

UNIVERSIDADE ESTADUAL DE CAMPINAS
SISTEMA DE BIBLIOTECAS DA UNICAMP
REPOSITÓRIO DA PRODUÇÃO CIENTÍFICA E INTELECTUAL DA UNICAMP

Versão do arquivo anexado / Version of attached file:

Versão do Editor / Published Version

Mais informações no site da editora / Further information on publisher's website:

<http://seer.upf.br/index.php/rbca/article/view/12163>

DOI: 10.5335/rbca.v13i2.12163

Direitos autorais / Publisher's copyright statement:

©2021 by Universidade de Passo Fundo. All rights reserved.

DIRETORIA DE TRATAMENTO DA INFORMAÇÃO

Cidade Universitária Zeferino Vaz Barão Geraldo



CEP 13083-970 – Campinas SP

Fone: (19) 3521-6493

<http://www.repositorio.unicamp.br>

ORIGINAL PAPER

Security analysis of the message queuing telemetry transport protocol

Matheus Ferraz Silveira ¹ and André L. S. Gradvohl ¹¹Faculdade de Tecnologia, Universidade Estadual de Campinas – Limeira, SP, Brasil

*matheus_ferraz@live.com; gradvohl@ft.unicamp.br

Received: 2020-12-26. Revised: 2021-07-07. Accepted: 2021-07-26.

Abstract

The internet of things aims to assign computational processing and connection to simple objects on a network to collect data and then perform analysis. However, due to its easy use, the simplified implementation has several information security problems. This paper presents attack procedures in an internet of things environment using the Message Queue Telemetry Transport protocol. We use the Low Orbit Ion Cannon and Wireshark programs for attack procedures, compromising the integrity, confidentiality, and availability of data and network connection. After carrying out the attack procedures, we implemented security methods on the network, such as data encryption and firewall, to protect data integrity and prevent network connection attacks.

Keywords: Encryption; firewall; information security; internet of Things; MQTT.

Resumo

A internet das coisas tem como objetivo atribuir a objetos simples um pouco de poder de processamento computacional e conexão com uma rede para coletar dados e, em seguida, realizar análises. Contudo, devido à facilidade de uso, a implementação simplificada apresenta diversos problemas de segurança da informação. Este artigo apresenta procedimentos de ataque em um ambiente na internet das coisas usando o protocolo Message Queue Telemetry Transport. Nós usamos os programas *Low Orbit Ion Cannon* e *Wireshark* para procedimentos de ataque, comprometendo a integridade, confidencialidade e disponibilidade de dados e da conexão de rede. Após realizar os procedimentos de ataque, implementamos métodos de segurança na rede, como criptografia de dados e *firewall*, para proteger a integridade dos dados e evitar ataques à conexão de rede.

Palavras-Chave: Encriptação; firewall; internet das coisas; MQTT; segurança da informação.

1 Introduction

The Internet of Things (IoT) is a concept that has been in evidence today. In short, the IoT aims to connect simple objects with little computational power for data processing and information collection (Sinha et al., 2017). The concept of IoT has recently gained prominence as the ability to include data processing into simple objects. With IoT, it is possible to gather information to analyze the information that travels between objects and make a more assertive decision making, generating more positive results (da Cunha et al., 2016). This large set of data stored, analyzed, and processed by institutions is called Big

Data, a concept that, along with IoT, has acquired much relevance in academia and the market due to the positive history of use (Jung et al., 2019, Yadav and Vishwakarma, 2018).

With the advancement of research and technology, the IoT network has become widespread in several different application segments, such as restaurants, commerce, agriculture, academic research, and other segments (Dholu and Ghodinde, 2018, Alghamdi, 2019, Mekruksavanich, 2019). Most of these activities use a wireless network due to its practicality. However, when combining this interest in implementation and ease of implementation, many IoT networks fall short in data

security, which is a failure during its implementation (Chen and Erfani, 2017). The lack of data traffic security allows attacks in several ways to capture sensitive data or interfere with the stored dataset, compromising its analysis.

For creating a connection between simple objects, such as stoves and refrigerators, these objects must have some data processing capacity, such as microcontrollers and sensors, along with a wireless network connection. These components are necessary for reading and exchanging information (Samsudin et al., 2018). This network fostered the exchange of information works with rules for communication, which are called communication protocols (Rouse, 2020). In the TCP/IP protocol stack, most IoT protocols are in the application layer, as shown in Fig. 1.

TCP/IP protocol architecture

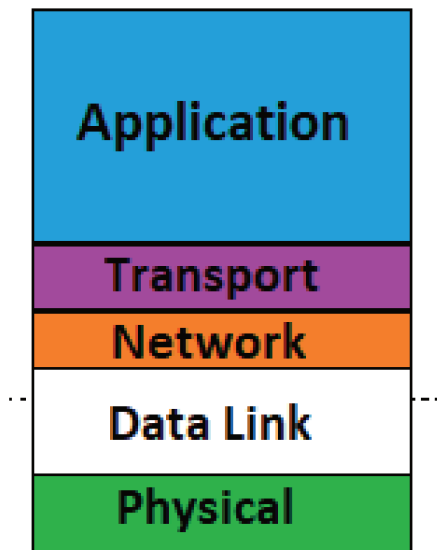


Figure 1: Illustration of the layers of the TCP / IP model. Adapted from Lopez (2003).

One of the most used protocols in an IoT network is Message Queuing Telemetry Transport (MQTT), as it is a protocol with low memory usage, low processing demand, and low bandwidth consumption (Yassein et al., 2017). IBM created the MQTT protocol to be a lightweight protocol, guaranteeing messages with data delivery, using small computational power, and high latency in the network (Yuan, 2017). It operates using messages between the publisher, the broker, and the client (subscriber).

In practice, the publisher generates the data and sends it through topics to the broker, who acts as the controller who signs the published data. Once published in the broker, the client sends the data's topic, subscribing to the publication and obtaining the data sent from the publisher (Cope, 2019).

Next, Fig. 2 illustrates the exchange of messages between the publisher and the client, with the topics controlled by a broker.

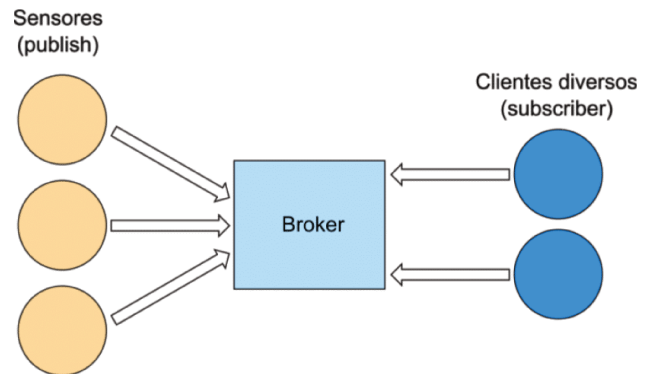


Figure 2: Illustration of the functioning of the MQTT protocol. Source: (da Cunha et al., 2016).

According to the presented context, this paper aims to demonstrate the MQTT protocol's fragility when implemented in an IoT network in the usual way. Thus, when using specific procedures to capture the data that travel on the network, we show that it is possible to compromise the customer's data.

We organized the rest of this paper as follows. Section 2 discusses some concepts regarding this work, and Section 3 presents a literature survey. Besides, Section 4 describes the methodology used to expose weaknesses and indicate solutions. Section 5 details the experiments and results. Finally, Section 6 shows the conclusions of this research.

2 Literature review

For this work's positioning concerning the others, we did a literature survey of the most relevant works in the area. However, first, we discuss the concepts that underlie IoT and the aspects related to security in the following sections.

2.1 Internet of Things

At the beginning of the discussion, in 1982, the International Telecommunication Union (2012) described the IoT as a global information infrastructure, which enables advanced services through the interconnection (physical and virtual) of objects based on information and communication technologies interoperable and evolving.

For this infrastructure to become real, it is necessary to explore the identification, data capture, processing, and communication resources, which we can see as small data packages for a large set of nodes. Later, practitioners considered that there were more objects connected to the internet with IoT than people. Cisco Systems estimated that in 2009 the proportion of connected objects to people

would increase, and that number will gradually increase in the following years (Raji, 1994).

The Internet of Things is an evolving concept. As such, it intends to cover several areas of research since the connection of devices to wide area networks or local area networks (LAN) becomes very simple, making activities that require specific monitoring or constant reading of data performs more efficiently and safely (Anil, 2016).

Usually, an IoT system uses an application layer protocol, such as MQTT, to transport light data, reduce network latency, and treat several devices connected to the IoT network. Besides, processing resources must be limited to avoid overloading them, supplying the data to a server with greater processing power to provide data on objects (Navani et al., 2017). For creating an IoT network, someone must use a sensor for data collection, a controller, and a client that will process the data, such as a desktop or smartphone.

2.2 Message Queuing Telemetry Transport Protocol

After creating the IoT network, one of the means of data transmission is using the Message Queuing Telemetry Transport (MQTT) protocol. Created by IBM in the 90s, this protocol allows data exchange between an object and other devices connected to a network. Moreover, it is capable of operating on limited hardware and networks with high latency.

In the MQTT protocol, the sensor and clients transfer data via topics. Assuming that several sensors and clients exist, the identifier will be the topic informed when sending data to the broker. For instance, in the type “publisher” topic, the object data is made available to all devices that subscribe to that topic. Therefore, it is necessary to use the “publisher” function to carry out the transfer. Then the publisher sends the data to the broker responsible for receiving and relaying the data. Finally, the function “subscriber” receives the data, identifying the data path and selecting it in the broker.

The TCP protocol with authentication and encryption options establishes the connection with the broker. The entire connection process determines the desired Quality of Service (QoS), indicating the relationship between client communication and a broker (Shinho Lee et al., 2013). Therefore, we may use one of the following three QoS levels:

- QoS 0 (maximum once): This service has no message delivery confirmation. Also, it does not store the message for future retransmissions.
- QoS 1 (at least once): In this service, there is a message delivery confirmation. Therefore, it can generate equal messages, depending on the non-confirmation of delivery, until it receives a confirmation of the message's delivery.
- QoS 2 (exactly once): It ensures delivery of the message only once, with confirmation in both directions of traffic. As long as the message is not confirmed, the sender keeps it.

The server sent a *connack* message answering a client's connect message for a connection between client and server. If the message from the client does not reach the server, the connection must be closed.

If the client does not receive the return message, he must restart the session by making a new request to the server and issuing a message. This rule includes messages that provide invalid protocol names or protocol version numbers.

If the server can perform a connection message analysis, it can return a message stating the connection error before ending the session with the client (OASIS Standard, 2019).

2.3 Information Security Test Procedures

After implementing the test environment, we will perform network attack procedures, which will focus on the data and the connection between the publisher and the broker. Finally, we describe the techniques we will use in the following sections.

2.3.1 Denial of Service Method

This type of attack aims to establish the connection to some point on the network inaccessible, obstructing the passage of data through the network. In addition, it is possible to overload any point on the network with data packets, such as the data collection sensor, impairing the sending of information to the next hop. For this procedure, there are specialized programs, most frequently found in the Kali Linux operating system, a Linux distribution made for information security (Chen et al., 2018).

A more elaborate tactic for the denial of service attack is the Distributed Denial of Service (DDoS). This tactic comprises a “boss” computer and several zombie computers connected to the “boss”. When the moment of service attack is determined, zombie computers send data packets to a specific service. As a result, the service may not handle the load of requests per session and become unusable since every service, such as the page servers on the web, for example, has a maximum data load limit (Nagpal et al., 2015).

2.3.2 Capture of network data packets

Another type of attack resource is to perform a network fragility analysis, such as obtaining an access password and, using specific tools, capturing data packets that travel on the network, obtaining the data, which may be confidential or sensitive information. For this procedure, a sniffer program, such as Wireshark, is generally used to intercept and register data packets that travel on the network and subsequently evaluate the content (Das and Tuna, 2017). With the knowledge of the topic used in an IoT network implemented simply, it is possible to use a fake publisher and inject fictitious data, compromising the collected dataset's analysis (Andy et al., 2017).

As stated before, one of the focuses of attacks on an IoT network can be to obtain the data that travels on the network to record sensitive data or make the network unusable. The work from Andy et al. (2017) demonstrates that an IoT-based network with the MQTT's protocol

implementation is superficial, as there is no security tool available, only an authentication tool.

With many devices and networks installed, this type of IoT network using the MQTT protocol is vulnerable to attacks. In the first scenario, the attack occurs so that the subscriber uses a generic topic, using the symbol “#”, to subscribe to all possible topics connected to the broker, obtaining sensitive data. Fig. 3 illustrates the attack scenario.

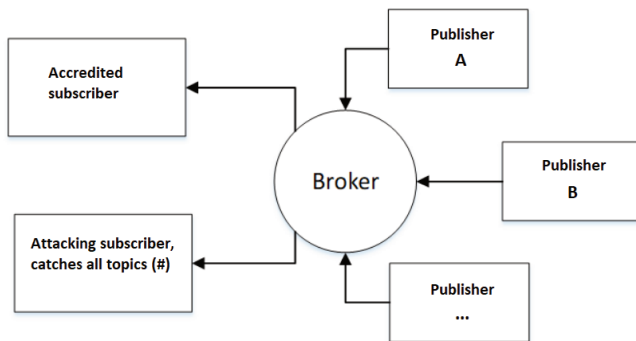


Figure 3: Typical scenario of an attack. Source: (Andy et al., 2017).

It is also possible to perform a reverse attack. It means that, instead of collecting sensitive data using the symbol “#”, it is possible to inject data into a broker, using the “publisher” function, informing false data to the broker and the possible reading of the topic by a subscriber.

We can perform the presented scenarios in a public IoT network if that network does not have authentication. That is because, in an IoT network using the MQTT protocol, authentication is not mandatory. Assuming that the attacker connects to the IoT network, he can analyze the data that travels on the network.

In this scenario, we used the Wireshark to verify the data that travels on the network, exploring the MQTT package’s privacy and integrity. By definition, MQTT does not have data encryption. As a result, the attacker can soon quickly check the data that travels over the network, according to the screen illustrated in Fig. 4.

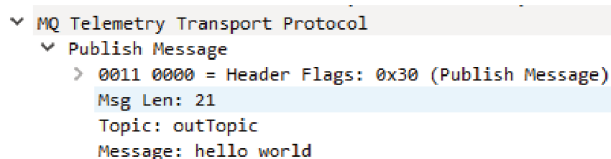


Figure 4: Adaptation of the figure presented in Andy et al. (2017), with attack on broker using Wireshark.

Although authentication in the IoT network is not mandatory using the MQTT protocol, using a user and password to authenticate with the broker is possible. Assuming the attacker is on the same network as the

publisher, one can analyze the network and check the data packets sent to the broker that contain authentication. Within this data package, authentication data is entered in text form by default if there is no encryption by network administrators. Fig. 5 illustrates the connect package collection in the broker, containing the user and password data.

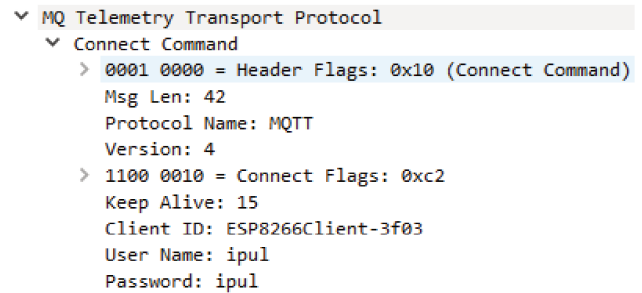


Figure 5: Demonstration of the connect package sent to broker for authentication. Source: (Andy et al., 2017).

Another way to attack the data packets’ integrity is to change the data traveling between the publisher and the broker. To perform this procedure, one can change the name of the topic published by another and perform a filter so that the subscriber reads the false topic’s data. Fig. 6 illustrates how we can do a fake topic using the Etterfilter application (Debian.org, 2020).

```
#owned.filter
if (ip.proto == TCP && tcp.dst == 1883 && ip.dst == 'IP Broker' &&
search(DATA.data, "outTopic")) {
    replace("outTopic", "outTopic");
    msg("payload replaced\n");
}
```

Figure 6: MQTT topic change and creation under another name. Source: (Andy et al., 2017).

After that, using an Etterfilter interface, the attacker, connected to the network, modifies the packet and sends it to the attacked computer. Fig. 7 informs that the subscriber received the changed topic.

As indicated, it is possible to use attack methods on data packets that travel on an IoT network and compromise its connection when someone implements the IoT network without precautions. The following section will demonstrate the testing environment, the experiments carried out, and the results obtained.

3 Related works

Harsha et al. (2018) analyze how security breaches in the use of the MQTT protocol and demonstrate the implementation of security measures using authorization

```

Transmission Control Protocol, Src Port: 1883, Dst Port: 1883
MQ Telemetry Transport Protocol
  Publish Message
    > 0011 0000 = Header Flags: 0x30 (Publish Message)
      Msg Len: 25
      Topic: outTopic
      Message: hello world #31

```

Figure 7: Demonstration of the reading of the altered package made by the subscriber. Source: (Andy et al., 2017).

and authorization techniques. Besides, that work addresses some adversities in the use of the MQTT protocol. Among them, as any customer can subscribe or publish any topic, they explore how to resend lost messages is challenging since there is no access control mechanism or other barriers. Therefore, this work uses Wireshark software to carry out an analysis and monitoring of data packages.

The work indicated that publisher authentication is important, making the broker less vulnerable to unauthorized publishers. Fig. 8 illustrates an attempt to authenticate in the broker with incorrect credentials.

```

MQ Telemetry Transport Protocol, Connect Ack
  Header Flags: 0x20 (Connect Ack)
  Msg Len: 2
  Acknowledge Flags: 0x00
  Return Code: Connection Refused: not authorized (5)

```

Figure 8: Attempted client authentication failure. Source: (Harsha et al., 2018).

To carry out and prevent non-authenticated publications, the authors configured the authentication option, limiting publications to customers who have access. Fig. 9 shows the authentication credentials in Wireshark's log.

```

MQ Telemetry Transport Protocol, Connect Command
  Header Flags: 0x10 (Connect Command)
  Msg Len: 27
  Protocol Name Length: 4
  Protocol Name: MQTT
  Version: MQTT v3.1.1 (4)
  Connect Flags: 0xc2
  Keep Alive: 60
  Client ID Length: 0
  Client ID:
  User Name Length: 4
  User Name: test
  Password Length: 7
  Password: test123

```

Username and Password in plain text

Figure 9: Client authentication on broker using credentials. Source: (Harsha et al., 2018).

It is possible to use encryption on the data and Transport Layer Security or Secure Sockets Layer on port 8883 to overcome this vulnerability. That procedure makes data transmission over a secure connection. However, overhead can occur when there are frequent reconnections to the broker.

As a resource, we can use the Access Control List (ACL) – a list configured in the broker, with authorized users for publishing data. For example, the first red rectangle of Fig. 10 shows a customer posting on a topic that has access. In the second rectangle, the same customer tries to post to another topic. This one, however, is refused.

```

pi@raspberrypi:~$ sudo mosquitto -p 1883 -v -c /etc/mosquitto/mosquitto.conf
1525012156: mosquitto version 1.3.4 (build date 2017-05-29 22:25:09+000)
1525012156: Config loaded from /etc/mosquitto/mosquitto.conf.
1525012156: Opening ipv4 listen socket on port 1883.
1525012156: Opening ipv6 listen socket on port 1883.
1525012161: New connection from ::1 on port 1883.
1525012161: New client connected from ::1 as mosqpub/1943-raspberrypi (c
k60, utest).
1525012161: Sending CONNACK to mosqpub/1943-raspberrypi (0)
1525012161: Received PUBLISH from mosqpub/1943-raspberrypi (d0, q0, r0,
0, test, ... (5 bytes))
1525012161: Received DISCONNECT from mosqpub/1943-raspberrypi
1525012178: New connection from ::1 on port 1883.
1525012178: New client connected from ::1 as mosqpub/1944-raspberrypi (c
k60, utest).
1525012178: Sending CONNACK to mosqpub/1944-raspberrypi (0)
1525012178: Denied PUBLISH from mosqpub/1944-raspberrypi (d0, q0, r0, m0
test1, ... (5 bytes))
1525012178: Received DISCONNECT from mosqpub/1944-raspberrypi

```

Figure 10: Authorization to publish topics on the broker. Source: Harsha et al. (2018).

Harsha et al. (2018) show that it is possible to create a list of authenticated clients in the broker for data publishing permission. However, it is still possible to capture the customer's name using Wireshark when sending publisher authentication. Furthermore, it is possible to use encryption to guarantee the integrity and confidentiality of the data. Our method shows that only the client can access authentic data using this procedure.

Another attack on MQTT is the denial of service procedure, as shown in Section 2. In Firdous et al. (2017), the authors show that mobile devices are attractive for performing hacking procedures, as they are always connected to the internet and can be controlled remotely. Another attractive target are web pages.

The work from Firdous et al. (2017) illustrates some scenarios of an attack on MQTT. Among them are the following:

- A user can create multiple TCP sessions in a broker, overloading and depleting the broker's resources.
- A user can send multiple CONNECT packages to the broker, overloading and depleting their resources.
- A user with privileges to send data packets can send many packages to the broker, overloading and depleting resources.
- An internal user can obtain access' data and make malicious publications, compromising the study's results.
- A user with access information can obtain sensitive data for a specific group of customers.

The mentioned procedures can be challenged with some information security implementation techniques, such as firewall, data encryption, and ACL to broker. In [Firdous et al. \(2017\)](#), the authors used the denial of service procedure using a virtual machine. The authors sent two thousand messages of type “publishers” in the proposed experiment, locking the broker for 30 seconds. [Fig. 11](#) shows that the CPU load increased, reaching a peak of 100% utilization.

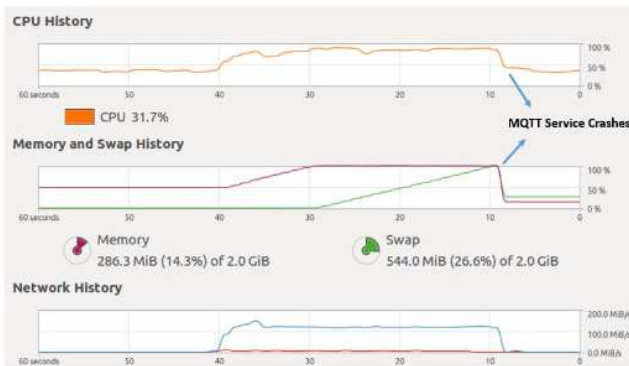


Figure 11: Peak usage of 100% of broker. Source: ([Firdous et al., 2017](#)).

An attack with TCP SYN packets is launched to overload the network bandwidth, increasing the network transfer rate to 300 MB/s, as shown in [Fig. 12](#).



Figure 12: Increased network throughput. Source: ([Firdous et al., 2017](#)).

Despite [Firdous et al. \(2017\)](#) show a practical application of the denial of service procedure, the authors restrict the procedure to a single application. Thus, it is different from this paper, which handles other forms of attack on the MQTT protocol, such as sending false data to the broker and presenting security techniques, such as encryption in the transmission of data.

In [Potrino et al. \(2019\)](#), the authors model and evaluate an information security system to mitigate the damage of a denial of service attack using an intrusion detection system (IDS), which applies a policy of discarding packets not authorized in the MQTT protocol. One can use the IDS to monitor only one server or leave a secure network. Constant monitoring and analysis of the network are necessary for its use, allowing quick decisions when the system detects an attack.

The data monitored by sensor nodes are sent periodically to the nebulizer node using the MQTT protocol. Thus, it is possible to accept some packets with limited frequency, monitor the buffer's integrity,

prioritize to authorized topics, and identify attacking nodes. Contrary to this work, [Potrino et al. \(2019\)](#) demonstrates a firewall's implementation to contain a denial of service attack.

In [Chifor et al. \(2017\)](#), the authors designed the security for the MQTT protocol, protecting messages against DoS attacks. The scenario is a smart city transport system. A base station receives messages from sensors, trusted and unreliable vehicles, aggregates the data, and transmits it to the cloud. Untrusted vehicles can provide helpful traffic information, but malicious devices can easily interrupt communication between the base station and trusted vehicles.

[Chifor et al. \(2017\)](#) propose dividing MQTT into two separate channels, one for data and one for security control. A device authenticates with the broker and reports message delays or applies security policies in the security channel. Then, if multiple authenticated devices report MQTT message delays, the broker will discard messages transmitted by unauthorized devices until the overall network delay is resolved. [Fig. 13](#) illustrates the architecture of the MQTT division.

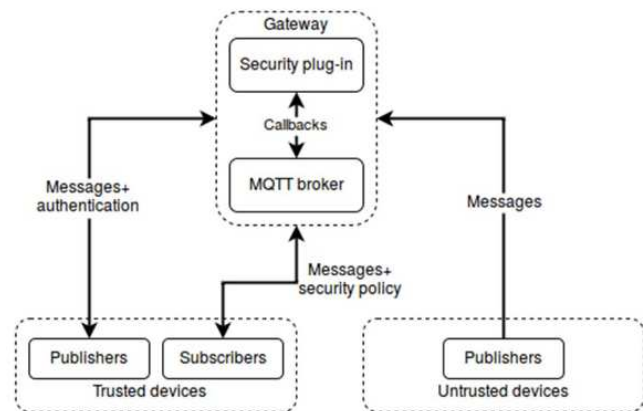


Figure 13: architecture of the MQTT division. Source: ([Chifor et al., 2017](#)).

We did a simulation to verify the consequence of delayed messages. In this simulation, several clients bulk send messages to a broker.

Furthermore, in the experiment, we found that the average delay time in the network increases dramatically. Because of this situation, the broker will discard messages from unauthorized clients and analyze network security policies. Thus, this study shows a more complex implementation of the MQTT protocol, different from the environment proposed by the paper, which analyzes the security of the MQTT protocol implemented in a simplified way.

In agreement with the works presented in this section, the study on information security using the MQTT protocol is quite extensive. In addition to attack procedures, this paper demonstrates the implementation of methods to make attacks more difficult. [Section 4](#) presents the research methodology adopted in this work, showing the concepts and methods we used.

4 Research Methodology

Focusing on the security analysis of the implementation of the MQTT protocol, this paper demonstrates that the protocol's security service quality is not satisfactory, containing loopholes that can be exploited to intercept data and create connection problems between objects in an IoT network. After the attack procedures described in the previous sections, we will implement security methods in a network, reducing a common IoT network's loopholes.

To achieve the aforementioned objectives, attack procedures will be carried out within the information security concept, focused on transmitting data between the publisher and the broker. The purpose of the attack procedures is to disable the connection between the publisher and broker, capture data packets, and send fake data packets. After the tests, we present the results' analyses.

One of the procedures, focusing on the connection between the devices, is the Denial of Service (DoS). That is a widespread method among attacks to interrupt a system's network connection, making it impossible to use. This practice is prevalent to prevent access to a web page (Chen et al., 2018).

To obtain data traveling on the network, we used the method of capturing and analyzing packets, also known as sniffing packets. Somebody can use this method to intercept, catalog, and even decrypt data packets that travel over a network (Dawson and McDonald, 2016).

After executing of the attack procedures mentioned earlier, we will implement a firewall in the broker and the publisher, increasing the difficulty of rendering the connection unusable. Besides, we shall implement data encryption, preventing packet interception, and performing an analysis of possible false data without compromising it.

After the security implementations, we carried out new IoT network attack tests, a new analysis of the results, and compared the results obtained before implementing better security measures.

We used an IoT network implementing the MQTT protocol to perform the procedures through the Python programming language (Nagpal and Gabrani, 2019, Lo et al., 2015).

This IoT network is composed of a smartphone acting as a client. The customer signs the data for a Raspberry Pi3 card containing a DHT11 temperature and humidity sensor, acting as a publisher (Sharmila et al., 2019). Besides, there is a notebook acting as a broker through the implementation of Mosquitto and a message controller program created to act as a broker in network implementations that use the MQTT protocol (Martins, 2019).

In addition to the equipment reported for that IoT network, we used a notebook with a Kali Linux operating system (OS) since it is an OS with several pre-installed software to test information security (Al Neyadi et al., 2020). Fig. 14 illustrates the testing environment.

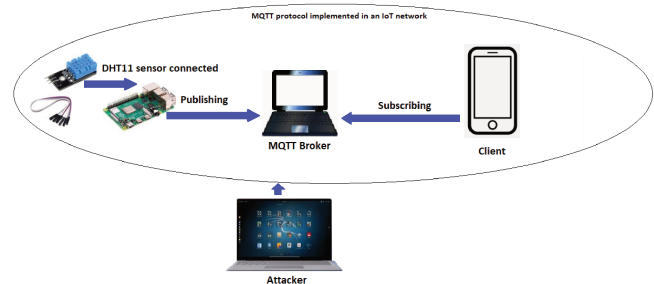


Figure 14: Test environment created for procedures related to the concept of information security.

5 Experiments and Results

This section demonstrates the implementation of information security concepts, both in attacking data and implementing security applications on the proposed IoT network. After the experiments, we discuss the analysis of the results, indicating the MQTT protocol's weakness when implemented in an incautious way.

5.1 Attack Procedures

This section will cover some attack strategies, including Denial of Service, data packet capture and encryption, and sending incorrect data packets.

5.1.1 Denial of Service

To apply the denial of service concept, a relatively large amount of data packets must be sent to the target, causing the bandwidth to experience very high latency in the connection, leading to loss of connection to the network (Liang et al., 2016).

In the proposed experiment, the targets for the denial of service procedure were the publisher containing the DHT11 sensor and the broker. We used the Low Orbit Ion Cannon (LOIC) program to carry out the attack. LOIC is an open-source program written in the C# programming language, aimed at a denial of service attack to test the network's quality (Patil et al., 2018). LOIC sends a large volume of User Datagram Protocol (UDP) request packets, overloading the target, causing it to stop responding to authentic requests.

Fig. 15 illustrates the configuration of the LOIC to perform the denial of service. First, we configured the target for the publisher's IP and later for the broker, both using port 1883. The request uses the UDP protocol and, due to hardware limitation, we use five tasks or threads for carrying out the procedure.

Figs. 16 and 17 show the loss of connection between the publisher and broker in the network, interrupting the data flow to the client.

As shown, the denial of service procedure achieved its objective in the IoT environment by implementing the MQTT protocol. Both attacks took 5 seconds, with a total of 2,132,072 and 3,176,341 requests, respectively, to reach the goal.

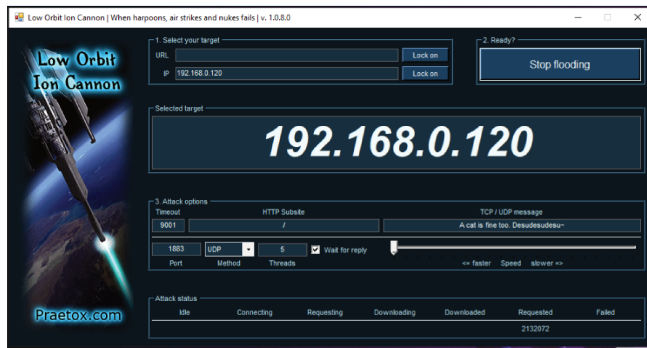


Figure 15: Default configuration of the LOIC program.

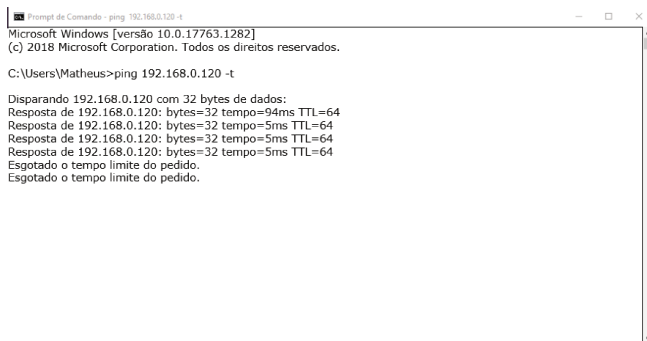


Figure 16: Demonstration of loss of connection with the publisher through the PING test.



Figure 17: Demonstration of the loss of connection with broker through the PING test.

5.1.2 Data Packet Capture

With specialized software, we perform methods for analyzing and capturing packets to obtain data with sensitive information. The best-known and most used software is Wireshark, as it has a simplified and easy-to-use interface (Wang et al., 2010, Das and Tuna, 2017).

In the experiment, Wireshark runs on the notebook and monitors the wireless network, registering all packets that travel on the network. It is possible to configure the Wireshark filter to display only the MQTT protocol. After configuring the filter, the program saves and interprets

the data packets captured on the network, revealing the data sent from the publisher to the broker, including access credentials to the broker.

Fig. 18 shows the credentials visible in the Wireshark when checking the connection line between the publisher and the broker. Also, Wireshark show when the publisher sends the credentials to the broker requesting the connection.

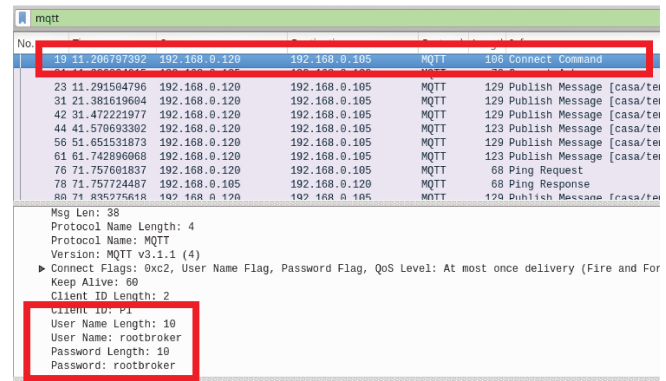


Figure 18: Capture packets with authentication credentials, using the Wireshark.

Fig. 19 shows the publisher's data after checking the line containing the topic in transit on the network. That is a severe security breach, as it is possible to obtain authentication data from the broker. However, depending on the context of the implementation of the IoT network, the data sent to the broker may be confidential and should not be exposed to registration and analysis.

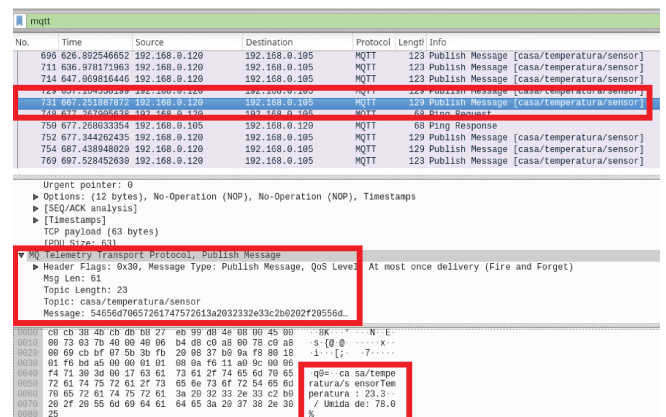


Figure 19: Capture of packages containing data sent from the publisher.

5.1.3 Sending incorrect data packets

Another advantage the attacker used is knowing the connection data, and the topic used to transmit the publisher's data. We can obtain this knowledge through tools aimed at capturing packets, previously mentioned.

After the intercepted packages' registration and analysis, it is possible to create an illusory publisher and send false data, impairing decision-making over the collected data set.

Fig. 18 illustrated the capture of data packets sent from the publisher with authentication credentials, allowing incorrect data to be sent to the broker using an unaccredited publisher (da Silveira, 2020).

Fig. 20 shows that the customer received the false data sent, characterized by the temperature value 31,000 and humidity 21,000. As the subscriber receives the false data, the analysis and interpretation may contain errors, harming the whole experiment.

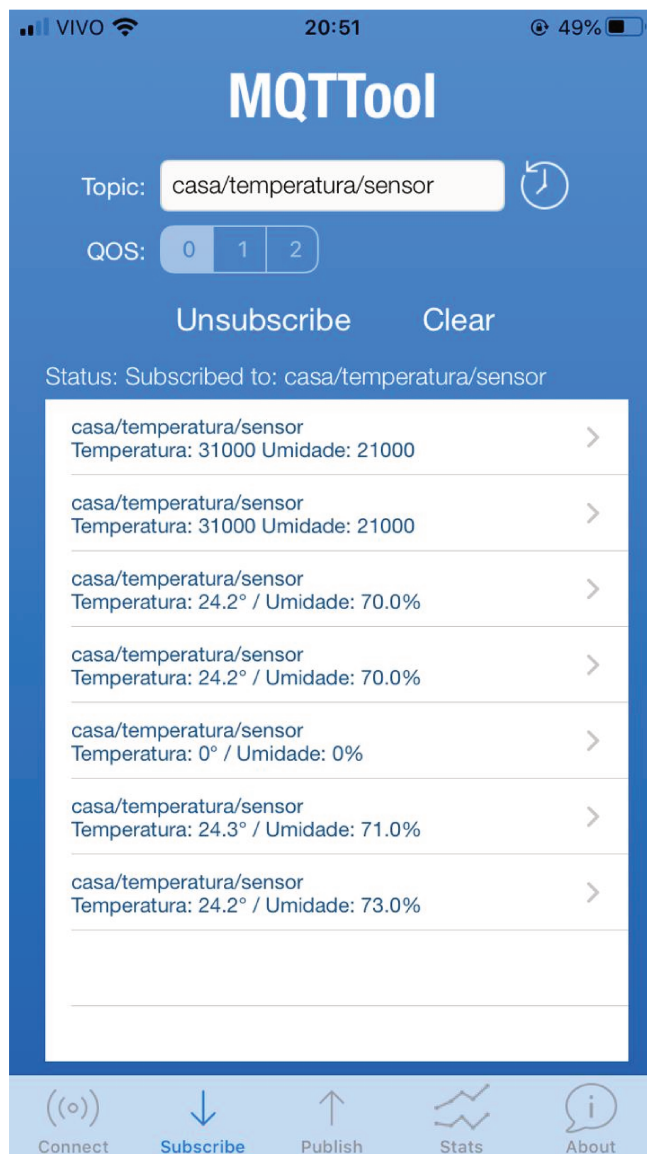


Figure 20: Demonstration that the customer signed the false data, sent by a non-authentic publisher.

5.2 Security measures

Following, we will discuss some strategies for increasing device security for IoT.

5.2.1 Data packet traffic encryption

Data encryption is a widely used technique for protecting data transmitted over a network (Carracedo et al., 2018). In this system, clients used a security key that only the publisher and an authentic subscriber have. Therefore, it is necessary to use the security key to read the data (Oak and Daruwala, 2018).

Fig. 21 shows that an attacker using the Wireshark software can still obtain the topic name. However, it is no longer possible to read the original data sent by the publisher. Instead, when reading the original data, it shows the encrypted data.

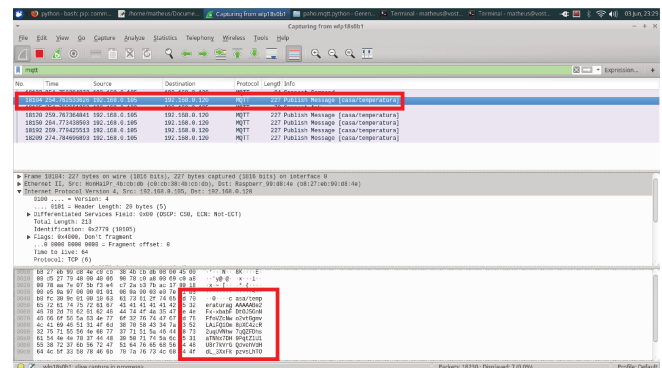


Figure 21: Capturing packages with data sent from the publisher.

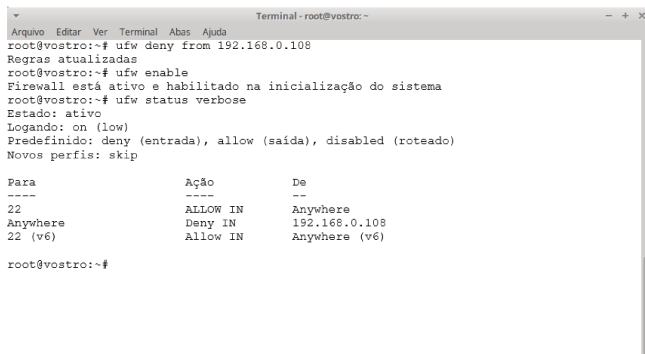
5.2.2 Firewall

A firewall is composed of software or even hardware, intending to implement security policies at a certain point in the network (Gupta et al., 2017). We can use the firewall to filter and analyze the data packets that travel on the network. We can also implement it via proxy, where it handles all requests and then sends them to the server (La Cruz and Goyzueta, 2016). An application firewall (WAF), widely used in web applications, creates a barrier between the business and the internet, filtering and blocking unauthorized access (Clincy and Shahriar, 2018).

For this application, we used the Uncomplicated Firewall (UFW), a firewall rules management interface that uses command lines and is available for Arch Linux, Debian, and Ubuntu distributions. In practice, UFW allows security rules and policies through commands in the Linux terminal (Kroust, 2019).

For the protection of denial of service attacks, two security policies can be implemented, for example. The first is to identify the attacker's IP and deny the receipt of data packets, as shown in Fig. 22. Another policy is to limit the volume of receipt of all data packets of a particular type of protocol, such as UDP, not allowing the target's resources to be rendered unusable. Fig. 23 illustrates the

data limitation rule of the UDP type, implemented in the UFW of a broker.



```

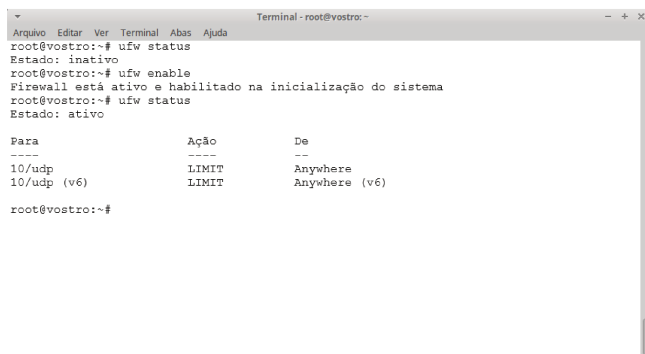
Terminal - root@vostro:~
Arquivo Editar Ver Terminal Abas Ajuda
root@vostro:~# ufw deny from 192.168.0.108
Regras atualizadas
root@vostro:~# ufw enable
Firewall está ativo e habilitado na inicialização do sistema
root@vostro:~# ufw status verbose
Estado: ativo
Logando: on (low)
Predefinido: deny (entrada), allow (saída), disabled (roteado)
Novos perfis: skip

Para      Ação      De
----      -
22        ALLOW IN  Anywhere
Anywhere  Deny IN   192.168.0.108
22 (v6)   Allow IN  Anywhere (v6)

root@vostro:~#

```

Figure 22: Denial of UFW packet received from a given IP address.



```

Terminal - root@vostro:~
Arquivo Editar Ver Terminal Abas Ajuda
root@vostro:~# ufw status
Estado: inativo
root@vostro:~# ufw enable
Firewall está ativo e habilitado na inicialização do sistema
root@vostro:~# ufw status
Estado: ativo

Para      Ação      De
----      -
10/udp    LIMIT     Anywhere
10/udp (v6) LIMIT     Anywhere (v6)

root@vostro:~#

```

Figure 23: Limitation of data packets received from the UDP protocol type.

5.3 Analysis of Results

Implementing the MQTT protocol presents a particular weakness for attacks when implemented in a simplified way, both in connecting devices and reading data traveling on the network. Therefore, we implemented methods to make the IoT network less fragile. That is, to identify and hinder attacks carried out by an external agent, such as data encryption and firewalls. After we carried out the attacks, we demonstrated that it is easily possible to succeed in an IoT network implemented in a simplified way.

With the data encryption method's adhesion, it is no longer possible to read the data, as the interpretation is impractical. Using a firewall, we kept the network connection stable, denying the connection to a specific IP and limiting the volume of data packets received.

6 Conclusion

An IoT network with the simplified implementation of the MQTT protocol has vulnerabilities that can be exploited in several different ways, compromising the confidentiality, integrity, and availability of data, impairing all work and analysis of results. However, using some resources in the network's installation and configuration, it is possible to make it more robust using information security policies. Furthermore, it is possible to prevent the presented vulnerabilities, such as using encryption for the data and implementing a firewall to stabilize the network connection.

6.1 Future Works

We will focus on implementing the IoT network's information security methods using the MQTT protocol, for future works. Such methods will concentrate on the following actions.

- Implement and make available an installation file or source code extraction of complete projects for implementing the MQTT protocol. The publisher, subscriber, and broker already have the appropriate security methods inserted, such as encryption and authentication mechanism.
- Perform scalability studies of the environment, adding several sensors at different credential levels, leaving some data more exposed to public access and others more confidential.
- Check the cost to implement a more robust information security system in the IoT network, depending on the data's criticality.
- Conduct a survey and study the broker's availability in the cloud, where only the publisher and subscriber's implementation in the IoT network is required.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

References

- Al Neyadi, E., Al Shehhi, S., Al Shehhi, A., Al Hashimi, N., Qbea'H, M. and Alrabae, S. (2020). Discovering Public Wi-Fi Vulnerabilities Using Raspberry pi and Kali Linux, *2020 12th Annual Undergraduate Research Conference on Applied Computing (URC)*, IEEE, Dubai, United Arab Emirates, pp. 1–4. Available at <https://doi.org/10.1109/URC49805.2020.9099187>.
- Alghamdi, S. (2019). Shopping and tourism for blind people using RFID as an application of IoT, *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, IEEE, Riyadh, Saudi Arabia, pp. 1–4. Available at <https://doi.org/10.1109/CAIS.2019.8769581>.

- Andy, S., Rahardjo, B. and Hanindhito, B. (2017). Attack scenarios and security analysis of MQTT communication protocol in IoT system, 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), IEEE, Yogyakarta, pp. 1–6. Available at <https://doi.org/10.1109/EECSI.2017.8239179>.
- Anil, K. (2016). Open source implementation of Internet of Things as a Network of Intranet of Things, 2016 International Conference on Information Technology (InCITe) - The Next Generation IT Summit on the Theme - Internet of Things: Connect your Worlds, IEEE, Noida, pp. 215–218. Available at <https://doi.org/10.1109/INCITE.2016.7857619>.
- Carracedo, J. M., Milliken, M., Chouhan, P. K., Scotney, B., Lin, Z., Sajjad, A. and Shackleton, M. (2018). Cryptography for Security in IoT, 2018 Fifth International Conference on Internet of Things: Systems, Management and Security, IEEE, Valencia, pp. 23–30. Available at <https://doi.org/10.1109/IoTSMS.2018.8554634>.
- Chen, L. and Erfani, S. (2017). A note on security management of the Internet of Things, 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), IEEE, Windsor, ON, pp. 1–4. Available at <https://doi.org/10.1109/CCECE.2017.7946616>.
- Chen, Q., Chen, H., Cai, Y., Zhang, Y. and Huang, X. (2018). Denial of Service Attack on IoT System, 2018 9th International Conference on Information Technology in Medicine and Education (ITME), IEEE, Hangzhou, pp. 755–758. Available at <https://doi.org/10.1109/ITME.2018.00171>.
- Chifor, B.-C., Bica, I. and Patriciu, V.-V. (2017). Mitigating DoS attacks in publish-subscribe IoT networks, 2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), IEEE, Targoviste, pp. 1–6. Available at <https://doi.org/10.1109/ECAI.2017.8166463>.
- Clincy, V. and Shahriar, H. (2018). Web Application Firewall: Network Security Models and Configuration, 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), IEEE, Tokyo, Japan, pp. 835–836. Available at <https://doi.org/10.1109/COMPSAC.2018.00144>.
- Cope, S. (2019). MQTT Protocol Packet Structure. Available at <http://www.steves-internet-guide.com/mqtt-protocol-messages-overview>.
- da Cunha, M. J., de Almeida, M. B., Fernandes, R. F. and Carrijo, R. S. (2016). Proposal for an IoT architecture in industrial processes, 12th IEEE International Conference on Industry Applications, IEEE, Curitiba, pp. 1–7. Available at <https://doi.org/10.1109/INDUSCON.2016.7874486>.
- da Silveira, M. F. (2020). MQTTPublisher. Available at <https://github.com/mathferraz/MQTTPublisher>.
- Das, R. and Tuna, G. (2017). Packet tracing and analysis of network cameras with Wireshark, 2017 5th International Symposium on Digital Forensic and Security (ISDFS), IEEE, Tirgu Mures, Romania, pp. 1–6. Available at <https://doi.org/10.1109/ISDFS.2017.7916510>.
- Dawson, J. and McDonald, J. T. (2016). Improving Penetration Testing Methodologies for Security-Based Risk Assessment, 2016 Cybersecurity Symposium (CYBERSEC), IEEE, Coeur d'Alene, ID, USA, pp. 51–58. Available at <https://doi.org/10.1109/CYBERSEC.2016.016>.
- Debian.org (2020). etterfilter – filter compiler for ettercap content filtering engine. Available at <https://manpages.debian.org/testing/ettercap-common/etterfilter.8.en.html>.
- Dholu, M. and Ghodinde, K. A. (2018). Internet of things (iot) for precision agriculture application, 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), IEEE, Tirunelveli, pp. 339–342. Available at <https://doi.org/10.1109/ICOEI.2018.8553720>.
- Firdous, S. N., Baig, Z., Valli, C. and Ibrahim, A. (2017). Modelling and Evaluation of Malicious Attacks against the IoT MQTT Protocol, 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE, Exeter, pp. 748–755. Available at <https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2017.115>.
- Gupta, N., Naik, V. and Sengupta, S. (2017). A firewall for Internet of Things, 2017 9th International Conference on Communication Systems and Networks (COMSNETS), IEEE, Bengaluru, India, pp. 411–412. Available at <https://doi.org/10.1109/COMSNETS.2017.7945418>.
- Harsha, M. S., Bhavani, B. M. and Kundhavai, K. (2018). Analysis of vulnerabilities in MQTT security using Shodan API and implementation of its countermeasures via authentication and ACLs, 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, Bangalore, pp. 2244–2250. Available at <https://doi.org/10.1109/ICACCI.2018.8554472>.
- International Telecommunication Union (2012). Overview of the internet of things, Recommendation ITU-T Y.4000, International Telecommunication Union, Geneva. Available at <http://handle.itu.int/11.1002/1000/11559-en>.
- Jung, J.-j., Kim, K. and Park, J. (2019). Framework of Big data Analysis about IoT-Home-device for supporting a decision making an effective strategy about new product design, 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC), IEEE, Okinawa, Japan, pp. 582–584. Available at <https://doi.org/10.1109/ICAIIIC.2019.8669086>.
- Krout, E. (2019). How to configure a firewall with UFW. Available at <https://www.linode.com/docs/security/firewalls/configure-firewall-with-ufw>.

- La Cruz, J. E. C. and Goyzueta, C. A. R. (2016). Design of a dynamic rules firewall to block avoidance internet censorship systems based on proxy, 2016 *IEEE XXIII International Congress on Electronics, Electrical Engineering and Computing (INTERCON)*, IEEE, Piura, Peru, pp. 1–4. Available at <https://doi.org/10.1109/INTERCON.2016.7815582>.
- Liang, L., Zheng, K., Sheng, Q. and Huang, X. (2016). A Denial of Service Attack Method for an IoT System, 2016 *8th International Conference on Information Technology in Medicine and Education (ITME)*, IEEE, Fuzhou, China, pp. 360–364. Available at <https://doi.org/10.1109/ITME.2016.0087>.
- Lo, C.-A., Lin, Y.-T. and Wu, C.-C. (2015). Which Programming Language Should Students Learn First? A Comparison of Java and Python, 2015 *International Conference on Learning and Teaching in Computing and Engineering*, IEEE, Taipei, pp. 225–226. Available at <https://doi.org/10.1109/LaTiCE.2015.15>.
- Lopez, N. G. (2003). Redes de computadores. Available at https://www.gta.ufrj.br/grad/03_1/ip-security/paginas/introducao.html.
- Martins, V. F. (2019). Automação residencial usando protocolo MQTT, Node-RED e Mosquitto Broker com ESP32 e ESP8266, *Technical report*, Graduação em Engenharia de Controle e Automação – Universidade Federal de Uberlândia, Uberlândia. Trabalho de Conclusão de Curso. Available at <https://repositorio.ufu.br/handle/123456789/28522>.
- Mekruksavanich, S. (2019). The Smart Shopping Basket Based on IoT Applications, 2019 *IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, IEEE, Beijing, China, pp. 714–717. Available at <https://doi.org/10.1109/ICSESS47205.2019.9040750>.
- Nagpal, A. and Gabrani, G. (2019). Python for Data Analytics, Scientific and Technical Applications, 2019 *Amity International Conference on Artificial Intelligence (AICAI)*, IEEE, Dubai, United Arab Emirates, pp. 140–145. Available at <https://doi.org/10.1109/AICAI.2019.8701341>.
- Nagpal, B., Sharma, P., Chauhan, N. and Panesar, A. (2015). DDoS tools: Classification, analysis and comparison, *International Conference on Computing for Sustainable Global Development*, IEEE, New Delhi, India, pp. 342–346. Available at <https://ieeexplore.ieee.org/document/7100270>.
- Navani, D., Jain, S. and Nehra, M. S. (2017). The Internet of Things (IoT): A Study of Architectural Elements, 2017 *13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, IEEE, Jaipur, India, pp. 473–478. Available at <https://doi.org/10.1109/SITIS.2017.83>.
- Oak, A. and Daruwala, R. (2018). Assessment of Message Queue Telemetry and Transport (MQTT) protocol with Symmetric Encryption, 2018 *First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, IEEE, Jalandhar, India, pp. 5–8. Available at <https://doi.org/10.1109/ICSCCC.2018.8703314>.
- OASIS Standard (2019). MQTT Version 5.0, *Technical report*, OASIS Open. Available at <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>.
- Patil, G. V., Pachghare, K. V. and Kshirsagar, D. D. (2018). Feature Reduction in Flow Based Intrusion Detection System, 2018 *3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, IEEE, Bangalore, India, pp. 1356–1362. Available at <https://doi.org/10.1109/RTEICT42901.2018.9012554>.
- Potrinio, G., de Rango, F. and Santamaria, A. F. (2019). Modeling and evaluation of a new IoT security system for mitigating DoS attacks to the MQTT broker, 2019 *IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, Marrakesh, Morocco, pp. 1–6. Available at <https://doi.org/10.1109/WCNC.2019.8885553>.
- Raji, R. (1994). Smart networks for control, *IEEE Spectrum* 31(6): 49–55. Available at <https://doi.org/10.1109/6.284793>.
- Rouse, M. (2020). TCP/IP (transmission control protocol/internet protocol). Available at <https://searchnetworking.techtarget.com/definition/TCP-IP>.
- Samsudin, M. F. A., Mohamad, R., Suliman, S. I., Anas, N. M. and Mohamad, H. (2018). Implementation of wireless temperature and humidity monitoring on an embedded device, 2018 *IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, IEEE, Penang, pp. 90–95. Available at <https://doi.org/10.1109/ISCAIE.2018.8405450>.
- Sharmila, F. M., Suryaganesh, P., Abishek, M. and Benny, U. (2019). Iot Based Smart Window using Sensor Dht11, 2019 *5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, IEEE, Coimbatore, India, pp. 782–784. Available at <https://doi.org/10.1109/ICACCS.2019.8728426>.
- Shinho Lee, Hyeonwoo Kim, Dong-kweon Hong and Hongtaek Ju (2013). Correlation analysis of MQTT loss and delay according to QoS level, *The International Conference on Information Networking 2013 (ICOIN)*, IEEE, Bangkok, pp. 714–717. Available at <https://doi.org/10.1109/ICOIN.2013.6496715>.
- Sinha, A., Sharma, S. and Mahboob, M. R. (2017). An Internet of Things based prototype for smart appliance control, 2017 *International Conference on Computing, Communication and Automation (ICCCA)*, IEEE, Greater Noida, pp. 1358–1363. Available at <https://doi.org/10.1109/CCAA.2017.8230009>.
- Wang, S., Xu, D. and Yan, S. (2010). Analysis and application of Wireshark in TCP/IP protocol teaching, *International Conference on E-Health Networking, Digital Ecosystems and Technologies*, Vol. 2, IEEE, Shenzhen, pp. 269–272. Available at <https://doi.org/10.1109/EDT.2010.5496372>.

- Yadav, P. and Vishwakarma, S. (2018). Application of Internet of Things and Big Data towards a Smart City, *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, IEEE, Bhimtal, pp. 1–5. Available at <https://doi.org/10.1109/IoT-SIU.2018.8519920>.
- Yassein, M. B., Shatnawi, M. Q., Aljwarneh, S. and Al-Hatmi, R. (2017). Internet of Things: Survey and open issues of MQTT protocol, *2017 International Conference on Engineering & MIS (ICEMIS)*, IEEE, Monastir, pp. 1–6. Available at <https://doi.org/10.1109/ICEMIS.2017.8273112>.
- Yuan, M. (2017). Conhecendo o MQTT. Available at <https://developer.ibm.com/br/technologies/iot/articles/iot-mqtt-why-good-for-iot>.