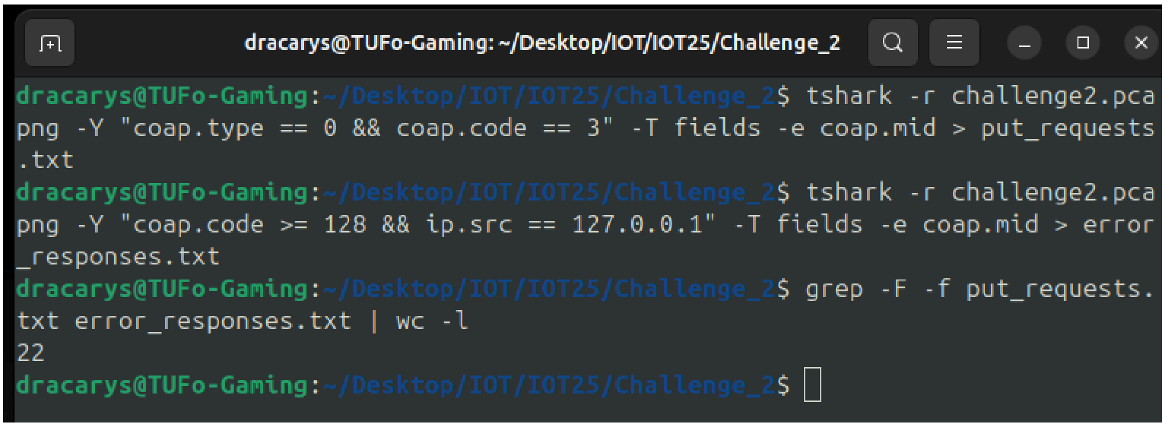
Project topic

The project consist on the **analysis** and **sniffing** of the packets captured on the "challenge2.pcap", the analysis have been done through Wireshark and TShark.

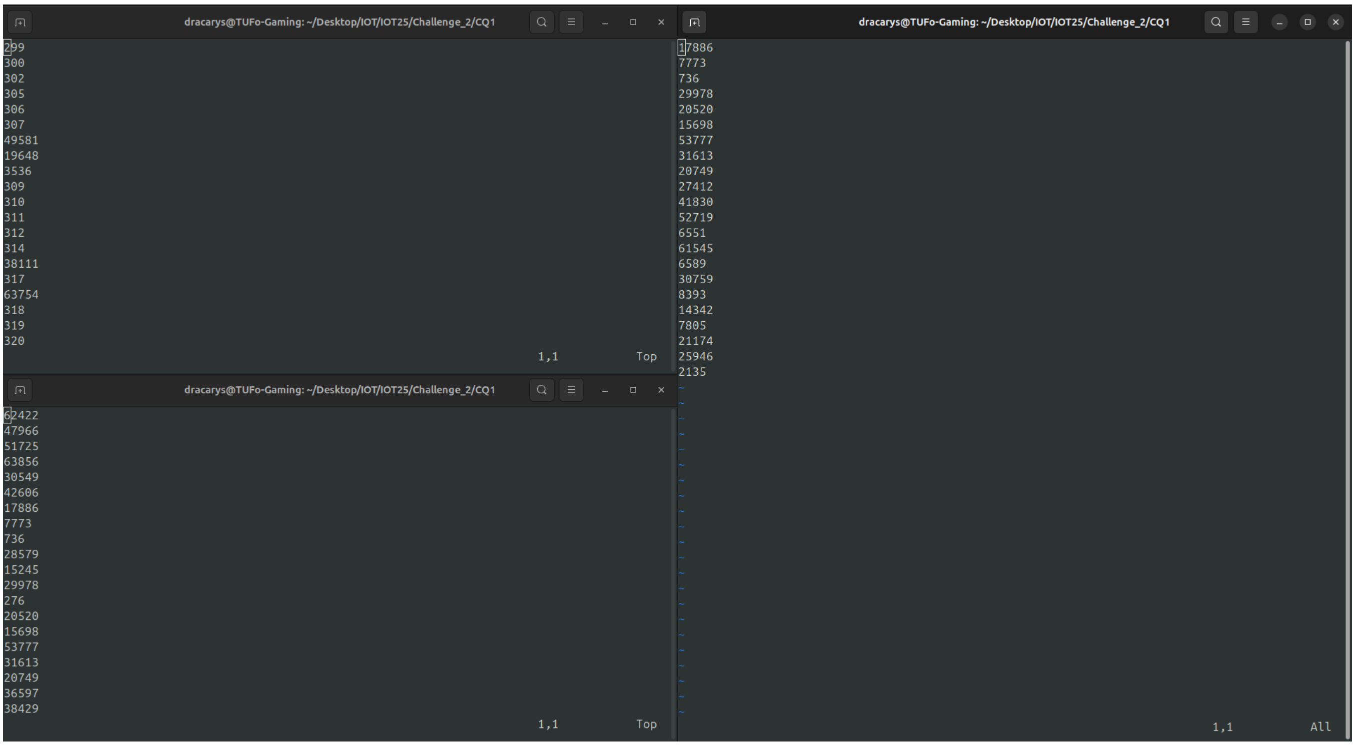
Questions

CQ1

"How many different Confirmable PUT requests obtained an unsuccessful response from the local CoAP server?"
Answer: 22
The answer is given by filtering before all the **CONfirmbale PUT requests**, and after that checked the **error response** sent from the server. Due the fact that's not possible to do combination of filter in Wireshark, to deal with this problem we used TShark.
Below it's possible to see the filters in TShark and the consequent result.



```
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ tshark -r challenge2.pca
png -Y "coap.type == 0 && coap.code == 3" -T fields -e coap.mid > put_requests
.txt
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ tshark -r challenge2.pca
png -Y "coap.code >= 128 && ip.src == 127.0.0.1" -T fields -e coap.mid > error
_responses.txt
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ grep -F -f put_requests.
txt error_responses.txt | wc -l
22
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ 
```


dracarys@TUFo-Gaming: ~/Desktop/IOT/IOT25/Challenge_2/CQ1
dracarys@TUFo-Gaming: ~/Desktop/IOT/IOT25/Challenge_2/CQ1
dracarys@TUFo-Gaming: ~/Desktop/IOT/IOT25/Challenge_2/CQ1

```
299
300
302
305
306
307
49581
19648
3536
309
310
311
312
314
38111
317
63754
318
319
320
                                                  1,1           Top
```

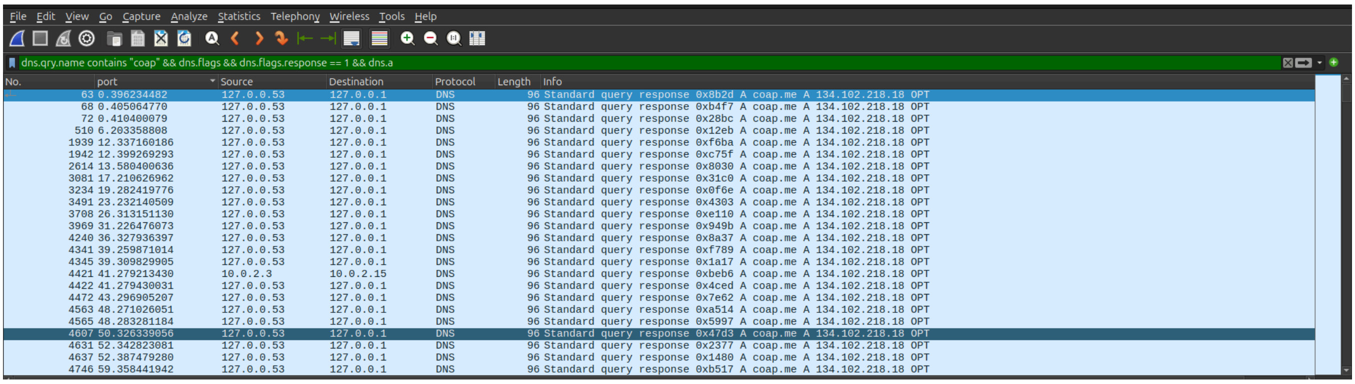
```
62422
47966
51725
63856
30549
42606
17886
7773
736
28579
15245
29978
276
20520
15698
53777
31613
20749
36597
38429
                                                  1,1           Top
```

```
17886
7773
736
29978
20520
15698
53777
31613
20749
27412
41830
52719
6551
61545
6589
30759
8393
14342
7805
21174
25946
2135
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                  1,1           All
```

Above it's possible to see the the matching between the MID of the several packets.

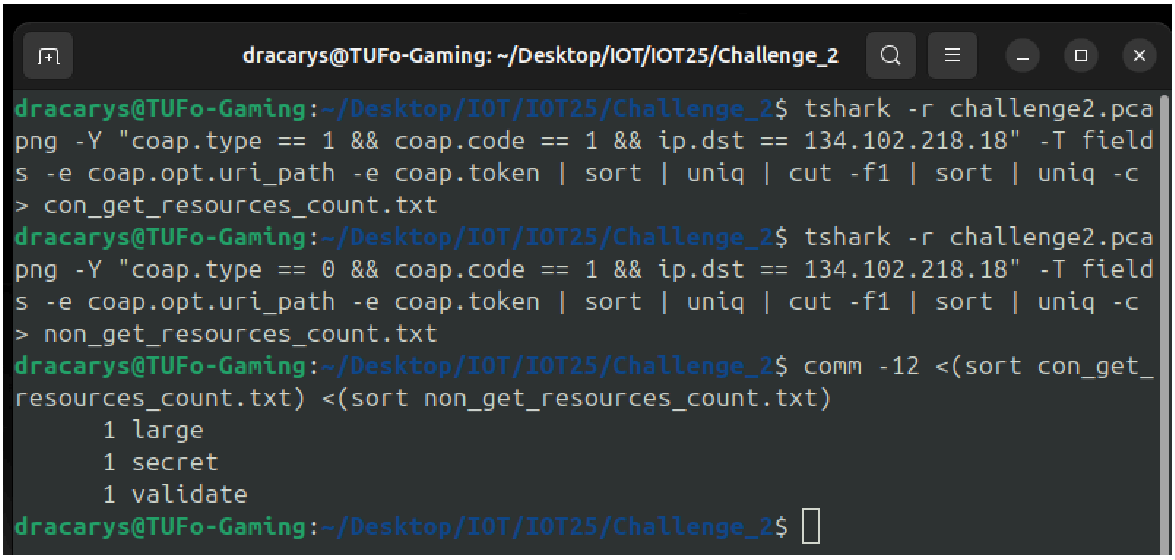
CQ2

"How many CoAP resources in the coap.me public server received the
same number of unique Confirmable and Non Confirmable GET requests?"
Assuming a resource receives X different CONFIRMABLE requests and Y different
NONCONFIRMABLE GET requests, how many resources have X=Y, with X>0?
Answer: 3 [large, secret, validate]
In order to solve the problem, before all on Wireshark has been done an analysis to find out the **address** of the "coap.me" server


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
dns.qry.name contains "coap" && dns.flags && dns.flags.response == 1 && dns.a
No.       port            Source        Destination  Protocol  Length Info
  63 0.396234482    127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x8b2d A coap.me A 134.102.218.18 OPT
  68 0.405064770    127.0.0.53    127.0.0.1    DNS         96 Standard query response 0xb4f7 A coap.me A 134.102.218.18 OPT
  72 0.410400079    127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x28bc A coap.me A 134.102.218.18 OPT
 510 6.203358808    127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x12eb A coap.me A 134.102.218.18 OPT
1939 12.337160186   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0xf6ba A coap.me A 134.102.218.18 OPT
1942 12.399269293   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0xc75f A coap.me A 134.102.218.18 OPT
2614 13.580400636   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x8030 A coap.me A 134.102.218.18 OPT
3081 17.210626962   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x31c0 A coap.me A 134.102.218.18 OPT
3234 19.282419776   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x0f6e A coap.me A 134.102.218.18 OPT
3491 23.232140509   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x4303 A coap.me A 134.102.218.18 OPT
3708 26.313151130   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0xe110 A coap.me A 134.102.218.18 OPT
3969 31.226476073   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x949b A coap.me A 134.102.218.18 OPT
4240 36.327936397   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x8a37 A coap.me A 134.102.218.18 OPT
4341 39.259871014   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0xf789 A coap.me A 134.102.218.18 OPT
4345 39.309829905   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x1a17 A coap.me A 134.102.218.18 OPT
4421 41.279213430   10.0.2.3      10.0.2.15    DNS         96 Standard query response 0xbeb6 A coap.me A 134.102.218.18 OPT
4422 41.279430031   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x4ced A coap.me A 134.102.218.18 OPT
4472 43.296905207   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x7e62 A coap.me A 134.102.218.18 OPT
4563 48.271026051   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0xa514 A coap.me A 134.102.218.18 OPT
4565 48.283281184   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x5997 A coap.me A 134.102.218.18 OPT
4607 50.326339056   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x47d3 A coap.me A 134.102.218.18 OPT
4631 52.342823081   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x2377 A coap.me A 134.102.218.18 OPT
4637 52.387479280   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0x1480 A coap.me A 134.102.218.18 OPT
4746 59.358441942   127.0.0.53    127.0.0.1    DNS         96 Standard query response 0xb517 A coap.me A 134.102.218.18 OPT
```

which is possible to see that the coap.me ip address is: 134.102.218.18 . After that on TShark has been done the filters and comparison to find out the result.

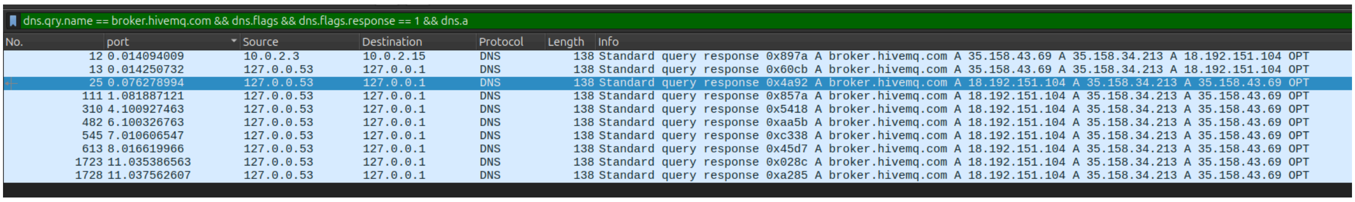


```
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ tshark -r challenge2.pca
png -Y "coap.type == 1 && coap.code == 1 && ip.dst == 134.102.218.18" -T field
s -e coap.opt.uri_path -e coap.token | sort | uniq | cut -f1 | sort | uniq -c
> con_get_resources_count.txt
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ tshark -r challenge2.pca
png -Y "coap.type == 0 && coap.code == 1 && ip.dst == 134.102.218.18" -T field
s -e coap.opt.uri_path -e coap.token | sort | uniq | cut -f1 | sort | uniq -c
> non_get_resources_count.txt
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ comm -12 <(sort con_get_
resources_count.txt) <(sort non_get_resources_count.txt)
      1 large
      1 secret
      1 validate
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ 
```

In the first filter will be returned the **GET NON-confirmable packets**, with an IP destination equals to the coap.me server. The second does the same but with the **CONfirmable packets**. At the end matches the outputs of the two previously obtained files and returns the common rows.

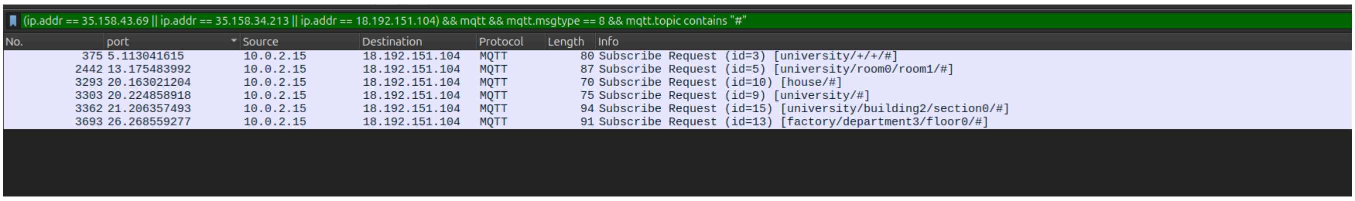
CQ3

"How many different MQTT clients subscribe to the public broker
HiveMQ using multi-level wildcards?"
Answer: 4 [38619- 38641-54449-57863]
Before all it has been made a filter on Wireshark to find out the **address** of the HiveMQ broker, as is possible to see below.



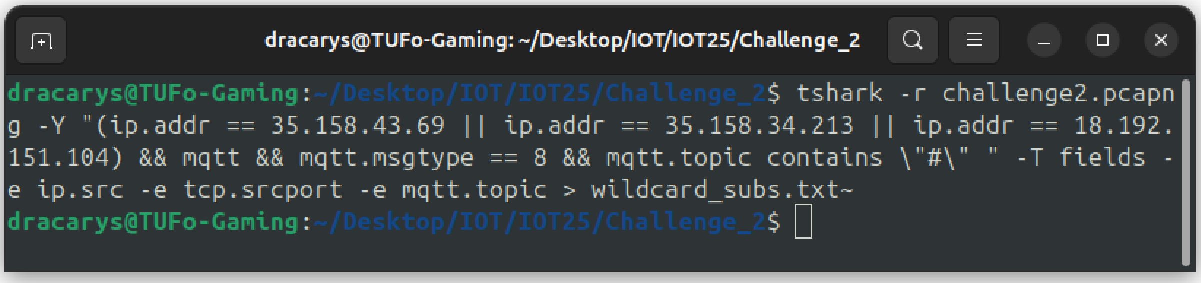
```
dns.qry.name == broker.hivemq.com && dns.flags && dns.flags.response == 1 && dns.a
No.       port            Source        Destination  Protocol  Length Info
  12 0.014094009    10.0.2.3      10.0.2.15    DNS        138 Standard query response 0x897a A broker.hivemq.com A 35.158.43.69 A 35.158.34.213 A 18.192.151.104 OPT
  13 0.014250732    127.0.0.53    127.0.0.1    DNS        138 Standard query response 0x60cb A broker.hivemq.com A 35.158.43.69 A 35.158.34.213 A 18.192.151.104 OPT
  25 0.076278994    127.0.0.53    127.0.0.1    DNS        138 Standard query response 0x4a92 A broker.hivemq.com A 18.192.151.104 A 35.158.34.213 A 35.158.43.69 OPT
 111 1.081887121    127.0.0.53    127.0.0.1    DNS        138 Standard query response 0x857a A broker.hivemq.com A 18.192.151.104 A 35.158.34.213 A 35.158.43.69 OPT
 310 4.100927463    127.0.0.53    127.0.0.1    DNS        138 Standard query response 0x5418 A broker.hivemq.com A 18.192.151.104 A 35.158.34.213 A 35.158.43.69 OPT
 482 6.100326763    127.0.0.53    127.0.0.1    DNS        138 Standard query response 0xaa5b A broker.hivemq.com A 18.192.151.104 A 35.158.34.213 A 35.158.43.69 OPT
 545 7.010606547    127.0.0.53    127.0.0.1    DNS        138 Standard query response 0xc338 A broker.hivemq.com A 18.192.151.104 A 35.158.34.213 A 35.158.43.69 OPT
 613 8.016619966    127.0.0.53    127.0.0.1    DNS        138 Standard query response 0x45d7 A broker.hivemq.com A 18.192.151.104 A 35.158.34.213 A 35.158.43.69 OPT
1723 11.035386563   127.0.0.53    127.0.0.1    DNS        138 Standard query response 0x028c A broker.hivemq.com A 18.192.151.104 A 35.158.34.213 A 35.158.43.69 OPT
1728 11.037562607   127.0.0.53    127.0.0.1    DNS        138 Standard query response 0xa285 A broker.hivemq.com A 18.192.151.104 A 35.158.34.213 A 35.158.43.69 OPT
```

There are **three IP addresses** used to connect to the broker. After that, MQTT subscribe messages sent to these IPs were analyzed to identify topics that contain the **multi-level wildcard**.



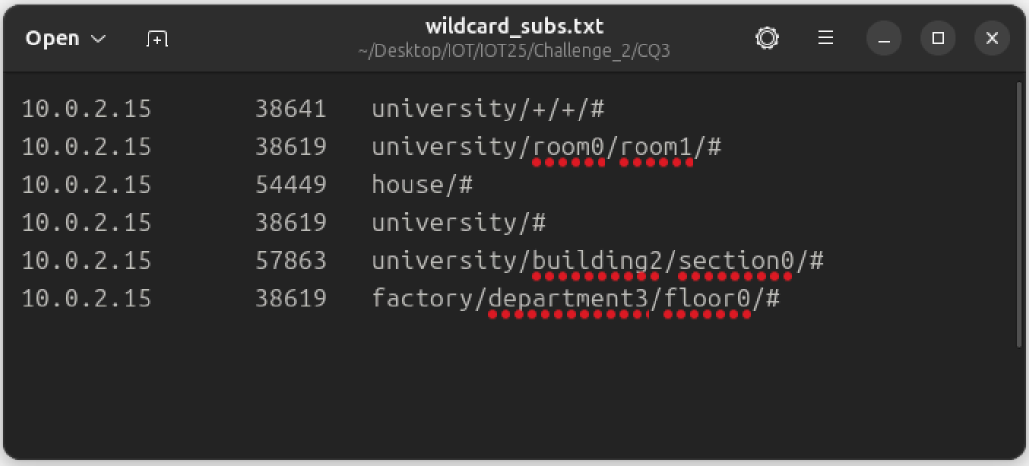
```
(ip.addr == 35.158.43.69 || ip.addr == 35.158.34.213 || ip.addr == 18.192.151.104) && mqtt && mqtt.msgtype == 8 && mqtt.topic contains "#"
No.      port             Source        Destination       Protocol  Length  Info
     375 5.113041615      10.0.2.15     18.192.151.104    MQTT          80 Subscribe Request (id=3) [university/+/+/#]
    2442 13.175483992     10.0.2.15     18.192.151.104    MQTT          87 Subscribe Request (id=5) [university/room0/room1/#]
    3293 20.163021204     10.0.2.15     18.192.151.104    MQTT          70 Subscribe Request (id=10) [house/#]
    3303 20.224858918     10.0.2.15     18.192.151.104    MQTT          75 Subscribe Request (id=9) [university/#]
    3362 21.206357493     10.0.2.15     18.192.151.104    MQTT          94 Subscribe Request (id=15) [university/building2/section0/#]
    3693 26.268559277     10.0.2.15     18.192.151.104    MQTT          91 Subscribe Request (id=13) [factory/department3/floor0/#]
```

The clients **differ** from each other by the **TCP port**, making it easy to compute the solution at this point.
It's easy to check the solution on TShark too



```
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ tshark -r challenge2.pcapn
g -Y "(ip.addr == 35.158.43.69 || ip.addr == 35.158.34.213 || ip.addr == 18.192.
151.104) && mqtt && mqtt.msgtype == 8 && mqtt.topic contains \"#\" " -T fields -
e ip.src -e tcp.srcport -e mqtt.topic > wildcard_subs.txt~
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ 
```

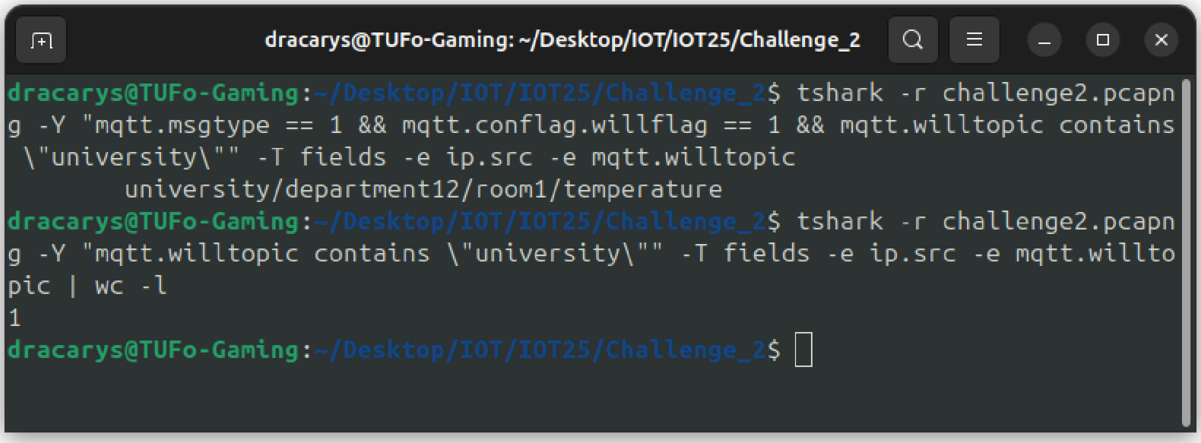
obtaining as a result the several clients, distinguished by their TCP port



```
10.0.2.15       38641   university/+/+/#
10.0.2.15       38619   university/room0/room1/#
10.0.2.15       54449   house/#
10.0.2.15       38619   university/#
10.0.2.15       57863   university/building2/section0/#
10.0.2.15       38619   factory/department3/floor0/#
```

CQ4

"How many different MQTT clients specify a last Will Message to be
directed to a topic having as first level "university" ?"
Answer: 1 [127.0.0.1 - university/department12/room1/temperature]
To solve this question is possible to use either Wireshark and TShark, will be showed
the TShark solution.
These conditions filters all the MQTT connect file, with the **willflag** on the **CONNACK** flags activated and a **Last Will Topic** that contains the string "university".

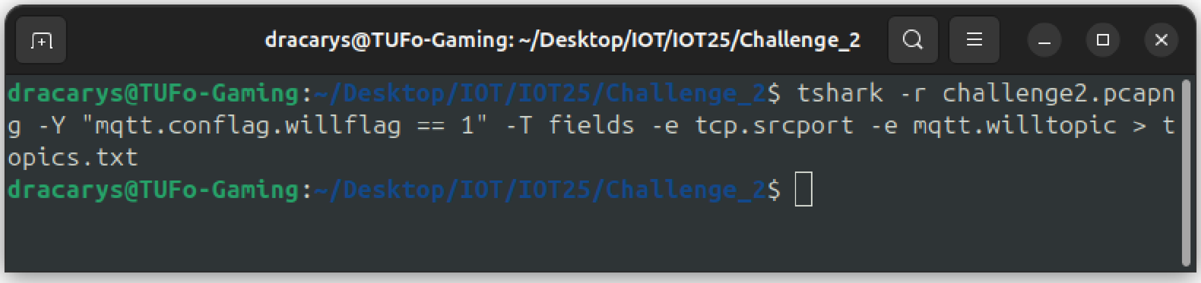


```
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ tshark -r challenge2.pcapn
g -Y "mqtt.msgtype == 1 && mqtt.conflag.willflag == 1 && mqtt.willtopic contains
 \"university\"" -T fields -e ip.src -e mqtt.willtopic
        university/department12/room1/temperature
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ tshark -r challenge2.pcapn
g -Y "mqtt.willtopic contains \"university\"" -T fields -e ip.src -e mqtt.willto
pic | wc -l
1
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ 
```

Note: In the first command, the Will Topic is shown for better clarity.

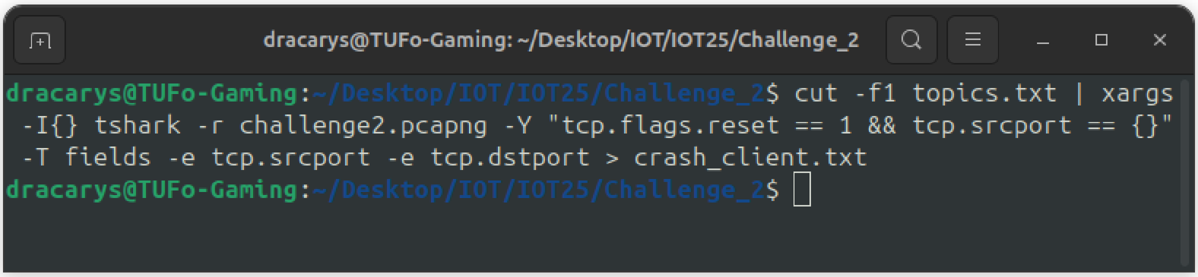
CQ5

"How many MQTT subscribers receive a last will message derived from a subscription without a wildcard?"
Answer: 3 [39551-53557-41789]
The answer for this question is divided in five parts.
At the beginning are searched all the **distinct topics** that have a flag set for last will message and clients which set them.



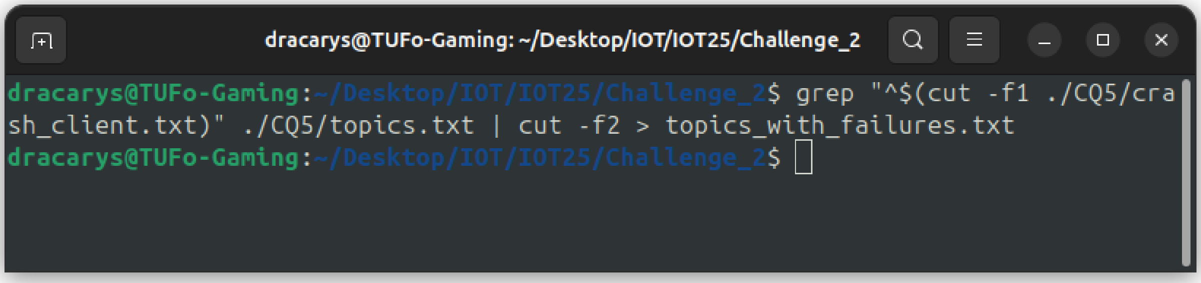
```
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ tshark -r challenge2.pcapn
g -Y "mqtt.conflag.willflag == 1" -T fields -e tcp.srcport -e mqtt.willtopic > t
opics.txt
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ 
```

output of the filter:
`38083 university/department12/room1/temperature 56285 metaverse/room2/floor4 53485 hospital/facility3/area3 42665 metaverse/room2/room2`
After that it's used this command that filters the connection reset packets by the TCP source port to determine if any **connection reset** has been sent by a client that has configured a **Last Will Topic** message.



```
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ cut -f1 topics.txt | xargs
 -I{} tshark -r challenge2.pcapng -Y "tcp.flags.reset == 1 && tcp.srcport == {}"
 -T fields -e tcp.srcport -e tcp.dstport > crash_client.txt
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ 
```

output of the filter:
`38083 1883`
Thus are matched the **Last Will Topic** with **failing clients**.

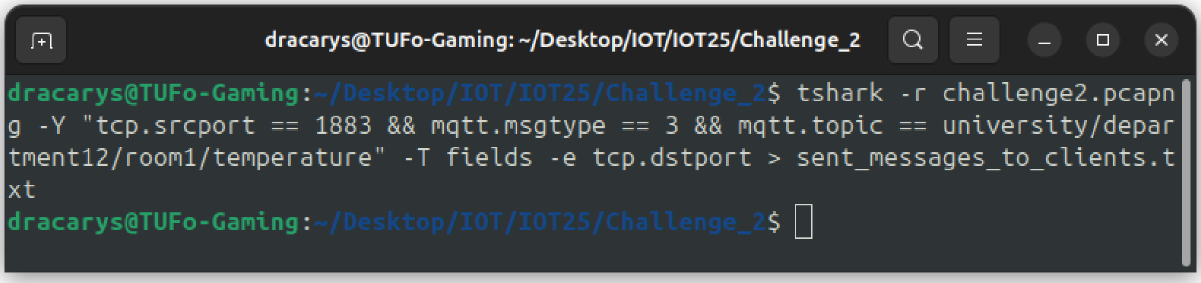


```
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ grep "^$(cut -f1 ./CQ5/crash_client.txt)" ./CQ5/topics.txt | cut -f2 > topics_with_failures.txt
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ 
```

output of the filter:

`university/department12/room1/temperature`

Now that we have the topic we can search the **messages** sent by the broker on the single topic found.

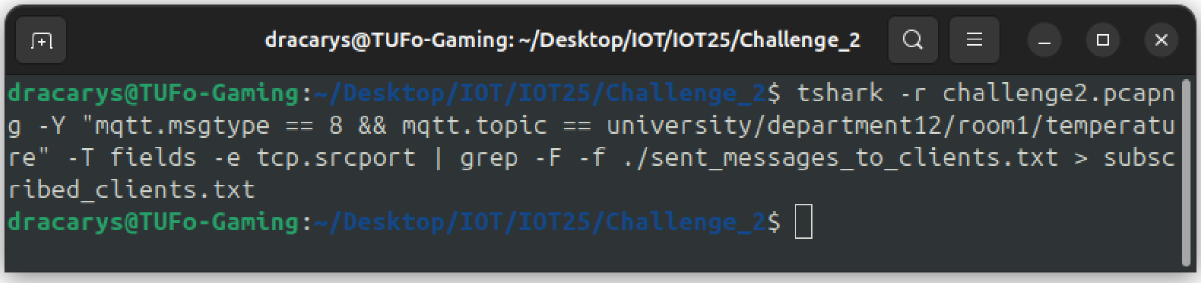


```
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ tshark -r challenge2.pcapng -Y "tcp.srcport == 1883 && mqtt.msgtype == 3 && mqtt.topic == university/department12/room1/temperature" -T fields -e tcp.dstport > sent_messages_to_clients.txt
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ 
```

output of the filter:

`39551 53557 51743 41789`

With clients IPs in hand we can finally check which of these received the message, subscribed to the topic without wildcards:



```
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ tshark -r challenge2.pcapng -Y "mqtt.msgtype == 8 && mqtt.topic == university/department12/room1/temperature" -T fields -e tcp.srcport | grep -F -f ./sent_messages_to_clients.txt > subscribed_clients.txt
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ 
```

obtaining as final answer to the question, the clients:

`39551 53557 41789`

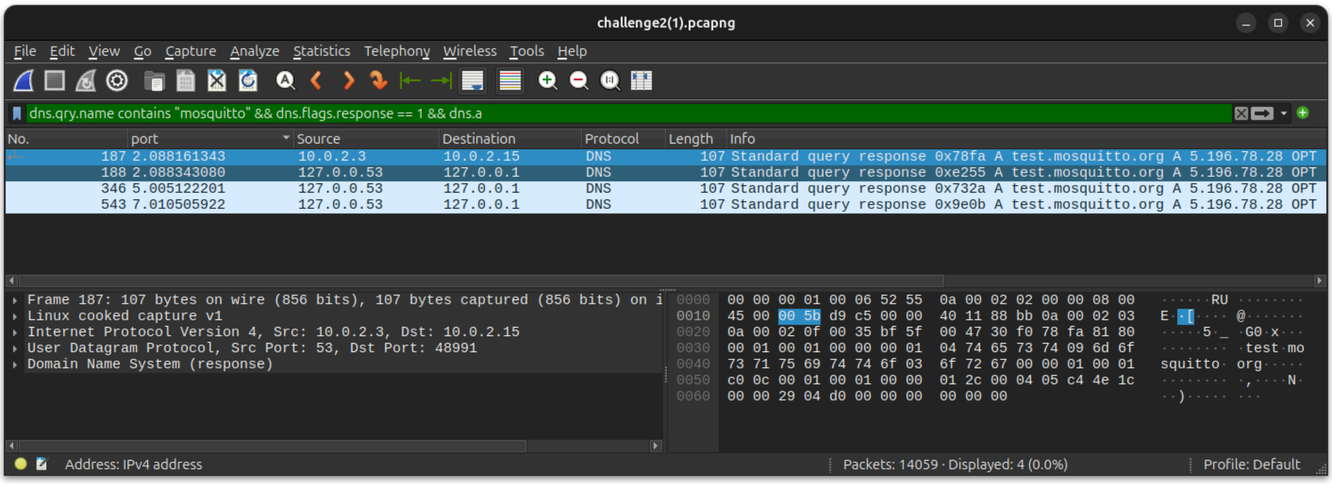
CQ6

"How many MQTT publish messages directed to the public broker Mosquitto are sent with the retain option and use QoS “At most once”? "

Answer: 208

To solve this question is possible to use either Wireshark and TShark, in this case will be showed the Wireshark solution, instead the TShark solution as proof.

Before all a filter was applied to find the address of the Mosquitto broker, as shown below


challenge2(1).pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
dns.qry.name contains "mosquitto" && dns.flags.response == 1 && dns.a
```

```
No.    port          Source       Destination  Protocol  Length  Info
   187 2.088161343   10.0.2.3     10.0.2.15    DNS        107 Standard query response 0x78fa A test.mosquitto.org A 5.196.78.28 OPT
   188 2.088343080   127.0.0.53   127.0.0.1    DNS        107 Standard query response 0xe255 A test.mosquitto.org A 5.196.78.28 OPT
   346 5.005122201   127.0.0.53   127.0.0.1    DNS        107 Standard query response 0x732a A test.mosquitto.org A 5.196.78.28 OPT
   543 7.010505922   127.0.0.53   127.0.0.1    DNS        107 Standard query response 0x9e0b A test.mosquitto.org A 5.196.78.28 OPT
```

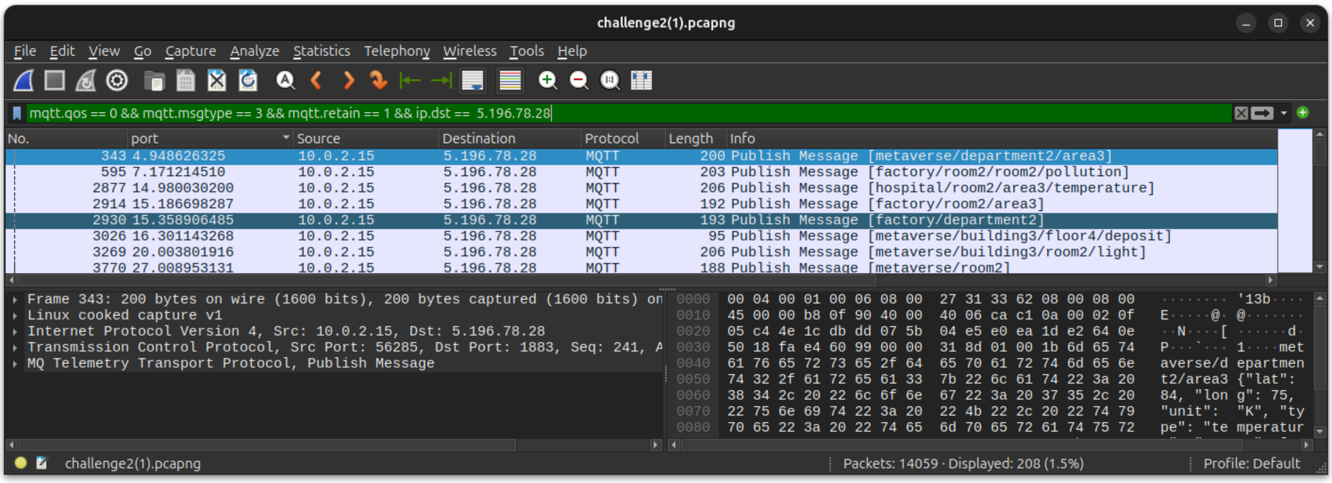
```
› Frame 187: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on i
› Linux cooked capture v1
› Internet Protocol Version 4, Src: 10.0.2.3, Dst: 10.0.2.15
› User Datagram Protocol, Src Port: 53, Dst Port: 48991
› Domain Name System (response)
```

```
0000  00 00 00 01 00 06 52 55  0a 00 02 02 00 00 08 00   ······RU ········
0010  45 00 00 5b d9 c5 00 00  40 11 88 bb 0a 00 02 03   E··[···· @·······
0020  0a 00 02 0f 00 35 bf 5f  00 47 30 f0 78 fa 81 80   ·····5·_ ·G0·x···
0030  00 01 00 01 00 00 00 01  04 74 65 73 74 09 6d 6f   ········ ·test·mo
0040  73 71 75 69 74 74 6f 03  6f 72 67 00 00 01 00 01   squitto· org·····
0050  c0 0c 00 01 00 01 00 00  01 2c 00 04 05 c4 4e 1c   ········ ·,····N·
0060  00 00 29 04 d0 00 00 00  00 00 00                  ··)····· ···
```

Address: IPv4 address | Packets: 14059 · Displayed: 4 (0.0%) | Profile: Default

it's possible to notice that the desired IP is: 5.196.78.28.

Thus is possible to analyze the traffic with QoS equals to zero (at most once), sent by the broker with the retain option activated.


challenge2(1).pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
mqtt.qos == 0 && mqtt.msgtype == 3 && mqtt.retain == 1 && ip.dst ==  5.196.78.28
```

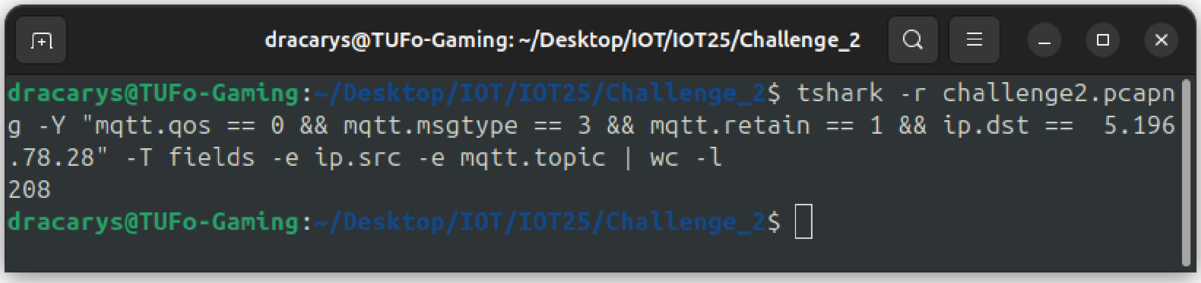
```
No.    port           Source      Destination  Protocol  Length  Info
   343 4.948626325    10.0.2.15   5.196.78.28  MQTT       200 Publish Message [metaverse/department2/area3]
   595 7.171214510    10.0.2.15   5.196.78.28  MQTT       203 Publish Message [factory/room2/room2/pollution]
  2877 14.980030200   10.0.2.15   5.196.78.28  MQTT       206 Publish Message [hospital/room2/area3/temperature]
  2914 15.186698287   10.0.2.15   5.196.78.28  MQTT       192 Publish Message [factory/room2/area3]
  2930 15.358906485   10.0.2.15   5.196.78.28  MQTT       193 Publish Message [factory/department2]
  3026 16.301143268   10.0.2.15   5.196.78.28  MQTT        95 Publish Message [metaverse/building3/floor4/deposit]
  3269 20.003801916   10.0.2.15   5.196.78.28  MQTT       206 Publish Message [metaverse/building3/room2/light]
  3770 27.008953131   10.0.2.15   5.196.78.28  MQTT       188 Publish Message [metaverse/room2]
```

```
› Frame 343: 200 bytes on wire (1600 bits), 200 bytes captured (1600 bits) on
› Linux cooked capture v1
› Internet Protocol Version 4, Src: 10.0.2.15, Dst: 5.196.78.28
› Transmission Control Protocol, Src Port: 56285, Dst Port: 1883, Seq: 241, A
› MQ Telemetry Transport Protocol, Publish Message
```

```
0000  00 04 00 01 00 06 08 00  27 31 33 62 08 00 08 00   ········ '13b····
0010  45 00 00 b8 0f 90 40 00  40 06 ca c1 0a 00 02 0f   E·····@· @·······
0020  05 c4 4e 1c db dd 07 5b  04 e5 e0 ea 1d e2 64 0e   ··N····[ ······d·
0030  50 18 fa e4 60 99 00 00  31 8d 01 00 1b 6d 65 74   P···`··· 1····met
0040  61 76 65 72 73 65 2f 64  65 70 61 72 74 6d 65 6e   averse/d epartmen
0050  74 32 2f 61 72 65 61 33  7b 22 6c 61 74 22 3a 20   t2/area3 {"lat": 
0060  38 34 2c 20 22 6c 6f 6e  67 22 3a 20 37 35 2c 20   84, "lon g": 75, 
0070  22 75 6e 69 74 22 3a 20  22 4b 22 2c 20 22 74 79   "unit":  "K", "ty
0080  70 65 22 3a 20 22 74 65  6d 70 65 72 61 74 75 72   pe": "te mperatur
```

challenge2(1).pcapng | Packets: 14059 · Displayed: 208 (1.5%) | Profile: Default

Below it's possible to see the proof of the result with TShark.



```
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ tshark -r challenge2.pcapng -Y "mqtt.qos == 0 && mqtt.msgtype == 3 && mqtt.retain == 1 && ip.dst ==  5.196.78.28" -T fields -e ip.src -e mqtt.topic | wc -l
208
dracarys@TUFo-Gaming:~/Desktop/IOT/IOT25/Challenge_2$ 
```

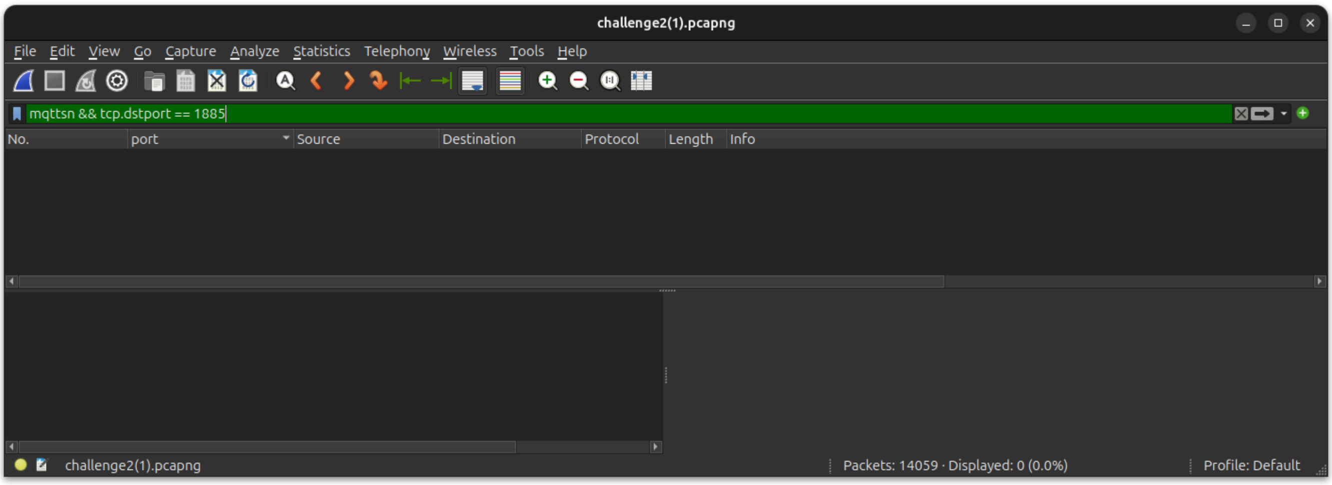
QC7

"How many MQTT-SN messages on port 1885 are sent by the clients to a broker in the local machine?"

Answer: 0

Before all we changed on the setting the port of the MQTT-SN on 1885.

After the filtering there weren't results, as is shown below.



challenge2(1).pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

mqttsn && tcp.dstport == 1885

Packets: 14059 · Displayed: 0 (0.0%)

Profile: Default