POLITECNICO
MILANO 1863

Internet of Things
Challenge 2

Andrea Pesciotti
10715428

Teachers:
Redondi Alessandro Enrico Cesare
Fabio Palmese
Boiano Antonio

April 2025

1 Packet Sniffing

1.1 CQ1

Q: How many different Confirmable PUT requests obtained an unsuccessful response from the local CoAP server?

1.1.1 Filter Confirmable PUT requests from localhost

The following Wireshark display filter was used to extract all PUT requests with Confirmable type, coming from the local server (127.0.0.1):

```
coap.code == 3 && coap.type == 0 && ip.addr == 127.0.0.1
```

This filter returned a total of **26** packets.

1.1.2 Identify requests with and without Uri-Path

CoAP requests without a `Uri-Path` are typically invalid and result in a `4.04 Not Found` response.
To isolate those requests, a refined filter was used:
Valid requests (with Uri-Path):

```
coap.code == 3 && coap.type == 0 && ip.addr == 127.0.0.1 && coap.
    opt.uri_path
```

This filter returned: **22 packets**

No.	Time	Source	Destination	Protocol	Lengtl	Info
66	0.404168588	127.0.0.1	127.0.0.1	CoAP	67	CON, MID:62422, PUT, TKN:12 a7 e7 28 fd 99 0c a5, /hello_post
5970	120.454709683	127.0.0.1	127.0.0.1	CoAP	62	CON, MID:30549, PUT, TKN:58 52 18 be bb fb 0e 63, /basic
6292	132.431648144	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:42606, PUT, TKN:10 3c ee 30 eb b6 62 b6, /living_room
6725	156.432747756	127.0.0.1	127.0.0.1	CoAP	73	CON, MID:7773, PUT, TKN:9b e6 70 9d 82 05 8e fc, /dining_room/door
8287	256.571556768	127.0.0.1	127.0.0.1	CoAP	66	CON, MID:29978, PUT, TKN:78 58 9c d0 0c c1 f6 73, /main_door
8368	261.575677345	127.0.0.1	127.0.0.1	CoAP	80	CON, MID:20520, PUT, TKN:9f 43 d8 0a 1f a8 24 49, /living_room/temperature
9320	304.672633300	127.0.0.1	127.0.0.1	CoAP	62	CON, MID:53777, PUT, TKN:45 ca c1 5b ab f2 da 2c, /basic
9370	308.632831307	127.0.0.1	127.0.0.1	CoAP	61	CON, MID:31613, PUT, TKN:3d 9b af ce 46 41 52 c9, /test
9808	341.669570149	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:20749, PUT, TKN:86 f0 81 3e af cf af b0, /dining_room
10792	407.830936149	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:27412, PUT, TKN:dd 02 50 2c e1 ce 9c 96, /hello_world
11000	421.795263082	127.0.0.1	127.0.0.1	CoAP	67	CON, MID:40890, PUT, TKN:34 1b d3 9f e0 bf 88 dd, /hello_post
11443	450.623824739	127.0.0.1	127.0.0.1	CoAP	62	CON, MID:41830, PUT, TKN:a9 19 3d 83 1f 5f 04 b8, /basic
13156	568.991003066	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:52719, PUT, TKN:90 ba 9c 68 c9 11 02 22, /living_room
13637	606.809045638	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:6551, PUT, TKN:24 cd 69 82 b5 50 8e 40, /hello_world
13826	620.566098890	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:6589, PUT, TKN:ab e6 91 f7 0f a1 95 16, /living_room/temperature
13840	621.546475735	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:30759, PUT, TKN:29 07 14 8f b1 68 ab f8, /living_room/temperature
13846	621.960299021	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:8393, PUT, TKN:a2 98 b9 3e 59 cd 04 a3, /living_room/temperature
13850	622.546404269	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:14342, PUT, TKN:3f 2a 9e 16 d0 f4 93 14, /living_room/temperature
13883	623.071715334	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:7805, PUT, TKN:39 2d 04 3b ec 68 bd f4, /living_room/temperature
13893	623.426307581	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:21174, PUT, TKN:d0 16 db 46 b3 51 0c a4, /living_room/temperature
13895	623.785089795	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:25946, PUT, TKN:ea 64 2b 05 bf fc 55 80, /living_room/temperature
14048	636.016454864	127.0.0.1	127.0.0.1	CoAP	62	CON, MID:2135, PUT, TKN:4a e5 b4 8a 0b 77 de 72, /basic

Figure 1: Filtered Confirmable PUT requests with valid Uri-Path

Result: Therefore, the number of Confirmable PUT requests that obtained a successful response from the local CoAP server is:

$$\boxed{22}$$

1.2 CQ2

Q: How many CoAP resources in the coap.me public server received the same number of unique Confirmable and Non Confirmable GET requests?

1.2.1 Packet Extraction in Wireshark

The provided `.pcapng` trace was opened in Wireshark, and two filters were applied separately to isolate the relevant packets:

- **Confirmable GETs:**

```
coap.code == 1 && coap.type == 0 && ip.dst == 134.102.218.18

```

- **Non-confirmable GETs:**

```
coap.code == 1 && coap.type == 1 && ip.dst == 134.102.218.18

```

Each filtered set was exported as a CSV file.
Only the `Uri-Path` and `Token` columns were considered, which are essential to identify the resource and distinguish unique requests.

1.2.2 Python Analysis

The exported CSV files (`CQ2_CON.csv` and `CQ2_NON.csv`) were processed using the following Python script.
Each request was considered unique based on its `Token`, and the goal was to count the number of unique tokens per resource in both datasets.

```python
import csv
from collections import defaultdict

CON_FILE = 'CQ2_CON.csv'
NON_FILE = 'CQ2_NON.csv'

def count_unique_tokens(file_path):
    resource_tokens = defaultdict(set)
    with open(file_path, 'r', encoding='utf-8') as f:
        reader = csv.reader(f)
        next(reader)
        for row in reader:
            try:
                uri = row[6].strip()
                token = row[7].strip()
                if uri and token:
                    resource_tokens[uri].add(token)
            except:
                continue
    return resource_tokens

con_data = count_unique_tokens(CON_FILE)
non_data = count_unique_tokens(NON_FILE)
```

```python

matching_resources = []
for resource in con_data.keys() & non_data.keys():
    if len(con_data[resource]) == len(non_data[resource]) > 0:
        matching_resources.append(resource)
```

The code returned the following matching resources:

- /large
- /secret
- /validate

Result: The number of resources that received the same number of unique Confirmable and Non-confirmable GET requests is:

$$\boxed{3}$$

1.3 CQ3

Q: How many different MQTT clients subscribe to the public broker HiveMQ using multi-level wildcards?

1.3.1 Identifying the IP Address of the HiveMQ Broker

The hostname of the broker is `broker.hivemq.com`, but to filter packets by IP, I first needed to resolve the IP address. To do this, I applied the following display filter in Wireshark:

```
dns
```

This filter shows all DNS packets. Among the results, I located the query for `broker.hivemq.com` and observed its A record response. In this case, the IP address returned was:

```
18.192.151.104
```

1.3.2 Filtering MQTT Subscriptions with Multi-Level Wildcards

To identify relevant packets, I filtered MQTT SUBSCRIBE messages that used multi-level wildcards ('#') in their topic and were directed to the HiveMQ broker:

```
mqtt.msgtype == 8 && mqtt.topic contains "#" && ip.dst ==
    18.192.151.104
```

Where:

- `mqtt.msgtype == 8` selects MQTT `SUBSCRIBE` messages.
- `mqtt.topic contains "#"` restricts to subscriptions using multi-level wildcards.
- `ip.dst == 18.192.151.104` limits the results to subscriptions sent to the HiveMQ server.

This filter returned 6 packets.

1.3.3 Identifying Unique Clients

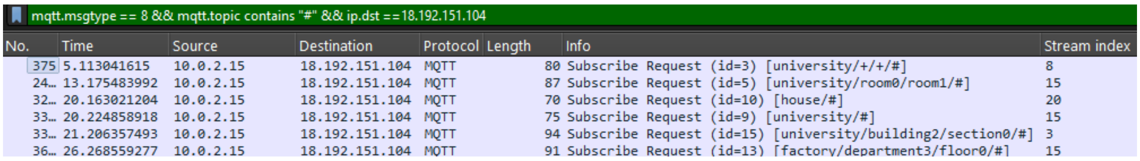
At first glance, it may seem that there are 6 different clients. However, based on a useful insight, MQTT clients can:

- Share the same IP address (e.g., behind NAT).
- Be uniquely identified by the TCP session they establish with the broker.

Wireshark assigns a unique `tcp.stream` index to each TCP connection. Therefore, to find the actual number of different clients, I added a custom column in Wireshark with the field:

```
tcp.stream
```

Then, I re-applied the previous filter and observed the `tcp.stream` values.
Out of the 6 packets, **3 of them shared the same stream index**. That means these packets were part of the same TCP session, and hence from the same MQTT client.


mqtt.msgtype == 8 && mqtt.topic contains "#" && ip.dst ==18.192.151.104

No.	Time	Source	Destination	Protocol	Length	Info	Stream index
375	5.113041615	10.0.2.15	18.192.151.104	MQTT	80	Subscribe Request (id=3) [university/+/+/#]	8
24…	13.175483992	10.0.2.15	18.192.151.104	MQTT	87	Subscribe Request (id=5) [university/room0/room1/#]	15
32…	20.163021204	10.0.2.15	18.192.151.104	MQTT	70	Subscribe Request (id=10) [house/#]	20
33…	20.224858918	10.0.2.15	18.192.151.104	MQTT	75	Subscribe Request (id=9) [university/#]	15
33…	21.206357493	10.0.2.15	18.192.151.104	MQTT	94	Subscribe Request (id=15) [university/building2/section0/#]	3
36…	26.268559277	10.0.2.15	18.192.151.104	MQTT	91	Subscribe Request (id=13) [factory/department3/floor0/#]	15

Figure 2: Requests with Stream Index filter

Result: The number of different MQTT clients that subscribed to the HiveMQ broker using multi-level wildcards is:

$$\boxed{4}$$

1.4 CQ4

Q: How many different MQTT clients specify a last Will Message to be directed to a topic having as first level "university"?

1.4.1 Filtering CONNECT Messages with Last Will

In MQTT, the Last Will Message is defined by the client during the initial connection to the broker using the `CONNECT` command.
I applied the following Wireshark display filter:

```
mqtt.msgtype == 1 && mqtt.willmsg && mqtt.willtopic
```

This filter selects all `CONNECT` packets that:

- Are of message type 1 (i.e., `CONNECT`).
- Define a Last Will Message payload.
- Specify a Last Will Topic.

1.4.2 Filtering by Topic Prefix

To narrow the results to only those clients whose Will Topic starts with `university`, the filter is defined as follows:

```
mqtt.msgtype == 1 && mqtt.willmsg && mqtt.willtopic && mqtt.
    willtopic contains "university"
```

After applying this filter, I checked that the Will Topic indeed starts with `university`, and not just containing it as a substring
The only identified Will Topic is:

- `university/department12/room1/temperature`

Result: The number of different MQTT clients that specified a Last Will Message directed to a topic under the `university` root is:

$$\boxed{1}$$

1.5 CQ5

Q: How many MQTT subscribers receive a last will message derived from a subscription without a wildcard?

1.5.1 Identify Will Topics

From the CONNECT packets in Wireshark, I inspected the `Will Topic` field and extracted the following topics:

- `university/department12/room1/temperature`
- `metaverse/room2/room2`
- `hospital/facility3/area3`
- `metaverse/room2/floor4`

However, only the following Will Topic was actually published later on as a `PUBLISH` message:

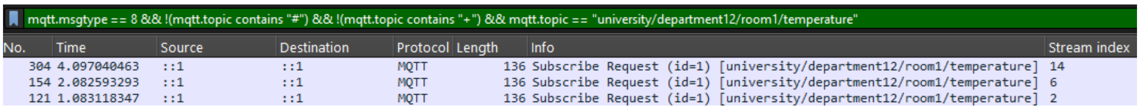
- `university/department12/room1/temperature`

1.5.2 Filter Exact Subscriptions to the Published Will Topic

To identify MQTT clients subscribed to this topic without using wildcards, I used the following Wireshark display filter:

```
mqtt.msgtype == 8 &&
mqtt.topic == "university/department12/room1/temperature"
```

This filter returned 3 packets.


mqtt.msgtype == 8 && !(mqtt.topic contains "#") && !(mqtt.topic contains "+") && mqtt.topic == "university/department12/room1/temperature"

No.	Time	Source	Destination	Protocol	Length	Info	Stream index
304	4.097040463	::1	::1	MQTT	136	Subscribe Request (id=1) [university/department12/room1/temperature]	14
154	2.082593293	::1	::1	MQTT	136	Subscribe Request (id=1) [university/department12/room1/temperature]	6
121	1.083118347	::1	::1	MQTT	136	Subscribe Request (id=1) [university/department12/room1/temperature]	2

Figure 3: MQTT SUBSCRIBE packets to the Will Topic without wildcards

Result: The number of different MQTT clients that would receive a Will Message **without using wildcards** is:

$$\boxed{3}$$

1.6 CQ6

Q: How many MQTT publish messages directed to the public broker mosquitto are sent with the retain option and use QoS "At most once"?

1.6.1 Identify the Mosquitto Broker IP

I first filtered DNS traffic using the display filter `dns` to capture hostname resolutions. Within the results, the following DNS query and response were found:

- Query: `Standard query A test.mosquitto.org`
- Response: `test.mosquitto.org A 5.196.78.28`

This confirmed that the IP address of the public Mosquitto broker is:

`5.196.78.28`

1.6.2 Apply Wireshark Filter

To isolate the required packets, the following Wireshark display filter was used:

```
mqtt.msgtype == 3 && ip.dst == 5.196.78.28 && mqtt.qos == 0 && mqtt
    .retain == 1
```

This filter extracts all MQTT PUBLISH messages that:

- are sent to `5.196.78.28` (Mosquitto),
- have QoS level 0 (`mqtt.qos == 0`),
- and are marked as retained (`mqtt.retain == 1`).

The number of matching packets corresponds to the final answer.
Result: The number of MQTT PUBLISH messages directed to the Mosquitto broker with the `retain` option and QoS "At most once" is:

$$\boxed{208}$$

1.7 CQ7

Q: How many MQTT-SN messages on port 1885 are sent by the clients to a broker in the local machine?

1.7.1 Filtering Strategy

- MQTT-SN operates over UDP rather than TCP.
- The default port for MQTT-SN is `1885`.
- It is assumed that the local broker runs on IP address `127.0.0.1` (IPv4).

The Wireshark display filter used was:

```
udp.port == 1885
```

This filter selects all UDP packets on port 1885, regardless of direction. In the trace, this resulted in **0 packets** matching the condition.

Result: The number of MQTT-SN messages sent by clients to a broker on the local machine is:

$$\boxed{0}$$