information

MDPI

Article

SlowTT: A Slow Denial of Service Against IoT Networks

Ivan Vaccari [1,2,*], Maurizio Aiello [1] and Enrico Cambiaso [1]

[1] Consiglio Nazionale delle Ricerche (CNR), IEIIT Institute, 16149 Genoa, Italy; maurizio.aiello@ieiit.cnr.it (M.A.); enrico.cambiaso@ieiit.cnr.it (E.C.)
[2] Department of Informatics, Bioengineering, Robotics and System Engineering (DIBRIS), University of Genoa, 16145 Genoa, Italy
* Correspondence: ivan.vaccari@ieiit.cnr.it; Tel.: +39-010-6475-215

Received: 18 August 2020; Accepted: 15 September 2020; Published: 18 September 2020

Abstract: The security of Internet of Things environments is a critical and trending topic, due to the nature of the networks and the sensitivity of the exchanged information. In this paper, we investigate the security of the Message Queue Telemetry Transport (MQTT) protocol, widely adopted in IoT infrastructures. We exploit two specific weaknesses of MQTT, identified during our research activities, allowing the client to configure the KeepAlive parameter and MQTT packets to execute an innovative cyber threat against the MQTT broker. In order to validate the exploitation of such vulnerabilities, we propose SlowTT, a novel "Slow" denial of service attack aimed at targeting MQTT through low-rate techniques, characterized by minimum attack bandwidth and computational power requirements. We validate SlowTT against real MQTT services, by considering both plaintext and encrypted communications and by comparing the effects of the attack when targeting different application daemons and protocol versions. Results show that SlowTT is extremely successful, and it can exploit the identified vulnerability to execute a denial of service against the IoT network by keeping the connection alive for a long time.

Keywords: Internet of Things, protocols security; cyber security; network security; slow DoS attack; MQTT

1. Introduction

Currently, the Internet of Things (IoT) is a consolidated technology and is integrated into daily life and communication systems. In Internet of Things applications, simple objects are able to elaborate, process, and communicate information of the surrounding applications and send it to other IoT devices or more complex systems. In the IoT world, IoT applications can be different depending on the user's needs. Some examples of applications concern the monitoring of workouts through IoT devices [1], the control of the perimeter of houses/industrial environments [2], or sensors to measure the temperature and humidity of a monitored area [3]. The adoption of IoT in our lives introduces new opportunities for a large number of applications, where the primary aim is to improve the quality of human life.

A very interesting and practical application of IoT technology is in Industry 4.0. Intelligent devices and machinery, implemented following the innovation brought by the IoT, are revolutionizing the scenario of the production and maintenance of machines, becoming increasingly innovative through the production of data and digital analysis. The Industrial Internet of Things (IIoT), the so-called the industrial application of the IoT, is a term coined to be adopted and applied exclusively in the new era of industry called Industry 4.0 [4]. The purpose of the IIoT is to optimize production processes, connecting machines together and supporting data processing to allow predictive analysis activities,

able for example to predict maintenance requirements and activities, thus potentially leading to the reduction of significant costs and more efficient production systems.

Due to the different nature of IoT applications, a standard that defines in detail the protocols or constraints to connect IoT devices has not yet been created. For this reason, IoT networks could be implemented by using different communication protocols. An ad hoc communication protocol available for implementing IoT networks is Message Queue Telemetry Transport (MQTT), a publisher/subscriber system introduced in 1999 [5] and adopted currently [6] in the IoT [7], the IIoT [8], and other contexts [9]. The MQTT protocol is based on the TCP/IP stack, and it is positioned at the application layer of such a stack. Being a protocol adopted in IoT and IIoT networks, communications security is a critical and delicate aspect, since the processed data/information are related to sensitive industrial contexts [10]. Moreover, the security of IoT systems is a critical topic since it is important to identify possible vulnerabilities and innovative threats to protect them efficiently as the information and data transmitted are extremely sensitive [11].

In this paper, we study and evaluate the security aspects of the MQTT protocol, since such a protocol is adopted both in industrial [12] and home automation applications [13], to identify possible vulnerabilities and design innovative attack approaches to exploit the possible weaknesses of the protocol. In particular, the proposed research work is contextualized in the topic of IoT security, where the exploitation of specifics parameters of MQTT are introduced, through the design, development, and validation of an improved version of the Slow DoS against Internet of Things Environments (SlowITe) attack [14] called SlowTT, the acronym of Slow denial of service against MQTT. SlowITe is a preliminary slow denial of service attack (SDA) against MQTT. In this work, we try to improve the SlowITe cyber threat, by improving the weaknesses of the tool, and to exploit other vulnerabilities identified inside the protocol and its parameters. Since, to the best of our knowledge, no previous SDA threats have been designed to target MQTT except SlowITe, while the main focus is HTTP and HTTPS [15–17], the proposed SlowTT threat should be considered a relevant advancement in the IoT security field.

The remainder of the paper is structured as follows: Section 2 reports the related work on the topic. Section 3 describes the MQTT protocol, while Section 4 describes in detail the SlowTT attack. Executed tests and obtained results are reported in Sections 5 and 6. Section 7 introduces instead protection approaches able to protect an IoT system from the proposed threat. Finally, Section 8 concludes the paper and reports further works on the topic.

2. Related Work

In order to detect possible vulnerabilities that could be exploited by an attacker in an IoT network, it is important to consider the security of wireless networks and the panorama of attacks against these networks. Given the low power consumption and the limited capacity of the IoT devices, security features are not always adequately implemented [18]. For this reason, security research activities related to IoT environments are important in order to improve the security of these devices and ensure greater reliability and efficiency in data transmission. An analysis about the security of IoT networks by identifying crucial aspects related to common vulnerabilities by considering both hardware and software aspects is investigated in [19,20], while a focus on security challenges in the IoT field, also proposing security solutions, in addressed in [21,22]. Likewise, security issues in environments such as healthcare, smart home, or vehicle management are analysized in [23]. Our work covers all IoT environments since the threat developed is able to target the central node of the network independently of the context.

By considering the security investigation of communication protocols for IoT networks, studies and comparisons between communication protocols available on this topic, by analyzing the characteristics, features, and functionalities, are investigated in [24–30]. The communication protocols available for IoT could be based on existing communication protocols, such as CoAP, MQTT, or the Advanced Message Queuing Protocol (AMQP), that are based on the TCP/IP stack; instead, other

protocols change the entire stack, for example, ZigBee 6LoWPAN or Z-Wave. The security aspects of these protocols are interesting research topics as demonstrated in [31–42]. Based on these considerations, the IoT requires more studies and works related to security aspects.

The Constrained Application Protocol (CoAP) is a specialized Internet application protocol for constrained devices [43]. It allows devices with limited computational power to communicate on the Internet. The protocol is specific for constrained hardware such as micro-controllers, low-power sensors, and IoT devices that cannot be executed over HTTP or TLS. It is a simplification of the HTTP protocol running on UDP, which helps to save bandwidth. It is designed for use between devices on the same constrained network (e.g., low-power and data loss networks), between devices and general nodes on the Internet, and between devices on different bound networks, both connected by an Internet network [44]. CoAP allows ensuring the security of the communication by implementing a Datagram Transport Layer Security (DTLS) , a TLS version on UDP [45], in order to protect the information [46,47]. Security aspects of CoAP by investigating common threats such as replay, DoS, and man-in-the-middle are analyzed in [48,49]. CoAP is vulnerable also to IP address spoofing.

The AMQP [50] is an open standard that defines an application-level protocol for message-oriented middleware. AMQP is defined in such a way as to guarantee messaging, queuing, routing (with point-to-point and publication-subscription paradigms), reliability, and security. AMQP is an application-level binary protocol designed to efficiently support a wide range of messaging applications and communication schemes. The AMQP standard consists mainly of [51]: message, the key element of the entire communication process; producer, creates a message and sends it; broker, distributes the message according to rules defined to different queues; and finally, consumer, which takes the message from the queue where it can access and reprocess it. The security aspects and threats were analyzed in [52]. In particular, the authors showed that AMQP could be exploited by using common threats such as replay [53], masquerade, modification, and denial of service [54].

MQTT is a lightweight, publish-subscribe ISO standard (ISO/IEC PRF 20922) protocol placed on top of TCP/IP [55]. It has been designed for situations where low impact is required and where bandwidth is limited. The publish-subscribe pattern requires a message broker. MQTT provides three types of quality of service [6]. Facebook currently uses MQTT for their messaging app, not only because the protocol retains battery power during cell-to-phone messaging, but also because, despite inconsistent Internet connections around the world, the protocol allows delivering messages efficiently in milliseconds [56]. Most major cloud service providers [57], including AWS, Google Cloud, IBM Bluemix, and Microsoft Azure, support MQTT, as well as the IoT Carriots, Everything, and ThingWorx platforms. MQTT is suitable for applications that use Machine-to-Machine (M2M) and IoT devices for real-time analysis, preventive maintenance, and monitoring, among other uses, in environments such as smart homes, healthcare, logistics, industry, and production [58]. The adoption of Shodan (available at: https://www.shodan.io/) to detect a vulnerable MQTT broker connected to the public Internet and how to exploit it is proposed in [59,60]. A distributed denial of service attack with the related detection system is implemented in [59–61]. Cyber threats against MQTT such as DoS, identity spoofing, information disclosure, the elevation of privileges, and data tampering are executed in [62,63]. By analyzing these works, our work is focused on another threat classification: the slow DoS attacks. These types of threats are not widely considered in the IoT context, so the proposed level brings an improvement in the security of IoT devices and networks.

A comparison of different communication protocols was accomplished by [64–67], where MQTT, CoAP, AMQP, and HTTP were investigated in terms of security, power consumption, resource requirements, Quality of Service (QoS), reliability, and the characteristics of the protocols.

The previous works mentioned highlighted the critical security of IoT protocols since a malicious user can exploit possible vulnerabilities or weaknesses in order to create damage or to steal sensitive information in the environments where IoT technologies are applied. During our work, we theorized about and implemented SlowTT, an innovative cyber threat against MQTT able to make an MQTT-based IoT network unreachable. It is based on SlowITe, the first version of the cyber

threat [14]. SlowTT is an advancement of SlowITe to optimize the threat in terms of bandwidth and efficiency and to solve SlowITe's limits (e.g., SlowITe is able to keep connections alive for a specific time [14]). The proposed attack should be considered an advancement in the field of the research of communication protocols used in the Internet of Things networks.

3. The MQTT Protocol

As mentioned in Section 2, IoT networks could be implemented by using different application protocols such as MQTT, AMQP, or CoAP or communication protocols that implement the entire communication stack such as ZigBee or Z-Wave. In this work, we focused on the MQTT protocol since this application communication protocol is widely adopted in industrial [68], automation [3], healthcare, and other different contexts [55]. Moreover, the Organization for the Advancement of Structured Information Standards (OASIS) has declared that IBM's MQTT protocol is the reference standard for communication for the Internet of Things [69]. For these reasons, the security of the MQTT protocol assumes a crucial and critical role in research activities since it is adopted in different contexts where sensitive information is exchanged.

MQTT is a lightweight messaging protocol, created to obtain a protocol able to minimize bandwidth in device communication and optimize energy consumption for applications that require low battery power or with reduced computational power.

The MQTT protocol is based on the TCP/IP stack, positioning itself at the application level [64]. The MQTT protocol stack is mainly composed of four levels: physical, network, transport, and application. The physical layer is associated with Ethernet or WiFi connections, while the network layer is tied to IPv4 or IPv6 addresses (depending on how the network is set up). The transport layer guarantees a stable connection thanks to TCP, and finally, at the application level, MQTT is positioned.

The protocol, which adopts a publish/subscribe communication model based on a central node hosting the MQTT server called the broker, is used for M2M and plays an important role in the IoT. In order to exchange information, the MQTT protocol uses a mechanism for publishing and subscribing messages through the broker, able to manage thousands of clients simultaneously. In MQTT, if a particular client wants to communicate with another client, it publishes a message on a certain channel (called the topic) for the message broker. The message broker has the task of filtering and distributing communications between publishers and subscribers. Each client can subscribe to multiple topics, and every time a new message is published in a specific topic, the message broker distributes it to all clients subscribed to that specific topic. An example of an MQTT network is shown in Figure 1. In this example, two different sensors publish two different messages, m_1 and m_2, on the same MQTT topic t. From the other side, when another device connects to the MQTT broker on the topic t, it receives the messages m_1 and m_2.

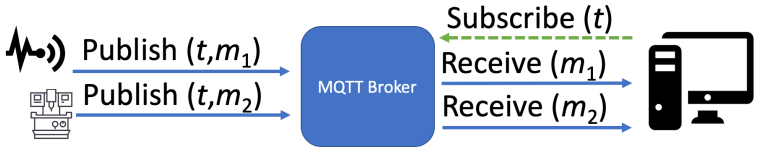

Figure 1. Sample of a MQTT network.

The MQTT protocol is an interesting solution for wireless networks since its features allow managing and mitigating possible connection issues. For example, if the communication from a subscribed client to a broker is interrupted, the broker will store the messages for that client and send messages when the client is connected again.

An MQTT session is divided into four phases: connection, authentication, communication, and termination. A client starts by creating a TCP/IP connection to the broker using a standard port or a custom port defined by the broker's operators. The standard ports are 1883 for unencrypted communications and 8883 for encrypted communications using SSL/TLS. During the SSL/TLS

handshake, the client validates the server certificate to authenticate it. The client can also provide a certificate to the broker server during the handshake. Since the MQTT protocol aims to be a protocol for devices with limited resources and the IoT [70], SSL/TLS may not always be an option [71]. In such cases, authentication could be implemented by using a username and plaintext password that is sent by the client to the server as part of the CONNECT/CONNACK packet sequence [72]. These packets are adopted by MQTT to authenticate a device on the broker. Otherwise, if devices have sufficient computational power, MQTT allows authenticating devices by using the username/password and communicating this information encrypted with SSL.

MQTT is considered a lightweight protocol because all its messages have a reduced code footprint. Each message consists indeed of a fixed header, an optional variable header, a message payload with a maximum size of 256 MB of information, and a QoS payload. The three different quality of service levels determine how the content is managed by the MQTT protocol [7]. Although higher levels of QoS are more reliable, they have more latency and bandwidth requirements, so subscriber customers can specify the highest QoS level they wish to receive.

During the communication phase, a client can perform different operations, such as publication, subscribe, unsubscribe, or ping. Considering the proposed work, the most important packets available in MQTT networks are:

- CONNECT/CONNACK: packets adopted to authenticate a device on the MQTT broker.
- SUBSCRIBE/SUBACK: packets adopted to subscribe a client on a specific topic.
- UNSUBSCRIBE/UNSUBACK: packets adopted to unsubscribe a client from the topic.
- PINGREQ/PINGRESP: packets adopted to ping the broker server [73].
- DISCONNECT: packets adopted to disconnect the client from the MQTT broker [74].

4. The SlowTT Attack

The proposed slow denial of service against MQTT (SlowTT) is an optimized and more efficient version of the SlowITe attack against the MQTT protocol. As described in [14], SlowITe has some limitations that SlowTT aims to optimize and improve. In particular, SlowITe is able to keep all the connections available on the network busy for the time defined in the KeepAlive parameter (by exploiting the 1.5 times management of the KeepAlive parameter), and SlowTT aims to keep connections alive for an infinite time by exploiting the parameters and network configurations adopted by the MQTT protocol in order to avoid re-establishment of the connections or to avoid the high value of the KeepAlive parameter that could be suspected (although defining a high KeepAlive value as not legitimate is not correct). In addition, SlowTT allows setting lower KeepAlive values as it keeps connections active by exploiting the PING packets to simulate an even more legitimate behavior of IoT sensors (think of a fire alarm that remains connected, but as long as there is no fire, it never communicates information). The value of KeepAlive does not affect the behavior of the attack as SlowTT is able to keep the connections open by exploiting the PING packet. Furthermore, the attack by keeping the connections open prevents a legitimate client from occupying them as soon as the broker closes them. In that case, the DoS status is still reached for each subsequent client, but this could lead to problems as the legitimate connection is not controlled by the attacker and should be occupied as soon as it becomes free [17]. With SlowTT, however, this does not happen as the connections are not closed by the broker. Therefore, starting from the first version of the attack, we carried out an in-depth study of the MQTT communication protocol to identify other parameters and network packets to be used to implement the innovative version of the attack.

From the first version of SlowITe, an innovative version was idealized, theorized, and implemented. The aim is to implement a denial of service attack against the MQTT protocol that makes a network unreachable by legitimate users and also that is difficult to identify and mitigate. Based on these features, we developed the SlowTT tool to send MQTT packets to the broker. The threat tool instantiates a high number of connections with the MQTT broker and tries to keep connections busy

to prevent the client's connection by exploiting the CONNECT and PING packets. In order to briefly summarize the SlowTT approach, the tool has the following flow:

1. Connection with the MQTT broker by exploiting the CONNECT packet.
2. Sending packets to keep the connection alive by exploiting the PING packet.

With this approach, the MQTT broker will never close the connection because through the PING packet, the attacker keeps instantiated connections alive. In order to develop an efficient version of the SlowTT tool, initially, three different versions of the thread were developed based on packets adopted to perform the attack.

Before proceeding with tests to validate the other versions of SlowTT, we further optimized the attack by working on two very important parameters: Wait Timeout (WT) and KeepAlive. The Wait Timeout is a client-side parameter configured by SlowTT to define the interval time between each PING packet to keep the connection alive. The KeepAlive parameter, on the other hand, is adopted in communication with the MQTT broker and used server-side. Such a parameter identifies the time MQTT connections remain alive (without an exchange of packets) before the MQTT broker closes the connection with the client. These parameters are related among themselves, as the Wait Timeout has to be lower than the server-side timeout (e.g., KeepAlive, for MQTT), in order to avoid server-side connection closures. After an in-depth study of the MQTT protocol, we noted two critical issues regarding the KeepAlive parameter that can be exploited by the attacker to perform SlowTT effectively and efficiently. The KeepAlive is not defined by the MQTT broker, but by the client through the CONNECT packet during the authentication phase; in addition to the client's ID, the KeepAlive is also transmitted. This can be used by an attacker to occupy the connection as much as possible without the MQTT broker closing it. The maximum time supported by the KeepAlive is 65,535 seconds. The second critical issue is that the protocol MQTT keeps the connections alive for 1.5 times the KeepAlive time, specifically with the following sentence "If the Keep Alive value is non-zero and the Server does not receive a Control Packet from the Client within one and a half times the Keep Alive time period, it MUST disconnect the Network Connection to the Client as if the network had failed" in the MQTT white paper (available at: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html). These two critical issues are fully exploited in SlowTT; as with the CONNECT packet, the attack tool sets a KeepAlive, and knowing that the MQTT broker closes the connection after 1.5 times the KeepAlive time, the tool sends the PING packet a few seconds before connection closure. These two aspects are very critical and important since by configuring in the right way the KeepAlive and Wait Timeout parameters, it is possible to drastically reduce the number of packets sent on the network by the attacker. In this way, the threat is very complex to detect since the tool simulates, in all its aspects, a legitimate connection that sends packets with a certain time.

Finally, in order to make the attack efficient and make the MQTT broker unreachable by legitimate clients, SlowTT must occupy all available connections. The number of connections that the MQTT broker is able to manage is defined within the configuration file. By default, a maximum number of connections is not set, but again, within the configuration file, it is described that the maximum number of connections is "around 1024" (available at: https://github.com/eclipse/mosquitto/blob/master/mosquitto.conf) since most operating systems have a default socket limit of 1024 for processes, so during the test phase, the SlowTT attack will create an number of connections equal to 1024 to execute the attack against the MQTT broker. Table 1 summarizes the improvement of SlowTT compared with SlowITe.

Table 1. Features comparison between Slow denial of service against MQTT (SlowTT) and SlowITe.

Feature	SlowTT	SlowITe
Duration time	Keep connections alive for infinite time	When the 1.5 times KeepAlive value expires, connections are closed
KeepAlive	The attack does not depend on the value of KeepAlive	The attack strictly depends on the value of KeepAlive
Behavior	More realistic behavior (CONNECT and PING)	Anomalous behavior (just CONNECT is a rare approach)

5. Testbed

In order to test and validate the innovative SlowTT attack proposed in Section 4, a dedicated IoT test network based on MQTT was implemented. The network contains an MQTT broker, in order to manage messages exchanged on the MQTT network and different connected clients. The victim of the SlowTT attack is the MQTT broker since SlowTT aims to make it unreachable and to avoid legitimate connections.

As demonstrated in [14,75–77], a slow DoS attack, against a web application or an IoT protocol such as MQTT, requires low performance, computational power, and bandwidth to target the victim. Furthermore, mobile smartphones, botnets, and basic IoT devices are able to execute this kind of threat. For these reasons, we decided to adopt an IoT device for the vector attack; in particular, we adopted a Raspberry Pi 3 Model B connected to the broker. The testbed network aims to represent a real network scenario where IoT devices with limited computing power and hardware are able to execute cyber threats against other devices connected to the real network.

The MQTT broker is implemented by using the Eclipse Mosquitto tool [78], an open-source message broker (EPL/EDL license) that implements the MQTT protocol Versions 5.0, 3.1.1, and 3.1. Mosquitto is lightweight and is suitable for use on all devices, from single board low-power computers to complete servers. In particular, Mosquitto v1.6.2 based on MQTT Version 3.1.1 was installed in the broker. The MQTT broker implements standard configuration (e.g., the MQTT listening port is 1883 for clear communication; instead, the listening port for encrypted communication is 8883), and no constraints on the maximum connection number were entered so that, as mentioned in Section 3, the MQTT broker is able to manage around 1024 connections (due to limits on the number of processes the server can manage). Furthermore, TLS encryption and MQTT Version 5 are considered in these tests to evaluate the new version of the SlowTT attack. In particular, communication is encrypted with SSL/TLS (TLS v1.2).

In order to validate and verify the efficiency of the SlowTT attack, validation software was developed to verify that a DoS attack is not simple because the MQTT broker should be, at the same time, alive and dead (based on the Schrödinger's cat paradox [79]). In order to verify if SlowTT is able to reach the DoS status, the validation tool alters the MQTT broker status by trying to connect a legitimate client to the broker. This solution checks every second if it is possible to connect to the broker. The validation tool developed is adopted to validate SlowTT's efficacy server-side: if the validation software is able to connect, that means that the DoS attack was not effective (in that time); instead, if it is not able to connect, this means that the attack is successful because all possible connections are instantiated by SlowTT.

The following scenario can be summarized in the following steps:

- The Raspberry Pi device starts to execute the SlowTT attack.
- In order to validate the attack, the MQTT broker checks if there are connections available on the MQTT broker by using the validation tool.

The tests and results reported in Section 6 were carried out with the following times: 20 min of execution of the attack, KeepAlive set at 65,535 s, and Wait Timeout set to 60 s. The results obtained

are extremely interesting and innovative as they demonstrate that the SlowTT attack is able to make an MQTT network inaccessible to legitimate clients.

6. Executed Tests and Obtained Results

In this section, we report the obtained results related to the performed tests. Initially, we developed different versions of SlowTT by combining possible packets available in MQTT communications. Then, we performed preliminary tests to identify the best approach to develop an efficient and performing version of the tool. Finally, we describe in detail the obtained results in terms of network traffic and bandwidth.

6.1. Identification of the Best Approach to Adopt

As mentioned in Section 4, we implemented three different versions of the cyber threat in order to achieve the purpose of the tool. The concept of the implementation is to test different versions of the tool by combining network packets available in MQTT. Based on these constraints, the developed versions are the following:

- Connect + Ping: both the CONNECT and PING packets adopted; in the following called *SlowTT*.
- Only Connect: only CONNECT packets adopted. This version is *SlowITe*.
- Only Ping: only PING packets adopted; in the following called *Slowping*.

The Slowping version is not able to perform the attack since without the CONNECT packet, it is impossible to complete the authentication process on the MQTT server based on the TCP three way handshake between the attacker and the broker, so this version is not able to establish a connection to execute the attack. The attack flow is reported in Figure 2. The Wait Timeout parameter is reported as *wt* in the figure.

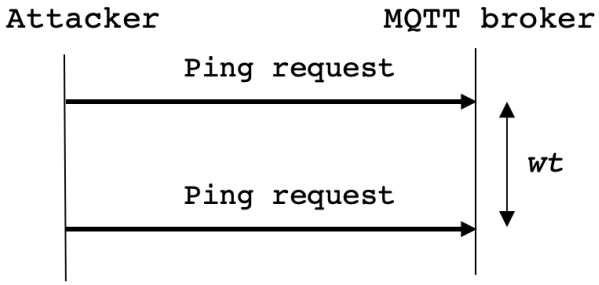

Figure 2. Slowping attack flow.

SlowITe instead is able to establish multiple connections with the MQTT server and to set the KeepAlive value since the authentication phase is completed by the attacker device. After this phase, as shown in Figure 3, SlowITe does not send any other packets to the MQTT server due to its nature. We further investigate and compare this attack with other versions in the rest of the paper.

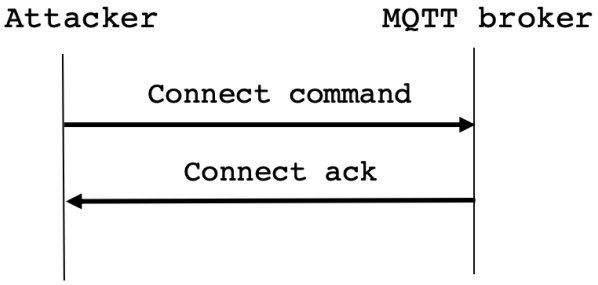

Figure 3. SlowITe attack flow.

Finally, the SlowTT version is the most promising version of SlowTT since, after establishing a connection by using the CONNECT packet and setting the KeepAlive parameter, it sends PING packets before the KeepAlive expired to keep the connection alive, as reported in Figure 4. The Wait Timeout parameter is reported as *wt* in the figures.

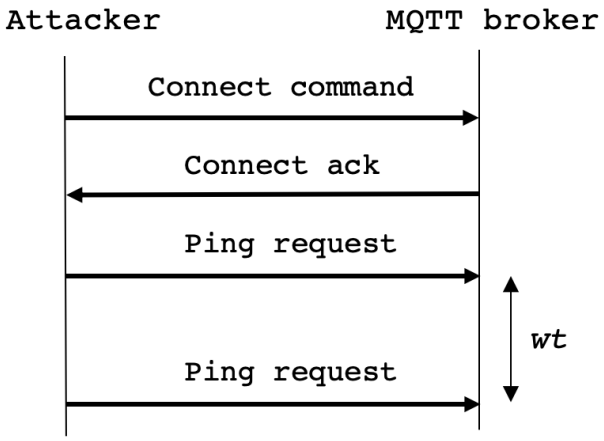

Figure 4. SlowTT attack flow.

Based on the previous consideration, we investigated and evaluated SlowITe and SlowTT since the Slowping version is not able to perform the attack since it is unable to create a TCP connection.

In order to validate the SlowTT attack, the first test executed is to demonstrate the difference between the SlowTT and SlowITe versions. For this test, we executed the attack for 120 s with the KeepAlive set to 60 s and with a PING message after 80 s in order to keep the connection established. The PING message is sent after 80 s since, as mentioned in Section 4, connections are kept alive for 1.5 times the KeepAlive value that, in this test equal to 90 s. Figure 5 shows the obtained result in terms of network traffic.

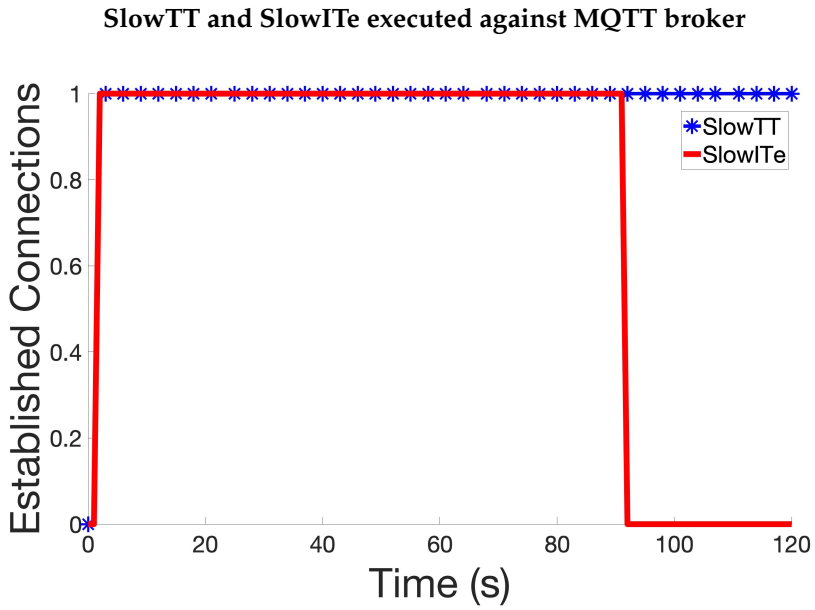

Figure 5. Comparison between the SlowTT and SlowITe attack tools.

The result demonstrate that the SlowTT attack (blue line in Figure 5) is able to keep the connection alive over the time set by the KeepAlive parameter by exploiting the PING packet. Instead, the SlowITe version of the tool (red line in Figure 5) closed the connection when the KeepAlive time expired. Furthermore, Figure 5 demonstrates that the MQTT protocol keeps the connection alive for 1.5 times

the value of the KeepAlive parameter, as mentioned in Section 3. Furthermore, the confirmation of MQTT's behavior on the KeepAlive parameter can be exploited by the attacker to optimize the SlowTT attack and drastically reduce the number of packets needed to make the MQTT broker unreachable, simulating normal network traffic and thus making the detection of the activities very complex.

6.2. Executed Tests and Results on the Testbed

In the previous section, we demonstrated the different approaches between SlowTT and SlowITe. Although the aim of both attacks is the same, the approach adopted in SlowTT is innovative since it is possible to keep alive instantiated connections for an infinite time, while in SlowITe, once the KeepAlive time expires, the broker automatically closes the connections.

After validating the SlowTT approach, in this section, we carried out further tests to validate the efficiency and performance of SlowTT compared with SlowITe. The tests were performed for both attacks by using the same scenario to avoid technical issues with configurations/parameters/features. In summary, the tests were performed for 20 min, with a KeepAlive value of 65,535 s and a Wait Timeout value of 60 s, as described in Section 4. The value of KeepAlive was not considered of significant importance in these tests since the aim is to demonstrate that the attack is able to make a broker unreachable, the target of DoS attacks; it is used in the previous test to prove that the SlowTT attack is able to keep the connection alive for a potentially infinite time. Firstly, we tested the SlowTT version of the attack. The attacker initialized the connections and every 60 s sent a PING packet against the MQTT broker. At the same time, on the MQTT Mosquitto broker, the broker checks every second if there are connections available in order to validate the efficacy of SlowTT. The broker reports zero if a connection is available on the broker or one if a connection is not available. The results obtained are shown in Figure 6.

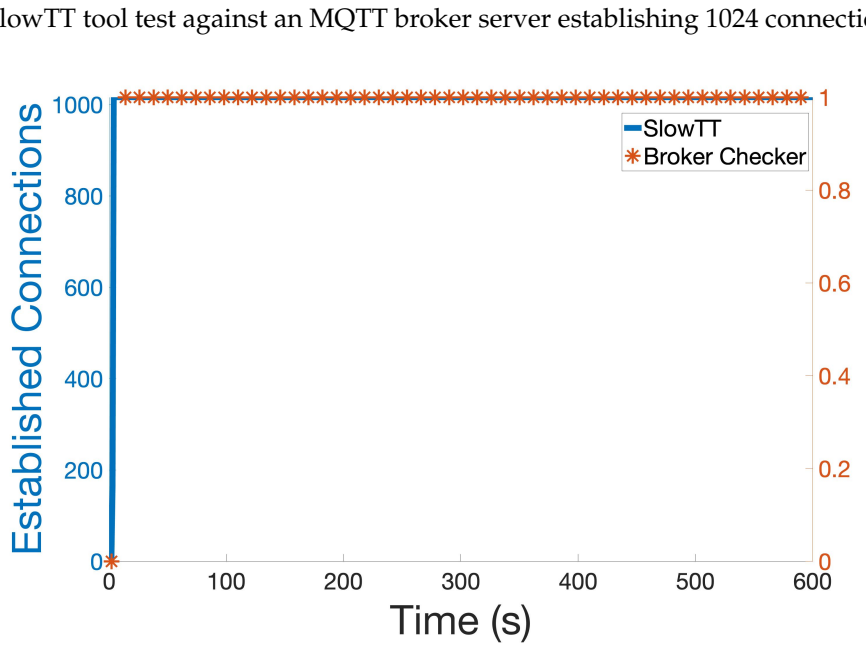

Figure 6. Network traffic of the SlowTT.

The results obtained demonstrate the efficiency and innovation of the SlowTT attack. As shown in Figure 6, the attacker creates 1016 connections; at the same time, the MQTT broker can no longer accept other requests since all connections are instantiated by the attack. The results are confirmed by the validation since it reported the value one. Analyzing Figure 6, when the attack begins, the maximum number of connections managed by the MQTT broker has not yet been reached, and the MQTT broker reports zero as there are connections available. By analyzing the obtained results, it is possible to confirm that SlowTT is able to perform a DoS attack against an MQTT broker, making the broker

unreachable by establishing all available connections on the MQTT server, and therefore, no clients are able to connect and communicate with other devices. From the bandwidth point of view, the SlowTT version requires 42.89 kbps. The execution of this attack against an MQTT broker can lead to serious damages, as MQTT is used in very critical environments (industrial, logistics, critical infrastructure monitoring, healthcare, smart homes, cyber physical systems, etc.) by IoT devices to communicate in real time sensitive data and information recovered from the environment in analysis, and in the case that the communication channel stops working, critical situations can arise with the loss of sensitive information.

After running the test using SlowTT, we performed the same tests with SlowITe in order to compare and evaluate the results. The scenario configuration was the same as the previous test; only the attack tool changed. The test was performed for 20 min, with a KeepAlive value of 65,535 s, a Wait Timeout value of 60 s, and several connections equal to 1024.

As for the test performed by using SlowTT, the attack is able to create 1016 connections that clog the network and make it unreachable by other clients (maybe legitimate ones). The figure related to the attack executed by using SlowITe is not reported since the network analysis reports a graph similar to Figure 6. In this case as well, as in the previous one, when the SlowITe attack reaches the maximum number of connections established by the MQTT broker, the validation tool reports the value of one while in the first seconds of the attack, the value of zero is reported as not all the connections were occupied. From the bandwidth point of view, the Connect version requires 9.53 kbps, and this value is lower than the SlowTT value as PING packets are not sent. However, comparing the two data, there is not a large discrepancy that could be used to identify attacks. As mentioned in Section 4, both attacks are very complex to detect in terms of bandwidth and data transmitted because, as these tests show, the network resources used are very minimal.

The real difference between the two versions is related to the value of the KeepAlive parameter. If the KeepAlive is set with a low value, the SlowTT attack is more efficient because by exploiting the PING packets, it is able to keep the server busy longer and therefore independent of the KeepAlive value. The Wait Timeout between the PING packets must be less than the KeepAlive. For the same reason, the SlowITe attack has a good effect only if the KeepAlive time is long; otherwise, the connections are closed when the time is expired.

6.3. Targeting an MQTT Service Running on SSL

As demonstrated in [14], SlowITe, a preliminary version of SlowTT, is able to target and exploit communication encrypted with SSL/TLS. In order to cover more IoT scenarios, we tested SlowTT also against an MQTT network encrypted with SSL/TLS. In this test, the testbed is composed of the same devices and configured with the same parameters of the previous tests, but the MQTT broker encrypts the communication by using TLS v1.2.

In order to instantiate connections, SlowTT must be able to communicate over SSL. In our tests, we hypothesized that the attacker is an external device to the network, that is without access to the certificates necessary to communicate legitimately with the broker. The first step is to authenticate the attacker on the MQTT broker by encrypting packets with SSL; otherwise, the MQTT broker is not able to authenticate the device. In order to establish a connection encrypted with SSL, we adopted an approach to retrieve the public key used by the MQTT broker to encrypt the communication. When the public key is retrieved, we imported this certificate into the attacker's keystore. By using this approach, SlowTT is able to encrypt packets with SSL and to communicate on the network.

When the attacker is authenticated on the broker, the workflow of SlowTT is the same as described in the plaintext tests. The results obtained once again demonstrate that SlowTT is able to correctly execute the attack even with encrypted communication with TLS v1.2. The network traffic analysis obtained during tests shows graphs equal to Figure 6, so we do not publish them here. In conclusion, what was previously demonstrated in [14] is confirmed: SlowTT is able to bypass the SSL encryption of the MQTT protocol and establish all the connections necessary for the attack.

6.4. SlowTT Against MQTT Version 5

In 2019, OASIS released Version 5 of MQTT, a refinement of Version 3.1.1 to improve protocol features and applications [80]. The new features of MQTT Version 5 include: better error reporting; a reason code has been added to responses to publications PUBACK/PUBREC; shared subscriptions: if the message rate on a subscription is high, shared subscriptions can be used to balance the load of messages; message properties, used to implement the other features; message and session expiry, an option to discard messages if not forwarded to other clients; other limits such as maximum packet size and the number of (QoS > 0) messages that are transmitted to the client.

Although this new version is not yet much adopted at the implementation level, we decided to test SlowTT against Version 5 of MQTT to highlight the innovation of the attack. In order to perform the attack on this new version of MQTT, studies were carried out initially on the level of network traffic and packets to compare the differences between the two versions. Based on the packets used by the attack, we noticed a difference in the CONNECT packet since, in addition to changing the version of the protocol defined within the packet, there is a new parameter called Properties. This property is used by all MQTT v5 packets and allows users to transfer additional information.

In order to execute the SlowTT attack against MQTT Version 5, we developed a dedicated version of the tool able to adopt the new CONNECT packet as a vector attack. In order to validate the new feature of SlowTT, we tested against an MQTT network based on Version 5 with Mosquitto as the broker since it supports MQTT Version 5 (available at: https://mosquitto.org/).

As the tests performed and fully described in the previous sections, also in this case, the attack is able to reach the DoS status and make the MQTT broker unreachable from legitimate connections.

7. Detection Systems and Algorithms

As can be deduced from the previous sections, the SlowTT attack can cause serious damage if performed against IoT systems adopted in sensitive contexts. For this reason, the implementation of a SlowTT attack detection and mitigation algorithms is a critical challenge in order to identify in real time the execution of the attack and instantiate a protection system to mitigate the attack and to limit the damages.

Currently, a detection approach is still an open issue, but in the research field, different works were performed to detect and mitigate the slow DoS attacks. In particular, a detection system based on the analyses of specific spectral features of network traffic is implemented in [81]. This approach allows identifying anomaly patterns of legitimate client behavior that is typically repetitive. Instead, slow DoS attacks are detected by using a Fourier transform and mutual information in [82]. Information are extracted by analyzing the features of the network traffic containing attacks in [83], and finally, an intrusion detection system to detect slow DoS attacks from real-time network packets is proposed in [84]. Such scientific works can be used as a starting point for developing a SlowTT attack detection system, and machine learning algorithms or artificial intelligence algorithms may be adopted to classify the attack. Considering identification of running SlowTT instances, we believe that the critical point for the detection of the attack is related to the characterization of legitimate traffic. In particular, while other similar solutions focus on traffic such as HTTP, FTP, or SMTP, where it is possible to model users' behavior, in the case of protocols such as MQTT (or SSH), it is not trivial to model the behavior of a client. Indeed, in this case, due to the nature of IoT environments, devices could, for instance, communicate periodically (like a temperature sensor designed to send updates every minute) or randomly (like a motion sensor communicating just when a motion is detected), based on their functionalities and activities.

A possible approach could be linked to the KeepAlive value set by the clients. If the KeepAlive is higher than a threshold defined in the network, it avoids the connection. However, this algorithm does not apply to the IoT context since, given the nature of this technology, the devices have different behaviors and actions to perform. Hence, a device with a high KeepAlive value could be legitimate and would not be authenticated on the network, losing sensitive and valuable information.

As for the detection phase of SlowITe, since defining parameters to identify legitimate connections is a complex task due to the behavior of IoT networks and devices, an initial approach is to analyze and manipulate packets that could be considered valid and legitimate. Due to the nature of the attacks, packets are considered legitimate, by this approach, if they contain data/information communicated by sensors to the central node. The packets that are considered valid and useful in this approach are the Publish and Subscribe packets. In the attacks that we developed and described in this work and in [14], these packets are not used. The proposed detection system could be based on the implementation of an algorithm that checks every second how many legitimate and valid bits are exchanged by a connection. If legitimate bits are zero after a certain time interval (based on the training of the model with traffic considered legitimate), the detection algorithm considers the connection as malicious. Using this approach, it is possible to perform the detection of SlowITe and SlowTT since in these attacks, there are never useful packets, while normal MQTT communication is based on Publish and Subscribe packets (these are considered useful packets in our approach).

Finally, a possible approach used in the context of detection systems is based on machine learning algorithms. While in normal network systems, there are rich and well-known datasets used to train these algorithms, at the moment, there are few datasets related to the IoT context, and they do not cover all the aspects necessary to carry out sufficient training to define whether a connection is malicious or legitimate.

8. Conclusions and Future Work

In this paper, we investigated the security of the Internet of Things, in particular related to the MQTT protocol. We focused on the MQTT application protocol, widely adopted in many IoT contexts, in order to identify its weakness to a slow denial of service attacks. We adopted low-rate DoS techniques and approaches to target the application protocol, by designing and implementing the SlowTT attack, an innovative version of the SlowITe attack aimed to exploit vulnerabilities inside the MQTT protocol. In particular, we exploited the weaknesses to set the Keep-Alive parameter of the server from the client itself and to adopt PING packets to keep a connection alive by avoiding the connection closure by the server. These should be considered as weaknesses of the MQTT protocol, validated in the proposed work, through the design and implementation of the SlowTT attack. Moreover, tests against SLL and MQTT Version 5 were performed to cover more scenarios related to the MQTT. In these tests as well, SlowTT is able to achieve its aims. The limitations are related to the low number of "legitimate" packets (e.g., PUBLISH or SUBSCRIBE) and regarding the security configuration of the MQTT broker since if the authentication based on username/password is implemented, SlowTT is not able to instantiate the connections without knowing the credentials. A first approach to the detection phase was presented in the work; future work concerns the implementation of the approach to detect the developed threat.

We tested the attack against a real network, targeting real MQTT services based on Eclipse Mosquitto with a low-power node represented by a Raspberry Pi 3 Model B. Tests were executed, firstly, to validate the improvement of SlowTT compared with SlowITe by analyzing the behavior of the attacks against the server on a single connection based on plaintext communication, encrypted communications, and MQTT Version 5 as well. We found that, after establishment, a single connection can be kept alive for an infinite time by exploiting the PING packets. By analyzing the results of the comparison between SlowTT and SlowITe, in SlowITe, the maximum time for a connection was 27 h; instead, in SlowTT, a connection could be alive for an infinite time. This is the added value of SlowTT because, by keeping the connections open for a long time, legitimate connections are discarded. Further work may be directed toward a refinement of SlowTT and to test against other possible brokers running on cloud-based solutions. Additional extensions of the topic may refine the attack in order to better emulate a legitimate client sending bogus data to the broker, with the aim to improve the attack in view of potential protection systems able to filter out connections not sending relevant data to the application. Regarding the detection system, future work may be the characterization of legitimate MQTT data to model a real-time detection system. Besides, since we focused on the MQTT protocol,

further investigations on the topic may consider other application layer IoT protocols like CoAP and AMQP [64].

Author Contributions: I.V. contributed to the study of vulnerability and testing and the drafting of the paper. M.A. contributed to the proof reading of the document. E.C. contributed to supervising the activities from a technical and scientific point of view and collaborated in refining the document. All authors read and agreed to the published version of the manuscript.

Funding: This work was supported by the following research project: the Integrated Framework for Predictive and Collaborative Security of Financial Infrastructures (FINSEC) project, which has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 786727.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, Y.; Chen, M.; Wang, X.; Chan, R.H.; Li, W.J. IoT for next-generation racket sports training. *IEEE Internet Things J.* **2018**, *5*, 4558–4566. [CrossRef]
2. Chihana, S.; Phiri, J.; Kunda, D. An IoT based warehouse intrusion detection (E-Perimeter) and grain tracking model for food reserve agency. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*. [CrossRef]
3. Minchev, D.; Dimitrov, A. Home automation system based on ESP8266. In Proceedings of the 2018 20th International Symposium on Electrical Apparatus and Technologies (SIELA), Bourgas, Bulgaria, 3–6 June 2018; pp. 1–4.
4. Manavalan, E.; Jayakrishna, K. A review of Internet of Things (IoT) embedded sustainable supply chain for industry 4.0 requirements. *Comput. Ind. Eng.* **2019**, *127*, 925–953. [CrossRef]
5. Soni, D.; Makwana, A. A survey on mqtt: A protocol of Internet of Things (iot). In Proceedings of the International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017), Chennai, India, 6–8 March 2017.
6. Lee, S.; Kim, H.; Hong, D.K.; Ju, H. Correlation analysis of MQTT loss and delay according to QoS level. In Proceedings of the International Conference on Information Networking 2013 (ICOIN), Bangkok, Thailand, 28–30 January 2013; pp. 714–717.
7. Luzuriaga, J.E.; Cano, J.C.; Calafate, C.; Manzoni, P.; Perez, M.; Boronat, P. Handling mobility in IoT applications using the MQTT protocol. In Proceedings of the 2015 Internet Technologies and Applications (ITA), Wrexham, UK, 8–11 September 2015; pp. 245–250.
8. Moustafa, N.; Turnbull, B.; Choo, K.K.R. Towards automation of vulnerability and exploitation identification in IIoT networks. In Proceedings of the 2018 IEEE International Conference on Industrial Internet (ICII), Seattle, WA, USA, 21–23 October 2018; pp. 139–145.
9. Kim, G.; Kang, S.; Park, J.; Chung, K. An MQTT-Based Context-Aware Autonomous System in oneM2M Architecture. *IEEE Internet Things J.* **2019**, *6*, 8519–8528. [CrossRef]
10. Sadeghi, A.R.; Wachsmann, C.; Waidner, M. Security and privacy challenges in industrial Internet of Things. In Proceedings of the 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 8–12 June 2015; pp. 1–6.
11. Al-Turjman, F.; Zahmatkesh, H.; Shahroze, R. An overview of security and privacy in smart cities' IoT communications. *Trans. Emerg. Telecommun. Technol.* **2019**, e3677, doi:10.1002/ett.3677. [CrossRef]
12. Seleznev, S.; Yakovlev, V. Industrial Application Architecture IoT and protocols AMQP, MQTT, JMS, REST, CoAP, XMPP, DDS. *Int. J. Open Inf. Technol.* **2019**, *7*, 17–28.
13. Cornel-Cristian, A.; Gabriel, T.; Arhip-Calin, M.; Zamfirescu, A. Smart home automation with MQTT. In Proceedings of the 2019 54th International Universities Power Engineering Conference (UPEC), Bucharest, Romania, 3–6 September 2019; pp. 1–5.
14. Vaccari, I.; Aiello, M.; Cambiaso, E. SlowITe, a Novel Denial of Service Attack Affecting MQTT. *Sensors* **2020**, *20*, 2932. [CrossRef]
15. Shorey, T.; Subbaiah, D.; Goyal, A.; Sakxena, A.; Mishra, A.K. Performance comparison and analysis of slowloris, goldeneye and xerxes ddos attack tools. In Proceedings of the 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, India, 19–22 September 2018; pp. 318–322.

16. Park, J.; Iwai, K.; Tanaka, H.; Kurokawa, T. Analysis of slow read DoS attack. In Proceedings of the 2014 International Symposium on Information Theory and its Applications, Melbourne, Australia, 26–29 October 2014; pp. 60–64.
17. Maciá-Fernández, G.; Díaz-Verdejo, J.E.; García-Teodoro, P.; de Toro-Negro, F. LoRDAS: A low-rate DoS attack against application servers. In *International Workshop on Critical Information Infrastructures Security*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 197–209.
18. Miorandi, D.; Sicari, S.; De Pellegrini, F.; Chlamtac, I. Internet of things: Vision, applications and research challenges. *Ad Hoc Netw.* **2012**, *10*, 1497–1516. [CrossRef]
19. Suo, H.; Wan, J.; Zou, C.; Liu, J. Security in the Internet of Things: A review. In Proceedings of the 2012 International Conference on Computer Science and Electronics Engineering, Hangzhou, China, 23–25 March 2012; Volume 3, pp. 648–651.
20. Hossain, M.M.; Fotouhi, M.; Hasan, R. Towards an analysis of security issues, challenges, and open problems in the Internet of Things. In Proceedings of the 2015 IEEE World Congress on Services, New York, NY, USA, 27 June–2 July 2015; pp. 21–28.
21. Matharu, G.S.; Upadhyay, P.; Chaudhary, L. The Internet of Things: Challenges and security issues. In Proceedings of the 2014 International Conference on Emerging Technologies (ICET), Islamabad, Pakistan, 8–9 December 2014; pp. 54–59.
22. Balte, A.; Kashid, A.; Patil, B. Security issues in Internet of things (IoT): A survey. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **2015**, *5*, 450–455.
23. Kumar, J.S.; Patel, D.R. A survey on Internet of things: Security and privacy issues. *Int. J. Comput. Appl.* **2014**, *90*, 20–26.
24. Hedi, I.; Špeh, I.; Šarabok, A. IoT network protocols comparison for the purpose of IoT constrained networks. In Proceedings of the 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 22–26 May 2017; pp. 501–505.
25. Yassein, M.B.; Shatnawi, M.Q. Application layer protocols for the Internet of Things: A survey. In Proceedings of the 2016 International Conference on Engineering & MIS (ICEMIS), Agadir, Morocco, 22–24 September 2016; pp. 1–4.
26. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [CrossRef]
27. Gazis, V.; Görtz, M.; Huber, M.; Leonardi, A.; Mathioudakis, K.; Wiesmaier, A.; Zeiger, F.; Vasilomanolakis, E. A survey of technologies for the Internet of Things. In Proceedings of the 2015 International Wireless Communications and Mobile Computing Conference (IWCMC), Dubrovnik, Croatia, 24–28 August 2015; pp. 1090–1095.
28. Dragomir, D.; Gheorghe, L.; Costea, S.; Radovici, A. A survey on secure communication protocols for IoT systems. In Proceedings of the 2016 International Workshop on Secure Internet of Things (SIoT), Heraklion, Greece, 26–30 September 2016; pp. 47–62.
29. Asim, M. A survey on application layer protocols for Internet of Things (IoT). *Int. J. Adv. Res. Comput. Sci.* **2017**, *8.3*.
30. Kraijak, S.; Tuwanut, P. A survey on IoT architectures, protocols, applications, security, privacy, real-world implementation and future trends. In Proceedings of the 11th International Conference on Wireless and Mobile Communications, Shanghai, China, 21–23 September 2015.
31. Vaccari, I.; Cambiaso, E.; Aiello, M. Remotely Exploiting AT Command Attacks on ZigBee Networks. *Secur. Commun. Netw.* **2017**, *2017*, 1723658. [CrossRef]
32. Vaccari, I.; Cambiaso, E.; Aiello, M. Evaluating Security of Low-Power Internet of Things Networks. *Int. J. Comput. Digit. Syst.* **2019**, *8*, 101–114. [CrossRef]
33. Vidgren, N.; Haataja, K.; Patino-Andres, J.L.; Ramirez-Sanchis, J.J.; Toivanen, P. Security threats in ZigBee-enabled systems: Vulnerability evaluation, practical experiments, countermeasures, and lessons learned. In Proceedings of the 2013 46th Hawaii International Conference on System Sciences, Wailea, Maui, HI, USA, 7–10 January 2013; pp. 5132–5138.
34. Plósz, S.; Farshad, A.; Tauber, M.; Lesjak, C.; Ruprechter, T.; Pereira, N. Security vulnerabilities and risks in industrial usage of wireless communication. In Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), Barcelona, Spain, 16–19 September 2014; pp. 1–8.

35. Khanji, S.; Iqbal, F.; Hung, P. ZigBee security vulnerabilities: Exploration and evaluating. In Proceedings of the 2019 10th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 11–13 June 2019; pp. 52–57.
36. Pongle, P.; Chavan, G. A survey: Attacks on RPL and 6LoWPAN in IoT. In Proceedings of the 2015 International Conference on Pervasive Computing (ICPC), Pune, India, 8–10 January 2015; pp. 1–6.
37. Hummen, R.; Hiller, J.; Wirtz, H.; Henze, M.; Shafagh, H.; Wehrle, K. 6LoWPAN fragmentation attacks and mitigation mechanisms. In Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks, Budapest, Hungary, 17–19 April 2013; pp. 55–66.
38. Sahay, R.; Geethakumari, G.; Modugu, K. Attack graph—Based vulnerability assessment of rank property in RPL-6LOWPAN in IoT. In Proceedings of the 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), Singapore, 5–8 February 2018; pp. 308–313.
39. Kasinathan, P.; Pastrone, C.; Spirito, M.A.; Vinkovits, M. Denial-of-service detection in 6LoWPAN based Internet of Things. In Proceedings of the 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Lyon, France, 7–9 October 2013; pp. 600–607.
40. Yassein, M.B.; Mardini, W.; Almasri, T. Evaluation of security regarding Z-Wave wireless protocol. In Proceedings of the Fourth International Conference on Engineering & MIS 2018, Istanbul, Turkey, 19–21 June 2018; p. 32.
41. Fuller, J.D.; Ramsey, B.W. Rogue Z-Wave controllers: A persistent attack channel. In Proceedings of the 2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops), Clearwater Beach, FL, USA, 26–29 October 2015; pp. 734–741.
42. Fouladi, B.; Ghanoun, S. Security evaluation of the Z-Wave wireless protocol. *Black Hat USA* **2013**, *24*, 1–2.
43. Randhawa, R.H.; Hameed, A.; Mian, A.N. Energy efficient cross-layer approach for object security of CoAP for IoT devices. *Ad Hoc Netw.* **2019**, *92*, 101761. [CrossRef]
44. Raza, S.; Shafagh, H.; Hewage, K.; Hummen, R.; Voigt, T. Lithe: Lightweight secure CoAP for the Internet of things. *IEEE Sens. J.* **2013**, *13*, 3711–3720. [CrossRef]
45. Kothmayr, T.; Schmitt, C.; Hu, W.; Brünig, M.; Carle, G. DTLS based security and two-way authentication for the Internet of Things. *Ad Hoc Netw.* **2013**, *11*, 2710–2723. [CrossRef]
46. Rahman, R.A.; Shah, B. Security analysis of IoT protocols: A focus in CoAP. In Proceedings of the 2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC), Muscat, Oman, 15–16 March 2016; pp. 1–7.
47. Capossele, A.; Cervo, V.; De Cicco, G.; Petrioli, C. Security as a CoAP resource: An optimized DTLS implementation for the IoT. In Proceedings of the 2015 IEEE International Conference on Communications (ICC), London, UK, 8–12 June 2015; pp. 549–554.
48. Andrea, I.; Chrysostomou, C.; Hadjichristofi, G. Internet of Things: Security vulnerabilities and challenges. In Proceedings of the 2015 IEEE Symposium on Computers and Communication (ISCC), Larnaca, Cyprus, 6–9 July 2015; pp. 180–187.
49. Meneghello, F.; Calore, M.; Zucchetto, D.; Polese, M.; Zanella, A. IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices. *IEEE Internet Things J.* **2019**, *6*, 8182–8201. [CrossRef]
50. Luzuriaga, J.E.; Perez, M.; Boronat, P.; Cano, J.C.; Calafate, C.; Manzoni, P. A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks. In Proceedings of the 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, 9–12 January 2015; pp. 931–936.
51. Fernandes, J.L.; Lopes, I.C.; Rodrigues, J.J.; Ullah, S. Performance evaluation of RESTful web services and AMQP protocol. In Proceedings of the 2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN), Da Nang, Vietnam, 2–5 July 2013; pp. 810–815.
52. Nebbione, G.; Calzarossa, M.C. Security of IoT application layer protocols: Challenges and findings. *Future Internet* **2020**, *12*, 55. [CrossRef]
53. Santhadevi, D.; Janet, B. Security challenges in computing system, communication technology and protocols in IoT system. In Proceedings of the 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), Kottayam, India, 21–22 December 2018; pp. 1–7.
54. Issarny, V.; Mallet, V.; Nguyen, K.; Raverdy, P.G.; Rebhi, F.; Ventura, R. Dos and don'ts in mobile phone sensing middleware: Learning from a large-scale experiment. In Proceedings of the 17th International Middleware Conference, Trento, Italy, 12–16 December 2016; p. 17.

55. Hunkeler, U.; Truong, H.L.; Stanford-Clark, A. MQTT-S—A publish/subscribe protocol for wireless sensor networks. In Proceedings of the 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08), Bangalore, India, 6–10 January 2008; pp. 791–798.
56. Pflanzner, T.; Kertész, A. A survey of IoT cloud providers. In Proceedings of the 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 30 May–3 June 2016; pp. 730–735.
57. Jukić, O.; Špeh, I.; Hedi, I. Cloud-based services for the Internet of Things. In Proceedings of the 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 21–25 May 2018; pp. 0372–0377.
58. Breivold, H.P.; Sandström, K. Internet of things for industrial automation–challenges and technical solutions. In Proceedings of the 2015 IEEE International Conference on Data Science and Data Intensive Systems, Sydney, Australia, 11–13 December 2015; pp. 532–539.
59. Andy, S.; Rahardjo, B.; Hanindhito, B. Attack scenarios and security analysis of MQTT communication protocol in IoT system. In Proceedings of the 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), Yogyakarta, Indonesia, 19–21 September 2017; pp. 1–6.
60. Harsha, M.; Bhavani, B.; Kundhavai, K. Analysis of vulnerabilities in MQTT security using Shodan API and implementation of its countermeasures via authentication and ACLs. In Proceedings of the 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, India, 19–22 September 2018; pp. 2244–2250.
61. Chifor, B.C.; Bica, I.; Patriciu, V.V. Mitigating DoS attacks in publish-subscribe IoT networks. In Proceedings of the 2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Targoviste, Romania, 29 June–1 July 2017; pp. 1–6.
62. Firdous, S.N.; Baig, Z.; Valli, C.; Ibrahim, A. Modelling and evaluation of malicious attacks against the iot mqtt protocol. In Proceedings of the 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), Exeter, UK, 21–23 June 2017; pp. 748–755.
63. Yassein, M.B.; Shatnawi, M.Q.; Aljwarneh, S.; Al-Hatmi, R. Internet of Things: Survey and open issues of MQTT protocol. In Proceedings of the 2017 International Conference on Engineering & MIS (ICEMIS), Monastir, Tunisia, 8–10 May 2017; pp. 1–6.
64. Naik, N. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In Proceedings of the 2017 IEEE International Systems Engineering Symposium (ISSE), Vienna, Austria, 11–13 October 2017; pp. 1–7.
65. Ammar, M.; Russello, G.; Crispo, B. Internet of Things: A survey on the security of IoT frameworks. *J. Inf. Secur. Appl.* **2018**, *38*, 8–27. [CrossRef]
66. Karagiannis, V.; Chatzimisios, P.; Vazquez-Gallego, F.; Alonso-Zarate, J. A survey on application layer protocols for the Internet of things. *Trans. IoT Cloud Comput.* **2015**, *3*, 11–17.
67. Fysarakis, K.; Askoxylakis, I.; Soultatos, O.; Papaefstathiou, I.; Manifavas, C.; Katos, V. Which iot protocol? Comparing standardized approaches over a common M2M application. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016.
68. Gündogan, C.; Kietzmann, P.; Schmidt, T.C.; Lenders, M.; Petersen, H.; Wählisch, M.; Frey, M.; Shzu-Juraschek, F. Information-centric networking for the industrial IoT. In Proceedings of the 4th ACM Conference on Information-Centric Networking, Berlin, Germany, 26–28 September 2017; pp. 214–215.
69. Kodali, R.K.; Soratkal, S. MQTT based home automation system using ESP8266. In Proceedings of the 2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), Agra, India, 21–23 December 2016; pp. 1–5.
70. Prada, M.A.; Reguera, P.; Alonso, S.; Morán, A.; Fuertes, J.J.; Domínguez, M. Communication with resource-constrained devices through MQTT for control education. *IFAC-PapersOnLine* **2016**, *49*, 150–155. [CrossRef]
71. Wukkadada, B.; Wankhede, K.; Nambiar, R.; Nair, A. Comparison with HTTP and MQTT in Internet of Things (IoT). In Proceedings of the 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 11–12 July 2018; pp. 249–253.

72. Bauer, J.; Aschenbruck, N. Measuring and adapting MQTT in cellular networks for collaborative smart farming. In Proceedings of the 2017 IEEE 42nd Conference on Local Computer Networks (LCN), Singapore, 9–12 October 2017; pp. 294–302.
73. Rausch, T.; Nastic, S.; Dustdar, S. Emma: Distributed qos-aware mqtt middleware for edge computing applications. In Proceedings of the 2018 IEEE International Conference on Cloud Engineering (IC2E), Orlando, FL, USA, 17–20 April 2018; pp. 191–197.
74. Atmoko, R.; Riantini, R.; Hasin, M. IoT real time data acquisition using MQTT protocol. *J. Phys. Conf. Ser.* **2017**, *853*, 012003. [CrossRef]
75. Cambiaso, E.; Papaleo, G.; Chiola, G.; Aiello, M. Slow DoS attacks: Definition and categorisation. *Int. J. Trust Manag. Comput. Commun.* **2013**, *1*, 300–319. [CrossRef]
76. Cambiaso, E.; Papaleo, G.; Aiello, M. Slowdroid: Turning a smartphone into a mobile attack vector. In Proceedings of the 2014 International Conference on Future Internet of Things and Cloud, Barcelona, Spain, 27–29 August 2014; pp. 405–410.
77. Farina, P.; Cambiaso, E.; Papaleo, G.; Aiello, M. Understanding ddos attacks from mobile devices. In Proceedings of the 2015 3rd International Conference on Future Internet of Things and Cloud, Rome, Italy, 24–26 August 2015; pp. 614–619.
78. Light, R.A. Mosquitto: Server and client implementation of the MQTT protocol. *J. Open Source Softw.* **2017**, *2*, 265. [CrossRef]
79. Bambauer, D.E. Schrodinger's Cybersecurity. *UCDL Rev.* **2014**, *48*, 791.
80. Glaroudis, D.; Iossifides, A.; Chatzimisios, P. Survey, comparison and research challenges of IoT application protocols for smart farming. *Comput. Netw.* **2020**, *168*, 107037. [CrossRef]
81. Aiello, M.; Cambiaso, E.; Mongelli, M.; Papaleo, G. An on-line intrusion detection approach to identify low-rate DoS attacks. In Proceedings of the 2014 International Carnahan Conference on Security Technology (ICCST), Rome, Italy, 13–16 October 2014; pp. 1–6.
82. Mongelli, M.; Aiello, M.; Cambiaso, E.; Papaleo, G. Detection of DoS attacks through Fourier transform and mutual information. In Proceedings of the 2015 IEEE International Conference on Communications (ICC), London, UK, 8–12 June 2015; pp. 7204–7209.
83. Aiello, M.; Cambiaso, E.; Scaglione, S.; Papaleo, G. A similarity based approach for application DoS attacks detection. In Proceedings of the 2013 IEEE Symposium on Computers and Communications (ISCC), Split, Croatia, 7–10 July 2013; pp. 000430–000435.
84. Cambiaso, E.; Papaleo, G.; Chiola, G.; Aiello, M. A Network Traffic Representation Model for Detecting Application Layer Attacks. *Int. J. Comput. Digit. Syst.* **2016**, *5*, doi:10.12785/IJCDS/050104. [CrossRef]