

CoAP and MQTT Traffic Analysis

The purpose of this challenge was to analyze the traffic of different communication protocols using Wireshark. In that way, as it is shown below, the implementation of filters was crucial to obtain the answer of the questions present in the current challenge.

1a. How many different CoAP clients sent a GET request to a temperature resource (.../temperature)?

Answer: After applying the filter `coap.code == 1 && coap.opt.uri_path == "temperature"`, generally found in the 8 packets sent from source to destination, which are the same IP so there is only **1** CoAP Client which sends GET requests to a temperature resource shown in the screen shot .

The figure shows a Wireshark interface with a packet capture named "example1.pcapng". The filter bar displays the selected packets: "coap.code == 1 && coap.opt.uri_path == "/>

1b. For each of the clients found in 1a), write the MID of the longest CoAP response (any response) received by the client.

Answer: The biggest packet related to the “Time delta from previous capture frame “ which shown in the picture, the MID if the “ time delta from previous capture frame “ is : **12425**

File Edit View Go Capture Analyze Statistics Telephony Analysis Tools Help
example1.pcapng

Copy code == 1 64 ccap.net url path == "temperature"

No	Time	Source	Destination	Protocol	Length	Info
31256	588.872934760	127.0.0.1	127.0.0.1	CoAP	80	NON, MID=47478, GET, TKN=a0 b0 74 14 68 c0 c2 8a, /living room.
4145	582.92323588	127.0.0.1	127.0.0.1	CoAP	88	NON, MID=28177, GET, TKN=c2 24 c4 c7 57 40 c8 c0, /living room.
4840	588.864306858	127.0.0.1	127.0.0.1	CoAP	80	NON, MID=62866, GET, TKN=01 ae 32 38 b7 ab c3 5c, /living room.
4897	513.594659545	127.0.0.1	127.0.0.1	CoAP	74	CON, MID=16587, GET, End of Block ab, /setting/temperature
4443	214.773784714	127.0.0.1	127.0.0.1	CoAP	77	CON, MID=10588, GET, TKN=89 b0, End of Block a0, /setting room.
4838	584.623836451	127.0.0.1	127.0.0.1	CoAP	88	NON, MID=7385, GET, TKN=c0 89 29 60 c3 c0, /living room.
16822	387.483485421	127.0.0.1	127.0.0.1	CoAP	80	NON, MID=20998, GET, TKN=87 a3 20 c8 a4 cc 02 f0, /living room.
11859	427.678487597	127.0.0.1	127.0.0.1	CoAP	74	CON, MID=47147, GET, End of Block ab, /setting/temperature
11150	809.365535523	127.0.0.1	127.0.0.1	CoAP	74	CON, MID=17420, GET, TKN=a1 a4 c0 c0 c0, /block ab, /living room.

* Frame 11150: 77 bytes on wire (616 bits), 77 bytes captured (616 bits) on interface any, id 0

- Interface id: 0 (any)
- Encapsulation type: Linux cooked-mode capture (20)
- Arrival Time: Jan 27, 2024 16:08:27.380955099 CEST
- Time shift for this packet: 0.000000000 seconds
- EPOCH Time: 1706387627.380955099 seconds
- Time delta from previous captured frame: 0.168236220 seconds
- Time delta from capture start: 0.851031880 seconds
- Time since reference or first frame: 430.276833557 seconds

Frame Number: 11150

Frame Length: 77 bytes (616 bits)

Capture Length: 77 bytes (616 bits)

[Frame is marked: False]

[Frame is loaded: False]

```

0000  00 00 03 04 00 00 00 00 00 00 00 00 20 22 66 00  ....x...x...
0010  45 00 00 38 4d 45 40 40 40 11 0f 0a 7f 00 00 01  ...E...E...E...
0020  7f 00 00 01 a1 38 16 33 30 29 74 3c 42 01 38 09  ...-B...-B...
0030  00 00 00 00 00 70 02 00 00 07 4f 74 0f 00 00 00  ...T...T...T...
0040  74 85 6d 70 85 72 61 74 74 75 82 01 c1 82
                                temperature:

```

Time delta from previous displayed frame: 0.168236220 seconds
Packets: 12824 - Displayed: 9 (0.1%)
Profile: Default

2a. How many CoAP POST requests directed to the "coap.me" server did NOT produce a successful result?

Answer: There are **15** non confirmable requests which satisfies this question, which can be reached with the filter `coap.type == 1`, used to find non-confirmable or not successful requests). In addition, it was implemented the filter `(coap.code == 2) && (ip.dst == 134.102.218.18) && (coap.type == 1)` to find CoAP POST requests directed to the "coap.me" server that did NOT produce a successful.

No.	Time	Source	Destination	Protocol	Length	Info
56	25.667497670	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:41294, POST, TKN:e6 83 17 39 e5 be 6d 11, /weird33
107	44.765093072	10.0.2.15	134.102.218.18	CoAP	61	NON, MID:4495, POST, TKN:b6 48 74 44 b1 e3 2d 01, /test
3196	160.819530847	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:35316, POST, TKN:fd ec 27 1c 81 d2 a1 ff, /weird55
4498	218.012907229	10.0.2.15	134.102.218.18	CoAP	58	NON, MID:3516, POST, TKN:ce 73 eb 58 6e 37 f6 13, /3
5335	261.911645270	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:50806, POST, TKN:d8 76 b7 bb 38 ae b2 30, /weird44
5640	273.949869320	10.0.2.15	134.102.218.18	CoAP	63	NON, MID:21142, POST, TKN:10 64 e3 b9 04 88 ce 63, /secret
9233	316.033444875	10.0.2.15	134.102.218.18	CoAP	62	NON, MID:538, POST, TKN:c1 c2 7b dd 15 96 2d 9e, /large
9444	328.067206289	10.0.2.15	134.102.218.18	CoAP	72	NON, MID:63135, POST, TKN:2c f9 cd 3a 15 ed 3b 4a, /location-
9967	356.085312077	10.0.2.15	134.102.218.18	CoAP	63	NON, MID:39365, POST, TKN:3b f2 e9 64 41 df d2 db, /broken
10278	376.095368701	10.0.2.15	134.102.218.18	CoAP	69	NON, MID:32440, POST, TKN:1c b6 09 83 83 44 88 76, /large-cre-
10369	382.130800715	10.0.2.15	134.102.218.18	CoAP	61	NON, MID:59486, POST, TKN:b8 49 41 c3 e0 fe 7c 31, /sink
10546	393.159636446	10.0.2.15	134.102.218.18	CoAP	69	NON, MID:38549, POST, TKN:e7 1f c5 d5 f5 93 68 50, /multi-for-
11035	425.210817907	10.0.2.15	134.102.218.18	CoAP	62	NON, MID:12224, POST, TKN:d9 ac 6b b4 99 61 99 76, /hello
11532	455.315849234	10.0.2.15	134.102.218.18	CoAP	66	NON, MID:55666, POST, TKN:e0 d6 7e d4 2e 5b 26 6d, /location1
11849	466.323531875	10.0.2.15	134.102.218.18	CoAP	62	NON, MID:64977, POST, TKN:1e 5f ec cc 9c c5 3e 7d, /weird

2b. How many requests from 2a) are directed to a "weird" resource? (resources like /weirdXX)?

Answer: There are **4** packets with these attributes.

Filter: `((coap.code == 2) && (ip.dst == 134.102.218.18) && (coap.type == 1)) && (coap.opt.uri_path contains "weird")`

No.	Time	Source	Destination	Protocol	Length	Info
56	25.667497670	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:41294, POST, TKN:e6 83 17 39 e5 be 6d 11, /weird33
3196	160.819530847	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:35316, POST, TKN:fd ec 27 1c 81 d2 a1 ff, /weird55
5335	261.911645270	10.0.2.15	134.102.218.18	CoAP	64	NON, MID:50806, POST, TKN:d8 76 b7 bb 38 ae b2 30, /weird44
11849	466.323531875	10.0.2.15	134.102.218.18	CoAP	62	NON, MID:64977, POST, TKN:1e 5f ec cc 9c c5 3e 7d, /weird

3a. How many MQTT Publish messages with qos=2 are RECEIVED by the clients running in the machine capturing the traffic?

Answer: 32 packets – 9 bad formed (no content packets) = 23 packets

Filter(s) :

`mqtt.msgtype == 3 && mqtt.qos == 2 && (ip.dst == 127.0.0.0/8 or ip.dst == 10.0.0.0/8)`

`mqtt.msgtype == 3 && mqtt.qos == 2 && sll.pkttype == 0`

Motivation: Both filters get the same results. The first gets all the messages received by clients which are in the auto-loop domain and in the router domain addresses. There could be other addresses but in this case they are not, so it is useful.

It is checked with another filter, which gets the packets labeled as "send to us"; In the 'Linux cooked capture v1 / Packet type' section with the value 'Unicast to us (0)'. As it was said the same number of packets are obtained.

3b. How many clients are involved in the messages found in 3a)?

Answer: With any of the filters used before look into the *Statistics > Conversations > UDP* section and check how many different lines are there (because the packets are sent just in one direction of the communication). It is not looked at the IPv4 section because there are clients with the same IP address but using different ports.

Ethernet	IPv4 - 2	IPv6	TCP - 4	UDP										
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A	
10.0.2.15	59385	3.65.168.153	1883	2	413	0	0	2	413	99.765350	6.7104	0	0	
127.0.0.1	51523	127.0.0.1	1883	10	1.691	10	1.691	0	0	68.002314	24.4586	553	0	
127.0.0.1	44887	127.0.0.1	1883	10	1.933	10	1.933	0	0	72.941460	19.8806	777	0	
127.0.0.1	32965	127.0.0.1	1883	10	1.790	10	1.790	0	0	73.590104	27.9408	512	0	

3c. What are the MQTT Message identifiers (ID) of the subscribe requests that let the client receive the messages found in 3a)?

mqtt.msgtype == 8 -> check manually for the topics we got in the 3a answer.

After implementing the filter mentioned before, it is shown below a table with the topics of the published messages in 3a, where each of them is associated with the topic for which each client is subscribed and its corresponding ID.

Topic Publication Message:	Topic Subscription Message	ID
<i>hospital/facility2/room4</i>	<i>hospital/+/+</i>	4
<i>hospital/facility2/section0</i>	<i>hospital/+/+</i>	4
<i>hospital/facility2/room4/temperature</i>	<i>hospital/+/#</i>	7
<i>hospital/building5/area2/temperature</i>	<i>hospital/+/#</i>	7
<i>hospital/department5</i>	<i>hospital/+</i>	6
<i>hospital/facility2/section0</i>	<i>hospital/+/+</i>	4
<i>hospital/facility2/room4/temperature</i>	<i>hospital/+/#</i>	7
<i>hospital/building5/area2/temperature</i>	<i>hospital/+/#</i>	7
<i>hospital/department5</i>	<i>hospital/+</i>	6

Topic Publication Message:	Topic Subscription Message	ID
<i>factory/facility2</i>	<i>factory/+ factory/#</i>	3 & 1 & 17
<i>factory/room2/area2</i>	<i>factory/#</i>	1&17 (2 subscriptions for the same topic)
<i>factory/room2/room4</i>	<i>factory/#</i>	1&17 (2 subscriptions for the same topic)

Topic Publication Message:	Topic Subscription Message	ID
<i>factory/building5</i>	<i>factory/+ factory/#</i>	3 & 1 & 17

Topic Publication Message:	Topic Subscription Message	ID
<i>metaverse/facility2/room4</i>	<i>metaverse/+/+</i>	3
<i>metaverse/facility2/floor1</i>	<i>metaverse/+/+</i>	3
<i>metaverse/room2</i>	<i>metaverse/+</i>	19
<i>metaverse/building5</i>	<i>metaverse/+</i>	19
<i>metaverse/facility2/room4</i>	<i>metaverse/+/+</i>	3
<i>metaverse/facility2/section0/hydraulic_valve</i>	-	-
<i>metaverse/facility2/area2/pollution</i>	-	-
<i>metaverse/facility2/section0/deposit</i>	-	-
<i>metaverse/building5/room4/deposit</i>	-	-
<i>metaverse/facility2/room4/deposit</i>	-	-

Topic Publication Message:	Topic Subscription Message	ID
<i>university/room2/area2</i>	-	-
<i>university/facility2/floor1/pollution</i>	-	-
<i>university/department5</i>	-	-
<i>university/building5/area2</i>	-	-

4a. How many MQTT clients sent a subscribe message to a public broker using at least one wildcard?

Filter: `mqtt.msgtype == 8 && (mqtt.topic contains "#" || mqtt.topic contains "+") && (ip.dst != 127.0.0.0/8) && (ip.dst != 10.0.0.0/8)`

Motivation: Considering messages to a public broker, it is considered public brokers all the addresses that are not the auto-loop and the router interface ones. For that reason, those directions were excluded and checked if those were actually delivered to public brokers.

Answer: 16

4b. Considering clients found in 4a), how many of them WOULD receive a publish message directed to the topic: "metaverse/facility4/area0/light"

mqtt.msgtype == 8 && (mqtt.topic contains "#") mqtt.topic contains "+" and ip.dst != 127.0.0.0/8 and ip.dst != 10.0.0.0/8 and (mqtt.topic contains "metaverse" mqtt.topic == "/#")						
Time	Source	Destination	Protocol	Length	Info	
202 57.684689968	10.0.2.15	91.121.93.94	MQTT	97	Subscribe Request (id=3) [metaverse/department2/+pollution]	
290 60.535760441	10.0.2.15	3.65.168.153	MQTT	80	Subscribe Request (id=4) [metaverse/room3/#]	
296 60.690118129	10.0.2.15	91.121.93.94	MQTT	87	Subscribe Request (id=6) [metaverse/+area0/light]	
419 65.540914755	10.0.2.15	3.65.168.153	MQTT	88	Subscribe Request (id=7) [metaverse/kcbplh/floor5/#]	
459 66.714654956	10.0.2.15	91.121.93.94	MQTT	82	Subscribe Request (id=10) [metaverse/+floor5]	
551 69.497044483	10.0.2.15	3.65.168.153	MQTT	90	Subscribe Request (id=5) [metaverse/facility4/light]	
789 75.513346282	10.0.2.15	3.65.168.153	MQTT	89	Subscribe Request (id=9) [metaverse/+area0/humidity]	
1356 87.531947583	10.0.2.15	3.65.168.153	MQTT	81	Subscribe Request (id=18) [metaverse/kcbplh/#]	

5. How many MQTT ACK messages in total are received by clients who connected to brokers specifying a client identifier shorter than 15 bytes and using MQTT version 3.1.1?

Answer: In total there are 2 MQTT ACK messages using version 3.1.1 and with a client identifier length shorter than 15 bytes. To find them, the filter `mqtt.ver == 4 and mqtt.clientid_len < 15` was used, where the mqtt.ver equal to 4 is corresponding to the 3.1.1 version and the client identifier was discriminated to be less than 15 bytes.

Filter: `mqtt.ver == 4 and mqtt.clientid_len < 15`

No.	Time	Source	Destination	Protocol	Length	Info
402	65.123715869	10.0.2.15	91.121.93.94	MQTT	81	Connect Command
570	70.007164573	127.0.0.1	127.0.0.1	MQTT	102	Connect Command

Frame 570: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface ^

- Linux cooked capture v1
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- Transmission Control Protocol, Src Port: 44887, Dst Port: 1883, Seq: 1, Ack: 1, Len: 102
- MQ Telemetry Transport Protocol, Connect Command
 - Header Flags: 0x10, Message Type: Connect Command
 - Msg Len: 32
 - Protocol Name Length: 4
 - Protocol Name: MQTT
 - Version: MQTT v3.1.1 (4)
 - Connect Flags: 0xc2, User Name Flag, Password Flag, QoS Level: At most once delivered
 - Keep Alive: 457
 - Client ID Length: 7
 - Client ID: apzsfux
 - User Name Length: 5
 - User Name: woody
 - Password Length: 4

6a. How many MQTT subscribe requests with message ID=1 are directed to the HiveMQ broker?*

Answer: First, the filter `dns.qry.name matches "HiveMQ"` was implemented, where it was found that the answer for the broker is given in the IPs 3.65.168.153 and 3.66.35.116 as it is shown in the following image.

Answers

- > broker.hivemq.com: type A, class IN, addr 3.65.168.153
- > broker.hivemq.com: type A, class IN, addr 3.66.35.116

Moreover, after applying a filter for the type of message equal to subscribe request (8) and the message identifier equal to 1, for the IP addresses of the HiveMQ broker, **3** subscribe requests were detected, for the topics “university/department2/floor5”, “house/kcbplh/section2” and “hospital/kcbplh/#”

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda						
mqtt.msgtype == 8 && mqtt.msgid == 1						
No.	Time	Source	Destination	Protocol	Length	Info
137	54.681549765	10.0.2.15	91.121.93.94	MQTT	90	Subscribe Request (id=1) [university/facility4/area0]
169	56.327726604	127.0.0.1	127.0.0.1	MQTT	96	Subscribe Request (id=1) [factory/kcbplh/floor5]
172	56.524975163	10.0.2.15	3.65.168.153	MQTT	92	Subscribe Request (id=1) [university/department2/floor5]
214	58.328498103	127.0.0.1	127.0.0.1	MQTT	87	Subscribe Request (id=1) [house/kcbplh]
261	59.331285698	127.0.0.1	127.0.0.1	MQTT	105	Subscribe Request (id=1) [factory/facility4/+pollution]
264	59.333780153	127.0.0.1	127.0.0.1	MQTT	85	Subscribe Request (id=1) [factory/#]
268	59.476446713	10.0.2.15	3.66.35.116	MQTT	85	Subscribe Request (id=1) [house/kcbplh/section2]
302	61.336290454	127.0.0.1	127.0.0.1	MQTT	100	Subscribe Request (id=1) [hospital/kcbplh/section2]
351	63.490223942	10.0.2.15	3.65.168.153	MQTT	80	Subscribe Request (id=1) [hospital/kcbplh/#]
409	65.343255142	127.0.0.1	127.0.0.1	MQTT	92	Subscribe Request (id=1) [house/department2]

6b. How many publish messages are received by the clients thanks to the subscribe requests found in 6a)

Answer: If only the topics of the subscribe request message in point 6a are considered, there are not published messages that were received. To confirm, the filter (*mqtt.msgtype* == 3 or *mqtt.msgtype* == 5) was used to discriminate just the message type under Publish messages or Publish Received, but after searching specifically the subscribed topics of point 6a, there was not any result.

7a. How many MQTT-SN (on port 1885) publish messages sent after the hour 3:59PM (Milan Time) are directed to topic 6?

Answer: To find all the published messages directed to topic 6, the filter *mqtt.sn.topic.id* == 6 was used, however only **3** published messages were sent after 3:59 PM for topic 6 in the MQTT-SN protocol, on January 27th 2024.

7b. Explain possible reasons why messages in 7a) are not handled by the server

Answer: One of the possible reasons the messages are not handled by the server is because the specific port required for the communication (in this case, port 1885 for MQTT-SN) is unreachable or closed. In addition, as MQTT-SN is implemented for a sensor network where the connection is not reliable, there could be connectivity issues between the MQTT-SN client and the server, such as network outages or firewall restrictions, thus the publish message may not reach the server.