A CYBER SECURITY APPLICATION FOR REPORTING MALICIOUS ATTACKS ON IOT NETWORKS THAT USETHE MQTT PROTOCOL

CUCURUZAC Andrei-Traian[1], ABAZA Bogdan Felician[2]
[1]Faculty of Industrial Engineering and Robotics, Specialization: Applied ComputerScience in Industrial Engineering, Year of study: 4, e-mail: andreicucuruzac99@gmail.com
[2] Faculty of Industrial Engineering and Robotics, Manufacturing Engineering Department, University POLITEHNICA of Bucharest

ABSTRACT: It is known that information and communication technology has advanced a lot in recent decades and with it, the risk of malicious attacks increased. New and abstract concepts have emerged that have improved and continue to improve human life. Whether we are talking about cities, factories, universities or even households, the need to automate and monitor the various processes that take place within them has increased, paving way for IoT networks. Unfortunately, in many cases this accelerated development forgot about securing devices that transmit valuable information. Thus, the data transmitted in the networks became an easy target for malicious people. This paper focuses on the creation of an IoT network on which we will simulate a set of possible attacks that shows us where and what these malicious people could hit.

KEYWORDS: Internet of Things, MQTT Protocol, Cyber Security, Network Attacks

1. Introduction

The rapid development of information and communication technology has made people's lives easier in many ways. But with the digitization of information has also come the danger that information can be stolen or obstructed, leading to economic loss or even loss of life. *"Every American depends - directly or indirectly - on our system of information networks. They are increasingly the backbone of our economy and our infrastructure, our national security and our personal well-being. But it's no secret that terrorists could use our computer networks to deal us a crippling blow ... As President, I'll make cyber security the top priority that it should be in the 21st century."* - Barack Obama, 44th President of the U.S. , "Summit on Confronting New Threats", Purdue University, 16 July 2008.

The present work entitled " Cybersecurity application for reporting malicious attacks in IoT networks running on the MQTT protocol " aims at securing MQTT servers in IoT architectures, which are known within the protocol as brokers, by generating a report that canhelp detect certain vulnerabilities and propose possible suggestions in their remediation.

The Internet of Things, or IoT for short, is a concept that involves the use of the Internet in interconnecting various devices, sensors, software or automated systems to form a network of objects that aims to provide various services easily accessible to humans [1]. With the appearance of IoT, other innovative concepts such as smart industry, smart city or smart home have emerged, all of which improve everyday human life. Smart industry or Industry 4.0 has led to the restructuring of factories thus increasing production while managing to decrease the number of employees, smart city has led to efficient air quality monitoring or smart traffic management, and smart home has in some cases even led to the disappearance of the common light switch, homeowners using only their voice to operate light bulbs or other devices. This is even more useful in the case of people with mobility problems. [2-5].

2. Architecture of IoT Networks running on the MQTT protocol

Accelerated progress in information and communications technology has forced the IoT concept to rapidly take shape and adopt certain types of architectures. Although there are several types of architectures, the main ones used are: three-layer architecture, service-based architecture and middleware-based architecture. At the core of this work is the service-based architecture, which in turn is divided into four layers: the perception layer, the network layer, the service layer and the application layer. For the last-mentioned layer, although there is currently no standardization, protocols for low bandwidth environments are used, for example: MQTT, CoAP, XMPP, DDS, AMQP [6].

The protocol used in this paper is MQTT or Message Queuing Telemetry Transport. This is a simple protocol that has been designed for limited networks that operates with devices with poor processing capabilities and high latencies. It is on layer seven of the OSI (Open Systems Interconnection) stack and on layer four in the service-based architecture (Fig.2.). On the main layer, the perception layer, are the sensors. They allow the conversion of information from the environment into digital information. Further on the next layer are the technologies for transmitting sensor data. In this paper, data will be transmitted via WIFI and Bluetooth.[7]

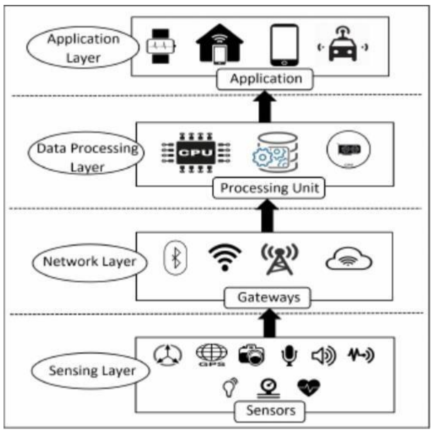


Fig. 2. Service-based architecture. [7]

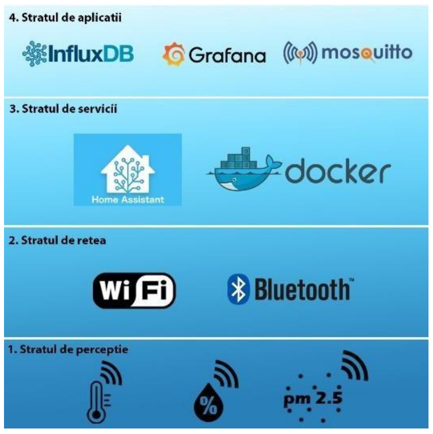


Fig.3. Architecture applied in this paper.

A difference can be seen on the service layer, also called the data processing layer. Where, the processing is not done directly by a physical computer but by a virtual machine with the HomeAssistant operating system, which is based on Docker containers. On the last layer, the application layer, there is the MQTT broker, Mosquitto, which makes it possible for clients to connect and transmit data. (Fig.3.)

MQTT is used today in different types of industries such as consumer goods manufacturing, the automotive industry or the oil industry. The protocol works on the principle of transmitting data through a publish-subscribe system where, as suggested by the name, IoT network devices send data to an MQTT broker to which they have subscribed [8] (Fig.4.).

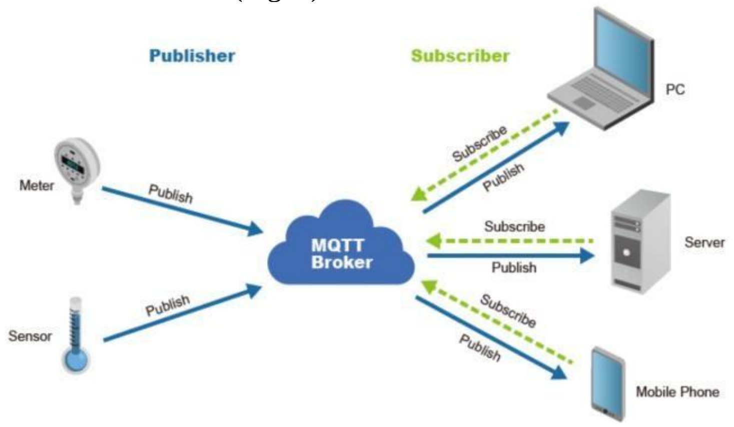


Fig.4. Principle of data transmission via MQTT. [9]

In order to carry out the present work and for a better understanding of the infrastructure ofan IoT network, it was necessary to create a test environment as similar as possible to a real case, in which we can connect different sensors, transmit via MQTT, visualize and store in a database the information received from them. The final objective being to test the created environment an understand the vulnerabilities to which the IoT network is exposed.

3. Defining the IoT network test environment

The test environment is currently fully and actively set up in one of the university rooms, collecting data from April 18, 2022 to present. Following the service based IoT architecture and considering what sensors can be found in industrial environments, but at the sametime taking into account the components accessible on the market, the environment created is divided as follows:

- Perception layer (Sensor layer)
  - a) PM 2.5 particle sensor.
  - b) Bluetooth humidity and temperature sensor
  - c) Humidity and temperature sensor
- Network layer
  - a) ESP32 – Microcontroller responsible for taking data from sensors andtransmitting them using the WIFI standard via MQTT.
- Service layer (Data processing layer)
  - b) Home Assistant OS - Operating system that embeds containerization services viaDocker to install applications in a controlled and secure environment.
- Application layer
  - c) Mosquitto broker - Application responsible for retrieving data transmitted viaMQTT.
  - d) Graph - The application responsible for the graphical representation of data.
  - e) Influx DB - Database.
  - f) Home Assistant Core - Module present in the operating system that allowsautomation based on intercepted data.

The PM 2.5 sensor used is a commercial ready-to-use sensor that is powered from the socket using a USB-C cable. To make the design as practical as possible, it had to be modified(Fig. 6). Thus, the modification of the sensor consists of powering the ESP32 microcontrollerand the DHT22 sensor from its motherboard and collecting data from the two sensors in serial mode by the microcontroller. At the same time the ESP board will receive temperature and humidity information via Bluetooth from two other nearby sensors.

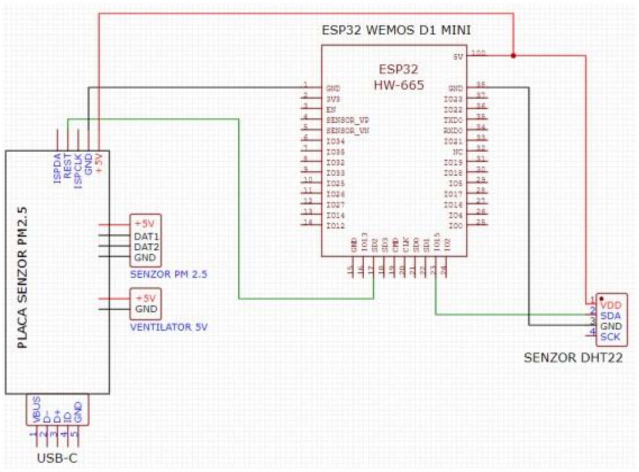


Fig. 5. Wiring diagram

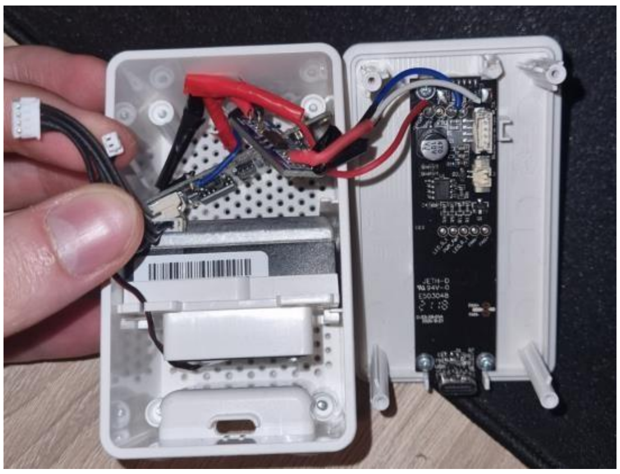


Fig. 6. Soldering components

In the case of the two Bluetooth sensors, a modification was also made by changing the firmware pre-installed by the manufacturer with a custom firmware already available on the internet [10]. This was necessary to make it possible to transmit parameters to the microcontroller via Bluetooth.

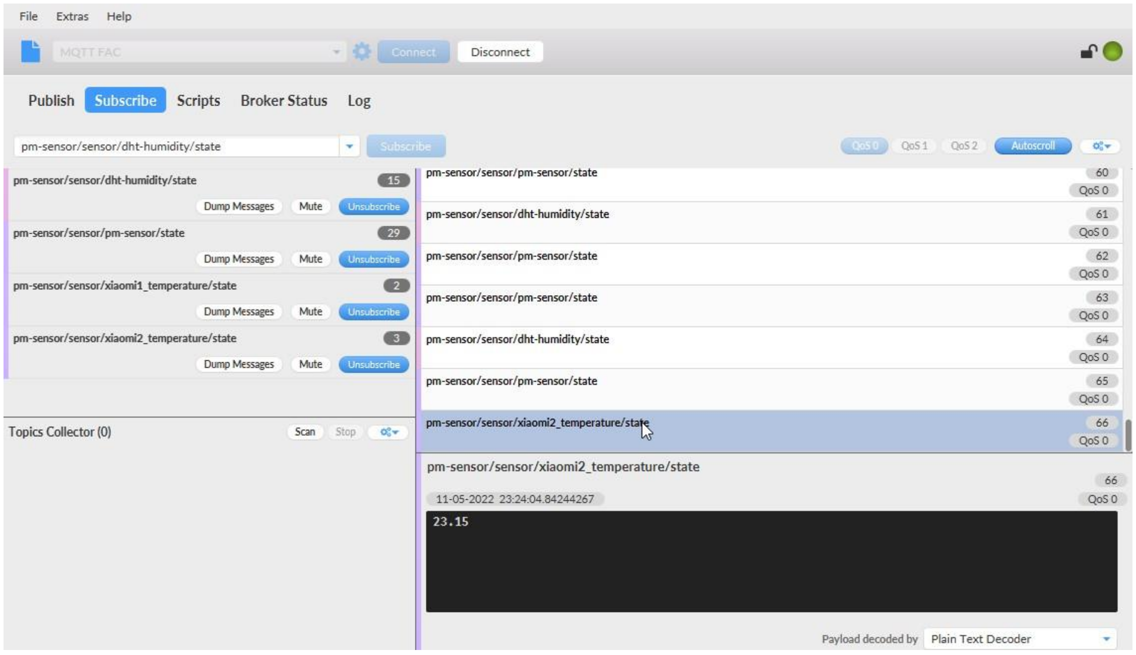
The ESP-Home environment was used to program the microcontroller, which is based on a YAML file translator. After translation, the program compiles a firmware based on the configurations made later in the file to be uploaded to the ESP board. The process proceeds asfollows:

- Configure the YAML file with the name of the microcontroller and its type.

- Activate the Bluetooth module and set the WIFI credentials that will allow the microcontroller to access the internet.
- Sensor configuration. This part involves choosing the pins over which serial communication will be done and setting the MAC addresses for the Bluetooth sensors.
- Configure the MQTT client that will transmit to the server. This involves setting the IP and port of the broker, as well as setting the username and password that were set when configuring the server.
- Generate firmware and install it on the ESP via USB

After programming the microcontroller, the MQTT is configured on the Home Assistant OS server. The broker available for this OS is Mosquitto, a broker supporting the latest version of the MQTT protocol. As the chosen OS works on docker containers, the configuration only involved setting the authentication credentials and setting the port through which the data exchange willbe performed.

Thus, with simple software that subscribes to the broker topic, [11] it is possible to visualizethe first data transmitted by the sensors. In the following picture, four topics are shown that have been selected to demonstrate data transmission via MQTT (Fig.7).



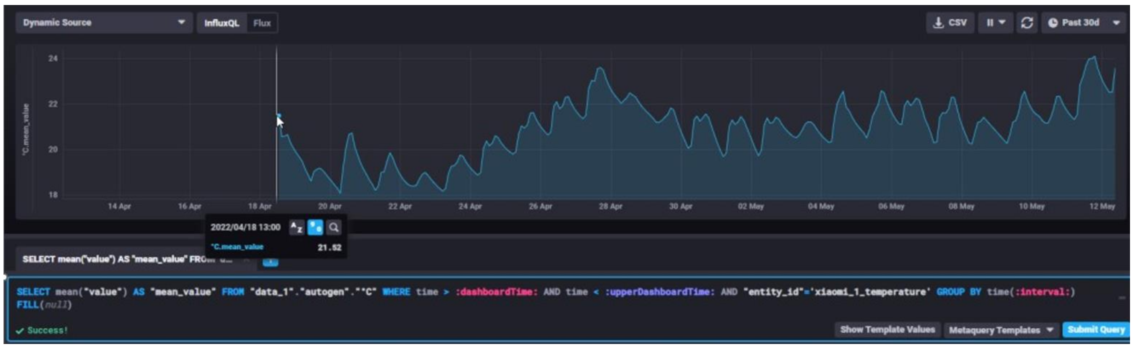
```
pm-sensor/sensor/dht-humidity/state               15
pm-sensor/sensor/pm-sensor/state                  29
pm-sensor/sensor/xiaomi1_temperature/state         2
pm-sensor/sensor/xiaomi2_temperature/state         3
```

```
pm-sensor/sensor/pm-sensor/state             60  QoS 0
pm-sensor/sensor/dht-humidity/state          61  QoS 0
pm-sensor/sensor/pm-sensor/state             62  QoS 0
pm-sensor/sensor/pm-sensor/state             63  QoS 0
pm-sensor/sensor/dht-humidity/state          64  QoS 0
pm-sensor/sensor/pm-sensor/state             65  QoS 0
pm-sensor/sensor/xiaomi2_temperature/state   66  QoS 0
```

```
pm-sensor/sensor/xiaomi2_temperature/state
66
11-05-2022 23:24:04.84244267   QoS 0
23.15
```

Fig.7. Sensor data captured with the MQTT.fx program [11]

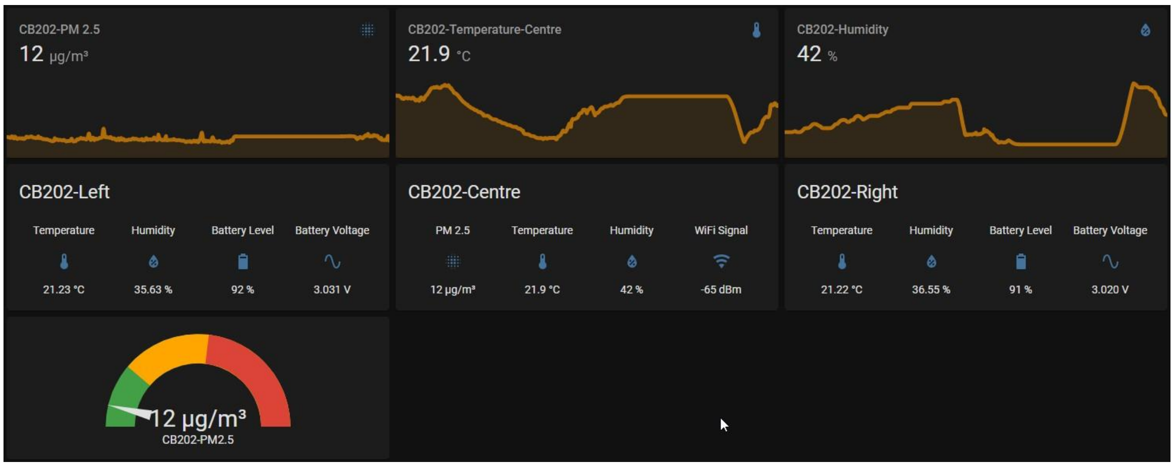
Once the necessary settings have been made, the storage of all the values sent on the protocols intercepted by Home Assistant will begin. The data will be kept on the server for 365 days in the database called "data_1". The following figure shows the temperature data from one of the two installed Bluetooth sensors. The data shown is stored from the time of installation on 18 April 2022 until the day this paper was created (Fig.8.).



```
SELECT mean("value") AS "mean_value" FROM "data_1"."autogen"."°C" WHERE time > :dashboardTime: AND time < :upperDashboardTime: AND "entity_id"='xiaomi_1_temperature' GROUP BY time(:interval:)
FILL(null)
✓ Success!
```

Fig.8. Querying the database

Finally, for data visualization we configured the main panel of Home Assistant so that the parameters were delimited by zones, according to how the sensors were placed in the room (Fig.9.).



CB202-PM 2.5
12 µg/m³

CB202-Temperature-Centre
21.9 °C

CB202-Humidity
42 %

CB202-Left

Temperature	Humidity	Battery Level	Battery Voltage
21.23 °C	35.63 %	92 %	3.031 V

CB202-Centre

PM 2.5	Temperature	Humidity	WiFi Signal
12 µg/m³	21.9 °C	42 %	-65 dBm

CB202-Right

Temperature	Humidity	Battery Level	Battery Voltage
21.22 °C	36.55 %	91 %	3.020 V

12 µg/m³
CB202-PM2.5

Fig. 9. Monitoring panel

Thus, a system was created based on an existing architecture, which takes temperature and humidity parameters both via Bluetooth and serial mode, as well as air quality parameters, to be transmitted by the MQTT client, via WIFI, using the ESP32 microcontroller to the virtual machine with Home Assistant OS, where the data is taken, processed and stored.

4. Security simulation and analysis

Like most WAN (Wide Area Network) connected networks, MQTT-based IoT networks can also be vulnerable to cyber-attacks. A retrospective analysis by Cloudflare Inc. shows that in 2016 the *"Mirai"* malware managed to infect 65,000 IoT devices on the first day of its release peaking in November 2016 when it reached 600,000 infections. Internet service provider Deutsche Telekom suffered massive signal disruptions and the compromise of hundreds of thousands of routers because of the "DoS" attack carried out by the IoT botnet created after the infections [12].

With the help of libraries for the Python programming environment and tools available on the Internet [13], a series of attacks were attempted. These attacks aim to load the MQTT broker with fake clients. As a result of the attack parameters such as CPU utilization, RAM utilization and system temperature were influenced. The biggest differences can be seen in the processor utilization parameters where the values increased from 5% normally to 63% during the attack (Fig.10.).

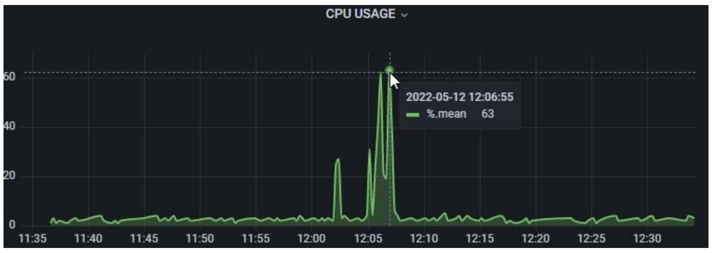


Fig. 10. Processor usage parameters

It must be mentioned that the attack was carried out from an old computer with a dual-core processor and 8 GB RAM. The higher the performance of the computer on which the attack is carried out, the greater the damage can be. In this case, the damage is not considerable, but only a slowdown of the system. However, if routine updates by the system operator are added to this scenario, concomitant with the attack, there is a possibility that the usage parameters may reach higher levels, leading to the broker's inability to function.

This test was carried out without encrypting the data, in the future it is intended to set up a server that will encrypt the data and the connection using known protocols such as TLS or SSL. The current security settings that have been made on the test environment relate to blocking other data ports on the network that communicate with the outside and using only those that are strictly necessary for remote system operation and actuation.

This is where the issue of studying the security of microcontroller update management came into play. Since other ports were blocked for security reasons, it made it inaccessible to send updates wirelessly to ESP. With any firmware update it is necessary to disassemble the sensorand connect via USB to the computer from where the action will be performed. As a result, ways to send these updates via MQTT, where port 1882 is available for external communication, have been investigated. An unfriendly but possible way is to compile the firmware and transmit it in small bit packets. But an unstable WIFI connection can lead to a data packet being lost *en route*, so there is the possibility of corrupting the ESP's memory and making it unusable.

5. Conclusions

In conclusion, it is easy to see that technology has advanced and at the same time the risks to users have increased, which is why a thorough analysis of the vulnerabilities of all information systems is necessary.

The system was developed from the desire to create and secure a functional IoT network in real scenarios, in this case monitoring temperature, humidity and air quality in a classroom according to an existing architecture. Following the service based IoT architecture, it was possible to capture sensor data using the ESP32 microcontroller, connect it to a network via WIFI, transmit the retrieved data to a server via MQTT, store and visualize the data. At the same time, a type of MQTT attack has been carried out which has been shown to slowdown the processing capacity of a server, and in other scenarios may even render it unusable.

6. Bibliography

[1]. Samuel Greengard (2015), The Internet of Things, The MIT Press, ISBN 978-0-262- 52773-6, Massachusetts.
[2]. Marcelo Tsuguio Okano (2017), IOT and Industry 4.0: The Industrial New Revolution, International Conference on Management and Information Systems, Chitkara University in association with AIMS-International and INFOMS, Bangkok, 25-26 September 2017.
[3]. Ashraf Tahat, Ruba Aburub, Aseel Al-Zyoude and Chamseddine Talhi (2018), A Smart City Environmental Monitoring Network and Analysis Relying on Big Data Techniques, ICSIM2018: Proceedings of the 2018 International Conference on Software Engineering and Information Management, January 2018.
[4] Yamuna Kaluarachchi (2022), Implementing Data-Driven Smart City Applications for Future Cities, MDPI Smart Cities, 30 March 2022.
[5]. Soojung Chang and Kyeongsook Nam(2021), Smart Home Adoption: The Impact of User Characteristics and Differences in Perception of Benefits, MDPI Buildings, 3 September 2021.
[6]. Marco Lombardi, Francesco Pascale and Domenico Santaniello (2021), Internet of Things: A General Overview between Architectures, Protocols and Applications, MDPI Information, 19 February 2021.
[7]. Sikder, Amit Kumar & Petracca, Giuseppe & Aksu, Hidayet & Jaeger, Trent & Uluagac, Selcuk. (2018) A Survey on Sensor-based Threats to Internet-of-Things (IoT) Devices and Applications.
[8]. MQTT - The Standard for IoT Messaging, 2022 - https://mqtt.org/
[9]. Image source - https://oringnet.com/en-global/tech/detail/93
[10]. Custom firmware for the Xiaomi Thermometers and Telink Flasher via USB to Serialconverter - https://github.com/pvvx/ATC_MiThermometer
[11]. MQTT.fx - https://mqttfx.jensd.de/
[12]. Cloudflare, Inside the infamous Mirai IoT Botnet: A Retrospective Analysis, 12/14/2017 - https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/
[13]. Stfbk (2019), Github, MQTTSA, https://github.com/stfbk/mqttsa