Internet Of Things (IOT) - Challenge 2

Mohammadreza Zamani (10869960) – Asal Abbasnejadfard (10974178)

Packet Sniffing

Part 1
Questions on the PCAP file

CQ1) How many different Confirmable PUT requests obtained an unsuccessful response from the local CoAP server?

Answer: 22

In the first step to solve this question, we should separate different confirmable put requests from others. We use this filter coap.type == 0 && coap.

code == 3. coap.type == 0 for separating confirmable requests and coap.code == 3 for put requests. This filter returns 45 packets, representing all Confirmable PUT requests.

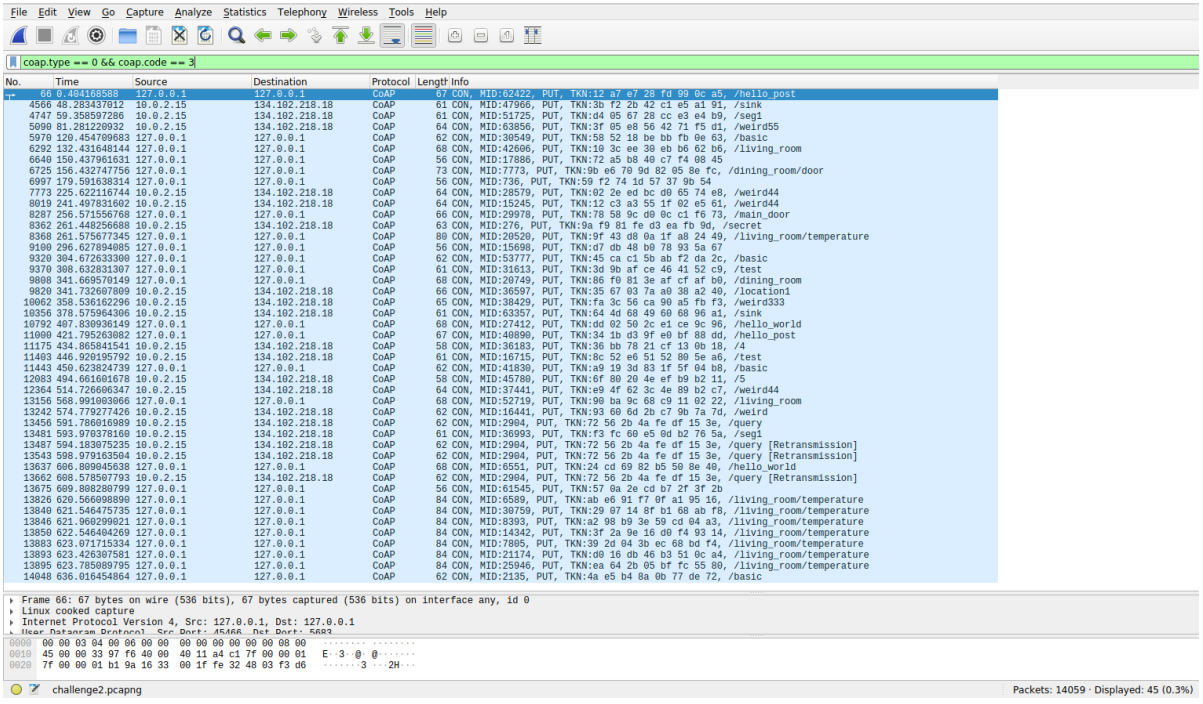


Fig. 1.

Then, we should only consider an unsuccessful response from the local CoAP server. It means their destination address should be 127.0.0.1 .
Then, we should consider token number for each pocket in the info section. For example, for the packet number 66, we use this filter coap.token == 12:a7:e7:28:fd:99:0c:a5 to find which confirmable put requests obtain an unsuccessful response. This filter helps us to see responses related to this token. In the first line of the figure below, you can see the request and in the second line the response to this request. Code 2.04 according to the CoAP standard means a successful response. So, we should not consider this response.

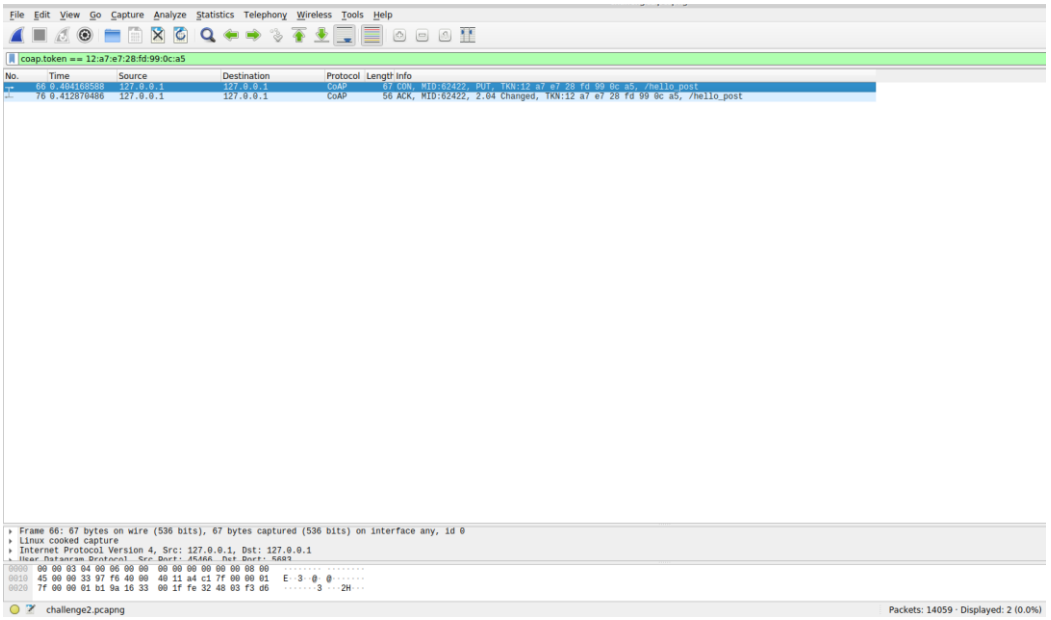


Fig. 2.

For packet number 6725, we obtain 4.04 not found and the response is unsuccessful.

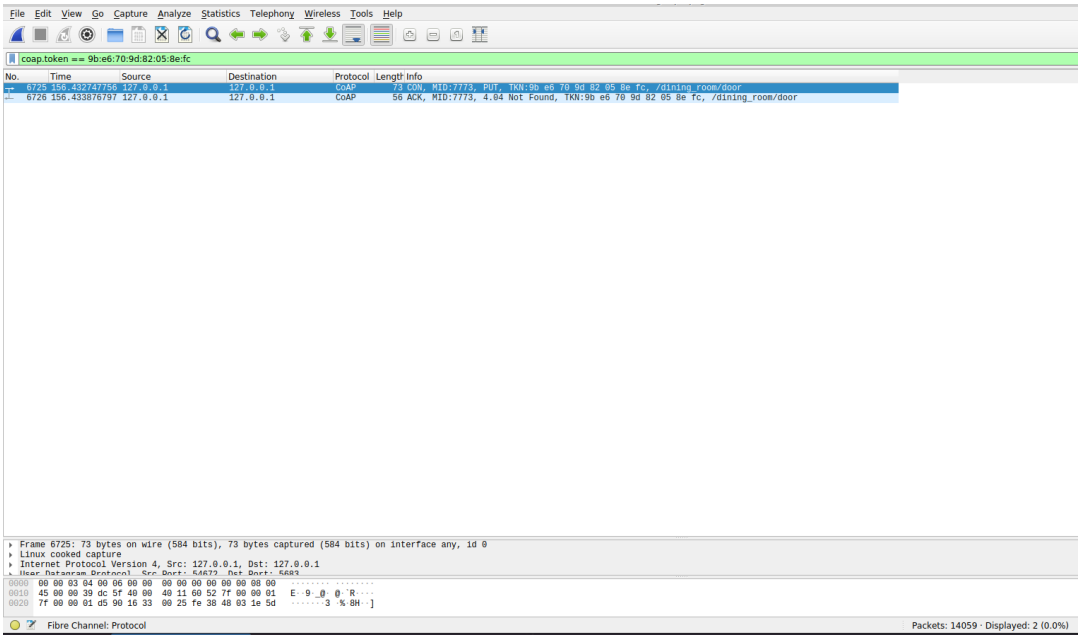


Fig. 3.

For packet number 10792, we obtain 4.05 method not allowed and the response is unsuccessful.

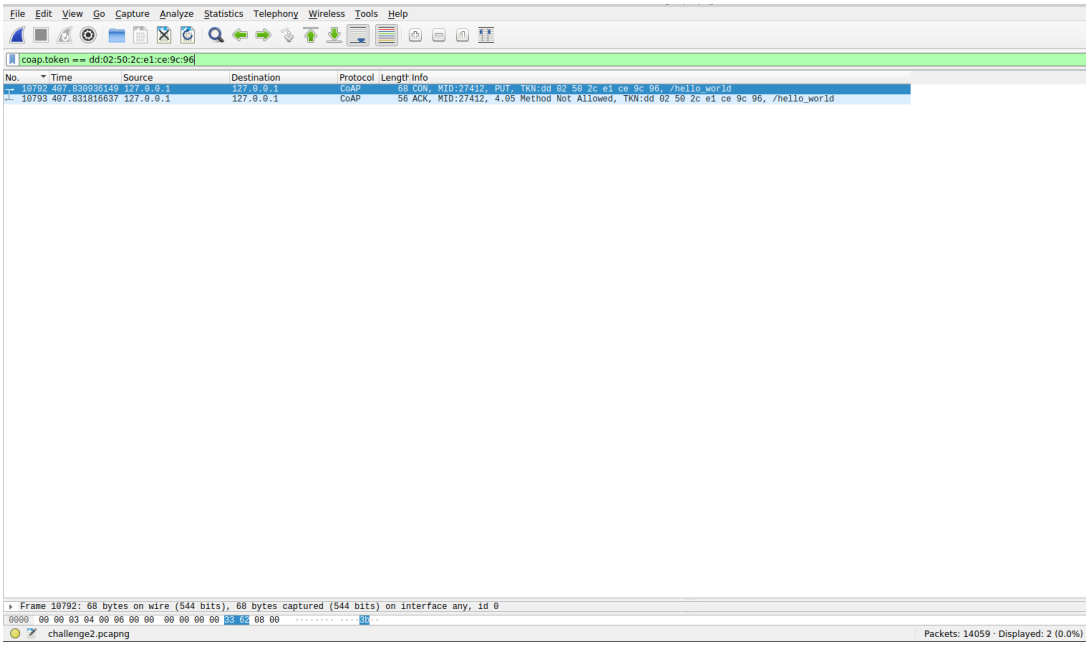


Fig. 4.

We do this process for all packets and finally find 22 different Confirmable PUT requests obtained an unsuccessful response from the local CoAP server.

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

First, we use this filter coap.type == 0 && coap.code == 1 && ip.dst == 134.102.218.18 .

We know that The IP address of the coap.me server is 134.102.218.18. also, coap.type == 0 use to filter confirmable request and coap.code == 1 use to filter get requests.

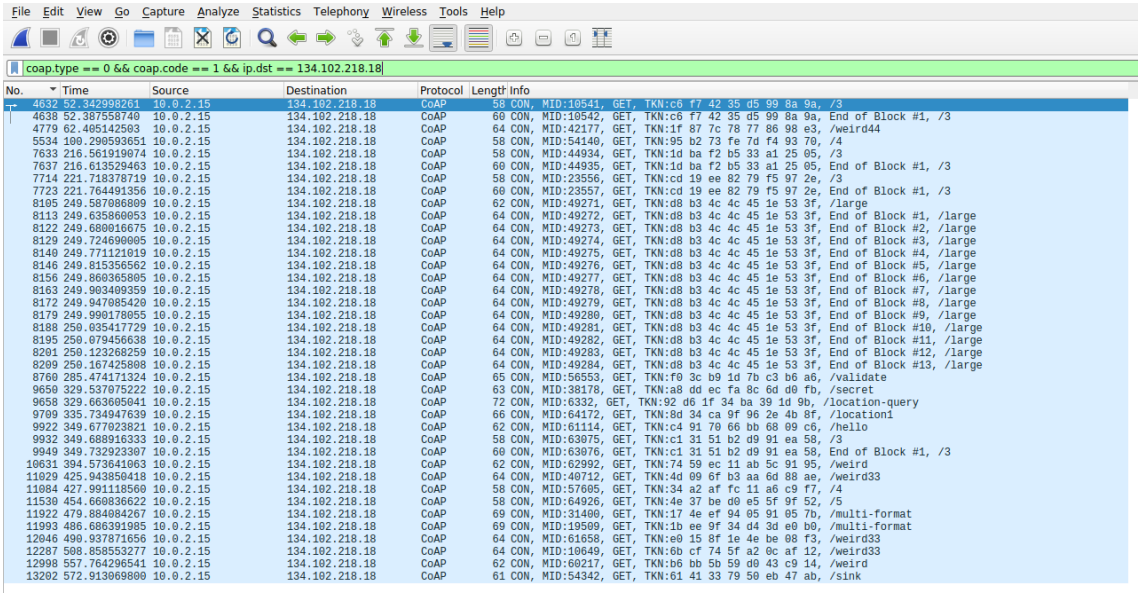


Fig. 5.

Then, we write from the info column the resource paths. The resource paths For confirmable get requests are:

1- /3

2- /weird44

3- /4

4- /large

5- /5

6- /secret

7- /validate

8- /location-query

9- /location1

10- /hello

11- /weird

12- /weird33

13- /multi-format

14- /sink

Next, we use this filter for non-confirmable request

coap.type == 1 && coap.code == 1 && ip.dst == 134.102.218.18

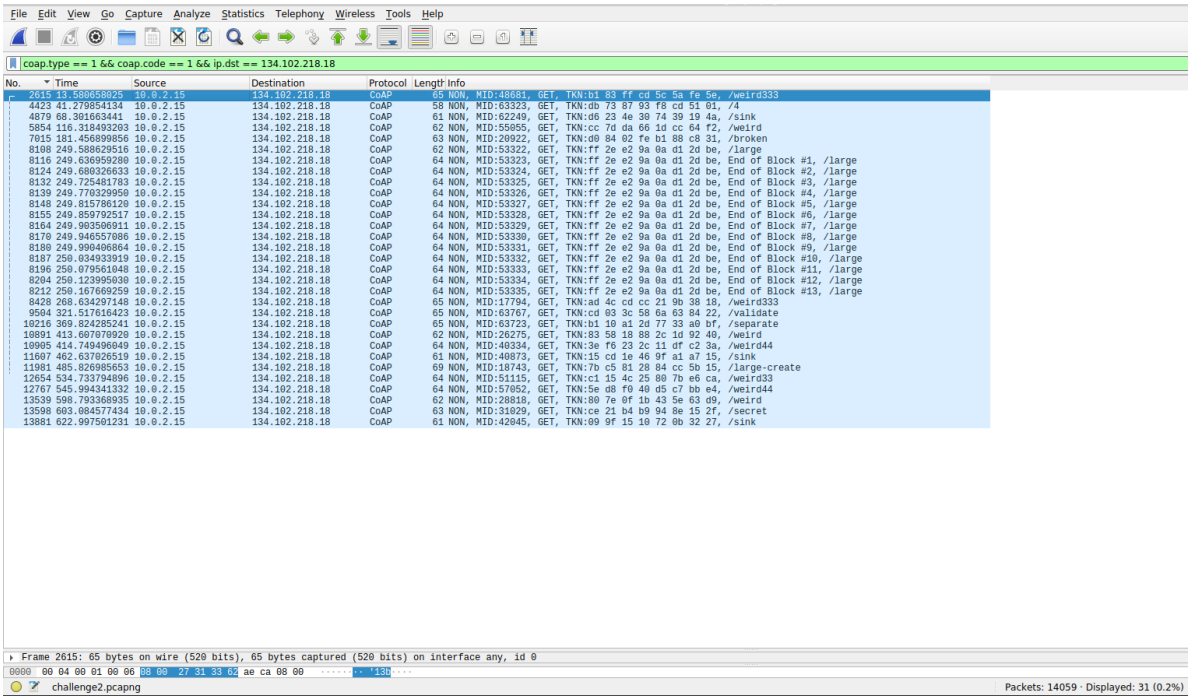


Fig. 6.

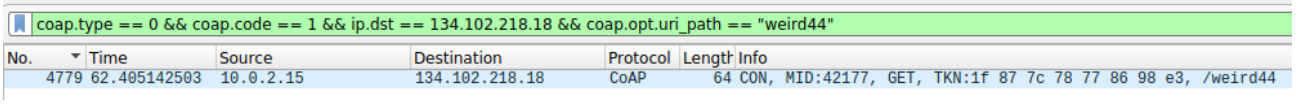
The resource paths For non-confirmable get requests are:

1. /weird333
2. /weird44
3. /4
4. /sink
5. /weird
6. /broken
7. /large
8. /validate
9. /separate
10. /weird33
11. /large-create
12. / secret

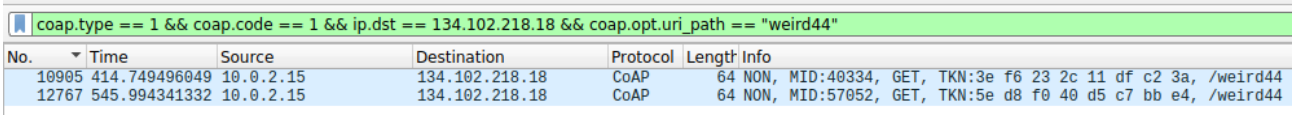
The number of resources that received both Confirmable and Non-confirmable GETs (with X=Y and X>0) is 8. These lists are:

1. /weird44
2. /4
3. /sink
4. /weird
5. /large
6. /validate
7. /weird33
8. /secret

Now, we should create two separate filters for each path above:
For /weird44 path we set this filter coap.type == 0 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "weird44" for confirmable get and this filter coap.type == 1 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "weird44" for non-confirmable get.

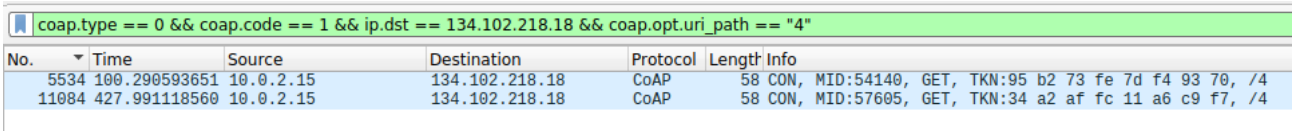


In the image above, there is only one packet, which shows that a confirmable GET request was sent to the resource /weird44.



In the image above, there are two packets, which show that a non-confirmable GET request was sent to the resource /weird44.So, X is not equal to Y.

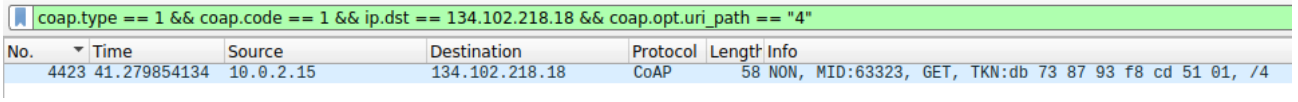
For /4 path:



coap.type == 0 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "4"

No.	Time	Source	Destination	Protocol	Length	Info
5534	100.290593651	10.0.2.15	134.102.218.18	CoAP	58	CON, MID:54140, GET, TKN:95 b2 73 fe 7d f4 93 70, /4
11084	427.991118560	10.0.2.15	134.102.218.18	CoAP	58	CON, MID:57605, GET, TKN:34 a2 af fc 11 a6 c9 f7, /4

There are two packets which show that a confirmable GET request was sent to the resource /4.

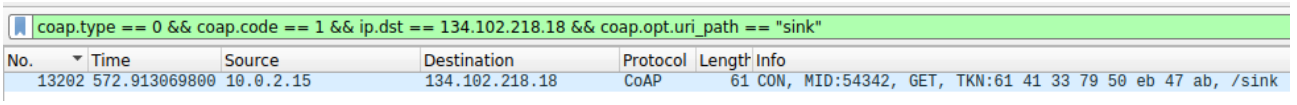


coap.type == 1 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "4"

No.	Time	Source	Destination	Protocol	Length	Info
4423	41.279854134	10.0.2.15	134.102.218.18	CoAP	58	NON, MID:63323, GET, TKN:db 73 87 93 f8 cd 51 01, /4

There is one packet which shows that a confirmable GET request was sent to the resource /4. X is not equal to Y.

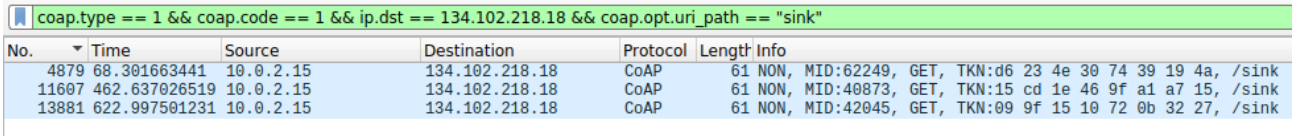
For /sink path:



coap.type == 0 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "sink"

No.	Time	Source	Destination	Protocol	Length	Info
13202	572.913069800	10.0.2.15	134.102.218.18	CoAP	61	CON, MID:54342, GET, TKN:61 41 33 79 50 eb 47 ab, /sink

There is one packet which shows that a confirmable GET request was sent to the resource /sink.

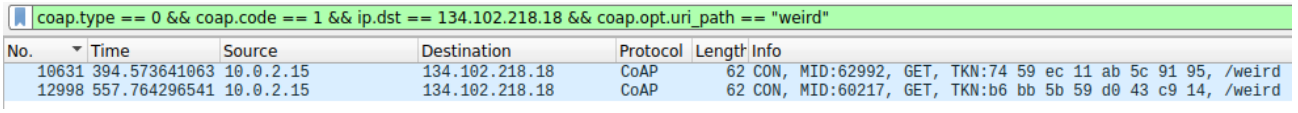


coap.type == 1 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "sink"

No.	Time	Source	Destination	Protocol	Length	Info
4879	68.301663441	10.0.2.15	134.102.218.18	CoAP	61	NON, MID:62249, GET, TKN:d6 23 4e 30 74 39 19 4a, /sink
11607	462.637026519	10.0.2.15	134.102.218.18	CoAP	61	NON, MID:40873, GET, TKN:15 cd 1e 46 9f a1 a7 15, /sink
13881	622.997501231	10.0.2.15	134.102.218.18	CoAP	61	NON, MID:42045, GET, TKN:09 9f 15 10 72 0b 32 27, /sink

There are three packets which show that a confirmable GET request was sent to the resource /sink. X is not equal to Y.

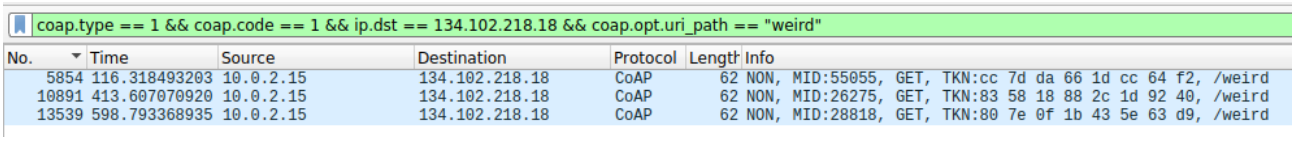
For /weird path:



coap.type == 0 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "weird"

No.	Time	Source	Destination	Protocol	Length	Info
10631	394.573641063	10.0.2.15	134.102.218.18	CoAP	62	CON, MID:62992, GET, TKN:74 59 ec 11 ab 5c 91 95, /weird
12998	557.764296541	10.0.2.15	134.102.218.18	CoAP	62	CON, MID:60217, GET, TKN:b6 bb 5b 59 d0 43 c9 14, /weird

There are two packets which show that a confirmable GET request was sent to the resource /weird.

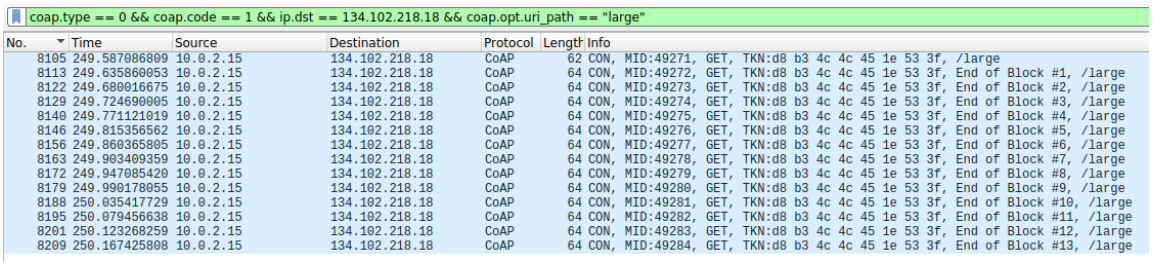


coap.type == 1 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "weird"

No.	Time	Source	Destination	Protocol	Length	Info
5854	116.318493203	10.0.2.15	134.102.218.18	CoAP	62	NON, MID:55055, GET, TKN:cc 7d da 66 1d cc 64 f2, /weird
10891	413.607070920	10.0.2.15	134.102.218.18	CoAP	62	NON, MID:26275, GET, TKN:83 58 18 88 2c 1d 92 40, /weird
13539	598.793368935	10.0.2.15	134.102.218.18	CoAP	62	NON, MID:28818, GET, TKN:80 7e 0f 1b 43 5e 63 d9, /weird

There are three packets which show that a confirmable GET request was sent to the resource /weird. X is not equal to Y.

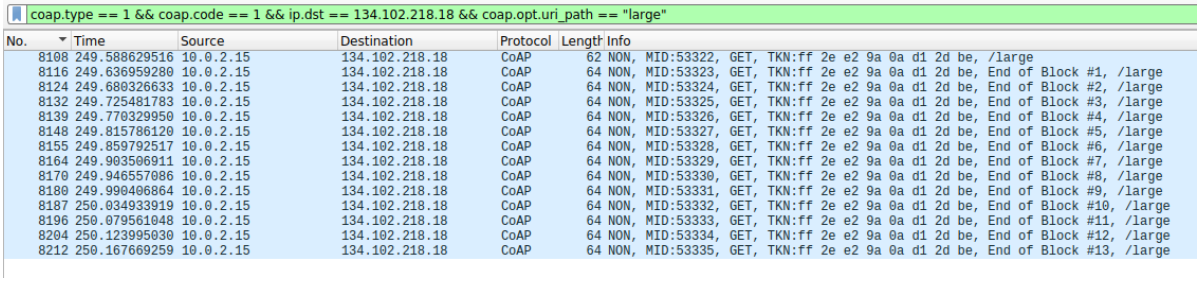
For /large path:



coap.type == 0 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "large"

No.	Time	Source	Destination	Protocol	Length	Info
8105	249.587086809	10.0.2.15	134.102.218.18	CoAP	62	CON, MID:49271, GET, TKN:d8 b3 4c 4c 45 1e 53 3f, /large
8113	249.635860053	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:49272, GET, TKN:d8 b3 4c 4c 45 1e 53 3f, End of Block #1, /large
8122	249.680016675	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:49273, GET, TKN:d8 b3 4c 4c 45 1e 53 3f, End of Block #2, /large
8129	249.724690005	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:49274, GET, TKN:d8 b3 4c 4c 45 1e 53 3f, End of Block #3, /large
8140	249.771121019	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:49275, GET, TKN:d8 b3 4c 4c 45 1e 53 3f, End of Block #4, /large
8146	249.815356562	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:49276, GET, TKN:d8 b3 4c 4c 45 1e 53 3f, End of Block #5, /large
8156	249.860365805	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:49277, GET, TKN:d8 b3 4c 4c 45 1e 53 3f, End of Block #6, /large
8163	249.903409359	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:49278, GET, TKN:d8 b3 4c 4c 45 1e 53 3f, End of Block #7, /large
8172	249.947085420	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:49279, GET, TKN:d8 b3 4c 4c 45 1e 53 3f, End of Block #8, /large
8179	249.990178055	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:49280, GET, TKN:d8 b3 4c 4c 45 1e 53 3f, End of Block #9, /large
8188	250.035417729	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:49281, GET, TKN:d8 b3 4c 4c 45 1e 53 3f, End of Block #10, /large
8195	250.079456638	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:49282, GET, TKN:d8 b3 4c 4c 45 1e 53 3f, End of Block #11, /large
8201	250.123268259	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:49283, GET, TKN:d8 b3 4c 4c 45 1e 53 3f, End of Block #12, /large
8209	250.167425808	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:49284, GET, TKN:d8 b3 4c 4c 45 1e 53 3f, End of Block #13, /large

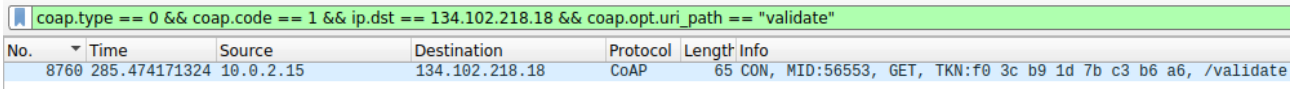
There are 14 packets which show that a confirmable GET request was sent to the resource /large.


coap.type == 1 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "large"

No.	Time	Source	Destination	Protocol	Length	Info
8108	249.588629516	10.0.2.15	134.102.218.18	CoAP	62	NON, MID:53322, GET, TKN:ff 2e e2 9a 0a d1 2d be, /large
8116	249.636959280	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:53323, GET, TKN:ff 2e e2 9a 0a d1 2d be, End of Block #1, /large
8124	249.680326633	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:53324, GET, TKN:ff 2e e2 9a 0a d1 2d be, End of Block #2, /large
8132	249.725481783	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:53325, GET, TKN:ff 2e e2 9a 0a d1 2d be, End of Block #3, /large
8139	249.770329950	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:53326, GET, TKN:ff 2e e2 9a 0a d1 2d be, End of Block #4, /large
8148	249.815786120	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:53327, GET, TKN:ff 2e e2 9a 0a d1 2d be, End of Block #5, /large
8155	249.859792517	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:53328, GET, TKN:ff 2e e2 9a 0a d1 2d be, End of Block #6, /large
8164	249.903506911	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:53329, GET, TKN:ff 2e e2 9a 0a d1 2d be, End of Block #7, /large
8170	249.946557086	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:53330, GET, TKN:ff 2e e2 9a 0a d1 2d be, End of Block #8, /large
8180	249.990406864	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:53331, GET, TKN:ff 2e e2 9a 0a d1 2d be, End of Block #9, /large
8187	250.034933919	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:53332, GET, TKN:ff 2e e2 9a 0a d1 2d be, End of Block #10, /large
8196	250.079561048	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:53333, GET, TKN:ff 2e e2 9a 0a d1 2d be, End of Block #11, /large
8204	250.123995030	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:53334, GET, TKN:ff 2e e2 9a 0a d1 2d be, End of Block #12, /large
8212	250.167669259	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:53335, GET, TKN:ff 2e e2 9a 0a d1 2d be, End of Block #13, /large

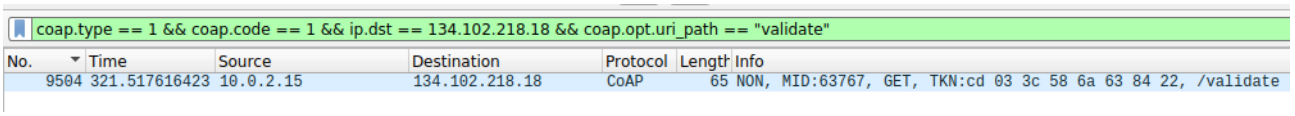
There are 14 packets which show that a confirmable GET request was sent to the resource /large.
X is equal to Y and X>0.

For /validate path:


coap.type == 0 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "validate"

No.	Time	Source	Destination	Protocol	Length	Info
8760	285.474171324	10.0.2.15	134.102.218.18	CoAP	65	CON, MID:56553, GET, TKN:f0 3c b9 1d 7b c3 b6 a6, /validate

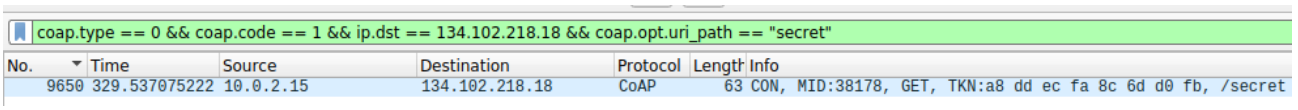
There is one packet which shows that a confirmable GET request was sent to the resource /validate.


coap.type == 1 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "validate"

No.	Time	Source	Destination	Protocol	Length	Info
9504	321.517616423	10.0.2.15	134.102.218.18	CoAP	65	NON, MID:63767, GET, TKN:cd 03 3c 58 6a 63 84 22, /validate

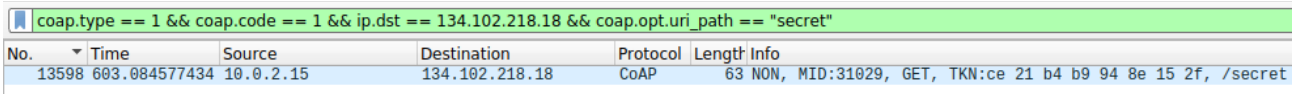
There is one packet which shows that a confirmable GET request was sent to the resource /validate.
X is equal to Y and X>0.

For /secret path:


coap.type == 0 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "secret"

No.	Time	Source	Destination	Protocol	Length	Info
9650	329.537075222	10.0.2.15	134.102.218.18	CoAP	63	CON, MID:38178, GET, TKN:a8 dd ec fa 8c 6d d0 fb, /secret

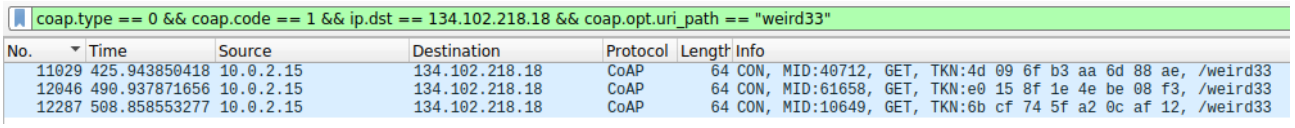
There is one packet which shows that a confirmable GET request was sent to the resource /secret.


coap.type == 1 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "secret"

No.	Time	Source	Destination	Protocol	Length	Info
13598	603.084577434	10.0.2.15	134.102.218.18	CoAP	63	NON, MID:31029, GET, TKN:ce 21 b4 b9 94 8e 15 2f, /secret

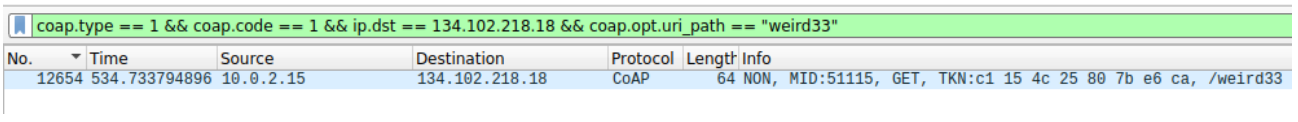
There is one packet which shows that a confirmable GET request was sent to the resource /secret.
X is equal to Y and X>0.

For /weird33 path:


coap.type == 0 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "weird33"

No.	Time	Source	Destination	Protocol	Length	Info
11029	425.943850418	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:40712, GET, TKN:4d 09 6f b3 aa 6d 88 ae, /weird33
12046	490.937871656	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:61658, GET, TKN:e0 15 8f 1e 4e be 08 f3, /weird33
12287	508.858553277	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:10649, GET, TKN:6b cf 74 5f a2 0c af 12, /weird33

There are three packets which show that a confirmable GET request was sent to the resource /weird33.


coap.type == 1 && coap.code == 1 && ip.dst == 134.102.218.18 && coap.opt.uri_path == "weird33"

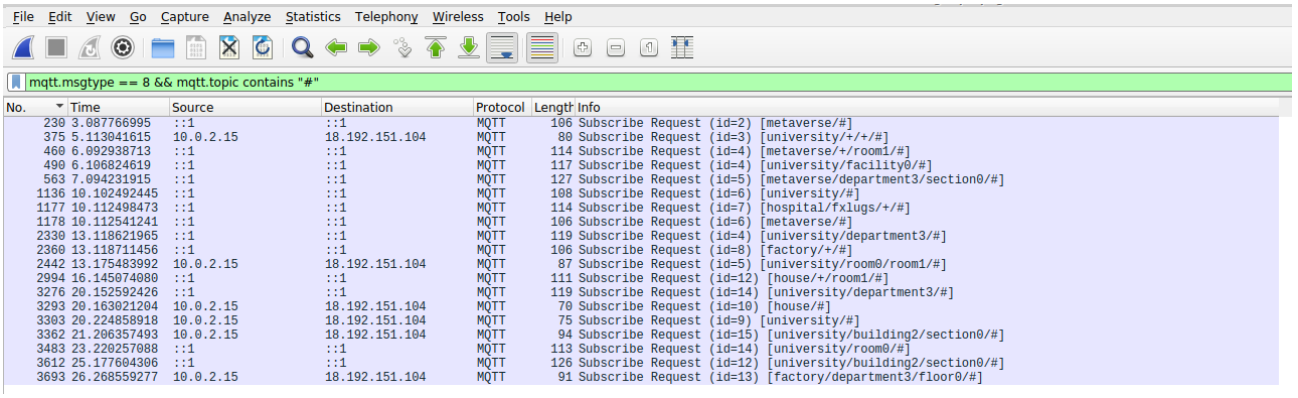
No.	Time	Source	Destination	Protocol	Length	Info
12654	534.733794896	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:51115, GET, TKN:c1 15 4c 25 80 7b e6 ca, /weird33

There is one packet which shows that a confirmable GET request was sent to the resource /weird33.
X is not equal to Y.

CQ3) How many different MQTT clients subscribe to the public broker HiveMQ using multi-level wildcards?

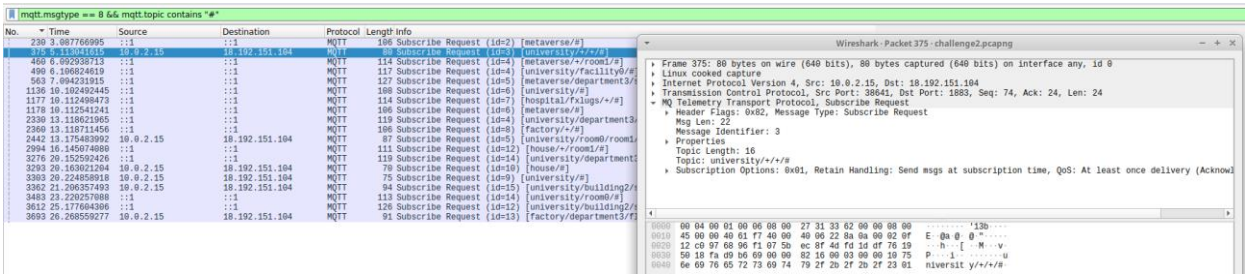
Answer: 4

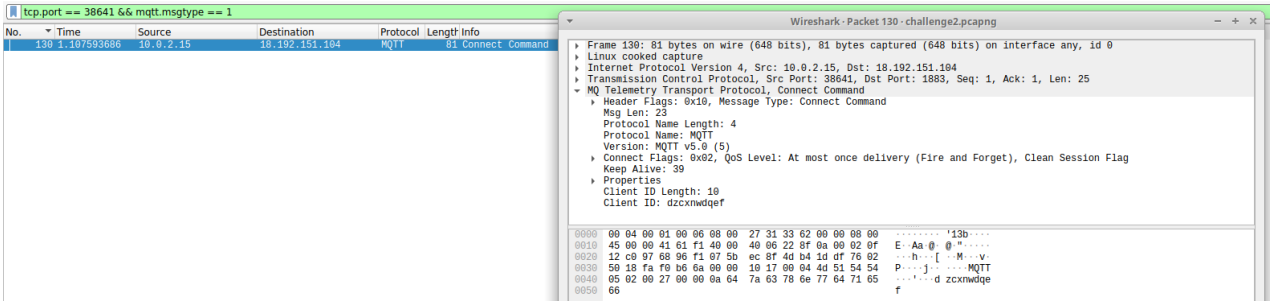
To solve this question, first we use this filter mqtt.msgtype == 8 && mqtt.topic contains "#".
This filter displays only MQTT SUBSCRIBE messages (mqtt.msgtype == 8) where the topic contains the multi-level wildcard #.



No.	Time	Source	Destination	Protocol	Length	Info
230	3.087766995	::1	::1	MQTT	106	Subscribe Request (id=2) [metaverse/#]
375	5.113041615	10.0.2.15	18.192.151.104	MQTT	80	Subscribe Request (id=3) [university/+/+/#]
460	6.092938713	::1	::1	MQTT	114	Subscribe Request (id=4) [metaverse/+/room1/#]
490	6.106824619	::1	::1	MQTT	117	Subscribe Request (id=4) [university/facility0/#]
563	7.094231915	::1	::1	MQTT	127	Subscribe Request (id=5) [metaverse/department3/section0/#]
1136	10.102492445	::1	::1	MQTT	108	Subscribe Request (id=6) [university/#]
1177	10.112498473	::1	::1	MQTT	114	Subscribe Request (id=7) [hospital/fxlugs/+/#]
1178	10.112541241	::1	::1	MQTT	106	Subscribe Request (id=6) [metaverse/#]
2330	13.118621965	::1	::1	MQTT	119	Subscribe Request (id=4) [university/department3/#]
2360	13.118711456	::1	::1	MQTT	106	Subscribe Request (id=8) [factory/+/#]
2442	13.175483992	10.0.2.15	18.192.151.104	MQTT	87	Subscribe Request (id=5) [university/room0/room1/#]
2994	16.145074080	::1	::1	MQTT	111	Subscribe Request (id=12) [house/+/room1/#]
3276	20.152592426	::1	::1	MQTT	119	Subscribe Request (id=14) [university/department3/#]
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/#]
3483	23.220257088	::1	::1	MQTT	113	Subscribe Request (id=14) [university/room0/#]
3612	25.177604306	::1	::1	MQTT	126	Subscribe Request (id=12) [university/building2/section0/#]
3693	26.268559277	10.0.2.15	18.192.151.104	MQTT	91	Subscribe Request (id=13) [factory/department3/floor0/#]

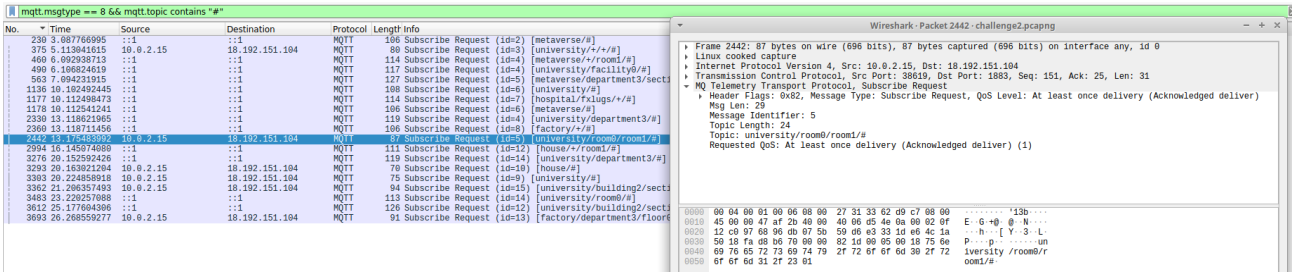
We can concern from above image that which clients connect to the HiveMQ public broker. Some clients are using IP ::1 (localhost). This means that some of these clients are not connected to the public broker at all but are connected to their own system.So, we should consider only those clients that connect to IP 18.192.151.104 .
For each of these six packets, we should do the same procedure. We should find different Client ID for these packets.
For packet **375**:
Use its source port (38641) and then use this filter tcp.port == <source port> && mqtt.msgtype == 1 to find Client ID.

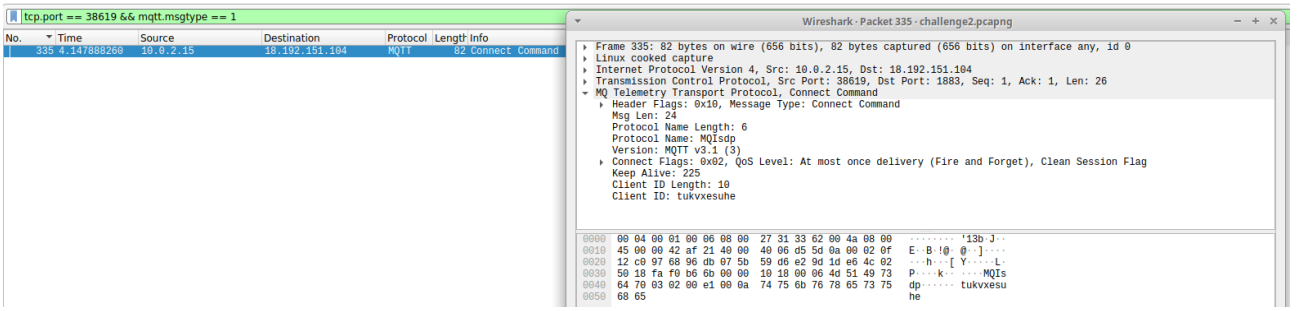




The client id for this packet is **dzcxnwdqef.**

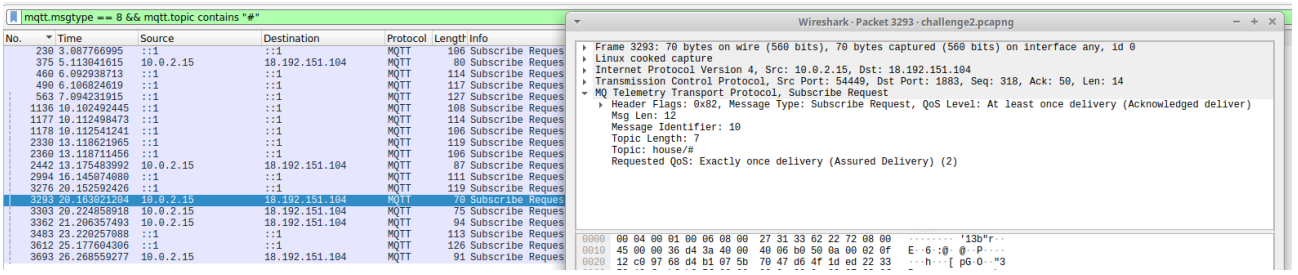
For packet **2442**:
Use its source port (38619).

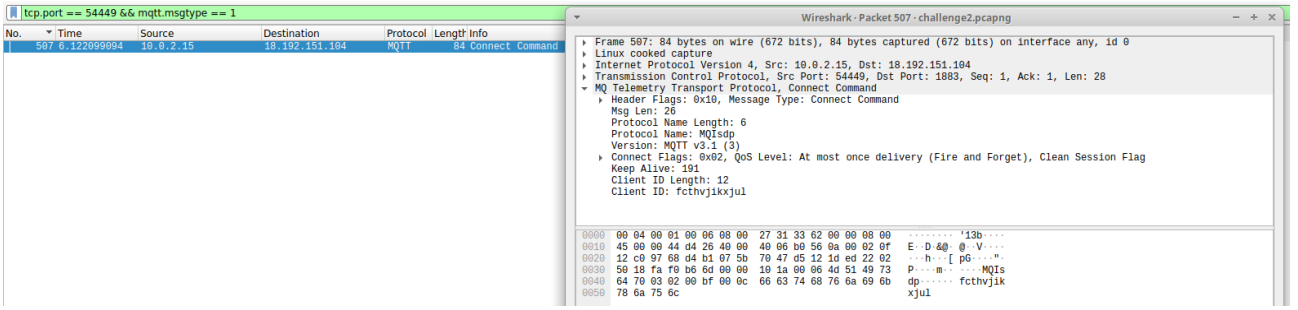




The client id for packet 2442 is **tukvxesuhe.**

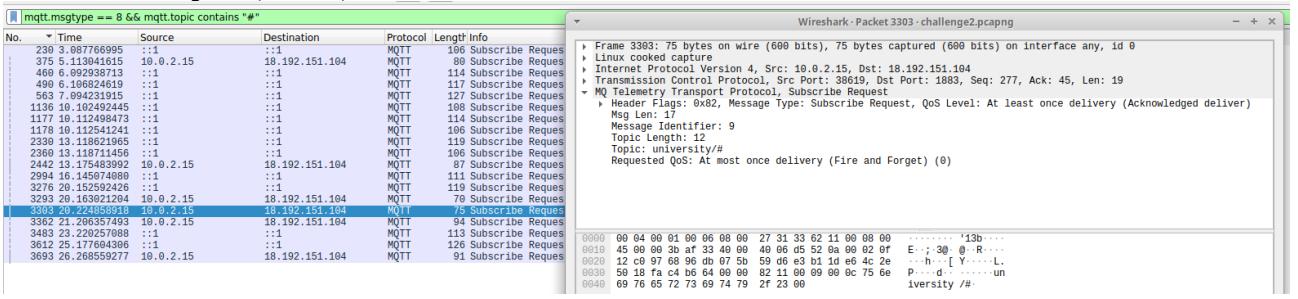
For packet **3293**:
Use its source port (54449).

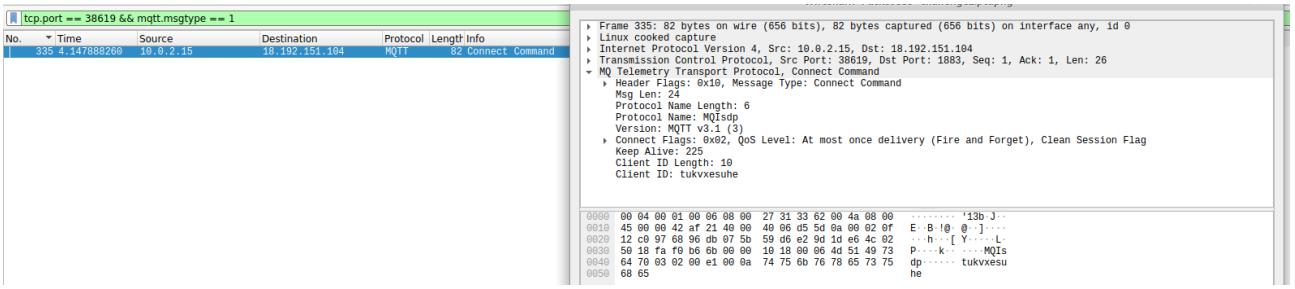




The client id for packet 3293 is **fcthvjkkxjul.**

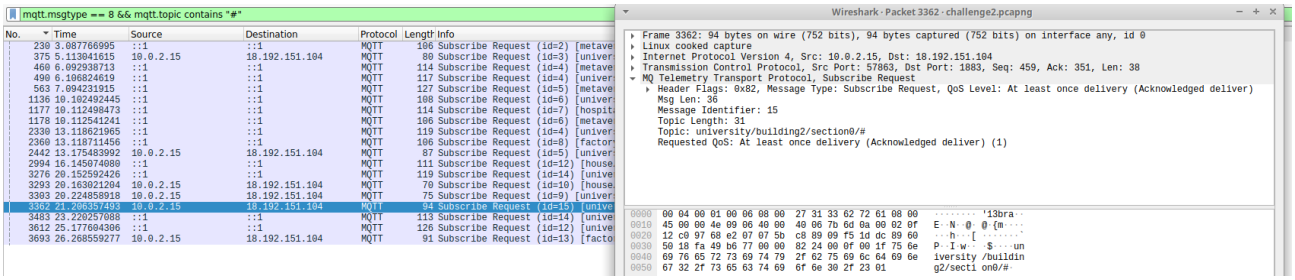
For packet **3303**:
Use its source port (38619).

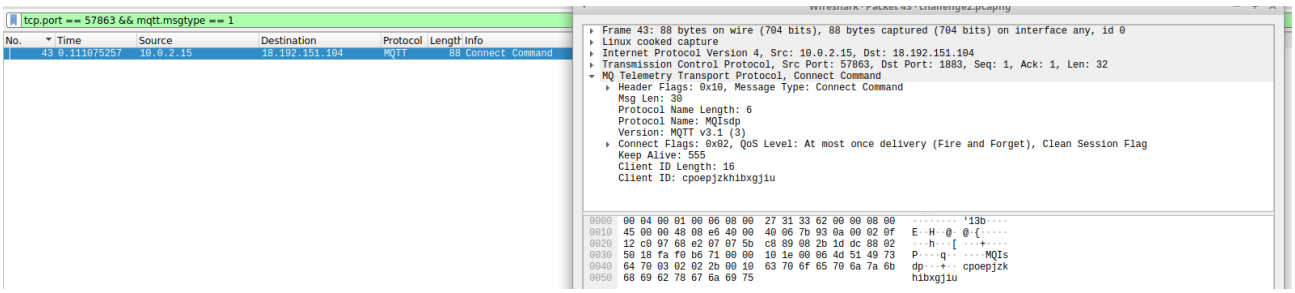




The client id for packet 3303 is **tukvxesuhe.**

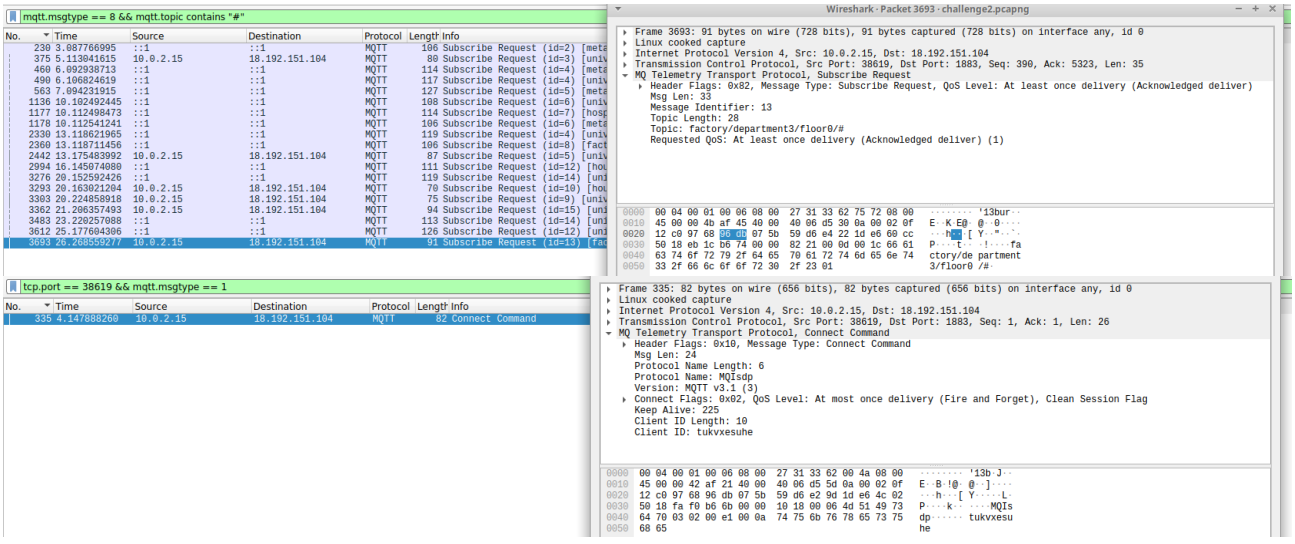
For packet **3362**:
Use its source port (57863).





The client id for packet 3362 is **cpoepjzkhibxgjiu.**

For packet **3693**:
Use its source port (38619).



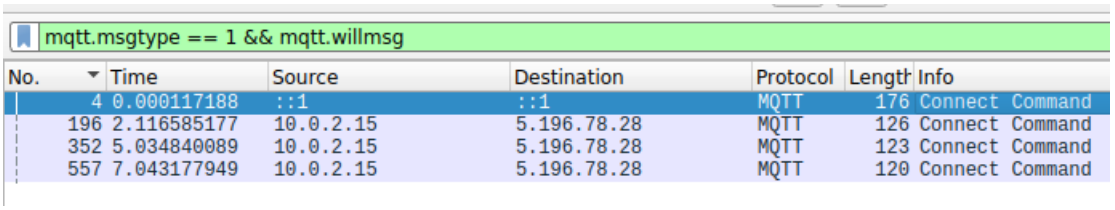
The client id for packet 3693 is **tukvxesuhe.**

The list of different client ids is **dzcxnwdqef, tukvxesuhe, fcthvjkkxjul, cpoepjzkhibxgjiu.**

CQ4) How many different MQTT clients specify a last Will Message to be directed to a topic having as first level "university"?

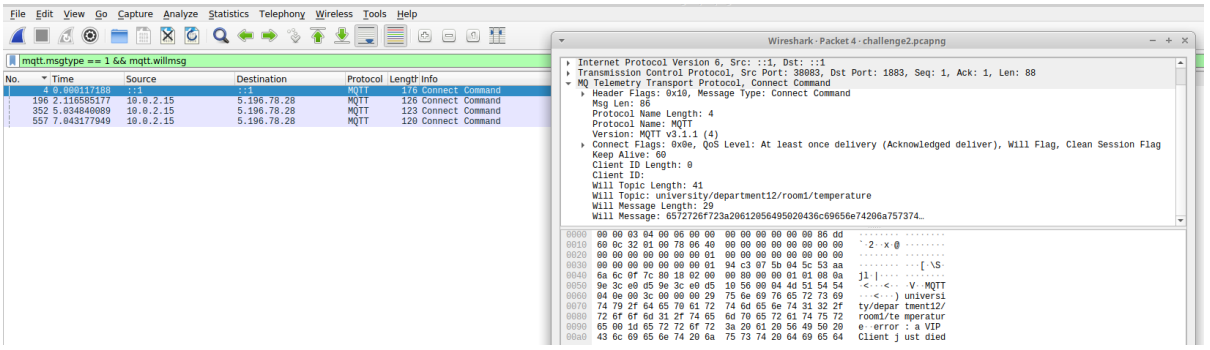
Answer: 1

We use this filter mqtt.msgtype == 1 && mqtt.willmsg. This filter shows only MQTT CONNECT packets (mqtt.msgtype == 1) that include a Last Will Message (mqtt.willmsg).



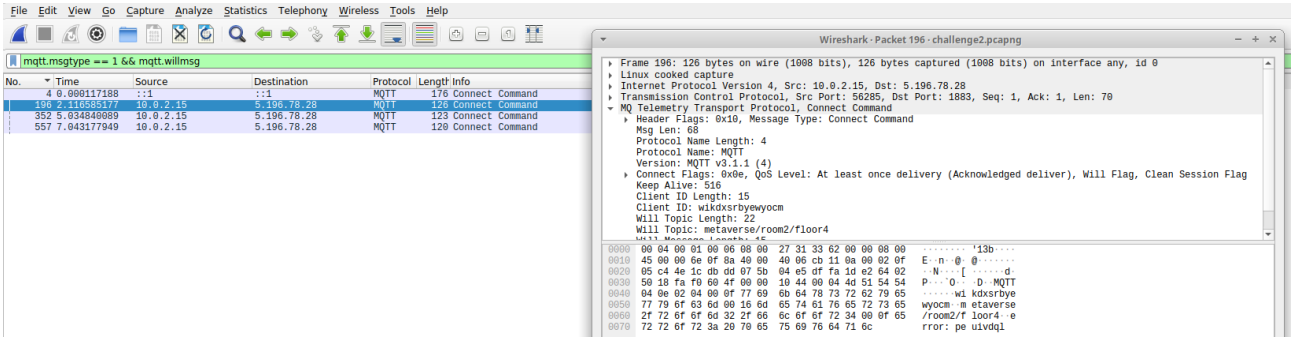
Now we need to check in each packet if Will Topic starts with "university/" or not.

For packet **4**:



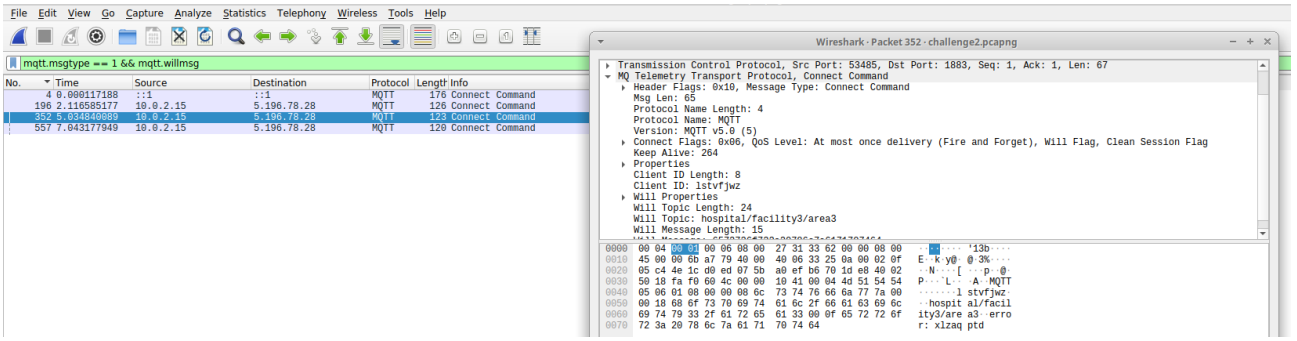
The Will Topic starts with /university.

For packet **196**:



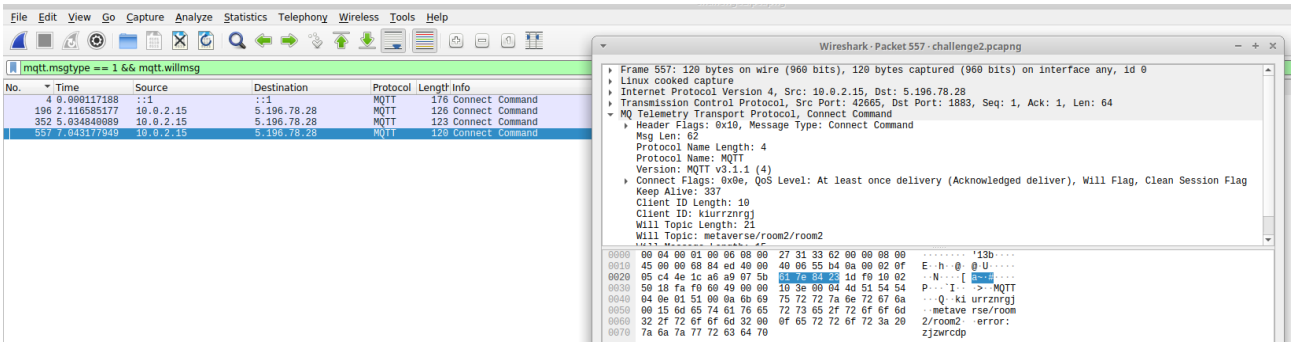
The will topic doesn't start with /university.

For packet **352**:



The Will Topic doesn't start with /university.

For packet **557**:



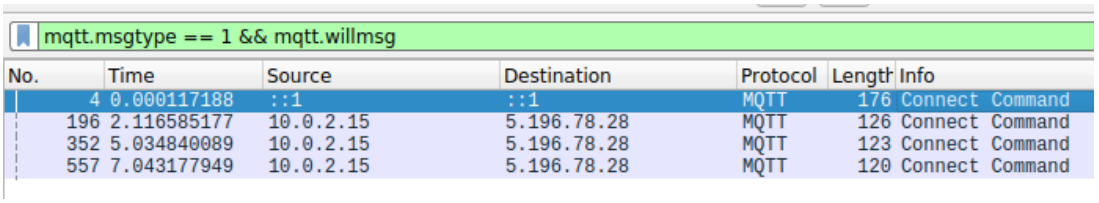
The Will Topic doesn't start with /university.

CQ5) How many MQTT subscribers receive a last will message derived from a subscription without a wildcard?

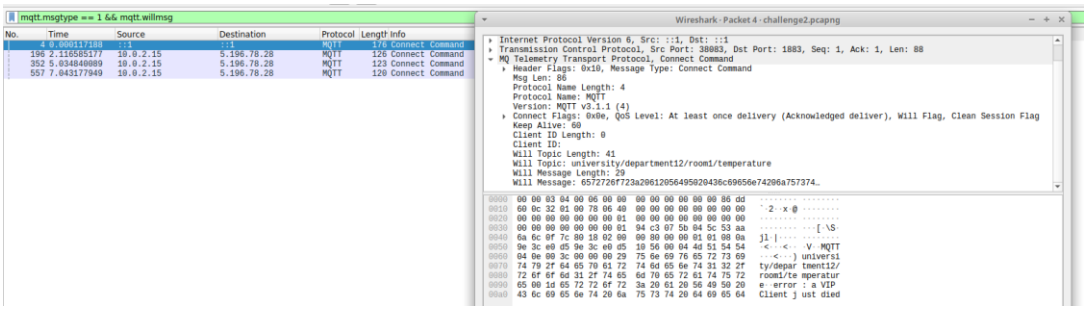
Answer: 3

First we use this filter mqtt.msgtype == 1 && mqtt.willmsg.

This filter identifies clients (mqtt.msgtype == 1) that defines a last will message (mqtt.willmsg) derived from a subscription without a wildcard. Now we need to check what will topic each packet has.

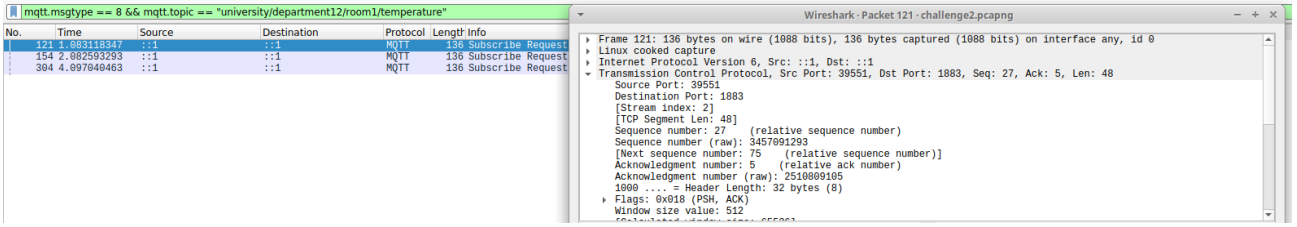


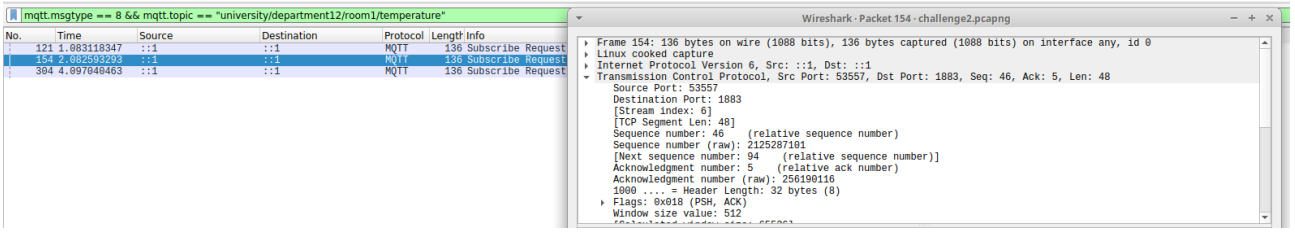
For packet **4**:

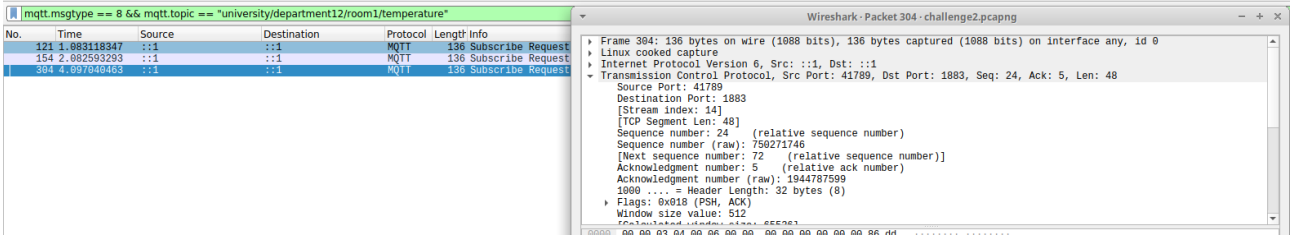


The Will Topic is **"university/department12/room1/temperature"**.

In this step, we check if a client has subscribed to this topic without using a wildcard or not. We use this filter mqtt.msgtype == 8 && mqtt.topic == "university/department12/room1/temperature". From above image we can see three packets, but we should analyze that these three packets belong to different TCP streams, meaning they are from different clients.

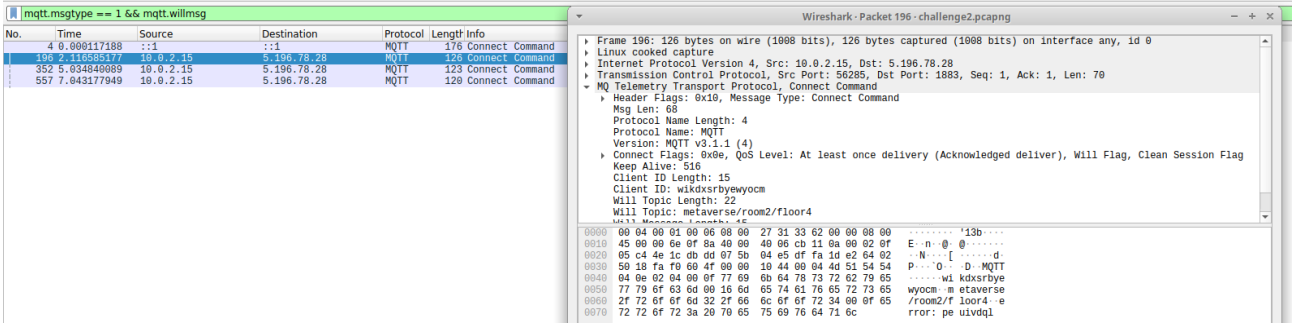




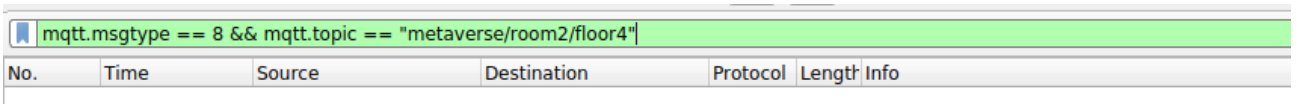


For this packet, three clients received messages without wildcards.

For packet **196**:

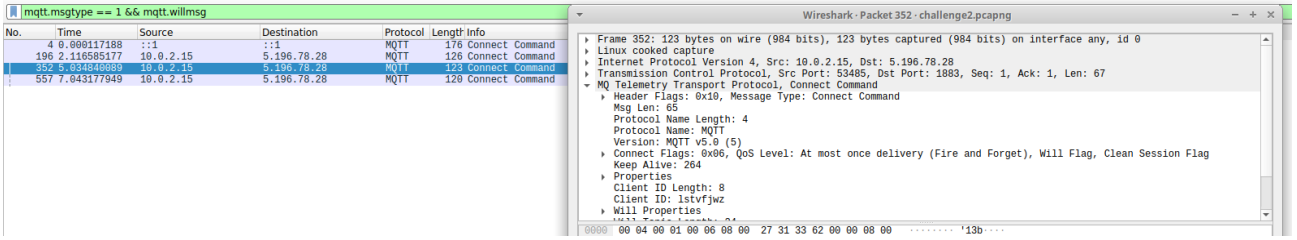


The Will Topic is **"metaverse/room2/floor4"**. We use this filter mqtt.msgtype == 8 && mqtt.topic == "metaverse/room2/floor4".

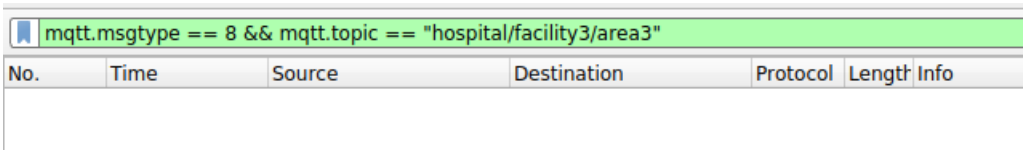


The number of subscriptions without wildcards on this topic is zero.

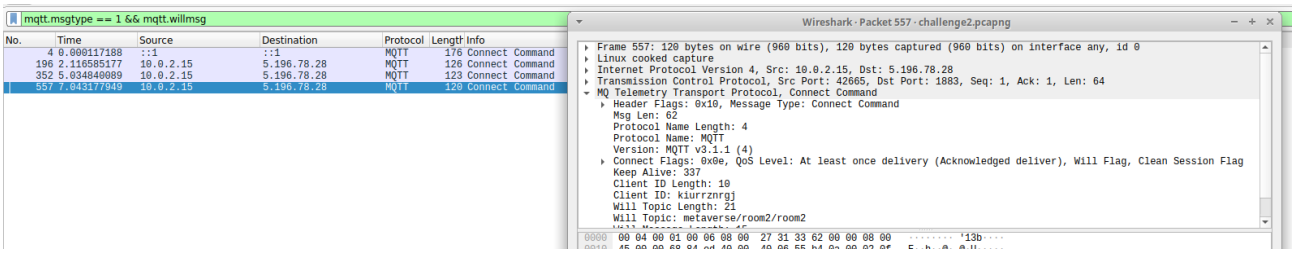
For packet **352**:



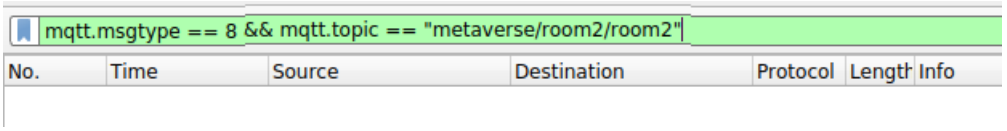
The Will Topic is "hospital/facility3/area3". We use this filter mqtt.msgtype == 8 && mqtt.topic == "hospital/facility3/area3".



For packet **557**:



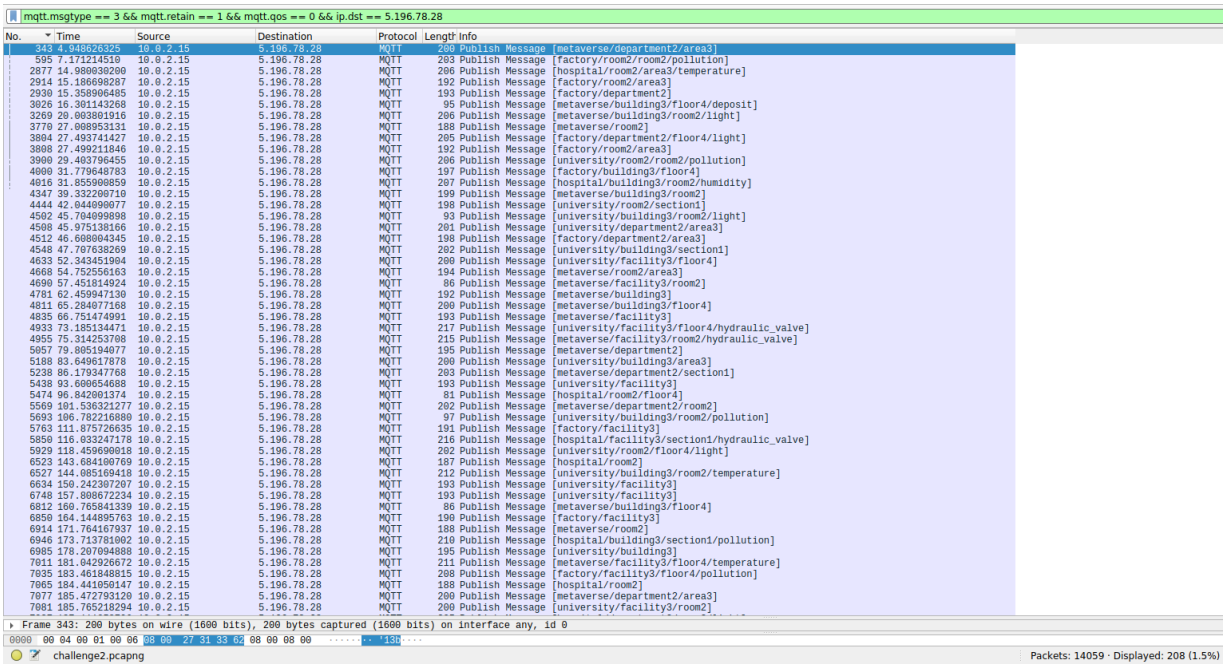
The Will Topic is "metaverse/room2/room2". We use this filter mqtt.msgtype == 8 && mqtt.topic == "metaverse/room2/room2".



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

I use this filter mqtt.msgtype == 3 && mqtt.retain == 1 && mqtt.qos == 0 && ip.dst == 5.196.78.28. This filter displays all MQTT PUBLISH messages sent to the Mosquitto broker (ip.dst == 5.196.78.28) that have the retain flag set (mqtt.retain == 1) and use QoS level 0 (mqtt.qos == 0), which means the message is delivered at most once, without any acknowledgment or retransmission if delivery fails.

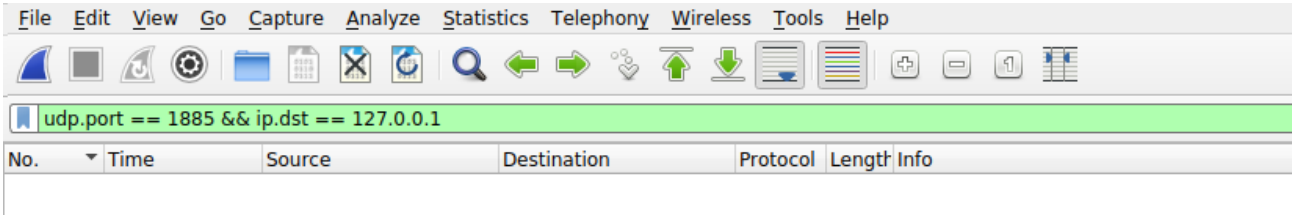


In the above image, the displayed show 208.

CQ7) How many MQTT-SN messages on port 1885 are sent by the clients to a broker in the local machine?

Answer: 0

To solve Question 7, we use this filter udp.port == 1885 && ip.dst == 127.0.0.1 .



The condition udp.port == 1885 ensures that only messages associated with the MQTT-SN service (which typically runs on UDP port 1885) are included. The condition ip.dst == 127.0.0.1 restricts the results to packets whose destination is the local machine (localhost), indicating that the messages are being sent to the broker.