IoT – Home Challenge 3

Arnab Mondal - 10714338
Carlo Pezzoli - 10500665
Massimiliano Sica - 10558133

Answer 1

First of all, we visualize only those two packets using the following filter:
coap.mid == 3978 || coap.mid == 22636
Then we can conclude that packet with id 3978 is a confirmable message while packet with id 22636 is a non-confirmable message.

Answer 2

Yes, the response comes back with message 6953 and the response time is 0.113150167 seconds. It is easy to find this since wireshark directly "links" CoAP req/resp messages.

Answer 3

Since the question asks for "replies of type confirmable", we need to filter for the ACKs received by localhost and their content. The filter used is:
ip.dst == 127.0.0.1 && coap.type == 2 && coap.code == 69
In total the packets are 8.

Answer 4

First of all we search for the messages that contains
mqtt.username == jane
and the results are 4 connect message. From those messages we retrieve the IP address and the port this specific client is using. With those info, we search for all its publish messages with the filter:
(ip.src == 127.0.0.1 && tcp.srcport == 50985 && mqtt.msgtype==3) || (ip.src == 127.0.0.1 && tcp.srcport == 40005 && mqtt.msgtype==3) || (ip.src == 127.0.0.1 && tcp.srcport == 40989 && mqtt.msgtype==3) || (ip.src == 127.0.0.1 && tcp.srcport == 42821 && mqtt.msgtype==3)
The result are 7 messages, but only 6 of them contain the topic "factory/department*/+".
Note: we are sure those publish message are of the client "jane" since using the same filter but with mqtt.msgtype==1 we can clearly see jane is the only one using those 4 sockets.

Answer 5

First of all we have to find the IP address of the hivemq broker, looking at the DNS reply. It turns out that hivemq broker has two IP addresses. Then we apply the filter:
(ip.dst == 3.120.68.56 || ip.dst == 18.185.199.22) && mqtt.conflag.willflag == 1
We have a total of 16 connect commands: only 4 have a client id (for a total of 3 distinct client id, since one client has sent 2 connect commands specifying a last will message) and the other 6 are without client id since they are not keeping the session with the broker.

Answer 6

We apply this filter in order to display all messages of publish with qos 1 and the ACKs:
(mqtt.qos == 1 && mqtt.msgtype == 3) || (mqtt.msgtype == 4)
Then we simply look at them, distinguishing between publish messages acked and not. It turns out that in total 56 publish messages with qos 1 were not acked.

Answer 7

First of all we display all the connect messages that contain a lastwill message with the filter:
mqtt.willmsg
Now we can see that only 56 clients set a last will message upon connection. By an exhaustive search based on each last will message content, we note that only in 5 cases the broker sends to the subscribers the lastwill message (so only in 5 cases the publisher dies). Of those 5 lastwill messages forwarded to the clients, 4 are with qos 1 and only 1 with qos 0.

Answer 8

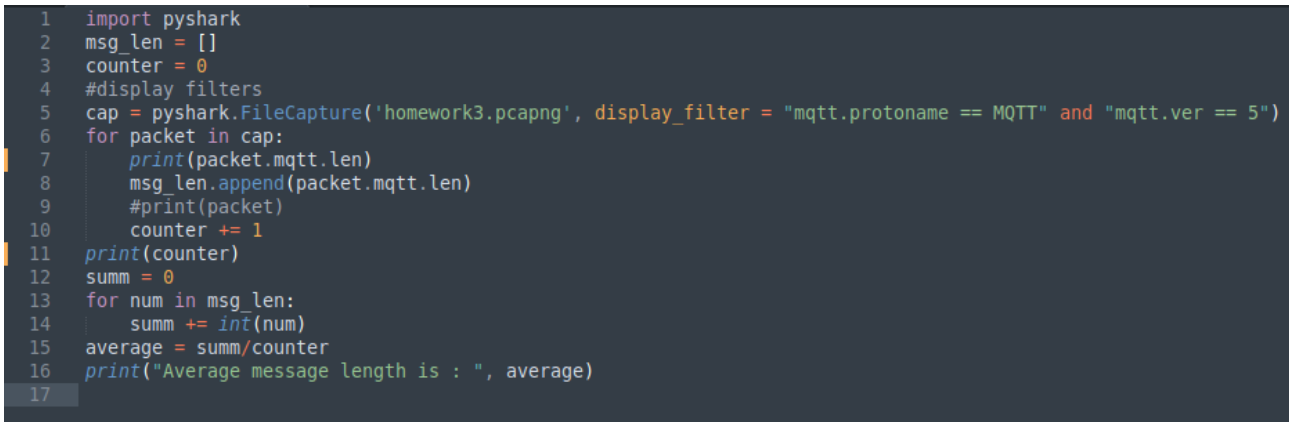
With this filter we acquire information about the socket of the client:
mqtt.clientid == 4m3DWYzWr40pce6OaBQAfk
Then we search for its publish message with qos >0 with this:
ip.src == 10.0.2.15 && tcp.srcport == 58313 && mqtt.msgtype==3 && mqtt.qos > 0
He publishes only 1 message with this characteristics, on topic "factory/department1/section1/deposit". Now we look at the subscribers of the topic with the following filter, that include also the IP of the broker where the client of our interest is publishing since there are other brokers with the same topic:
(mqtt.msgtype == 8 && mqtt.topic == "factory/department1/section1/deposit" && ip.dst == 5.196.95.208)
It turns out that when our client publishes the messages, the topic has no subscriber. Only later on there will be a client that will subscribe to that topic, but since the publish message has no retain flag, he will not receive that message.

Answer 9

We display only the messages of interest with the filter:
(mqtt.msgtype==1) && (mqtt.ver == 5)
Now we calculate the average length and it is 91.079 bytes if we consider the entire frame and 30.222 bytes if we consider the mqtt message only (read from mqtt.len, see python code below). The length of those messages is different since some messages can include additional optional information, such as lastwill message or client authentication information.

Answer 10

The capture provided to us only cover the interval (0, 160.5). Let's see the keepalive parameter set when a client connects with the filter:
mqtt.msgtype == 1
There are 175 messages. Messages with a keep alive time higher than 161 won't need to do the req/resp ping, so we exclude them through the filter:
mqtt.msgtype == 1 && mqtt.kalive <= 161
Now only 60 messages remain. For the same reasoning we can exclude the packets for which the time they subscribe plus the keep alive time is >= 161, in this way only 27 packets remain. Now we look at those 27 clients, and it turns out that they transmit or disconnect before the keepalive timer expires, so that it is resetted. So, there are no ping req/resp since there is no need to use them.



```python
import pyshark
msg_len = []
counter = 0
#display filters
cap = pyshark.FileCapture('homework3.pcapng', display_filter = "mqtt.protoname == MQTT" and "mqtt.ver == 5")
for packet in cap:
    print(packet.mqtt.len)
    msg_len.append(packet.mqtt.len)
    #print(packet)
    counter += 1
print(counter)
summ = 0
for num in msg_len:
    summ += int(num)
average = summ/counter
print("Average message length is : ", average)
```