10595640_10638165_challenge1

1st Question

WireShark filter = " coap.mid == 53533 || coap.mid == 42804 " #used to found the packets

1. MID 42804 is a non Confirmable Message, whereas MID 53533 is Confirmable message. In fact, we can notice that for the second one there's a response message with the same MID flagged as ACK-message.
   In addition, analyzing the response for the MID 53533 we can notice:
   - the code of the packet starts with '2', so the GET request was successful
   - the code is 2.05 since the packet has the content requested by the GET message
2. MID 42804 uses DELETE method (code = 4), whereas MID 53533 uses a GET (code = 1) method. More specifically:
   - MID 42804 asks for the deletion of the resource link2;
   - MID 53533 asks for the resource large;
3. MID 53533 has also one option more that is the Observation option. It is set to 0 that is the code number for the 'register' mode.
4. MID 53533 has option Block2 = (NR = 5, M=1, SZX = 256). This means that the packet has been fragmented in 6 packets. More in detail, the analyzed packet is the fifth one, and another packet will arrive. Size of each block is 256 byte. MID 42804 has option Block2 = (NR=0, M=0, SZX=64). This means that the packet has not been fragmented, and there's just a single packets with size=64 byte.

2nd Question

The packet with No. 2428 has MID = 12935. If we use the filter = 'coap.response_to == 2428', we find a packet (No. 2429) with the same token of the packet number 2428, that is 67c7229a

3rd Question

Used filter = 'coap.type == 2 && coap.code == 69 && ip.dst == 127.0.0.1', where:

- coap.type == 2 means type ACK (since it is a response to a confirmable message)
- coap.code = 69 means code = 2.05
- ip.dst = 127.0.0.1 is localhost

Total number of responses = 8

4th Question

We used the following python script:

```
import subprocess as sp

query = "tshark -r homework1.pcapng -Y 'coap.code == 1 && not coap.opt.observe'"
output = sp.getoutput(query)

packets = output.split('\n')

#Getting the packet No.
for i in range(0, len(packets)):
    packets[i] = (packets[i].split(' '))[1]

count = 0
filter = '\'coap.code == 132 && coap.response_to == '
query = 'tshark -r homework1.pcapng -Y '

for pk in packets:

    pk_filter = filter+pk+'\''
    pk_query = query + pk_filter

    result = sp.getoutput(pk_query)
    if (result!=''):
        count += 1

print(count)
```

By running the script, count = 6

5th question

By using the filter `mqtt.passwd == admin` we have found all the users with that password. Than we saved all the ports number from which the ‘Connect message’ came from.

Using the filter

```
mqtt.msgtype == 3 &&  (tcp.srcport == 51565 || tcp.srcport == 41869 ||  tcp.srcport ==
 60395 || tcp.srcport == 40989 || tcp.srcport == 47135 || tcp.srcport == 44429 || tcp.s
rcport == 49649 || tcp.srcport == 60419 || tcp.srcport==49041 || tcp.srcport == 55953 |
| tcp.srcport == 56927)
```

we find 13 PUBLISH packets coming from clients with that password. Now, we have two ways of proceeding:

1. if we consider the '+' as a multilevel wildcard, we must considered all the 13 found publishes;
2. Instead, if we consider the '+' as a single level wildcard, all the 13 packets are to be discarded, and the number of publishes is 0.

6th Question

Using the filter `dns.resp.name == "test.mosquitto.org"` we can find the ip of the public mosquitto broker. Then, using the `mqtt.willtopic != "" && ip.dst == 5.196.95.208 && mqtt.msgtype == 1` filter we get all the Connect messages sent to that broker in which the will message has been specified.

Taking into account also the emptyID, different clientsID are :

- 4 different connect messages with emptyID
- 4m3DWYzWr40pce6OaBQAfk
- 4JwBgQDhNaBGrj98UYIZ0i
- 7uU0RtQermBqJwhvPRsI6J
- 6M5H8y3HJD5h4EEscWknTD
- 3N3BoDJ96jvhsaDeo5Jg6g

Therefore, we have 9 different clients connecting to a public mosquitto broker specifying a will topic message since we considered all the clients with emptyID as different clients.

7th question

Since there's no PUBREL message (using `mqtt.msgtype == 6` as filter) we can conclude that all the PUBLISH message with qos = 2 has no PUBREL message

associated.

The total number of this kind of packets is 94 and we can find them using the filter : `mqtt.msgtype == 3 && mqtt.qos == 2`

8th question

Solving this exercise, we considered only those clients, with empty clientID, that had specified as empty the field will topic.

The python script used is:

```
import os
import json

os.system('tshark -r homework1.pcapng -T json > source.json')

with open('source.json', 'rb') as fp:
    source= json.load(fp)

mqtt_packets = []
for pk in source:
    if 'mqtt' in pk['_source']['layers'].keys():
        mqtt_packets.append(pk)


_sum = 0
_count = 0
for pk in mqtt_packets:
    if pk['_source']['layers']['mqtt']['mqtt.hdrflags_tree']['mqtt.msgtype'] ==  '1' and\
        pk['_source']['layers']['mqtt']['mqtt.clientid_len'] ==  '0':
        try:
            _sum += int(pk['_source']['layers']['mqtt']['mqtt.willtopic_len'])
            _count += 1
        except:
            pass

print(_sum/_count)
```

Total Number of messages with empty client_id, and a will topic message specified = 37

→ Average Will Topic Length specified by clients with empty Client ID = 37.054

Another interpretation may be that in which all the connect messages with clientid_len = 0 are taken into account, considering a missing will_topic field as a will_topic message with length = 0.

It suffices to slightly modify the last part of the previous script as follows:

```
_sum = 0
_count = 0
for pk in mqtt_packets:
    if pk['_source']['layers']['mqtt']['mqtt.hdrflags_tree']['mqtt.msgtype'] ==  '1' and\
        pk['_source']['layers']['mqtt']['mqtt.clientid_len'] ==  '0':
        try:
            _sum += int(pk['_source']['layers']['mqtt']['mqtt.willtopic_len'])
            _count += 1
        except:
            _count += 1

print(_sum/_count)
```

Total Number of messages with empty client_id = 123

→ Average Will Topic Length specified by clients with empty Client ID = 11.146

9th question

We can leverage the NAT process.
In fact :

- with the filter `mqtt.clientid == 6M5H8y3HJD5h4EEscWknTD` we find the port binding that clientId to the source IP 10.0.2.15 (that could be a router, for instance)
- sthan with `tcp.dstport == 46295 && ip.dst == 10.0.2.15 && mqtt && (mqtt.msgtype==2 ||mqtt.msgtype==4 || mqtt.msgtype==9 || mqtt.msgtype==11)` we find all the ACKs received by that ClientID.

Total number of ACKS = 5. More in detail:

- 1 connect ACK
- 3 subscribe ACK
- 1 publish ACK

10th question

First we have filtered on : “mqtt.msgtype == 1 && mqtt.ver == 3” (note that type = 1 represent the connect messages, while ver = 3 is used to indicate the version 3.1 as requested); Then all the packets are displayed and therefore it is possible to compute: cumulative summation = 2989 total_message = 47 avg = 63,595

The messages are of different length because the some fields of the Connect messages may be of different length.