IoT 2023 CHALLENGE 1

(1) Name: Lorenzo Campana Person Code: 10605775

(2) Name: Irio Castrignano Person Code: 10656379

Referring to the file challenge2023_1.pcap, answer the following questions:

1. How many CoAP GET requests are directed to non-existing resources in the local CoAP server? How many of these are of type Non confirmable? **(0.2 pts)**

   Using a python script leveraging on pyshark (a python wrapper over wireshark), I discovered all the mid (for ack) and tokens (for NON) for responses from the localhost with bad status (4.04). So now I was able to filter all the request sent to the local server, and get only those with an mid or token among the one selected before. To sum up, 11 packets were found, for which 6 were of type NON.

   Results: (showing request message ids)
   CON: ['53038', '33524', '41903', '41947', '49850']

   NON: ['16289', '24158', '54308', '64436', '4501', '37241']

2. How many CoAP DELETE requests directed to the "coap.me" server did not produce a successful result? How many of these are directed to the "/hello" resource? **(0.2 pts)**

   Again, with the help of the python script, I discovered all the 10 responses with status 2.02(delete successfully) from the coap.me server host. Then, I filtered all the request towards coap.me that were not among those 10. In the end, 105 request, were 5 of them are directed to the "/hello" resource.

3. How many different MQTT clients subscribe to the public broker mosquitto using single-level wildcards? How many of these *clients* **WOULD** receive a publish message issued to the topic "hospital/room2/area0" **(0.2 pts)**

   First I discovered all the clients subscribing to a topic using a wildcard, identifying clients based on the port for which the request were made. Then I implemented a isSimilar(topic1, topic2) function, so I were able to filter all the client among this list subscribed to the topic "hospital/room2/area0". [Detailed implementation on the source code of the script]

   {('10.0.2.15', '51531'), ('10.0.2.15', '35239'), ('10.0.2.15', '43133')} -> 3

   [(('10.0.2.15', '43133'), 'hospital/+/area0'), (('10.0.2.15', '35239'), 'hospital/room2/area0')] -> 2

4. How many MQTT clients specify a last Will Message directed to a topic having as first level "university"? How many of these Will Messages are sent from the broker to the subscribers? **(0.2 pts)**

   So, since the will topic cannot contain the wildcard, we can simply use a regex expression with the matches clause from Wireshark as follow:

   mqtt and mqtt.conflag.willflag == 1 and mqtt.willtopic matches "^university/.*"

Then for each of those three clients:
{('10.0.2.15', '41083', 'university/building2/room6'), ('127.0.0.1', '44101', 'university/building3/floor2'), ('10.0.2.15', '58887', 'university/facility2/floor2')}-> 3

Then since if the broker send a last will message it will mark them as last will
mqtt and mqtt.conflag.willflag == 1 and mqtt.msgtype == 3 -> 0
which in our case doesn't return any result. In case it would, then we would have to check for each broker we previously sent a last will request to a topic starting with "university" if some messages were sent.

5. How many Publish messages with QoS = 1 are received by the MQTT clients connected to the HiveMQ broker with MQTT version 5? **(0.1 pts)**
   First we were able to find ip addresses of the broker "broker.hivemq.com" with a MQTT version 5, by finding all the clients connected to it:
   mqtt and mqtt.msgtype == 1 and ip.dst_host == "broker.hivemq.com" and mqtt.ver == 5

   Returning
   {('10.0.2.15', '60609'), ('10.0.2.15', '45635'), ('10.0.2.15', '47549'), ('10.0.2.15', '46967'), ('10.0.2.15', '42827'), ('10.0.2.15', '37401'), ('10.0.2.15', '36665'), ('10.0.2.15', '47723'), ('10.0.2.15', '57265')}

   Then for all the publish messages sent by the broker.hivemq.com broker with qos = 1
   mqtt and mqtt.msgtype == 3 and ip.src in {3.65.137.17, 52.29.173.150} and mqtt.qos == 1

   we filter all the ones that are directed towards those clients, keeping in mind that a a packet can incorporate multiple publish message mqtt layer inside. In the end the number of messages sent are 60.

6. How many MQTT-SN (on port 1885) publish messages sent after the hour 3.16PM (Milan Time) are directed to topic 9? Are these messages handled by the server? **(0.1 pts)**

   Firstly, we change the port on our wireshark application to listen to 1885 for mqtt-sn protocol. The we filter all the packets using the following filter:
   mqttsn and mqttsn.topic.id == 9 and frame.time >= "Mar 14, 2023 15:15:59.0". -> 15

   Then we can immediately notice that every publish request has a Port Unreachable response, which means that the mqtt-sn broker gateway responsible for routing messages to the proper subscriber is not running or listening on that port. So no, those messages are not handled by the server.

   For completeness, I leave here below the python scripts that I used to answer question 1, 2, 3, 5, using a well-known library pyshark that easily connect via tshark.

   Github repo: https://github.com/lorecampa/IOT_Wireshark_First_Challenge

question_1.py

```python
import pyshark

coap_code = {'GET': '1', 'POST': '2', 'PUT': '3', 'DELETE': '4'};
coap_response_code = {'NOT_FOUND': '132'}
coap_type = {'CON': '0', 'NON': '1', 'ACK': '2', 'RST': '3'};

# Iterate through each packet in the capture file
ack_mid = []
non_tokens = []
first_filter = 'coap and ip.src == 127.0.0.1 and coap.code == 132 and coap.type in {1,
2}'
first_capture = pyshark.FileCapture('challenge.pcapng', display_filter=first_filter)
for packet in first_capture:
    if (packet.coap.type == coap_type['ACK']):
        ack_mid.append(packet.coap.mid)
    elif (packet.coap.type == coap_type['NON'] and hasattr(packet.coap, 'token')):
        non_tokens.append(packet.coap.token);

second_filter = 'coap and ip.dst == 127.0.0.1 and coap.code == 1 and coap.type in {0,
1}'
second_caputure = pyshark.FileCapture('challenge.pcapng',
display_filter=second_filter)
con_mid = []
non_mid = []
for packet in second_caputure:
    if packet.coap.mid in ack_mid:
        con_mid.append(packet.coap.mid);
    elif hasattr(packet.coap, 'token') and packet.coap.token in non_tokens:
        non_mid.append(packet.coap.mid)



# Print the results
print(len(con_mid) + len(non_mid), 'CON:', con_mid, 'NON:', non_mid)

# Close the capture file
first_capture.close()
second_caputure.close()
```

question_2.py

```
from xml.etree import ElementTree
import pyshark

coap_code = {'GET': '1', 'POST': '2', 'PUT': '3', 'DELETE': '4'};
coap_response_code = {'NOT_FOUND': '132', 'DELETED': '66'}
coap_type = {'CON': '0', 'NON': '1', 'ACK': '2', 'RST': '3'};



#successfully deleted
first_filter = 'coap and coap.code == 66 and coap.type in {1, 2} and ip.src ==
134.102.218.18'
first_capture = pyshark.FileCapture('challenge.pcapng', display_filter=first_filter)
ack_mid = set();
non_token = set();

for packet in first_capture:
    if packet.coap.type == coap_type['ACK']:
        ack_mid.add(packet.coap.mid);
    elif packet.coap.type == coap_type['NON'] and hasattr(packet.coap, 'token'):
        non_token.add(packet.coap.token)


print(len(ack_mid) + len(non_token));

second_filter = 'coap and coap.code == 4 and coap.type in {0, 1} and ip.dst ==
134.102.218.18'
second_capture = pyshark.FileCapture('challenge.pcapng', display_filter=
second_filter)

result_mid = set();
hello_resource_mid = set();
n = 0;
n_hello = 0;

for packet in second_capture:
    if ((packet.coap.type == coap_type['CON'] and packet.coap.mid not in ack_mid) or
    (packet.coap.type == coap_type['NON'] and hasattr(packet.coap, 'token') and
packet.coap.token not in non_token)):
        n+=1;
        if (hasattr(packet.coap, 'opt_uri_path') and packet.coap.opt_uri_path ==
'hello'):
            hello_resource_mid.add(packet.coap.mid);
            n_hello += 1;
        else:
            result_mid.add(packet.coap.mid)


#Print results
print(n, n_hello)

# Close the capture file
```

question_3.py

```python
import pyshark


def areSimilar(t1, t2):
    #assuming that t1 is formatted correctly
    tl1 = t1.split('/');
    tl2 = t2.split('/');

    for i in range(min(len(tl1), len(tl2))):
        if tl1[i] == '#':
            return True;
        if (tl1[i] != '+' and tl1[i] != tl2[i]):
            return False;

    return len(tl1) == len(tl2);



first_filter = 'mqtt and ip.dst ==91.121.93.94  and mqtt.msgtype == 8 and mqtt.topic 
contains "+"'
first_capture = pyshark.FileCapture('challenge.pcapng', display_filter=first_filter)
clients = set();
for packet in first_capture:
    clients.add((packet.ip.addr, packet.tcp.srcport))

second_filter = 'mqtt and ip.dst == 91.121.93.94 and mqtt.msgtype == 8'
second_capture = pyshark.FileCapture('challenge.pcapng', display_filter=second_filter)
topic = "hospital/room2/area0";
registered_clients = []
for packet in second_capture:
    p_topic = packet.mqtt.topic;
    client = (packet.ip.addr, packet.tcp.srcport);
    if (areSimilar(p_topic, topic) and client in clients):
        registered_clients.append((client, p_topic))

# Print the results
print(clients, registered_clients)



# Close the capture file
first_capture.close()
second_capture.close()
```

```
        first_capture.close()
        # second_caputure.close()
```

question_5.py

```
import pyshark

first_filter = 'mqtt and mqtt.msgtype == 1 and ip.dst in {3.65.137.17, 52.29.173.150}
and mqtt.ver == 5'
first_capture = pyshark.FileCapture('challenge.pcapng', display_filter=first_filter)
clients = set();
for packet in first_capture:
    client = (packet.ip.addr, packet.tcp.srcport);
    clients.add(client);


second_filter = 'mqtt and mqtt.msgtype == 3 and ip.src in {3.65.137.17, 52.29.173.150}
and mqtt.qos == 1'
second_capture = pyshark.FileCapture('challenge.pcapng', display_filter=second_filter)

n = 0;
for packet in second_capture:
    client = (packet.ip.dst, packet.tcp.dstport);
    if (client in clients):
        mqtt_layers = packet.layers[3:]
        for mqtt in mqtt_layers:
            if (hasattr(mqtt, 'qos') and mqtt.qos == '1'):
                n += 1;

# Print the results
print(clients, n)



# Close the capture file
first_capture.close()
second_capture.close()
```