Home Challenge #1

Francesco Pallotta

COAP

Question 1

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

Answer

by filtering the given capture with Wireshark we can see the details of the request packets with MID 53533 and MID 42804:

- **Packet 6294**

  It's a COAP request packet, which is marked as confirmable and it's using a GET method:

MID	53533
Source IP	`10.0.2.15`
Destination IP	`134.102.218.18`
Source Port	`50593`
Destination Port	`5683`
Protocol	COAP
Method	`GET`
Type	Confirmable
Token	`242f92f0`
Endpoint	`/large`

- **Packet 6276**

  It's also a COAP request packet, but it's using a delete method and it's non-confirmable:

MID	42804
Source IP	`10.0.2.15`
Destination IP	`104.196.15.150`

MID	42804
Source Port	58700
Destination Port	5683
Protocol	COAP
Method	DELETE
Type	Non-confirmable
Token	6dbdd020
Endpoint	/link2

The main difference between the two lies in the used method and in the fact that the first packet is confirmable while the second one is not, which means that the first packet has a much higher reliability since it requires an ACK response from the target destination.

Question 2

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

Answer

First we need to filter out packet with frame number 2428 in Wireshark with the following filter:

```
frame.number == 2428
```

By doing that we can find a COAP request with the following characteristics: Constrained Application Protocol, Non-Confirmable, DELETE, MID:12935. Since its marked as non-confirmable, an ACK from the target machine is not requested.

I found a matching response by looking for a packet with the same token, which is 67c7229a. The following result came up: Constrained Application Protocol, Non-Confirmable, 2.02 Deleted, MID:844

MID	12935
Source IP	localhost
Destination IP	localhost
Source Port	5683
Destination Port	37584
Protocol	COAP
Response	2.02 Deleted
Type	Non-confirmable
Token	67c7229a
Endpoint	/living_room/door

Question 3

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

Answer

By applying the following filter:

```
((((ip.dst == 127.0.0.1)) && (coap))) && (coap.code == 69)
```

Which is filtering the capture by setting `localhost`as destination address, COAP as protocol and `2.05 Content` as type, we get the following result:

```
127.0.0.1   127.0.0.1   CoAP    69  ACK, MID:63229, 2.05 Content, /living_room
127.0.0.1   127.0.0.1   CoAP    69  ACK, MID:4920, 2.05 Content,  /living_room
127.0.0.1   127.0.0.1   CoAP    69  ACK, MID:23246, 2.05 Content, /living_room
127.0.0.1   127.0.0.1   CoAP    69  ACK, MID:13240, 2.05 Content, /living_room
127.0.0.1   127.0.0.1   CoAP    69  ACK, MID:29961, 2.05 Content, /living_room
127.0.0.1   127.0.0.1   CoAP    69  ACK, MID:25273, 2.05 Content, /living_room
127.0.0.1   127.0.0.1   CoAP    69  ACK, MID:48882, 2.05 Content, /living_room
127.0.0.1   127.0.0.1   CoAP    69  ACK, MID:21099, 2.05 Content, /living_room
```

Which means that there are 8 packets in total which respect the given constraints.

Question 4

How many GET requests, excluding `OBSERVE` requests, have been directed to non existing resources?

Answer

Assuming that with non existing resources we are referring to responses which have a `4.04 Not Found (132)` response code, we need to find how many non `OBSERVE` requests have been sent to match such responses. I have built the following script after exporting the capture as `.json` in Wireshark:

```python
from json import loads
from difflib import Differ
from IPython import embed


def pkt_parser(pkt, rel_path):
    out = pkt["_source"]["layers"]
    for el in rel_path.split("/"):
        out = out[el]
    return out
```

```python
def q4(cap):
    res = []
    for pkt in cap:
        if (
            "coap" in pkt_parser(pkt, "frame/frame.protocols")
            and pkt_parser(pkt, "coap/coap.code") == "132"
        ):
            res.append(pkt)
    print("found {} 404 response packets".format(len(res)))

    reqs = []
    for r in res:
        token = pkt_parser(r, "coap/coap.token")
        for pkt in cap:
            if (
                "coap" in pkt_parser(pkt, "frame/frame.protocols")
                and pkt_parser(pkt, "coap/coap.token") == token
                and pkt_parser(pkt, "coap/coap.code") == "1"
            ):
                try:
                    if pkt_parser(pkt, "coap/opt.observe") == "0":
                        reqs.append(pkt)
                except KeyError as e:
                    reqs.append(pkt)
    print("found {} matching responses".format(len(reqs)))


def main():
    cap = loads(open("cap.json").read())
    q4(cap)


if __name__ == "__main__":
    main()
```

By running the script we get the following result:

```
$ p app.py
found 8 404 response packets
found 6 matching responses
```

6 matches were found.

MQTT

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], e.g. factory/department1/+, factory/department2/+ and so on?

Answer

We can easily modify the previous script by replacing q4 with q5 which does the computation we need:

```
def q5(cap):
    res = []
    for pkt in cap:
        try:
            if (
                "factory/department" in pkt_parser(pkt, "mqtt/mqtt.topic")
                and pkt_parser(pkt, "mqtt/mqtt.passwd") == "admin"
            ):
                res.append(pkt)
        except KeyError as e:
            pass
    print("found {} matching packets".format(len(res)))
```

It gives the following output:

```
$ p app.py
found 0 matching packets
```

No matching packets with the desired constraints. There are some packets with password set as "admin" which have factory/department*/+ as will topic:

```
$ p app.py
replaced topic with will topic
found 5 matching packets
```

Question 6

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

Answer

First we look for the DNS queries in Wireshark by using the following filter:

```
dns.qry.name == "test.mosquitto.org"
```

Which gives us a series of packets whose responses all return the same answer: test.mosquitto.org: type A, class IN, addr 5.196.95.208. Now by filtering out packets that have that IP set as destination and that have the will flag set:

```
(ip.dst == 5.196.95.208) && (mqtt.conflag.willflag == 1)
```

We get 9 hits:

```
10.0.2.15   5.196.95.208    MQTT    133 Connect Command
10.0.2.15   5.196.95.208    MQTT    125 Connect Command
10.0.2.15   5.196.95.208    MQTT    149 Connect Command
10.0.2.15   5.196.95.208    MQTT    149 Connect Command
10.0.2.15   5.196.95.208    MQTT    157 Connect Command
10.0.2.15   5.196.95.208    MQTT    157 Connect Command
10.0.2.15   5.196.95.208    MQTT    145 Connect Command
10.0.2.15   5.196.95.208    MQTT    133 Connect Command
10.0.2.15   5.196.95.208    MQTT    133 Connect Command
```

Question 7

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

Answer

To answer this question we need a slight modification of the usual script which does the following operations:

1. Looks for publishes with QoS 2
2. annotes their msgid
3. For each of them, checks for PUBREL messages with the same id

```python
def q7(cap):
    res = []
    pubs = []
    for pkt in cap:
        try:
            if pkt_parser(pkt, "mqtt/mqtt.hdrflags_tree/mqtt.qos") == "2":
                pubs.append(pkt)
        except KeyError:
            pass
    print("found {} packets with QoS=2".format(len(pubs)))

    for p in pubs:
        msgid = pkt_parser(p, "mqtt/mqtt.msgid")
        for pkt in cap:
            try:
                if (
                    pkt_parser(pkt, "mqtt/mqtt.msgid") == msgid
                    and pkt_parser(pkt, "mqtt/mqtt.mgstype") == "6"
                ):
                    res.append(pkt)
            except KeyError:
                pass
    print("found {} matching packets".format(len(res)))
```

```
$ p app.py
found 92 packets with QoS=2
found 0 matching packets
```

Again we got no matching packets.

Question 8

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

Answer

This function loops trough the capture, selects only packets with empty `clientId` and computes the average of their `willTopic` length.

```
def q8(cap):
    sum = 0
    count = 0
    for pkt in cap:
        try:
            if pkt_parser(pkt, "mqtt/mqtt.clientid") == "":
                try:
                    sum += int(pkt_parser(pkt, "mqtt/mqtt.willtopic_len"))
                    count += 1
                except KeyError:
                    pass
        except KeyError:
            pass
    print("average will topic length: {}".format(sum / count))
```

```
$ p app.py
average will topic length: 37.054054054054056
```

Question 9

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

Answer

First of all we need to find the address of such client. We can do that by applying the following filter in Wireshark:

```
(mqtt.clientid == "6M5H8y3HJD5h4EEscWknTD")
```

As a result we get the packet that the client we are looking for used to establish a connection with the MQTT broker:

```
4659    109.331682932   10.0.2.15   5.196.95.208    MQTT    157 Connect Command
```

So we know that our client has address `10.0.2.15` and is using port `46295`. Now we can filter packets that are using the MQTT protocol and that have the same IP and port as the source of the packet containing our client id:

```
mqtt && ip.dst == 10.0.2.15 && tcp.dstport == 46295
```

We get a lot of results. We need to filter out only ACK packets. We can have the following types of ACK messages:

- Connect ACK `(2)`
- Subscribe ACK `(9)`
- Publish ACK `(4)`

```
mqtt && (ip.dst == 10.0.2.15 && tcp.dstport == 46295)
&& (mqtt.msgtype == 4 || mqtt.msgtype == 2 || mqtt.msgtype == 9)
```

Returning three packets:

No.	Type
4764	Connect ACK
4766	Subscribe ACK
4768	Publish ACK

Question 10

What is the average MQTT message length of the `CONNECT` messages using MQTT v3.1 protocol? Why messages have different size?

Answer

We can filter out packets with MQTT version 3 and message of type `CONNECT`, and then compute the average of their message length:

```
def q10(cap):
    sum = 0
    count = 0
    for pkt in cap:
        try:
            if (
                pkt_parser(pkt, "mqtt/mqtt.ver") == "3"
                and pkt_parser(pkt, "mqtt/mqtt.hdrflags_tree/mqtt.msgtype") == "1"
            ):
                sum += int(pkt_parser(pkt, "mqtt/mqtt.len"))
                count += 1
        except KeyError:
            pass
    print("average message length: {}".format(sum / count))
```

```
$ p app.py
average message length: 63.59574468085106
```

The difference in message length between packets may be caused by the fact that MQTT packets may have lots of different optional fields, which all contribute to determine a noticeable difference in message length.