Internet of Things

Challenge 01.

***Student**: Marco Bendinelli* ***ID**: 10673478*

***Required Software**: Wireshark, Python*

Note: In all the answers I don't explicitly write the use of the filter "!_ws.malformed" that I have used to filter all the not malformed messages.

Answer the following questions:

Question 1:

What are the differences between the request with MID:53533 and the one with MID:42804?

The request with MID:53533 is a GET request of type Confirmable so it needs to be acknowledged.
Instead, the request with MID:42804 is a DELETE request of type Non-Confirmable, in fact there is not ACKS for such message.
In order to filter the messages having a specific Message ID I have used the Wireshark filter *coap.mid == message_ID*.

Question 2:

What is the response of message No. 2428, if any?

The message number 2428 is a DELETE request of type Non-Confirmable having ID:12935 and token:67c7229a. Knowing that, I don't expect to find and acknowledge having the same message ID, instead I expect to find a response associated with the token of the request.
So I have used the filter *coap.token == 67:c7:22:9a*, to find such response.
The response of the message No. 2428 is the message No. 2429 which is a Non-Confirmable response with code 66: it is informing us that the DELETE request is successfully executed.

Question 3:

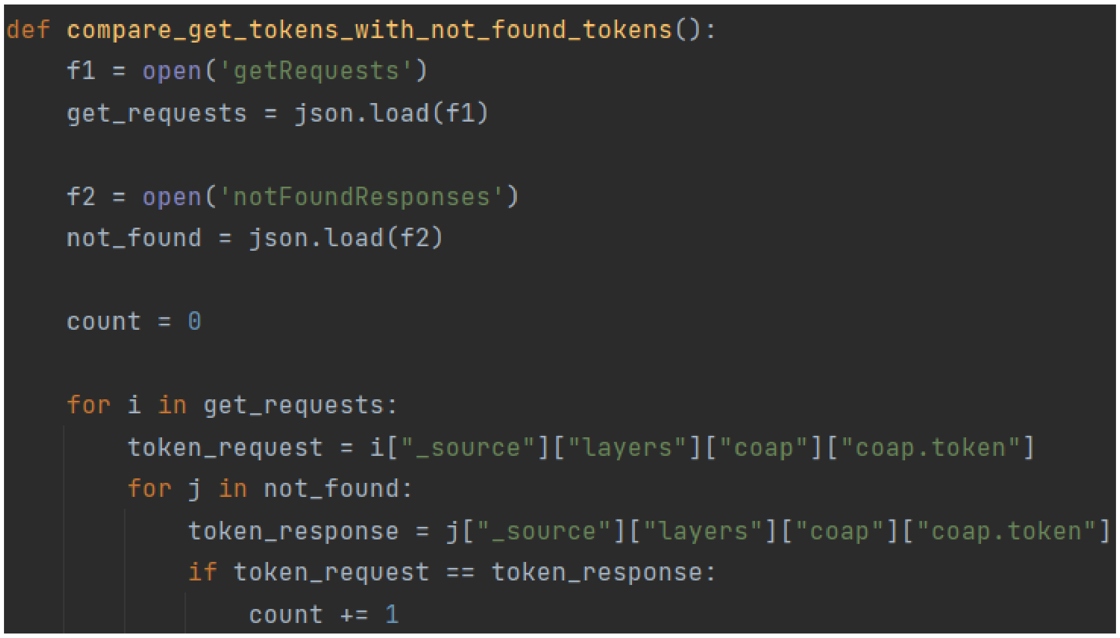
How many replies to requests of type confirmable, having result code "Content" are received by the client "localhost"?

The number of requests sent by the local host of type confirmable are 14, then I expect at most 14 replies. Used filters: *coap.type == 0 && ip.src == 127.0.0.1*.

Instead, the number of replies having result code “Content” which are received by the local host are 8. Used filters: *coap.code == 69 && ip.dst == 127.0.0.1*.
Then the answer is 8.

Question 4:
How many GET requests, excluding OBSERVE requests, have been directed to non existing resources?

I initially filter all the GET requests and I export them into a JSON file by using the following filter: *coap.code == 1*.
Then I filter all the Not Found responses by using: *coap.code == 132*, and I export also them.
For removing all the responses to the OBSERVE requests I write this simple snippet of Python code that compares the tokens of the GET requests with the tokens of the Not Found responses.

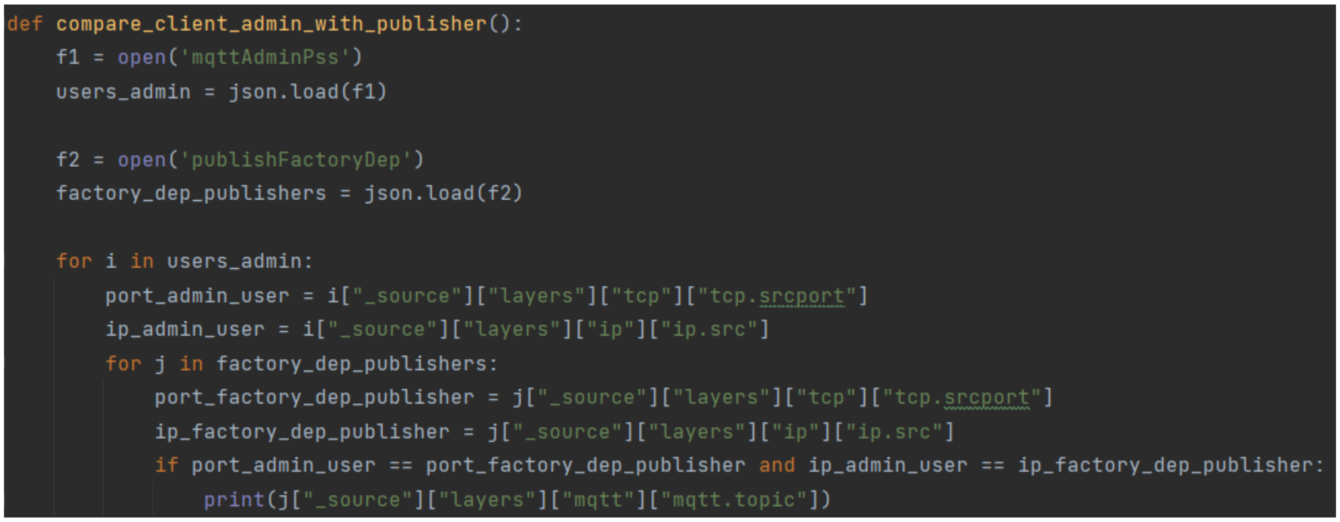


```python
def compare_get_tokens_with_not_found_tokens():
    f1 = open('getRequests')
    get_requests = json.load(f1)

    f2 = open('notFoundResponses')
    not_found = json.load(f2)

    count = 0

    for i in get_requests:
        token_request = i["_source"]["layers"]["coap"]["coap.token"]
        for j in not_found:
            token_response = j["_source"]["layers"]["coap"]["coap.token"]
            if token_request == token_response:
                count += 1
```

In this way I find that the GET requests which have been directed to non-existing resources are 6.

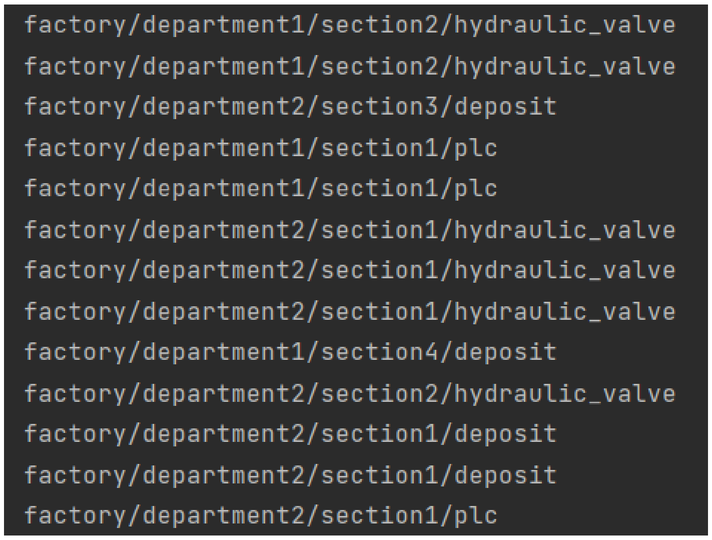
Question 5:
How many messages containing the topic “factory/department*/+” are published by a client with user password: “admin”? Where * replaces only the dep. number [0 9]

I find all the connection messages that include the field “Password: admin” by using the filter: *mqtt.passwd == "admin"*, so I export them.
By using the filters *mqtt.msgtype == 3 && mqtt.topic contains "factory/department"*, I find all the publish messages that contain the topic factory/department.
With this simple snippet of code, I find all the Users who published on topic “factory/department/#” and who logged in by using the password “admin”. For that purpose, I have just compared the socket of the Clients.



```python
def compare_client_admin_with_publisher():
    f1 = open('mqttAdminPss')
    users_admin = json.load(f1)

    f2 = open('publishFactoryDep')
    factory_dep_publishers = json.load(f2)

    for i in users_admin:
        port_admin_user = i["_source"]["layers"]["tcp"]["tcp.srcport"]
        ip_admin_user = i["_source"]["layers"]["ip"]["ip.src"]
        for j in factory_dep_publishers:
            port_factory_dep_publisher = j["_source"]["layers"]["tcp"]["tcp.srcport"]
            ip_factory_dep_publisher = j["_source"]["layers"]["ip"]["ip.src"]
            if port_admin_user == port_factory_dep_publisher and ip_admin_user == ip_factory_dep_publisher:
                print(j["_source"]["layers"]["mqtt"]["mqtt.topic"])
```

The printed topics are:



```
factory/department1/section2/hydraulic_valve
factory/department1/section2/hydraulic_valve
factory/department2/section3/deposit
factory/department1/section1/plc
factory/department1/section1/plc
factory/department2/section1/hydraulic_valve
factory/department2/section1/hydraulic_valve
factory/department2/section1/hydraulic_valve
factory/department1/section4/deposit
factory/department2/section2/hydraulic_valve
factory/department2/section1/deposit
factory/department2/section1/deposit
factory/department2/section1/plc
```

The “+” operator is a single level wildcard, so the number of topics having the string “factory/department*/+” as a topic is 0.

Question 6:

How many clients connected to the public broker “mosquitto” have specified a “will” message?

With this filter: *dns.resp.name == "test.mosquitto.org"*, I find all the DNS messages that involved the “mosquitto” broker, in this way I discover that there is a public broker with IPv4 address equal to 5.196.95.208. I discover also that there is an available IPv6 address (2001:41d0:a:fed0::1) for such broker, which is not used by our Clients that are characterized by an IPv4 address.
Finally, applying the filters *ip.dst == 5.196.95.208 && mqtt.conflag.willflag == 1*, I discover that there are 9 messages.

Question 7:

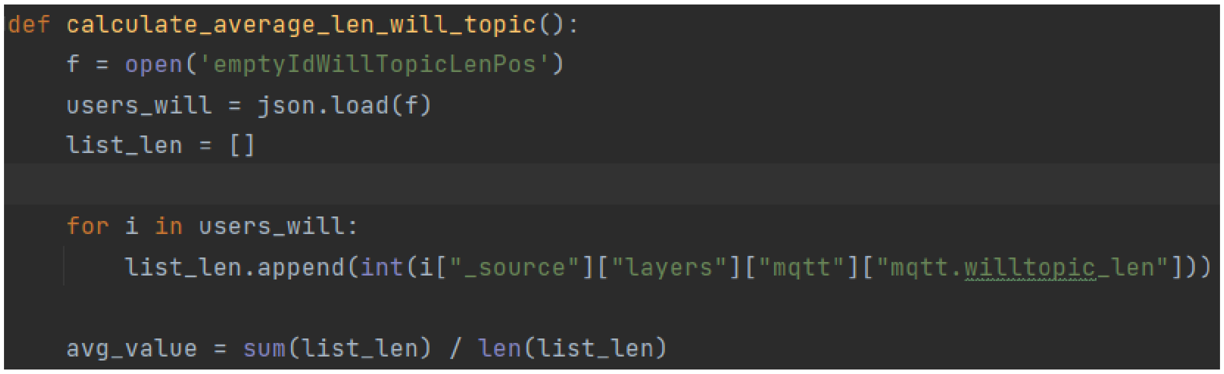
How many publishes with QoS 2 don’t receive the PUBREL?

With the filters *mqtt.qos == 2 && mqtt.msgtype == 3,* I find all the Publish message with Quality of Service equal to 2, and they are 94.
Instead, with the filter *mqtt.msgtype == 6*, I have tried to find all the PUBREL messages and I don’t find any. So, I can conclude that the answer is 94: all the publishes with QoS = 2 don’t receive a PUBREL message.

Question 8:

What is the average Will Topic Length specified by clients with empty Client ID?

With the filters *mqtt.clientid == "" && mqtt.willtopic_len* > *0*, I find all the messages with empty ID and with Will Topic Length which is strictly greater than zero, I export them and I use this snippet of code to calculate the average:



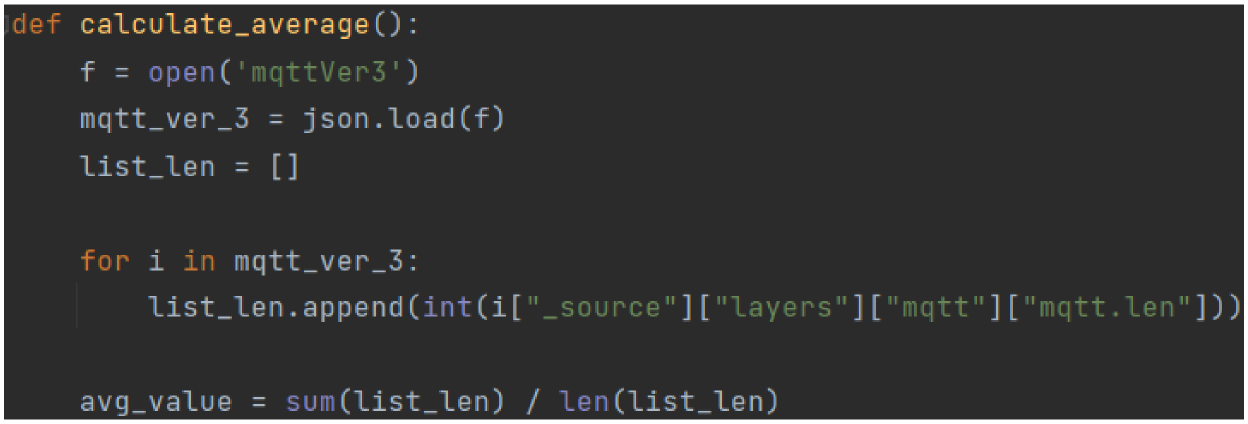
```python
def calculate_average_len_will_topic():
    f = open('emptyIdWillTopicLenPos')
    users_will = json.load(f)
    list_len = []

    for i in users_will:
        list_len.append(int(i["_source"]["layers"]["mqtt"]["mqtt.willtopic_len"]))

    avg_value = sum(list_len) / len(list_len)
```

Then I find this result: 37.77272727272727

Question 9:
How many ACKs received the client with ID "6M5H8y3HJD5h4EEscWknTD"? What type(s) is(are) it(them)??

By using the filter *mqtt.clientid == "6M5H8y3HJD5h4EEscWknTD"*, I discover the IP address and the port of the Client which are 10.0.2.15:46295.
So, by using the filter *ip.dst == 10.0.2.15 && tcp.dstport == 46295 && (mqtt.msgtype == 2 || mqtt.msgtype == 9 || mqtt.msgtype == 4)*, I find all the ACK messages that are received by such Client. In particular, I find 1 Connect ACK, 3 Subscribe ACK into one single packet and 1 Publish ACK. So, in total the Client receives 5 ACK messages.

Question 10:
What is the average MQTT message length of the CONNECT messages using mqttv3.1 protocol? Why messages have different size?

With the filter *mqtt.ver == 3 && mqtt.msgtype == 1*, I find all the messages of type CONNECT that use the mqttv3.1 protocol. I export the results and I calculate the average:

```
def calculate_average():
    f = open('mqttVer3')
    mqtt_ver_3 = json.load(f)
    list_len = []

    for i in mqtt_ver_3:
        list_len.append(int(i["_source"]["layers"]["mqtt"]["mqtt.len"]))

    avg_value = sum(list_len) / len(list_len)
```
The obtained result is 63.59574468085106.
Regarding the second part of the question, the messages can have different size because they can contain optional fields such as: User Name, Password, Will Topic, Will Message, …