IoT Challenge 2

Leader : Ni Giovanni 10831328

Gu Xinyue 10840236

CQ1

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

Firstly, we filter packets Confirmable PUT requests to the local CoAP server:

ip.dst == 127.0.0.1 and coap.type == 0 and coap.code == 3

ip.dst == 127.0.0.1 and coap.type == 0 and coap.code == 3

No.	Time	Source	Destination	Protocol	Length	Info
66	0.404168588	127.0.0.1	127.0.0.1	CoAP	67	CON, MID:62422, PUT, TKN:12 a7 e7 28 fd 99 0c a5, /hello_post
5970	120.454709683	127.0.0.1	127.0.0.1	CoAP	62	CON, MID:30549, PUT, TKN:58 52 18 be bb fb 0e 63, /basic
6292	132.431648144	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:42606, PUT, TKN:10 3c ee 30 eb b6 62 b6, /living_room
6640	150.437961631	127.0.0.1	127.0.0.1	CoAP	56	CON, MID:17886, PUT, TKN:72 a5 b8 40 c7 f4 08 45
6725	156.432747756	127.0.0.1	127.0.0.1	CoAP	73	CON, MID:7773, PUT, TKN:9b e6 70 9d 82 05 8e fc, /dining_room/door
6997	179.591638314	127.0.0.1	127.0.0.1	CoAP	56	CON, MID:736, PUT, TKN:59 f2 74 1d 57 37 9b 54
8287	256.571556768	127.0.0.1	127.0.0.1	CoAP	66	CON, MID:29978, PUT, TKN:78 58 9c d0 0c c1 f6 73, /main_door
8368	261.575677345	127.0.0.1	127.0.0.1	CoAP	80	CON, MID:20520, PUT, TKN:9f 43 d8 0a 1f a8 24 49, /living_room/temperature
9100	296.627894085	127.0.0.1	127.0.0.1	CoAP	56	CON, MID:15698, PUT, TKN:d7 db 48 b0 78 93 5a 67
9320	304.672633300	127.0.0.1	127.0.0.1	CoAP	62	CON, MID:53777, PUT, TKN:45 ca c1 5b ab f2 da 2c, /basic
9370	308.632831307	127.0.0.1	127.0.0.1	CoAP	61	CON, MID:31613, PUT, TKN:3d 9b af ce 46 41 52 c9, /test
9808	341.669570149	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:20749, PUT, TKN:86 f0 81 3e af cf af b0, /dining_room
10792	407.830936149	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:27412, PUT, TKN:dd 02 50 2c e1 ce 9c 96, /hello_world
11000	421.795263082	127.0.0.1	127.0.0.1	CoAP	67	CON, MID:40890, PUT, TKN:34 1b d3 9f e0 bf 88 dd, /hello_post
11443	450.623824739	127.0.0.1	127.0.0.1	CoAP	62	CON, MID:41830, PUT, TKN:a9 19 3d 83 1f 5f 04 b8, /basic
13156	568.991003066	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:52719, PUT, TKN:90 ba 9c 68 c9 11 02 22, /living_room
13637	606.809045638	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:6551, PUT, TKN:24 cd 69 82 b5 50 8e 40, /hello_world
13675	609.808280799	127.0.0.1	127.0.0.1	CoAP	56	CON, MID:61545, PUT, TKN:57 0a 2e cd b7 2f 3f 2b
13826	620.566098890	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:6589, PUT, TKN:ab e6 91 f7 0f a1 95 16, /living_room/temperature
13840	621.546475735	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:30759, PUT, TKN:29 07 14 8f b1 68 ab f8, /living_room/temperature
13846	621.960299021	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:8393, PUT, TKN:a2 98 b9 3e 59 cd 04 a3, /living_room/temperature
13850	622.546404269	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:14342, PUT, TKN:3f 2a 9e 16 d0 f4 93 14, /living_room/temperature
13883	623.071715334	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:7805, PUT, TKN:39 2d 04 3b ec 68 bd f4, /living_room/temperature
13893	623.426307581	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:21174, PUT, TKN:d0 16 db 46 b3 51 0c a4, /living_room/temperature
13895	623.785089795	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:25946, PUT, TKN:ea 64 2b 05 bf fc 55 80, /living_room/temperature
14048	636.016454864	127.0.0.1	127.0.0.1	CoAP	62	CON, MID:2135, PUT, TKN:4a e5 b4 8a 0b 77 de 72, /basic

Since each request has an unique token, we filter by that and see if there are any unsuccessful response with that token from the local coap server_

Ip.src == 127.0.0.1 and coap.type == 2 and coap.code >= 128 and coap.code < 192 and coap.token== ...

Token	Received unsuccessful response?
59f2741d57379b54	yes
abe691f70fa19516	yes
24cd6982b5508e40	yes
12a7e728fd990ca5	no
570a2ecdb72f3f2b	yes
45cac15babf2da2c	yes
90ba9c68c9110222	yes
103cee30ebb662b6	no
a9193d831f5f04b8	yes
341bd39fe0bf88dd	no
3d9bafce464152c9	yes
2907148fb168abf8	yes
585218bebbfb0e63	no
78589cd00cc1f673	yes
dd02502ce1ce9c96	yes
ea642b05bffc5580	yes
4ae5b48a0b77de72	yes
d016db46b3510ca4	yes
86f0813eafcfafb0	yes
9f43d80a1fa82449	yes
72a5b840c7f40845	yes
d7db48b078935a67	yes

3f2a9e16d0f49314	yes
9be6709d82058efc	yes
a298b93e59cd04a3	yes
392d043bec68bdf4	yes

22 requests receive unsuccessful response, so the answer is 22.

CQ2

How many CoAP resources in the coap.me public server received the same number of unique Confirmable and Non Confirmable GET requests?

Firstly, we find the ip address of coap.me using (the same could be done with “nslookup” in Windows)

dns.qry.name==”coap.me”

```
coap.me: type A, class IN, addr 134.102.218.18
    Name: coap.me
    Type: A (1) (Host Address)
    Class: IN (0x0001)
    Time to live: 40 (40 seconds)
    Data length: 4
    Address: 134.102.218.18
```

Which is 134.102.218.18

Now, we start by filtering non-confirmable GET requests to the coap.me public server:

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

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

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
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
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
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
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
7015	181.456899856	10.0.2.15	134.102.218.18	CoAP	63	NON, MID:20922, GET, TKN:d0 84 02 fe b1 88 c8 31, /broken
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
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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, Block #13, /large
10905	414.749496049	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:40334, GET, TKN:3e f6 23 2c 11 df c2 3a, /weird44
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
12767	545.994341332	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:57052, GET, TKN:5e d8 f0 40 d5 c7 bb e4, /weird44
2615	13.580658025	10.0.2.15	134.102.218.18	CoAP	65	NON, MID:48681, GET, TKN:b1 83 ff cd 5c 5a fe 5e, /weird333
8428	268.634297148	10.0.2.15	134.102.218.18	CoAP	65	NON, MID:17794, GET, TKN:ad 4c cd cc 21 9b 38 18, /weird333
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
10216	369.824285241	10.0.2.15	134.102.218.18	CoAP	65	NON, MID:63723, GET, TKN:b1 10 a1 2d 77 33 a0 bf, /separate
11981	485.826985653	10.0.2.15	134.102.218.18	CoAP	69	NON, MID:18743, GET, TKN:7b c5 81 28 84 cc 5b 15, /large-create

Then, we filter confirmable get request to che coap.me public server:

ip.dst == 134.102.218.18 and coap.code == 1 and coap.type == 0

ip.dst == 134.102.218.18 and coap.code == 1 and coap.type == 0

No.	Time	Source	Destination	Protocol	Len	Info
4632	52.342998261	10.0.2.15	134.102.218.18	CoAP	58	CON, MID:10541, GET, TKN:c6 f7 42 35 d5 99 8a 9a, /3
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
7633	216.561919074	10.0.2.15	134.102.218.18	CoAP	58	CON, MID:44934, GET, TKN:1d ba f2 b5 33 a1 25 05, /3
7714	221.718378719	10.0.2.15	134.102.218.18	CoAP	58	CON, MID:23556, GET, TKN:cd 19 ee 82 79 f5 97 2e, /3
9932	349.688916333	10.0.2.15	134.102.218.18	CoAP	58	CON, MID:63075, GET, TKN:c1 31 51 b2 d9 91 ea 58, /3
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
11530	454.660836622	10.0.2.15	134.102.218.18	CoAP	58	CON, MID:64926, GET, TKN:4e 37 be d0 e5 5f 9f 52, /5
4638	52.387558740	10.0.2.15	134.102.218.18	CoAP	60	CON, MID:10542, GET, TKN:c6 f7 42 35 d5 99 8a 9a, Block #1, /3
7637	216.613529463	10.0.2.15	134.102.218.18	CoAP	60	CON, MID:44935, GET, TKN:1d ba f2 b5 33 a1 25 05, Block #1, /3
7723	221.764491356	10.0.2.15	134.102.218.18	CoAP	60	CON, MID:23557, GET, TKN:cd 19 ee 82 79 f5 97 2e, Block #1, /3
9949	349.732923307	10.0.2.15	134.102.218.18	CoAP	60	CON, MID:63076, GET, TKN:c1 31 51 b2 d9 91 ea 58, Block #1, /3
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
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
9922	349.677023821	10.0.2.15	134.102.218.18	CoAP	62	CON, MID:61114, GET, TKN:c4 91 70 66 bb 68 09 c6, /hello
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
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
4779	62.405142503	10.0.2.15	134.102.218.18	CoAP	64	CON, MID:42177, GET, TKN:1f 87 7c 78 77 86 98 e3, /weird44
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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, Block #13, /large
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
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
9709	335.734947639	10.0.2.15	134.102.218.18	CoAP	66	CON, MID:64172, GET, TKN:8d 34 ca 9f 96 2e 4b 8f, /location1
11922	479.884084267	10.0.2.15	134.102.218.18	CoAP	69	CON, MID:31400, GET, TKN:17 4e ef 94 05 91 05 7b, /multi-format
11993	486.686391985	10.0.2.15	134.102.218.18	CoAP	69	CON, MID:19509, GET, TKN:1b ee 9f 34 d4 3d e0 b0, /multi-format
9658	329.663605041	10.0.2.15	134.102.218.18	CoAP	72	CON, MID:6332, GET, TKN:92 d6 1f 34 ba 39 1d 9b, /location-query

Having these two tables, we compare the number of Confirmable and Non Confirmable GET requests received by each resource:
only resources: /large, /secret, /validate received the same number of both types of GET requests.
Therefore, the answer is 3.

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

Firstly, determine ip address of “broker.hivemq.com”:

dns.qry.name == “broker.hivemq.com”

```
▾ broker.hivemq.com: type A, class IN, addr 35.158.43.69
    Name: broker.hivemq.com
    Type: A (1) (Host Address)
    Class: IN (0x0001)
    Time to live: 60 (1 minute)
    Data length: 4
    Address: 35.158.43.69
▾ broker.hivemq.com: type A, class IN, addr 35.158.34.213
    Name: broker.hivemq.com
    Type: A (1) (Host Address)
    Class: IN (0x0001)
    Time to live: 60 (1 minute)
    Data length: 4
    Address: 35.158.34.213
▾ broker.hivemq.com: type A, class IN, addr 18.192.151.104
    Name: broker.hivemq.com
    Type: A (1) (Host Address)
    Class: IN (0x0001)
    Time to live: 60 (1 minute)
    Data length: 4
    Address: 18.192.151.104
```

We obtain three different ip: 35.158.43.69; 35.158.34.213; 18.192.151.104.

Now, we can filter packets from MQTT clients that subscribe to the public broker HiveMQ using multi-level wildcards:

ip.dst == 18.192.151.104 and mqtt.msgtype == 8 and mqtt.topic contains “#”

ip.dst == 35.158.43.69 and mqtt.msgtype == 8 and mqtt.topic contains “#”

ip.dst == 35.158.34.213 and mqtt.msgtype == 8 and mqtt.topic contains “#”

ip.dst == 18.192.151.104 and mqtt.msgtype == 8 and mqtt.topic contains "#"

No.	Time	Source	Destination	Protocol	Len	Info
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/#]
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/#]
3693	26.268559277	10.0.2.15	18.192.151.104	MQTT	91	Subscribe Request (id=13) [factory/department3/floor0/#]
3362	21.206357493	10.0.2.15	18.192.151.104	MQTT	94	Subscribe Request (id=15) [university/building2/section0/#]

The second and third filters don’t return any packets, since there are no subscribers using those two IP addresses.

The first filter returns 6 packets. However, since the exercise asks for *different* clients, we need to check the source port of each packet to determine whether the requests were sent by the same client.

Let’s just open as example the first packet

```
    Source Address: 10.0.2.15
    Destination Address: 18.192.151.104
    [Stream index: 3]
▸ Transmission Control Protocol, Src Port: 57863, Dst Port: 1883, Seq: 459, Ack: 351, Len: 38
```

The source port of the first packet is 57863.

Continuing this way, we find that there are four different source ports: 54449, 38619 (which sent 3 requests), 38641, and 57863.

So the answer is 4.

CQ4

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

Filter CONNECT packet with topic having as first level “university”:

mqtt.msgtype == 1 and mqtt.willtopic matches “^university/”

mqtt.msgtype == 1 and mqtt.willtopic matches "^university/"

No.	Time	Source	Destination	Protocol	Len	Info
4	0.000117188	::1	::1	MQTT	176	Connect Command

Answer: 1

CQ5

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

Firstly, we identified the resources that have a last will message. In these resources, the broker notifies

subscribers of an unexpected shutdown of the publisher.

mqtt.msgtype == 1 and mqtt.willmsg_len > 0

mqtt.msgtype == 1 and mqtt.willmsg_len > 0

No.	Time	Source	Destination	Protocol	Leng	Info
4	0.000117188	::1	::1	MQTT	176	Connect Command
196	2.116585177	10.0.2.15	5.196.78.28	MQTT	126	Connect Command
352	5.034840089	10.0.2.15	5.196.78.28	MQTT	123	Connect Command
557	7.043177949	10.0.2.15	5.196.78.28	MQTT	120	Connect Command

We found 4 packets, and inspecting each packet, we found 4 resources:

- university/department12/room1/temperature
- metaverse/room2/floor4
- hospital/facility3/area3
- metaverse/room2/room2

Now for each resource, we filter all subscribers who subscribed without using a wildcard:

mqtt.msgtype == 8 and mqtt.topic == university/department12/room1/temperature

mqtt.msgtype == 8 and mqtt.topic == metaverse/room2/floor4

mqtt.msgtype == 8 and mqtt.topic == hospital/facility3/area3

mqtt.msgtype == 8 and mqtt.topic == metaverse/room2/room2

The first filter returns 3 clients (different source ports: 39551, 53557, 41789)

mqtt.msgtype == 8 and mqtt.topic == university/department12/room1/temperature

No.	Time	Source	Destination	Protocol	Leng	Info
304	4.097040463	::1	::1	MQTT	136	Subscribe Request (id=1) [university/department12/room1/temperature]
154	2.082593293	::1	::1	MQTT	136	Subscribe Request (id=1) [university/department12/room1/temperature]
121	1.083118347	::1	::1	MQTT	136	Subscribe Request (id=1) [university/department12/room1/temperature]

The remaining filters return 0 result.

So the answer is 3.

CQ6

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

First, we found the IP address of the public Mosquitto broker using:

dns.qry.name == “test.mosquitto.org”

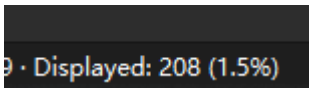
```
▾ test.mosquitto.org: type A, class IN, addr 5.196.78.28
    Name: test.mosquitto.org
    Type: A (1) (Host Address)
    Class: IN (0x0001)
    Time to live: 300 (5 minutes)
    Data length: 4
    Address: 5.196.78.28
```

Test.mosquitto.org corresponds to 5.196.78.28

Now, we can proceed by filtering PUBLISH messages directed to the public Mosquitto broker that use the

retain option and QoS level 0:

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



Answer: 208

CQ7

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

First, we can try using the filter mqttsn to check how many MQTT-SN messages are present.
However, there are no messages found.
Next, we apply the filter udp.port == 1885, but this also returns 0 results.
Therefore, we can confirm that the answer is 0.

we can cofirm that the answer is 0.