POLITECNICO DI TORINO
Repository ISTITUZIONALE

Exploiting Internet of Things Protocols for Malicious Data Exfiltration Activities

Original
Exploiting Internet of Things Protocols for Malicious Data Exfiltration Activities / Vaccari, I., Narteni, S., Aiello, M., Mongelli, M., Cambiaso, E.. - In: IEEE ACCESS. - ISSN 2169-3536. - ELETTRONICO. - 9:(2021), pp. 104261-104280. [10.1109/ACCESS.2021.3099642]

Availability:
This version is available at: 11583/2966701 since: 2022-06-14T10:40:55Z

Publisher:
Institute of Electrical and Electronics Engineers

Published
DOI:10.1109/ACCESS.2021.3099642

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

27 November 2026

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 30, 2021, accepted July 18, 2021, date of publication July 26, 2021, date of current version July 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3099642

Exploiting Internet of Things Protocols for Malicious Data Exfiltration Activities

IVAN VACCARI[1,2], SARA NARTENI[1], MAURIZIO AIELLO[1], MAURIZIO MONGELLI[1], (Member, IEEE), AND ENRICO CAMBIASO[1]

[1]Consiglio Nazionale delle Ricerche (CNR), IEIIT Institute, 16149 Genoa, Italy
[2]Department of Informatics, Bioengineering, Robotics and System Engineering (DIBRIS), University of Genoa, 16145 Genoa, Italy

Corresponding author: Ivan Vaccari (ivan.vaccari@ieiit.cnr.it)

ABSTRACT Internet of Things is a widely adopted and pervasive technology, but also one of the most relevant in cyber-security, given the volume and sensitivity of shared data and the availability of affordable but insecure products. In this paper, we propose a novel cyber-threat exploiting the Message Queue Telemetry Transport (MQTT) protocol to implement a tunneling attack. In IoT networks, sensitive and critical information are exchanged between devices or external systems to perform data analysis. For this reason, a tunneling threat could be adopted by a malicious user to steal information. In this context, a tunneling system based on MQTT can be considered since this communication protocol could be allowed to pass through enterprise firewalls because it is widely adopted in this IoT world. An attacker can exploit the MQTT protocol for various purposes such as steal information or access to not-allowed websites/servers. In particular in this work, we contribute in two main points: initially we demonstrate how the proposed threat is able to encapsulate messages through the MQTT protocol, by also comparing it with other tunneling systems exploiting different communication protocols. Obtained results show that exploiting MQTT for tunneling purposes is a good choice, compared to other communication protocols, especially for payloads up to 3000 bytes. Then, we propose and validate an initial machine learning based approach able to detect the proposed MQTT tunnel, by comparing different detection algorithms tested with and without a hyperparameter optimization, in terms of accuracy, F1 score and Receiver Operating Characteristic (ROC) curve. In this case, obtained results show that some algorithms are able to identify the attack, with an accuracy exceeding 95%, while others lack of such capability.

INDEX TERMS Cyber-security, network security, IoT, MQTT, tunneling systems, machine learning, detection algorithms, data exfiltration.

I. INTRODUCTION

Internet of Things (IoT) is a terminology adopted in 1999 by Kevin Ashton [1], referring to networks of physical objects with the capability to exchange data between them through Internet. Such objects, known as smart devices, may be either in fixed or mobile locations and they are integrated with sensors, actuators and communication modules.

In recent years, we have assisted to huge advancements in the design of both network architectures and smart devices. As a consequence, IoT is today a popular and trending technology: it can be found in many different applications as a way to provide assistance to humans in their everyday life. Such applications range from smart cities, smart factories, smart health, objects like household appliances controlled via smartphones, up to large-scale infrastructures, such as power grids and industrial systems [2], [3]. Banking and financial services are ranked at the top of IoT market share, followed by Information and Communication Technology (ICT), healthcare and public administrations. As reported by Ericsson [4], the number of interconnected IoT devices is estimated to grow up to 18 billions by 2022.

Because of such a rapid development, which led to the arising of several new attacks against networks and objects, IoT security has become a matter of great concern. In fact, IoT-related services have been recognized as critical infrastructures by national and international organizations [5]: attacks targeting or making use of IoT could then be very serious and lead to severe damages in the social contexts in which IoT operates. As an example, cyber-attacks to

The associate editor coordinating the review of this manuscript and approving it for publication was Yassine Maleh.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

medical imaging instrumentation, like computed tomography machines, can compromise their correct functioning, with dangerous consequences for patients [6].

Based on the previous security considerations, in this work we investigate the feasibility of an innovative cyber-threat that exploits an ad-hoc communication protocol used for implementing IoT networks, called Message Queue Telemetry Transport (MQTT). Such communication protocol is a publish/subscribe system and it is adopted nowadays in different IoT contexts such as home automation [7], healthcare [8], logistics [9] or industry [10].

The aim of this work is to design, develop and validate a tunneling system architecture based on MQTT such system could be adopted against critical infrastructures, e.g., to steal sensitive information. In addition, in order to introduce possible protection systems from the proposed threat, a potential detection system through machine learning is proposed. Although other tunneling systems exist and can be found in literature [11]–[14], the choice of the protocol to exploit (MQTT, in our case) depends on the possibility to bypass network restrictions or being undetected on the network. In computer science, the idea of tunneling consists in disguising a protocol into another one to secretly exchange information between network nodes. Thus, tunneling threats are able to encapsulate malicious data on apparently legitimate network packets. Being based on hidden data conveyance, a tunneling system can be categorized as an implementation of covert channel that exploits network communication. A covert channel is a broader class of cyber-threats, based on a secret communication between different entities that may lead to data exfiltration, but not necessarily based on network protocols. Moreover, a tunneling system could be adopted to perform a penetration testing activity to validate whether, in the network under analysis, it is possible to communicate externally using a different communication protocol from those used at the network side.

To the best of our knowledge, in the field of IoT security, approaches aimed to create tunneling systems for malicious purposes are very limited.

The remaining of the paper is structured as follows: Section II reports the related work about security on IoT tunneling systems. Section III introduces the proposed attack, while Section IV describes the testbed. Executed tests and obtained results are reported in Section V. Section VI discusses a possible machine learning based detection approach to protect from the proposed threat. Finally, Section VII concludes the paper and reports further works on the topic.

II. RELATED WORK

The massive development of IoT in recent times has exposed it to several different vulnerabilities and threats. In this context, several researches [4], [15]–[19] have been carried out to collect and provide taxonomies about cyber-security threats in IoT, mainly based on the architecture levels characterizing such environments. A recent detailed survey [4] has categorized IoT attacks into physical, network, software and data attacks, with a focus on Industrial IoT, also giving related countermeasures. In particular, traffic analysis, Sinkhole and Man-In-The-Middle (MiTM) attacks are mentioned as means for data leakage. In [20], the problem of IoT security is studied in terms of IoT-enabled attacks, referring to those threats which exploit IoT vulnerabilities to compromise critical systems (healthcare, industry, etc) that are strictly connected to IoT. In this work, examples of covert channels, built through smart IoT devices (smart TVs, smart LEDs), are reported as an instrument for data leakage. In [18] are described network security issues that could occur in IoT environments such as smart home, healthcare and transportation, classified through different criteria (e.g. device property, information damage level, location, strategy, host-based, access level, communication stack protocols, protocol-based). Instead, [19] describes IoT attacks in general, regardless the underlying protocol. This approach exposes similar attacks to different IoT protocols, by adopting an approach focusing on packets, protocols or the whole system. For this categorization, several protocols are considered (e.g. ZigBee, LoRaWAN, SigFox, etc.), with no mention of MQTT protocol. References [21] addresses the major privacy and security concerns in smart city applications instead, including DoS, malware, eavesdropping or traffic analysis, but without any reference on MQTT. References [16] discusses instead possible attacks coming from IoT vulnerabilities at hardware, network and smart applications levels. As regards the network level, protocol tunneling is mentioned as a threat coming from wireless communications, without specifying which protocols may be involved. In this context, it is relevant to mention the Mirai attack, occurred in 2017, exploiting billions of IoT devices to carry out a distributed attack against DynDNS [22]. As demonstrated by the presented works, the IoT security is a trending topic needing investigation. By following such research field, we focus on the exploitation of IoT data exchanging protocols for malicious activities, by focusing in particular on the exploitation of the MQTT protocol.

Particularly, by focusing on the MQTT protocol, in [23] a threat model is built in order to identify and mitigate possible attacks addressing MQTT brokers and IoT devices. Different attack scenarios are considered as parts of five main categories: DoS, identity spoofing, information disclosure, elevation of privileges, data tampering. References [24] considers other MQTT attack classes: attacks against data privacy, authentication and data integrity, port obscurity and botnet over MQTT. References [25], [26] propose instead novel security attacks against MQTT, by implementing low-rate denial of service attacks, an emerging category of threats making use of low-bandwidth rate to make a network service unavailable to its intended users. Authors in [17] propose instead a new reference model for the IoT stack, in which each layer is divided into different components (building blocks). Following the proposed architecture, different attacks are surveyed, including denial of service (DoS), MiTM, buffer overflow and authentication attack for MQTT. Also, in [27], three security approach to the MQTT are presented. The

first one is related to a solution to avoid tampered data in the MQTT communication, the second one is focused on privacy protection of data send and received, the third one is focused on a mutual authentication of the entities involved in the communication to protect the data. Instead in [28], geolocation information are exchanged between devices in order to improve security by monitoring the source of the data in case, e.g., of a cyber-attack. Although such threats focus on IoT environments, they do not consider IoT tunneling activities perpetrated for malicious purposes.

In particular, regarding IoT tunneling activities, several research works can be found in literature, proposing implementations of systems theoretically near the concept of a tunnel. In this context, [29] introduces an IP tunneling using the TCP with Network Coding (TCP/NC tunnel) as a way to optimize goodput in heterogeneous IoT lossy networks, hence, without any kind of malicious intention. Instead, [30] proposes a cloud-based framework for the setup of virtual networks among IoT nodes: the system includes a reverse tunnel over WebSocket protocol, as a way to provide server-initiated connectivity to any board-hosted service, or any other node on a contributed resource network, e.g., a Wireless Sensor Network. References [31] mentions Traffic Analysis as one of the main attacks against smart homes (aimed at device identification) and tunneling techniques (VPN tunnel) as the most common countermeasure, which allows to hide the metadata and mask the flow information. Moreover, the paper introduces a novel Signature based Tunneled Traffic Analysis (STTA) attack, that can be effective even when tunneling is implemented. Again, in this case, the proposed tunneling method should not be considered a novel threat. Another interesting approach could be the adoption of the Host Identity Protocol [32] to establish tunneling system. By exploiting this protocol, a tunnel to control data and channel could be implemented [33] by considering also IPv4 and IPv6 addresses [34]. By using this approach, users can control and manage the control channel adopted to exchange information [35]. In [36], authors explore the feasibility of tunneling HTTP or HTTP/2 over SSH as a valid alternative to TLS, in order to establish a transparent, secure, bidirectional communication between IoT devices and platforms, with increased performance. By considering HTTP and SSH tunnels, implementations of such tunneling methods are compared with the proposed MQTT based tunnel, in order to evaluate their performance. Although such works focus on the proposal of novel IoT tunneling activities, the design of a malicious system is not considered, while the final aim of the proposed solutions is benign.

Focusing on malicious tunneling activities, in [37] the feasibility of IoT covert timing channels (CTCs) over mobile 4G/5G networks is investigated. Besides inter-packet delay based CTCs, other techniques are illustrated (e.g. packet-reordering, rate-switching, packet-loss, retransmission, and scheduling-based CTCs). Another interesting covert channel in IoT is presented in [38], where Sequential Probability Ratio Test (SPRT) is dynamically applied to sensor data to build a covert channel between two distinct trojan applications running in two different IoT devices. Covert channels can also be achieved in IoT applications, such as for Android wearables [39], where hidden information is conveyed through changes to notifications content. The MQTT protocol has been exploited for covert channels too: in [40], an exhaustive categorization of covert channels affecting MQTT is provided. Authors focus on the usage of such protocol to exploit the unaware MQTT broker by sending hidden messages through steganographic approaches [41]. Unlike [40], the proposed work focuses on the exploitation of MQTT for tunneling purposes instead, by embedding the application payload on the MQTT publish message itself.

If we consider protection from malicious tunneling activities, as previously underlined, such threats are often dangerous for the security of the network. Therefore, adopting countermeasures to prevent the attacks is necessary. According to [42], the identification of a covert channel is required as a preliminary stage, then it's possible to apply countermeasures, which can be divided in four categories: eliminating the covert channel, limiting its bandwidth, auditing or documenting the covert channel (so that everybody is aware of its presence). As highlighted in [43], most of the existing detection approaches depend on the recognition of an abnormal behavior. Typically, the warden knows the normal traffic behavior in a certain network, so it can easily detect abnormalities caused by covert communication. However, if the normal traffic includes considerable variations, then these methods will fail to detect covert traffic. Moreover, any covert traffic that looks similar to normal traffic will be hard to detect. It is clear that covert channel detection is crucial. To this purpose, in the last years machine learning (ML) has come to help, providing different methods to analyze and discover covert channels. One of the most used classifiers is Support Vector Machine (SVM), as highlighted - between others - in [44]–[47]. Machine learning approaches are investigated in the research community to detect cyber-attacks. In [48], an approach based on Isolation forest (iForest) and Local Outlier Factor (LOF) are implemented to detect syn flood, SSH brute force, HTTP brute force and port scanning attacks. While in [49], a detection of DoS, MiTM, reconnaissance and replay attack is investigated by comparing different machine learning algorithms (such as Naive Bayes, Bayesian, J48, ZeroR,OneR,Simple logistic,SVM,Multi-layer perceptron and Random forest). Another interesting work is [50] where authors adopted an hierarchical semi-supervised k-means algorithm to detect cyber-attacks by using the KDDCUP99 dataset. By considering artificial neural networks approach instead, [51], [52] discussed the detection backdoor, dos, reconnaissance and worm attacks by using neural networks. Neural networks, logistic regression and decision trees also have been tested in the same studies. Other detection approaches involve deep learning [11], such as Recurrent Neural Networks ([53] or Reinforcement Learning [54]). References [55] puts into evidence that all the above mentioned algorithms were trained

and tested on traffic data obtained from just a single covert channel tool. As an advance to this limitation, three ML models (SVM, k-Nearest Neighbors, Deep Neural Networks) on covert traffic generated by nine different tools are discussed to demonstrate that k-NN performed better than the others. Moreover, statistical methods could be adopted to detect specific tunneling systems [56]. Another possible approach to detect a tunneling system is to adopt machine learning and artificial intelligence algorithms. In IoT applications, unsupervised machine learning approach [57], supervised [58] and ad-hoc algorithms [59] are adopted. Also, an innovative approach based on hinge classification algorithm focused on mini-batch gradient descent with an adaptive learning rate and momentum could be adopted in the IoT context [60]. Other interesting approaches are based on exploiting formal verification applied to hybrid machine learning algorithm to detect malicious behavior in IoT [61] or using Long Short-Term Memory (LSTM) algorithm [62]. Finally, also classic algorithm such as Naïve Bayes, Random Forest, SVM or KNN are widely adopted and validated in the research community [63], [64]. Also, by combining different technologies, such as blockchain, it is possible to detect and mitigate information leakage [65]. Moreover, by integrating blockchain and cloud solutions, a decentralized approach to protect data integrity and accurately arbitrate service disputes is discussed in order to allow users to verify data access and manipulation [66]. Moreover, [67] proposes a new authentication and access control mechanism by segregating in virtual domains to segregate and to limit devices' communication domain and confines the possible infected domains. Also, authors restricting nodes' capabilities of sending and receiving messages, which limits the capacities of possible infected nodes. By considering tunneling detection activities, in the proposed work, we analyze the efficacy and efficiency of different machine learning algorithms to identify the proposed MQTT tunneling attack.

By considering the previous works on the topic, to the best of our knowledge, the proposal of a tunneling system exploiting MQTT is not found in literature. In virtue of this, the proposed attack should be considered an advancement to the state of the art. In addition, in our work, we also discuss possible protection from the proposed threat.

III. TUNNEL SYSTEM ARCHITECTURE

In the context of computer networks, a tunneling protocol is a communication protocol architecture adopted to communicate through an hidden way. It is one of the major threats affecting networks security, as it can lead to the exfiltration of sensitive data [13], [68]. Tunneling is adopted to achieve different purposes, usually either to exfiltrate data to the outside of the organization (as implementation of a covert channel) or to bypass network restrictions deployed by the organization. In both scenarios, the tunneling systems are potentially unnoticed by network administrators, as they exploit allowed protocols to send apparently legitimate packets on the network. Different communication protocols are adopted to implement a tunneling infrastructure, such as HTTP [12], ICMP [13], SSH [14], UDP [69], VPN [70] or DNS [71].

The proposed approach is common for several tunneling systems, where the attacker controls both an host inside of the exploited network (the client) and one or more nodes outside the network (the server). The goal is indeed to bypass network restrictions or communicate in a malicious way outside of the organization network. This behavior is generally related to the insider threat problem [72].

Due to the nature of IoT networks, widely adopted to elaborate and communicate sensitive information, this attack network architecture could be adopted by a malicious user to extrapolate and steal sensitive information from an IoT network [73]. For example, in critical IoT environments, the network firewall could permit communications only on specific ports or for a predefined set of IoT protocols (e.g. port 1883 for MQTT plain text communication or 5683 for CoAP). In such scenario, a malicious user could exploit IoT protocols to set up a tunneling system aimed to communicate, and, potentially, exfiltrate sensitive data outside of the organization. Concerning tunneling activities, the choice of which protocol to exploit often depends on the configuration of the network or system under attack. Indeed, known tunneling systems or specific network protocols may be monitored by network firewall or intrusion detection system (IDS) installed on the targeted organization.

Based on such concepts, our aim is to implement a novel tunneling system based on IoT protocols. In particular, we make use of the Message Queue Telemetry Transport (MQTT) protocol, as it is considered an IoT standard protocol by the OASIS group [74]. MQTT is a lightweight messaging protocol, able to exchange packets on the network easily, through a publish/subscribe mechanism. Moreover, MQTT security is based on access control list or TLS 1.2 and TLS 1.3 algorithms.

In order to evaluate and validate a tunnel approach for IoT networks, we designed and implemented a tunneling architecture based on MQTT and reported in Figure 1, where the tunnel server implements a specific functionality of the application use case, such as communication with a web server or data storage. Similarly, other protocols may be used (e.g. SSH, SMTP, etc.).

Our tunneling framework is composed by three components: a tunnel client, a tunnel server and the MQTT broker. Particularly, the tunnel client embeds a Socks server, adopted by the client to initialize the tunneling communication with the MQTT broker and the tunnel server. We assume that the tunnel client is installed inside of the private targeted network and used by the malicious node to initiate the tunneling system. Instead, in our scenario, the MQTT broker and the tunnel server, both under the control of the attacker, as the tunnel client, are instead located outside of the targeted network and are adopted to communicate with the tunnel client.

The proposed framework of the tunneling system based on MQTT can achieve different purposes, due to its modular infrastructure: the tunnel client component, installed inside of

the target network or system, can intercept communications by the malicious client (for instance, aimed to sensitive data theft, or to bypass the restrictions of the network), by sending them outside of the network. In our scenario, the MQTT broker connects the tunnel client to the tunnel server, and vice-versa. The tunnel server, on the other hand, receives the data sent by the client and performs the activities requested by it, such as connecting to a web server or storing received data.

The proposed framework can be customized based on the needs of the malicious user: it is not bounded to a single application (for example, web browsing or data theft), but can achieve different objectives of the attacker.

The MQTT topics adopted in the framework are uniquely identified by a special key in order to map each TCP socket and its relative direction to a topic title string. The special key, adopted as topic name, is composed by the tuple *port_src:ip_dst:port_dst* that identify uniquely the transport-layer connection between the client and the server. Instead, the reverse communication, from the server to the client is identified by the topic name *feedback-port_src:ip_dst:port_dst*.

The application layer payload, sent on the channel identifying the relative socket is encapsulated in MQTT packets and sent through MQTT *publish* messages. By using this approach, that should be considered one of the possible approaches used to set up a MQTT tunnel, the proposed system is able to vehiculate/encapsulate TCP-based communications or other data in MQTT messages, independently on the adopted application protocol (e.g. HTTP, HTTPS, SSH, SMTP, etc.).

Such application may be used, for instance, to exfiltrate sensitive data outside of the targeted organization (e.g., by uploading files on a remote host), or to bypass network restrictions of the organization (e.g., by accessing blocked websites accessible from the tunnel service), by encapsulating malicious payloads on apparently legitimate MQTT packets.

The workflow of our tunneling system can be summarized in the following steps, by considering the tunnel client as C and the tunnel server as S:

- *Tunnel Client:* In the $C \rightarrow S$ pathway, the tunnel client processes the information received by the client (through the Socks proxy) and builds an MQTT packet containing the data to be communicated through the tunneling system to the tunnel server. According to the definition above, the communication is identified by a specific topic key, adopted to characterize each connection between the server and the client. Once the information is sent, the client goes into listening mode, to receive feedback from the server, referring to the $S \rightarrow C$ pathway.
- *Tunnel Server:* In the $C \rightarrow S$ pathway, it receives the malicious MQTT packets and elaborates such information, by converting received messages to TCP-based packets, hence forwarding them to the Internet server. During the $S \rightarrow C$ communication, it forwards the relative answer to the client .

As described above, the tunneling system can achieve different purposes. In order to validate the new tunneling architecture, we implemented an example scenario. In our sample scenario, the malicious client establishes a TCP-based connection with an Internet server, through the proposed tunneling system based on MQTT. We decided to test the tunneling system to establish a communication with an external server in order to describe and detail a possible application. In particular, the tunnel is established to communicate outside of the network for web browsing activities. Similarly, as previously mentioned, it may be used to communicate with an external service (e.g., a web service) to transfer sensitive data stolen from the internal network. In our case, HTTP requests are encapsulated inside of MQTT packets, sent from our client to the malicious MQTT broker. For simplicity, payload compression is not implemented by the tunneling system, although it may be applied by the client, for instance by the web browser.

Based on the general workflow, suppose $C \rightarrow S$ and $S \rightarrow C$ being request and response pathways, respectively, involving the tunnel client C and the tunnel server S. We provide in the following a more detailed overview of the single modules:

- *Tunnel Client:* In the $C \rightarrow S$ pathway, the tunnel client receives the HTTP/HTTPS raw request from the web browser and builds a MQTT packet string containing the request, which is then sent to the tunnel server. In details, the client sends a message on the main/general topic (subscribed by the tunnel server) specifying the name of the (new) topic related to the new connection; topics are uniquely identified, for each $C \rightarrow S$ connection, as they map to the underlying TCP socket. Hence, the tunnel client sends the raw conent of the web request on the just created topic. Similarly, as previously described, a secondary topic refers to $S \rightarrow C$ connections and communications coming from the tunnel server.
- *Tunnel Server:* This module is able to reach the web server. In the $C \rightarrow S$ pathway, it receives the topic name generated by the tunnel client on the main/general topic, subscribes to such new topic and waits for an incoming web request, represented as a sequence of MQTT packets. Once received, such request are forwarded to the destination web address. Simultaneously, the tunnel server creates a new topic, related to $S \rightarrow C$ communications for the current connection/socket. Instead, during the $S \rightarrow C$ phase, the tunnel server waits for the HTTP/HTTPS answer from the web server, hence, creates response MQTT packets containing the web response and send them to the new topic created by the tunnel service.

By adopting the multi-topic approach, where each topic is related to a specific web request, the MQTT tunneling system

is able to manage multiple connections in real time without the risk to blend MQTT packets requests for connections managed by the tunneling system.

In the proposed tunneling scheme, we assume that the communication between the client and the web server is not allowed or monitored by the targeted organization. Because of this, to adopt a tunneling approach may hinder the activity of the attacker, to the eyes of the targeted/exploited network.

In order to understand the components involved in the proposed tunneling system based on MQTT, a practical example is reported and described. The sequence diagram of the example is reported in Figure 2.

Assuming that the Tunnel server subscribes to the `general` topic at the beginning of its activity, by following the sequence diagram, the following steps are performed:

1) The Client (a PC) indirectly performs a web request (`request_payload`) to the Web server (identified by `web_ip` and `web_port`), through the tunnel, hence, by communicating with the Tunnel client
2) The Tunnel client univocally maps the `web_ip` and `web_port` couple to string used to identify the request (in case of multiple clients, it may be needed to include the Client IP address to the mapping); such mapping results in the `request_topic` on the MQTT broker to assign for the current $C \rightarrow S$ connection
3) The Tunnel client subscribes to the `general` topic on the MQTT broker
4) The Tunnel client publishes[1] a message on the `general` topic by specifying that a new `request_topic` was created
5) The Tunnel server receives (through the `general` topic previously subscribed) the message specifying that the new `request_topic` was initiated
6) The Tunnel server subscribes to the `request_topic` topic to receive incoming $C \rightarrow S$ data
7) The Tunnel client publishes on the `request_topic` the web request payload `request_payload`
8) The Tunnel client subscribes on a `response_topic` used to receive the web answer
9) The MQTT broker forwards the `request_payload` received on the `request_topic` to the Tunnel server
10) The Tunnel server extracts the `web_ip` and `web_port` information for the `request_topic`
11) The Tunnel server connects to the Web server on `web_ip:web_port`
12) The Tunnel server sends the received `request_payload` through the socket established with the Web server
13) After it is processed, the Web server sends the `response_payload` to the Tunnel server
14) The Tunnel server subscribes to the `response_topic`
15) The Tunnel server publishes the received `response_payload` on the `response_topic`
16) The MQTT broker forwards the `response_payload` received on the `response_topic` to the Tunnel client
17) The Tunnel client forwards `response_payload` to the Client

[1]In the model, after each publish, an unsubscription from the topic is accomplished, in order to save bandwidth in case of further publications.

TABLE 1. Adopted tools for comparing different tunneling approaches.

Protocol	Tool	Version
MQTT	Mosquitto	v2.0
HTTP	Chisel	v1.7.3
ICMP	ptunnel-ng	v1.42
SSH	OpenSSH	8.1p1
DNS	dns2tcp	v0.5.2

In order to validate the effective functioning of proposed tunneling system based on MQTT and to analyse its performance, we compared it with other tunneling communication protocols tools used to exploit different well-known communication protocols.

IV. TESTBED

In order to validate and test the new tunneling system based on MQTT, we decided to compare it with other communication protocols adopted to implement a tunneling network. After an analysis on the research community [75]–[77], we decided to compare our MQTT scenario with other protocols exploited for tunneling purposes. In particular, we selected HTTP, ICMP, DNS and SSH protocols. All of them are very popular protocols used for everyday communications. The tools used for the tests are able to encapsulate data to be transmitted through the tunnel within the application layer of the protocol used by the tunneling system. The behavior of each tool is similar, apart the exploited protocol. In addition, unlike other papers like [71], which focus on the comparison of multiple tools exploiting the same protocol (e.g. DNS), as the proposed work introduces the exploitation of a new protocol (never exploited before, for tunneling reasons, to the best of our knowledge), we focus on the comparison of tools exploiting different protocols. Because of their characteristics and their adoption in a wide variety of communication flows [71], such protocols are particularly important in the tunneling context. The architectural difference is related only to the communication protocol used to perform the tunneling: by comparing the system proposed on MQTT with the other tunneling systems, the only difference is the presence of the broker, on MQTT (which could be embedded in the same host of the tunnel server). All proposed systems encapsulate the data exchanged at the application layer of the protocol and send it from the client (internal to the network) to the server (external to the network). According to our choice, Table 1 reports details on the tools we adopted, along with their version, for our tests.

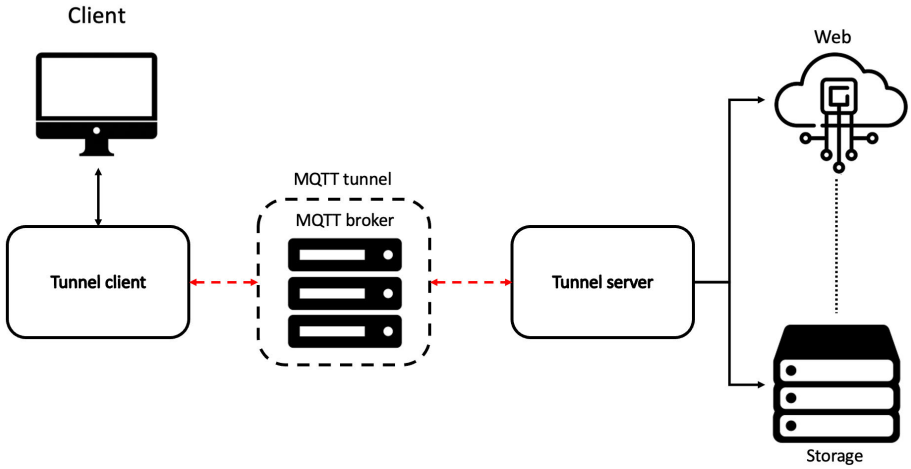

FIGURE 1. The considered MQTT tunnel architecture.

Referring to Mosquitto, according to Figure 1 consider that the MQTT tunneling is composed by different components implemented during our research work, Mosquitto is adopted to communicate on MQTT network.

In order to perform the tests for each tunneling system, a network testbed has been implemented. According to Figure 1, the concept of the network testbed is to simulate a real scenario where the client requires a resource to an external web server through the implemented tunneling system. For our tests, the adopted devices are a Raspberry Pi 3 model B acting as client,[2] a Linux based virtual machine acting as tunnel server, and a physical Linux based server running the MQTT broker. All hosts are part of the same local 1Gbps network.

For the executed tests, in order to simulate a client/server communication, the client generates and sends a random string to the server. Hence, the server answers back to the client, sending the same string received. In order to enhance tests reproducibility and to use the same payload for each protocol exploited, random strings are generated, by using the same seed. Therefore, by fixing the length of the generated string, for each test, the same string is sent/received to/from the server.

Moreover, additional tests, reported in Section V-E, are executed to demonstrate possible application of the tunneling system to encapsulate payloads related to application protocols different from HTTP. In particular, performed tests consider the data exfiltration scenario, referring to the malevolent share of sensitive files (of different size) outside of the organization. In this case, communication with an external SSH service (through the proposed tunneling system) is established, by transferring files through SCP. Such application demonstrates how the MQTT-based tunneling system can be used to encapsulate different TCP-based protocols.

[2]In this case, the client includes both the tunnel client and the web browser surfing the web.

V. EXECUTED TESTS AND OBTAINED RESULTS

In this section of the paper, we first describe the evaluation metrics adopted to compare the tunneling communication protocols described in Section IV. Hence, we report executed tests and, finally, obtained results.

A. EVALUATION METRICS

In order to compare and evaluate the different tunneling systems under analysis, we decided to define new metrics providing us the possibility to measure the performance of each tool. Such metrics are mainly focused on the overhead introduced by the tunneling system, referred to the initial payload to be sent. Being packets/data encapsulated, this metric is important as, in tunneling systems, during encapsulation, each protocol exploited introduces specific headers and footers necessary to communicate across the network through that protocol.

Let's define T the duration of our test, in seconds. S_s identifies the overall size of data (in Bytes) sent by the attacker between the beginning of the attack (assumed to begin at time 0) and its end (assumed to end at time T). Similarly, S_r identifies the overall size of data received by the attacker during the attack execution. Let's also define P_s the payload length (in Bytes) referred to data sent over the tunnel during the attack. Similarly, P_r refers to the payload length of data received through the tunnel.

We define the amplification factor F as follows:

$$F = \frac{S_s + S_r}{P_s + P_r} \tag{1}$$

By defining S as follows:

$$S = S_s + S_r \tag{2}$$

hence, as the overall size of data exchanged during the tunneling attack, similarly, we define P as follows:

$$P = P_s + P_r \tag{3}$$

as the overall size of payload sent through the tunnel.

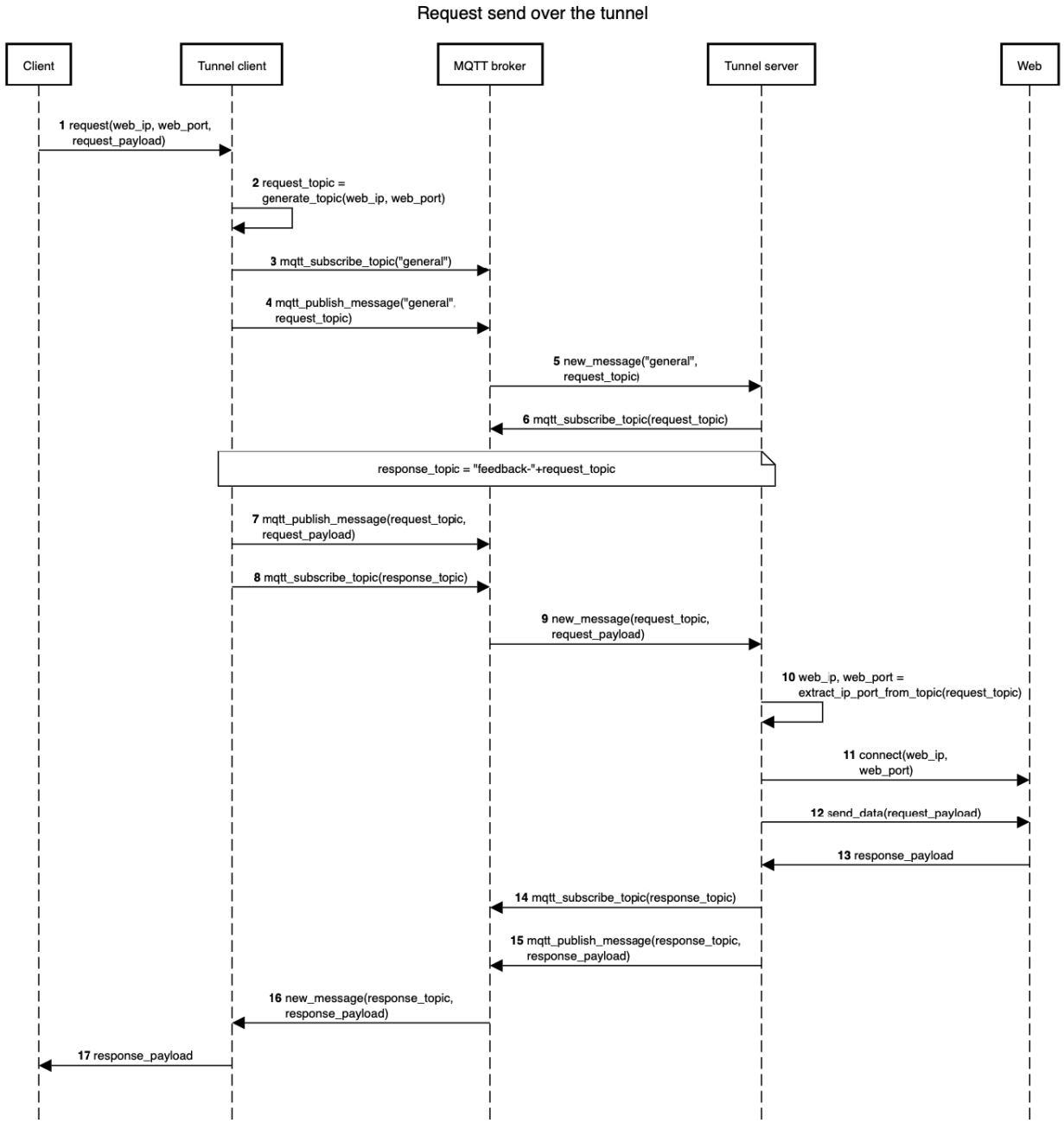

FIGURE 2. Sequence diagram of a practical tunneling system adoption.

We can now simplify Equation 1 to:

$$F = \frac{S}{P} \tag{4}$$

Having defined the amplification factor F, we carried out tests for each considered protocol by varying the payload size P to verify how S and F vary accordingly. Subsequently, for each protocol, a regression function was defined, in order to analyze the trend of the underlying curves associated to each scenario considered. Finally, the different regression functions were compared and analyzed, to identify the most efficient tunneling system, also in function of the payload size P.

B. EXECUTED TESTS

The tests performed refer to the execution of each single scenario, by varying P_s. During each attack execution, we monitored the network traffic of the adopted tunnel from the client side (in our case, the Raspberry Pi).

Particularly, during our tests, assumed I identifies the increment of payload considered between one test and the next one, we adopted the following payload size ranges:

- $P_s \in [100, 1000]$, with $I = 100$
- $P_s \in [1000, 10000]$, with $I = 1000$
- $P_s \in [10000, 100000]$, with $I = 10000$

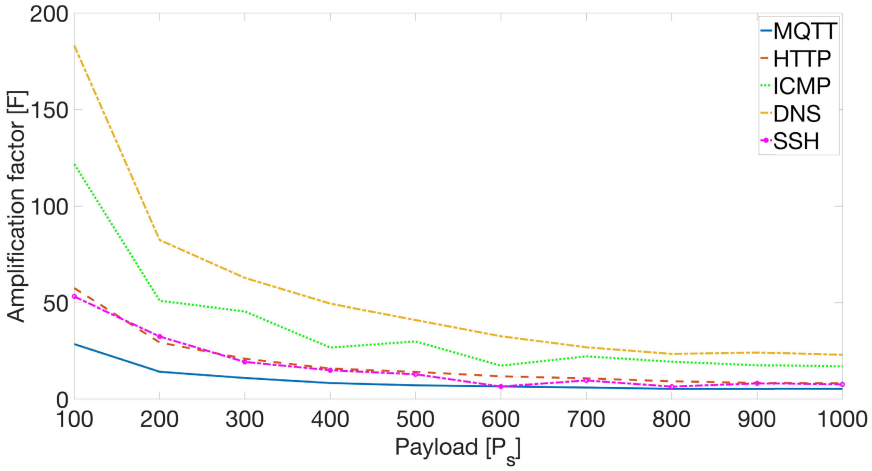

FIGURE 3. Comparison of the amplification factor F obtained for each tunneling method, considering $P_S \in [100, 1000]$.

As reported above in Section IV, we randomly generated the payload used for each scenario. Particularly, in order to make each test on the protocol payload-independent, by fixing P_s, we always generate the same payload. In addition, for each message received by our server, the same message is sent back to the client, hence, having Equation 5 satisfied, for our tests.

$$P_r = P_s \tag{5}$$

C. OBTAINED RESULTS

Tests was executed to compare different tunneling methods exploiting different communication protocols. Our aim is to analyze the behaviour of each tool considered, by varying P_s, the size of the payload generated.

The first test case refers to $P_s \in [100, 1000]$, with $I = 100$. Obtained results, in terms of overall exchanged data volume S, are reported in Table 2. Instead, Figure 3 shows the trends of the F value related to the considered tunnels, by varying the value of P_s.

Results show how, by increasing P_s, F decreases. This is due to the overhead introduced by network packets headers and footers introducedin the communication. As it may be expected, F decreases in percentage with the growth of P_s. In addition, results show how the proposed MQTT-based tunneling method is more efficient than the others, considering adopted P_s values. Similarly, the DNS case results to be the least efficient one.

The second test performed refers instead to $P_s \in [1000, 10000]$, with $I = 1000$. Similarly to the previous test, results are reported in Table 3 and Figure 4.

Obtained results show the MQTT tunneling system is still the best solution, but only until $P_s = 3000$ (excluded). Then, for $P_s >= 3000$, SSH and HTTP tunneling systems should be preferred.

Finally, the last test case is related to $P_s \in [10000, 100000]$, with $I = 10000$. Results are reported in Table 4 and Figure 5.

Results show that, in this range, the HTTP tunneling system turns out to be the best option, along with SSH, that is more efficient for $P_s \in \{10000, 50000\}$.

By analyzing obtained results, especially for the proposed MQTT tunneling system, we found that the proposed attack is the most efficient one (hence, lower F values) for $P_s < 3000$. For higher values of F, other approaches (particularly, SSH and HTTP) resulted more performant. Nevertheless, it is important to mention that the proposed threat does not implement compression capabilities, that may lead to different results, for high values of P_s. Such analysis is in line with further work on the topic. In addition, it should be considered that the choice to adopt a specific tunneling method is often driven by the needs to pass unobserved to the protection systems (e.g. firewall, intrusion detection system (IDS), intrusion prevention system (IPS), etc.) deployed on the targeted organization.

D. TRENDS ANALYSIS

Starting from the results obtained during our tests, we decided to model them, by computing the fitting/regression function related to each of the considered tunnels. The fitting function is a function that represents an approximation of the trend of real/measured data. Generally speaking, by increasing the degree of the equation, the fitting increases. Nevertheless, we have chosen a proper degree of the equation in order to have a function that is not too dependent on the input data, but able to represent the trend of the underlying curve. In particular, functions were calculated over the entire range of payloads (hence, referring to our tests, with $P_s \in [100, 100000]$), to provide a fitting function that better represents our attack models. Also, the R^2 value was calculated for each function. Such value represents the determination coefficient that is used as an indicator of the goodness of the fit [78]. Equations computed are reported in Equation 5.1 (MQTT tunnel),

TABLE 2. Obtained S values for payload size $P_S \in [100, 1000]$.

	P_s									
Protocol	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
MQTT	5716	5713	6645	6767	7269	8067	8579	8725	9773	10677
HTTP	11486	11772	12616	12756	14212	14336	15232	14992	15326	16408
ICMP	24338	20362	27286	21438	29984	20928	31174	31176	31784	33910
DNS	36627	32919	37649	39535	41061	39193	37723	37656	43574	45870
SSH	10616	13002	11616	12052	12998	7972	13734	10562	14897	15312

TABLE 3. Obtained S values for payload size $P_S \in [1000, 10000]$.

	P									
Protocol	**1000**	**2000**	**3000**	**4000**	**5000**	**6000**	**7000**	**8000**	**9000**	**10000**
MQTT	10677	15655	21041	25955	28746	30596	33536	35180	37791	40563
HTTP	16408	22012	20838	23358	24886	36792	29618	31932	34006	36624
ICMP	33910	37958	44048	49674	55454	60966	67046	72704	78118	82708
DNS	45870	64098	59507	77318	75749	84482	94463	102319	111267	119387
SSH	15312	21134	20574	22186	24050	26762	28446	30662	32730	35244

TABLE 4. Obtained S values for payload size $P_S \in [10000, 100000]$.

	P									
Protocol	**10000**	**20000**	**30000**	**40000**	**50000**	**60000**	**70000**	**80000**	**90000**	**100000**
MQTT	40563	62013	82570	102640	126203	146871	169514	189232	210182	233615
HTTP	36624	54792	65072	92058	101966	112732	160528	171294	182138	207600
ICMP	82708	134114	191586	146528	257108	278320	364924	432102	413296	525846
DNS	119387	200757	221906	307299	341222	433190	517069	613494	706652	802597
SSH	35244	56656	86746	99694	101820	152142	164032	185080	197144	226224

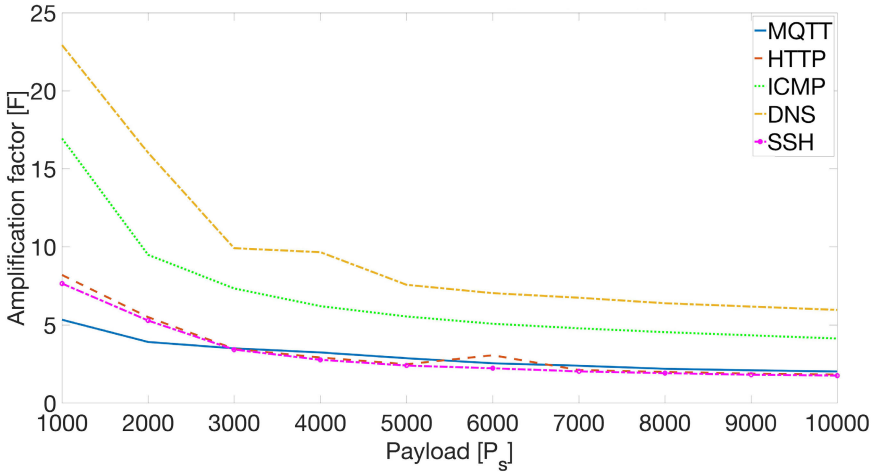

FIGURE 4. Comparison of the amplification factor F obtained for each tunneling method, considering $P_S \in [1000, 10000]$.

TABLE 5. Fitting functions.

Equation	**Protocol**	**Function**	R^2
5.1	MQTT	$y = 100.34 \cdot x^{-0.409}$	0.9582
5.2	HTTP	$y = 431.2 \cdot x^{-0.563}$	0.9521
5.3	ICMP	$y = 645.93 \cdot x^{-0.522}$	0.933
5.4	DNS	$y = 929.53 \cdot x^{-0.52}$	0.9263
5.5	SSH	$y = 292.94 \cdot x^{-0.522}$	0.9206

Equation 5.2 (HTTP tunnel), Equation 5.3 (ICMP tunnel), Equation 5.4 (DNS tunnel), and Equation 5.5 (SSH tunnel).

In order to compare real and fit functions, we reported the behaviour of the fit functions in Figure 6, for $P_s \in [100, 1000]$.

Figure 6 results are in line with Figure 3. In fact, as shown in figure, even in this case, the proposed MQTT tunneling system turns out to be the most effective one, for small payloads. This confirms our statement and the results reported in Table 2.

Fit functions with $P_s \in [1000, 10000]$ are instead reported in Figure 7.

In this case, results are different from our expectations. Particularly, analyzing the graph, the MQTT tunneling system results the best choice also for $P_s \in [1000, 10000]$. Compared to the real data and F values found and reported in Table 3, we find that the impact of HTTP and SSH is not relevant in this case. This may depend to the fitting function

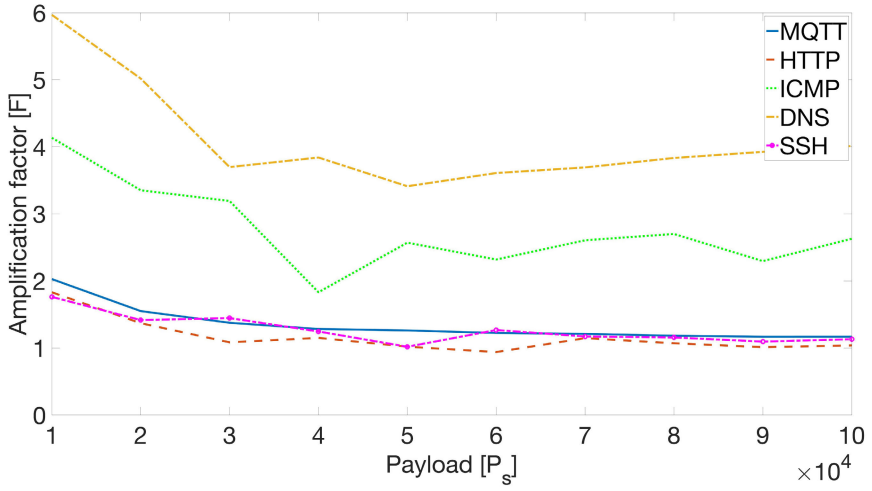

FIGURE 5. Comparison of the amplification factor F obtained for each tunneling method, considering $P_S \in [10000, 100000]$.

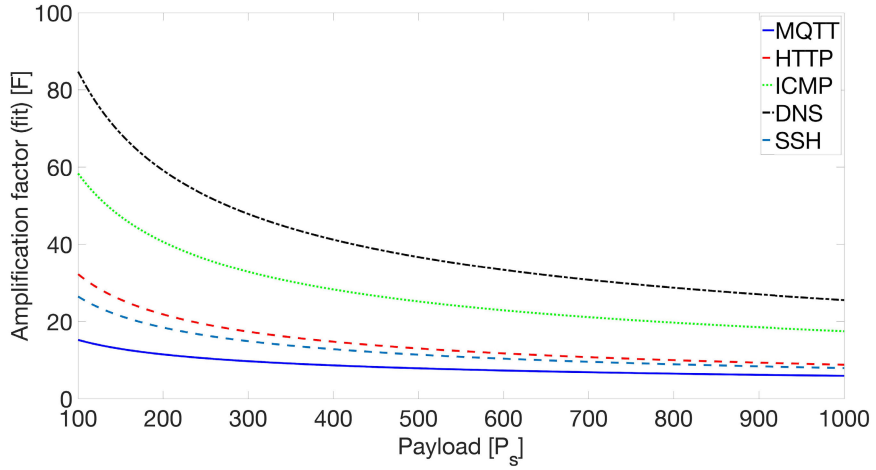

FIGURE 6. Comparison of the fit functions obtained for each tunneling method, considering $P_S \in [100, 1000]$.

considered and its fit to real data, the accuracy of the fitting function (R^2), and/or to the singularity introduced by the considered data, generated randomly, and potentially able to affect compression algorithms in place.

Finally, we report the behaviour of the fit functions applied to $P_s \in [10000, 100000]$ in Figure 8.

By analyzing the figure, the MQTT tunneling system is the most convenient solutions until $P'_s \in [10000, 20000]$. For P_s values higher than P'_s, the HTTP tunneling system first, hence, the SSH tunneling system, turns out to be a better choice than the MQTT tunnel.

By considering the fitting functions we have reported above, we will now try to define the limits of MQTT. In particular, as previously stated, MQTT results the best choice, among the considered ones, for $P_s \leq P'_s$. In order to identify the exact value of P'_s considering two tunneling systems a and b, let's define $L_{a,b}$ the crossing value between two of the proposed algorithms. In particular:

$$L_{a,b} = \{x | y_a = y_b\} \tag{6}$$

For instance, by comparing the MQTT tunnel with the SSH one, we will have:

$$L_{MQTT,SSH} = \{x | y_{MQTT} = y_{SSH}\} \tag{7}$$

hence, in our case:

$$L_{MQTT,SSH} = \{x | 100.34 \cdot x^{-0.409} = 292.94 \cdot x^{-0.522}\} \tag{8}$$

where we find $L_{MQTT,SSH} = 13114$.

As Figure 8 shows potentially similar values for $L_{MQTT,HTTP}$ and $L_{MQTT,SSH}$, we also computed such latter value.

$$L_{MQTT,HTTP} = \{x | y_{MQTT} = y_{HTTP}\} \tag{9}$$

In particular, we found $L_{MQTT,HTTP} = 12933$. Therefore, as we found:

$$L_{MQTT,HTTP} < L_{MQTT,SSH} \tag{10}$$

we find that the fitting function related to MQTT crosses with HTTP one before the SSH one.

Therefore, due to Equation 10, we can state that:

$$P'_s = L_{MQTT,HTTP} \tag{11}$$

Generally speaking, we found that the MQTT tunnel is the best choice for $P_s \in [0, Ps']$, hence, $P_s \in [0, 12933]$. For $P_s > P'_s$, we found that the HTTP tunnel is more performant.

In this context, further work on the topic may be focused on validating the fitting functions found, also evaluating other fitting approaches.

E. ADDITIONAL TESTS BY USING THE MQTT TUNNELING FOR DATA EXFILTRATION ACTIVITIES

In order to demonstrate the possibility of using the tunneling framework for different applications, we tested the architecture to extract data from a possible network protected by MQTT. In particular, the client transfers files of different sizes to the tunnel server by exploiting a SCP communication, which stores them on the disk, as would happen in a potential data exfiltration application.

In detail, the communication flow is based on the scheme shown in Section III. The tunnel client encapsulates the file inside an MQTT payload and sends it to the MQTT broker on a specific communication topic. The tunnel server listens on the broker and waits for incoming packets: when it receives messages from the broker on the specific topic, it starts to consume the MQTT packets to receive the file and stores it on the disk. In order to validate this scenario, we calculated the amplification factor defined in Section V. The transferred files have sizes: 1KB, 10KB, 100KB, 1MB, 3MB. Table 6 reports the amplification factor due to the exfiltration scenario and the related size of the data exchanged.

During our tests, the files have all been received correctly, also verifying correct transmission by calculating the MD5 function. The behavior of the tunneling system complies with what has been demonstrated previously. The overhead introduced by the tunneling system decreases as the file size increases.

The obtained results demonstrate how this tunneling system can also be applied to contexts of exfiltration of sensitive data as the infrastructure and the approach adopted are independent of the single application as all payloads are transmitted via MQTT.

VI. DETECTION OF THE PROPOSED THREAT THROUGH MACHINE LEARNING

In order to provide a first possible protection from the presented MQTT tunneling system, we focused on the adoption of machine learning techniques. Although the main scope of this work is to present the tunneling system based on MQTT, a first approach to identify the attack is necessary to demonstrate the importance of protection systems against cyber-attacks. Detection of attacks occurring in IoT network is a particularly important topic, due to the sensitive information exchanged in networks [79]. In recent years, the application of machine learning (ML) and artificial intelligence (AI) algorithms for monitoring network traffic and detecting attacks on networks or services has become widespread [80], [81]. These algorithms are able to prevent and therefore avoid attacks on infrastructures of different nature but to function correctly they require a training phase, where the quality of the datasets plays an important role [82]. These algorithms can also be applied to the IoT context, to monitor networks and communications to avoid possible attacks on systems.

For this purpose, we adopted the MQTTset dataset [83], a machine learning dataset for MQTT networks implemented by using different sensors. MQTTset simulates a real scenario where IoT sensors communicate on different topic, with different time range and purposes. The considered dataset could be associated to a smart home context, where sensors exchanged information related to temperature, humidity, light intensity, smoke, door opening/closure, CO-Gas, motion, and fan status during a normal day. The dataset is extrapolated from a MQTT network composed by 8 different sensors communicating through a publish/subscribe approach. The features of the dataset are directly extrapolate from the raw data of the communication, in order to retrieve characteristic information relating to the communications of the MQTT protocol. In particular, we selected a portion of the legitimate traffic dataset of MQTTset, in order to use legitimate and tunnel traffic to train machine learning models and to evaluate their efficacy and accuracy. The features adopted in this approach are not focused on the data exchanged but on the behavior of the communications to make the dataset independent from the exchanged content.

As scope of this work, we selected different machine learning algorithms adopted in the cyber-security context to detect cyber-attacks. In particular, we selected a support vector machine (SVM), a decision tree, gradient boost, K-nearest neighbors, logistic regression and random forest. In order to ensure good efficiency of the algorithms, the pre-processing phase of the dataset is critical and very important as the quality of the algorithms depends mainly on the features used in the learning phase. In this work, raw network data was extracted and processed to allow machine learning algorithms to learn the behavior of network traffic directly from the data exchanged by IoT sensors. An incorrect selection of the features could lead to bad training of the network, with no identification of an attack in progress, which is why it is very critical and important.

Regarding the malicious traffic, we simulated a real scenario where a malicious user adopts the MQTT tunnel to navigate on the web. After the initialization phase of the tunnel system, the malicious user visited the following web pages by using a web browser redirected on the Socks5 proxy to

TABLE 6. Obtained values for files with size $P_s \in \{1KB, 10KB, 100KB, 1MB, 3MB\}$.

	P				
Measure	**1KB**	**10KB**	**100KB**	**1MB**	**3MB**
Amplification factor (F)	79804.5	6336.8	3267.24	2764.2415	2773.329
Size data exchanged (S)	129481	110212	610664	5293771	15822656

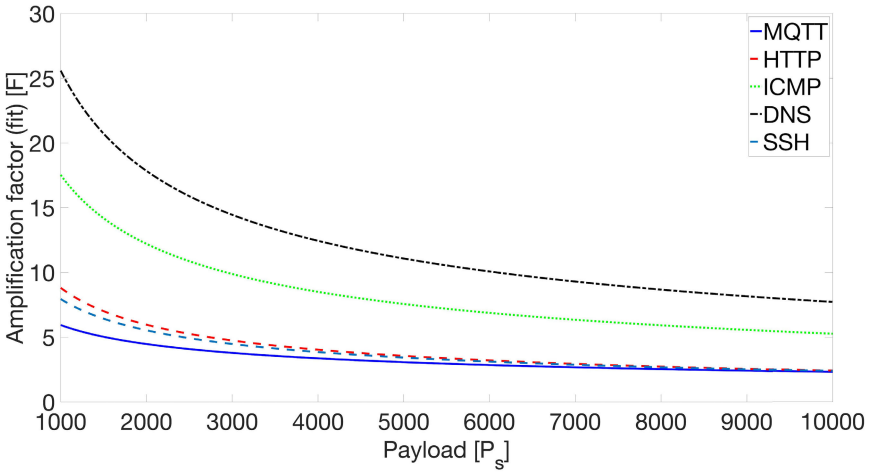

FIGURE 7. Comparison of the fit functions obtained for each tunneling method, considering $P_s \in [1000, 10000]$.

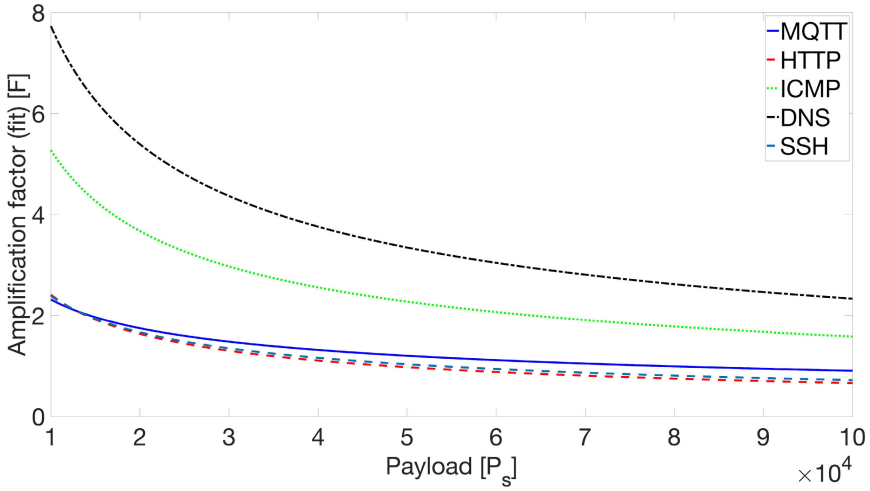

FIGURE 8. Comparison of the fit functions obtained for each tunneling method, considering $P_s \in [10000, 100000]$.

accomplish web surfing activities on the following websites: Google, Amazon, Wikipedia, Google Scholar, Stackoverflow, CNR-IEIIT (our institute website at http://www.ieiit.cnr.it). HTTP/HTTPS packets have been encapsulated inside of MQTT packets, so all traffic is seen as MQTT, while the encapsulated HTTP/HTTPS content is included in the payload of the MQTT message/packet.

The generated network traffic was sniffed between the Tunnel client component and MQTT broker, to resemble a network perimeter firewall installed on the targeted organization able to monitor such communications, in order to elaborate the requests and responses of the malicious user. After generating the malicious traffic and preparing the dataset for the machine learning algorithms, we tested the proposed detection systems and evaluated the obtained results.

A. TESTBED AND OBTAINED RESULTS

After the phase of creating the dataset and the extraction of the features necessary to implement the machine learning algorithms with the aim of classifying the network traffic as malicious or legitimate, we validated the proposed machine learning algorithms. The algorithms were

TABLE 7. Obtained results to detect MQTT tunnel attack.

Algorithm	Accuracy	F1 Score	Training Time (s)	Testing Time (s)
Decision tree	0.49970588235294117	0.3332025887428907	0.039	0.001
Random forest	0.5087254901960784	0.3525997157767825	1.625	0.128
Logistic regression	0.9845588235294118	0.984555637535868	0.530	0.002
K-nearest neighbors	0.4986274509803922	0.3327227528457412	1.212	1.50
Gradient boost	0.49950980392156863	0.33311539718862376	0.725	0.009
Support vector machine	0.875343137254902	0.8744453974088445	2.428	0.01

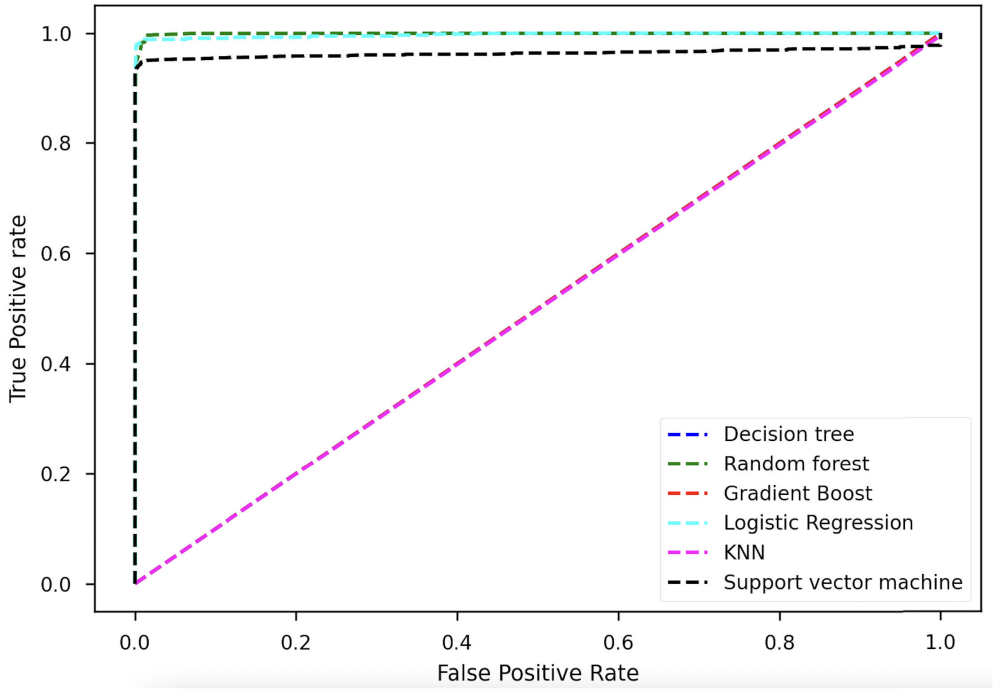

FIGURE 9. ROC curve about machine learning algorithms adopted to detect the MQTT tunneling system.

TABLE 8. Confusion matrices calculated for each machine learning algorithms.

		Predicted 0	Predicted 1
True	0	0	10.200
	1	0	10.200

(a) Decision tree matrix

		Predicted 0	Predicted 1
True	0	180	10.020
	1	2	10.198

(b) Random forest matrix

		Predicted 0	Predicted 1
True	0	10.189	11
	1	304	9896

(c) Logistic regression matrix

		Predicted 0	Predicted 1
True	0	0	10.200
	1	28	10.172

(d) KNN matrix

		Predicted 0	Predicted 1
True	0	0	10.200
	1	10	10.190

(e) Gradient boost matrix

		Predicted 0	Predicted 1
True	0	8066	2134
	1	409	9791

(f) Support vector matrix

implemented through the Sklearn [84] library, an open source machine learning library for the Python programming language. The tests were performed with the same dataset and on the same machine to avoid deviations in results due to different hardware or data between the different tests to maintain consistency on the tests and results. Moreover, the quantities of the legitimate and malicious datasets are of the same order of measurement in order to have a balanced dataset to avoid possible high accuracy or F1 score values, due to an imbalance of one of the two datasets. Since the dataset generated is composed of legitimate (MQTTset) or malicious (tunnel) network traffic, the detection system must

TABLE 9. Optimized hyperparameter obtained with Optuna.

Algorithm	Hyperparameters
Decision tree	max_depth = **199** , min_samples_split = **152**, max_leaf_nodes = **54**, splitter = **best** criterion = **entropy**, min_samples_leaf = **285**
Random forest	max_depth = **256**, min_samples_split = **118**, n_estimators = **179**, max_leaf_nodes = **262** criterion = **gini**, min_samples_leaf = **40**
Logistic regression	C parameter = **0.012707648116840513**, max_iter = **191**, solver = **newton-ng**
K-nearest neighbors	n_neighbors = **1**, leaf_size = **1**, p parameter = **2**, algorithm = **kd_tree**, weights = **distance**
Gradient boost	max_depth = **323**, min_samples_split = **147**, n_estimators = **50** criterion = **friedman_mse**, min_samples_leaf = **371**
Support vector machine	C parameter = **3.8631191813112075e-05**, max_iter = **2941**, loss = **squared_hinge**

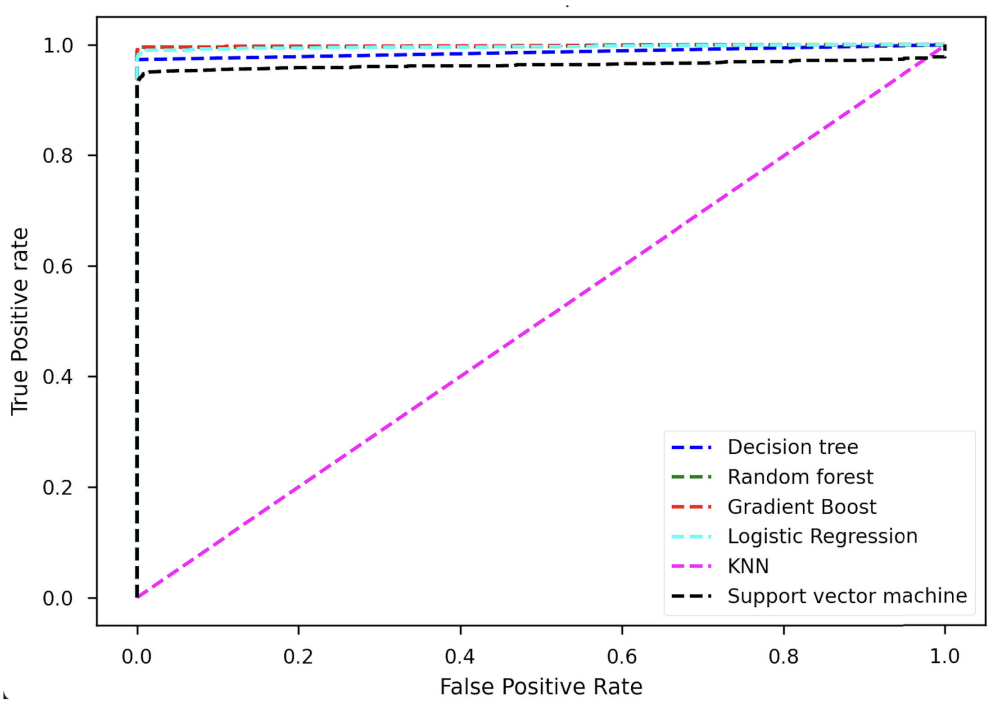

FIGURE 10. ROC curve with machine learning algorithms optimized.

TABLE 10. Obtained results to detect MQTT tunnel attack with hyperparameter optimization.

Algorithm	Accuracy	F1 Score	Training Time (s)	Testing Time (s)
Decision tree	0.9864705882352941	0.9864681112909595	0.050	0.002
Random forest	0.9968627450980392	0.9968627353280108	2.284	0.211
Logistic regression	0.9842647058823529	0.9842609532225192	5.818	0.003
K-nearest neighbors	0.4991666666666665	0.3329627570872707	1.394	0.896
Gradient boost	0.9572549019607843	0.9571867361173693	1.969	0.0158
Support vector machine	0.9792156862745098	0.9792153666338584	6.705	0.003

solve a binary classification problem as it must identify if the traffic is legitimate or not. The final dataset is composed by 160.000 records. The aim of the intrusion detection system is to identify possible malicious behavior in real time and with a certain reliability to protect networks from possible cyber-attacks.

In the performed tests, the algorithms based machine learning techniques are implemented with default configuration provided by Sklearn. The decision tree (DecisionTreeClassifier) is implemented with *gini* criterion, *best* splitter and maximum depth set until all leaves are pure, while the random forest classifier (RandomForestClassifier) is tested with 100 estimators, *gini* criterion and at least 1 leaf. The Logistic Regression (LogisticRegression) is executed with *l2* penalty, *lbfgs* solver and C equals to 1. The K-nearest neighbors (KNeighborsClassifier) is implemented with 5 neighbors and leaf size equal to 30 while the Gradient Boost (GradientBoostingClassifier) has 100 estimators, a *friedman_mse* criterion and deviance loss. Finally, Support Vector Machine (LinearSVC) is composed by a linear kernel and a *squared_hinge* parameter as loss. In order to evaluated the intrusion detection system based on machine learning

TABLE 11. Confusion matrices for algorithms optimized.

		Predicted 0	Predicted 1
True	0	10.200	0
	1	276	9924

(a) Decision tree matrix optimized

		Predicted 0	Predicted 1
True	0	10.186	14
	1	2	10.198

(b) Random forest matrix optimized

		Predicted 0	Predicted 1
True	0	10.197	3
	1	318	9882

(c) Logistic regression matrix optimized

		Predicted 0	Predicted 1
True	0	0	10.200
	1	17	10.183

(d) KNN matrix optimized

		Predicted 0	Predicted 1
True	0	9357	843
	1	29	10.171

(e) Gradient boost matrix optimized

		Predicted 0	Predicted 1
True	0	10.028	172
	1	252	9948

(f) Support vector matrix optimized

algorithms, the dataset is splitted in: 70% of training data and 30% of test data. In order to reproduce the same tests, a seed is fixed. Finally, the intrusion detection system is first trained and then tested on the train and test dataset as usually procedure in machine learning application [85]. Table 7 reports the obtained results for the different machine learning algorithms based on accuracy, F1 score [86] and performing time while Figure 9 reports ROC curves of the algorithms.

Moreover, the confusion matrix of each algorithm is reported in Table 8 to demonstrate a balanced dataset able to validate the obtained results in terms of accuracy, F1 score and ROC curve.

Accuracy is considered as the proportion between the correct predictions with the size of input. Instead, the F1 score is mean between precision and recall, where precision is the ratio between true positives and all positive results, while recall is ratio between true positives and all positive tests [87]. Instead, the ROC curve is adopted to demonstrate between true alarms (hit rate) and false alarms in terms of False and True Positive Rate [88].

Analyzing the results obtained, we note that the metrics report low values between about 45% and 80% of accuracy and F1 score. In particular, the decision tree, random forest and k-nearest neighbors algorithm have 49% of accuracy and 35 % as F1 score which are very low value to detect a cyber-attack. Instead, gradient boost, logistic regression and support vector machine are between 67-79% of accuracy and 64-78% of F1 score which is more precise but not good enough to this topic since a cyber-attack has many possibilities to be executed without countermeasures. These results actually demonstrate that the tunnel is difficult to identify as it simulates real behavior, with packet exchange between the different components of the MQTT network as if they were legitimate communications (such as when a sensor sends a request to another device to request a resource and the second sensor transmits the requested resource).

The obtained results also demonstrate that the basic algorithms are not efficient in detecting this attack but to carry out a correct identification, detailed and meticulous configurations must be tested and validated. For this reason, we decided to perform an optimization for all the configuration parameter algorithms, to validate if the attack can be correctly identified. In the next section, parametric optimization and the obtained results will be presented.

B. HYPERPARAMETERS OPTIMIZATION

In order to optimize the machine learning detection system to obtain better results, we worked on the optimization of hyperparameter of the algorithms. The hyperparameter optimization challenge is to choose a set of optimal hyperparameter for a learning algorithm to improve the quality of the training and test metrics.

In order to achieve these results, we adopted the hyperparameter optimization tool called Optuna [89]. Optuna formulates the hyperparameter optimization as a process of minimizing/maximizing an objective function that takes a set of hyperparameter as an input and returns its score. By using this tool, we defined for each algorithm, the parameters to be optimize:

- Decision tree: max_depth, min_samples_split, max_leaf_nodes, splitter, criterion, min_samples_leaf
- Random forest: max_depth, min_samples_split, n_estimators, max_leaf_nodes, criterion, min_samples_leaf
- Logistic regression: C parameter, max_iter, solver
- K-nearest neighbors: n_neighbors, leaf_size, p parameter, algorithm, weights
- Gradient boost: max_depth, min_samples_split, n_estimators, criterion, min_samples_leaf
- Support vector machine: C parameter, max_iter, loss

Once the parameters to be optimized to improve the algorithm metrics have been defined, Optuna starts combining the different parameters and testing the algorithm to see if the combination of the parameters leads to an algorithm improvement. At the end of the parameter testing process, the optimal machine learning model returns, i.e. the one with the best metrics. In our tests, to obtain efficient results and to test a good

number of parameter combinations, 400 parameter combinations with different values (chosen by Optuna according to a tool logic) were performed for each algorithm. The final and optimal configuration of each algorithm obtained with Optuna is reported in Table 9.

Once the optimal parameters for the algorithms were obtained, accuracy metrics, F1 score, ROC curves and confusion matrices were calculated again to validate the algorithm improvement. Obtained results are reported in Table 10 for accuracy and F1 score, Figure 10 and Table 11 for ROC curve and decision matrices.

By analyzing the obtained results with the optimized hyperparameters, it is possible to highlight an improvement of the matrices for the machine learning algorithms. In particular, the decision tree and random forest resulted in 98% of accuracy and F1 score, gradient boost obtained a 95% of accuracy and F1 score, logistic regression is improved with 98% of accuracy and F1 score, while support vector machine reached 97%. Compared to the previous results, the metrics indicate a more precise and efficient approach to detect the MQTT tunnel. Instead, k-nearest neighbors obtained a little improvement but the performance metrics values are still too low to detect a possible attack in an efficient way. In summary, the hyperparameter approach with Optuna led to an important improvement of some machine learning algorithms that with the following configurations, can be applied in a real context to perform the detection of the MQTT tunnel with high efficiency and precision.

VII. CONCLUSION

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 several IoT contexts, in order to exploit its features to execute a cyber-attack. Particularly, we implemented a novel tunneling system based on MQTT exploitation, to demonstrate that it is possible to use MQTT for nefarious purposes.

In order to validate the proposed tunneling system, we compared it with other tunnels exploiting HTTP, DNS, ICMP and SSH protocols. Particularly, in order to accurately compare such scenarios, we defined new metrics able to validate the efficiency of the tunnel, in function of the amplification factor metric we introduced. As shown in the results, the proposed MQTT tunneling system results the best choice for payloads up to 3000 bytes, while, for higher payloads, an HTTP tunnel results more efficient. Moreover, the MQTT tunneling system is validated to transfer file to an external server with malicious behaviour similar to steal sensitive data. The obtained results achieved the initial goals of the work: the MQTT protocol can be exploited to execute a tunneling system in order to communicate outside of the IoT network with payload up to 3000 bytes. Here, it should be considered that compression was not implemented in the proposed attack, while further work on the topic may include such functionality.

Finally, a first approach to design a detection system able to counter the proposed threat was presented, by using and comparing different machine learning algorithms to identify the introduced attack. The first detection approach based on default configurations of the machine learning models shows that the MQTT tunneling system is complex to classify and detect, due to its stealth nature. Only the SVM algorithm obtained an accuracy near to 87%. Instead, after the hyperparameter optimization, most of the algorithms obtained an accuracy near to 95%-99%, only the KNN is not able to detect the tunneling system. As expected, we found better results. Nevertheless, the time required to find the optimal configuration of the algorithm parameters is very high.

Further work on the topic may be directed to refine the proposed detection methodology and analyze its efficiency on a relevant scenario, by focusing on features selection and ranking activities. Considering the proposed attack, encryption and compression will be evaluated as scope of further work, in order to optimize the attack to be even more stealthy and difficult to detect. In particular, encryption could be evaluated to guarantee confidentiality of communications. Instead, compression could be adopted to reduce required bandwidth. Similarly, a better representation of the data, in terms of fitting curves and regression methodologies may be in the scope of further work on the topic, as well as investigating the gap between real and fitting functions. Alternatively, an extension of the proposed work may be directed toward a refinement of the MQTT tunneling system, comparing it with other possible tunneling solutions, like VPN, or by exploiting other communication protocols to implement malicious tunneling systems such as Host Identity Protocol. Regarding the machine learning detection system, we will investigate the detection of other communication protocols by using machine learning algorithms. Another detection approach, not based on machine learning, is to investigate virtual domains and limit IoT devices capabilities to avoid possible cyber-attacks. Moreover, a possible future work is related to the control of the data exchanged in the network by using a specific node able to elaborate the MQTT packets similar to a firewall or IDS system.

REFERENCES

[1] K. Ashton, "That 'Internet of Things' thing," *RFID J.*, vol. 22, no. 7, pp. 97–114, Jun. 2009.

[2] D. Conzon, P. Brizzi, P. Kasinathan, C. Pastrone, F. Pramudianto, and P. Cultrona, "Industrial application development exploiting IoT vision and model driven programming," in *Proc. 18th Int. Conf. Intell. Next Gener. Netw.*, 2015, pp. 168–175.

[3] P. Varga, J. Peto, A. Franko, D. Balla, D. Haja, F. Janky, G. Soos, D. Ficzere, M. Maliosz, and L. Toka, "5G support for industrial IoT applications—Challenges, solutions, and research gaps," *Sensors*, vol. 20, no. 3, p. 828, Feb. 2020.

[4] J. Sengupta, S. Ruj, and S. D. Bit, "A comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT," *J. Netw. Comput. Appl.*, vol. 149, Jan. 2020, Art. no. 102481.

[5] T. G. Lewis, *Critical Infrastructure Protection in Homeland Security: Defending a Networked Nation*. Hoboken, NJ, USA: Wiley, 2019.

[6] T. Mahler, N. Nissim, E. Shalom, I. Goldenberg, G. Hassman, A. Makori, I. Kochav, Y. Elovici, and Y. Shahar, "Know your enemy: Characteristics of cyber-attacks on medical imaging devices," 2018, *arXiv:1801.05583*. [Online]. Available: http://arxiv.org/abs/1801.05583

[7] A. Cornel-Cristian, T. Gabriel, M. Arhip-Calin, and A. Zamfirescu, "Smart home automation with MQTT," in *Proc. 54th Int. Universities Power Eng. Conf. (UPEC)*, Sep. 2019, pp. 1–5.

[8] A. Lohachab and Karambir, "ECC based inter-device authentication and authorization scheme using MQTT for IoT networks," *J. Inf. Secur. Appl.*, vol. 46, pp. 1–12, Jun. 2019.

[9] R. Zitouni, J. Petit, A. Djoudi, and L. George, "IoT-based urban traffic-light control: Modelling, prototyping and evaluation of MQTT protocol," in *Proc. Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData)*, Jul. 2019, pp. 182–189.

[10] C. Patel and N. Doshi, "A novel MQTT security framework in generic IoT model," *Procedia Comput. Sci.*, vol. 171, pp. 1399–1408, Jan. 2020.

[11] F. Palau, C. Catania, J. Guerra, S. Garcia, and M. Rigaki, "DNS tunneling: A deep learning based lexicographical detection approach," 2020, *arXiv:2006.06122*. [Online]. Available: http://arxiv.org/abs/2006.06122

[12] Y. He, Y. Zhu, and W. Lin, "HTTP tunnel trojan detection model based on deep learning," *J. Phys., Conf. Ser.*, vol. 1187, no. 4, Apr. 2019, Art. no. 042055.

[13] A. Singh, O. Nordström, C. Lu, and A. L. Dos Santos, "Malicious ICMP tunneling: Defense against the vulnerability," in *Proc. Australas. Conf. Inf. Secur. Privacy*. Berlin, Germany: Springer, 2003, pp. 226–236.

[14] M. Dusi, F. Gringoli, and L. Salgarelli, "A preliminary look at the privacy of SSH tunnels," in *Proc. 17th Int. Conf. Comput. Commun. Netw.*, Aug. 2008, pp. 1–7.

[15] J. Deogirikar and A. Vidhate, "Security attacks in IoT: A survey," in *Proc. Int. Conf. I-SMAC (IoT Social, Mobile, Analytics Cloud) (I-SMAC)*, Feb. 2017, pp. 32–37.

[16] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of Things security: A survey," *J. Netw. Comput. Appl.*, vol. 88, pp. 10–28, Jun. 2017.

[17] H. Akram, D. Konstantas, and M. Mahyoub, "A comprehensive IoT attacks survey based on a building-blocked reference model," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 3, pp. 355–373, Apr. 2018.

[18] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, "Internet of Things (IoT): Taxonomy of security attacks," in *Proc. 3rd Int. Conf. Electron. Design (ICED)*, Aug. 2016, pp. 321–326.

[19] J. Tournier, F. Lesueur, F. L. Mouël, L. Guyon, and H. Ben-Hassine, "A survey of IoT protocols and their security issues through the lens of a generic IoT stack," *Internet Things*, Jul. 2020, Art. no. 100264.

[20] I. Stellios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz, and J. Lopez, "A survey of IoT-enabled cyberattacks: Assessing attack paths to critical infrastructures and services," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3453–3495, 4th Quart., 2018.

[21] F. Al-Turjman, H. Zahmatkesh, and R. Shahroze, "An overview of security and privacy in smart cities' IoT communications," *Trans. Emerg. Telecommun. Technol.*, vol. e3677, Jul. 2019.

[22] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, and D. Kumar, "Understanding the mirai botnet," in *Proc. 26th USENIX Secur. Symp. (USENIX Secur.)*, 2017, pp. 1093–1110.

[23] S. N. Firdous, Z. Baig, C. Valli, and A. Ibrahim, "Modelling and evaluation of malicious attacks against the IoT MQTT protocol," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData)*, Jun. 2017, pp. 748–755.

[24] S. Andy, B. Rahardjo, and B. Hanindhito, "Attack scenarios and security analysis of MQTT communication protocol in IoT system," in *Proc. 4th Int. Conf. Electr. Eng., Comput. Sci. Informat. (EECSI)*, Sep. 2017, pp. 1–6.

[25] I. Vaccari, M. Aiello, and E. Cambiaso, "SlowITe, a novel denial of service attack affecting MQTT," *Sensors*, vol. 20, no. 10, p. 2932, May 2020.

[26] I. Vaccari, M. Aiello, and E. Cambiaso, "SlowTT: A slow denial of service against IoT networks," *Information*, vol. 11, no. 9, p. 452, Sep. 2020.

[27] L. Malina, G. Srivastava, P. Dzurenda, J. Hajny, and R. Fujdiak, "A secure publish/subscribe protocol for Internet of Things," in *Proc. 14th Int. Conf. Availability, Rel. Secur.*, Aug. 2019, pp. 1–10.

[28] R. Bryce, T. Shaw, and G. Srivastava, "MQTT-G: A publish/subscribe protocol with geolocation," in *Proc. 41st Int. Conf. Telecommun. Signal Process. (TSP)*, Jul. 2018, pp. 1–4.

[29] N. V. Ha and M. Tsuru, "On the characteristics of TCP/NC tunneling in heterogeneous environments," in *Advances in Intelligent Networking and Collaborative Systems*, F. Xhafa, L. Barolli, and M. Greguš, Eds. Cham, Switzerland: Springer, 2019, pp. 340–349.

[30] G. Merlino, D. Bruneo, S. Distefano, F. Longo, and A. Puliafito, "Enabling mechanisms for cloud-based network virtualization in IoT," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 268–273.

[31] A. Alshehri, J. Granley, and C. Yue, "Attacking and protecting tunneled traffic of smart home devices," in *Proc. 10th ACM Conf. Data Appl. Secur. Privacy*. New York, NY, USA: ACM, Mar. 2020, pp. 259–270, doi: 10.1145/3374664.3375723.

[32] P. Nikander, A. Gurtov, and T. R. Henderson, "Host identity protocol (HIP): Connectivity, mobility, multi-homing, security, and privacy over IPv4 and IPv6 networks," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 2, pp. 186–204, 2nd Quart., 2010.

[33] I. Ahmad, M. Liyanage, M. Ylianttila, and A. Gurtov, "Analysis of deployment challenges of host identity protocol," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2017, pp. 1–6.

[34] T. R. Henderson, J. M. Ahrenholz, and J. H. Kim, "Experience with the host identity protocol for secure host mobility and multihoming," in *Proc. IEEE Wireless Commun. Netw. (WCNC)*, vol. 3, Mar. 2003, pp. 2120–2125.

[35] M. Liyanage, M. Ylianttila, and A. Gurtov, "Securing the control channel of software-defined mobile networks," in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw.*, Jun. 2014, pp. 1–6.

[36] J. D. D. Hoz, J. Saldana, J. Fernandez-Navajas, J. Ruiz-Mas, R. G. Rodríguez, and F. D. J. M. Luna, "SSH as an alternative to TLS in IoT environments using HTTP," in *Proc. Global Internet Things Summit (GIoTS)*, Jun. 2018, pp. 1–6.

[37] Y.-A. Tan, X. Zhang, K. Sharif, C. Liang, Q. Zhang, and Y. Li, "Covert timing channels for IoT over mobile networks," *IEEE Wireless Commun.*, vol. 25, no. 6, pp. 38–44, Dec. 2018.

[38] J.-W. Ho, "Covert channel establishment through the dynamic adaptation of the sequential probability ratio test to sensor data in IoT," *IEEE Access*, vol. 7, pp. 146093–146107, 2019.

[39] K. Denney, A. S. Uluagac, K. Akkaya, and S. Bhansali, "A novel storage covert channel on wearable devices using status bar notifications," in *Proc. 13th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2016, pp. 845–848.

[40] A. Velinov, A. Mileva, S. Wendzel, and W. Mazurczyk, "Covert channels in the MQTT-based Internet of Things," *IEEE Access*, vol. 7, pp. 161899–161915, 2019.

[41] W. Mazurczyk and L. Caviglione, "Steganography in modern smartphones and mitigation techniques," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 334–357, 1st Quart., 2015.

[42] S. Zander, G. Armitage, and P. Branch, "A survey of covert channels and countermeasures in computer network protocols," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 3, pp. 44–57, 3rd Quart., 2007.

[43] M. Elsadig and Y. Fadlalla, "Survey on covert storage channel in computer network protocols: Detection and mitigation techniques," *Int. J. Adv. Comput. Netw. Secur.*, vol. 6, pp. 11–17, Dec. 2016.

[44] P. L. Shrestha, M. Hempel, F. Rezaei, and H. Sharif, "A support vector machine-based framework for detection of covert timing channels," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 2, pp. 274–283, Apr. 2015.

[45] G. Xu, W. Yang, and L. Huang, "Supervised learning framework for covert channel detection in LTE—A," *IET Inf. Secur.*, vol. 12, no. 6, pp. 534–542, Nov. 2018.

[46] M. A. Ayub, S. Smith, and A. Siraj, "A protocol independent approach in network covert channel detection," in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE), IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Aug. 2019, pp. 165–170.

[47] A. Almusawi and H. Amintoosi, "DNS tunneling detection method based on multilabel support vector machine," *Secur. Commun. Netw.*, vol. 2018, pp. 1–9, Jan. 2018.

[48] M. Eskandari, Z. H. Janjua, M. Vecchio, and F. Antonelli, "Passban IDS: An intelligent anomaly-based intrusion detection system for IoT edge devices," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6882–6897, Aug. 2020.

[49] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home IoT devices," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 9042–9053, Oct. 2019.

[50] H. Yao, D. Fu, P. Zhang, M. Li, and Y. Liu, "MSML: A novel multilevel semi-supervised machine learning framework for intrusion detection system," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1949–1959, Apr. 2018.

[51] N. Moustafa, B. Turnbull, and K.-K.-R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4815–4830, Jun. 2019.

[52] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, "Deep learning-based intrusion detection for IoT networks," in *Proc. IEEE 24th Pacific Rim Int. Symp. Dependable Comput. (PRDC)*, Dec. 2019, pp. 256–25609.

[53] Y. Sun, L. Zhang, and C. Zhao, "A study of network covert channel detection based on deep learning," in *Proc. 2nd IEEE Adv. Inf. Manage., Communicates, Electron. Autom. Control Conf. (IMCEC)*, May 2018, pp. 637–641.

[54] P. Zhang, H. Li, Q. P. Ha, Z.-Y. Yin, and R.-P. Chen, "Reinforcement learning based optimizer for improvement of predicting tunneling-induced ground responses," *Adv. Eng. Informat.*, vol. 45, Aug. 2020, Art. no. 101097.

[55] M. Chourib, "Detecting selected network covert channels using machine learning," in *Proc. Int. Conf. High Perform. Comput. Simul. (HPCS)*, Jul. 2019, pp. 582–588.

[56] M. Aiello, M. Mongelli, and G. Papaleo, "DNS tunneling detection through statistical fingerprints of protocol messages and machine learning," *Int. J. Commun. Syst.*, vol. 28, no. 14, pp. 1987–2002, Sep. 2015.

[57] T. de Toledo and N. Torrisi, "Encrypted DNP3 traffic classification using supervised machine learning algorithms," *Mach. Learn. Knowl. Extraction*, vol. 1, no. 1, pp. 384–399, Jan. 2019.

[58] S. Rathore and J. H. Park, "Semi-supervised learning based distributed attack detection framework for IoT," *Appl. Soft Comput.*, vol. 72, pp. 79–89, Nov. 2018.

[59] X. C. Yin, Z. G. Liu, L. Nkenyereye, and B. Ndibanje, "Toward an applied cyber security solution in IoT-based smart grids: An intrusion detection system approach," *Sensors*, vol. 19, no. 22, p. 4952, Nov. 2019.

[60] X. Yan, Y. Xu, X. Xing, B. Cui, Z. Guo, and T. Guo, "Trustworthy network anomaly detection based on an adaptive learning rate and momentum in IIoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 6182–6192, Sep. 2020.

[61] A. Souri, A. S. Mohammed, M. Y. Potrus, M. H. Malik, F. Safara, and M. Hosseinzadeh, "Formal verification of a hybrid machine learning-based fault prediction model in Internet of Things applications," *IEEE Access*, vol. 8, pp. 23863–23874, 2020.

[62] A. Makkar and N. Kumar, "An efficient deep learning-based scheme for web spam detection in IoT environment," *Future Gener. Comput. Syst.*, vol. 108, pp. 467–487, Jul. 2020.

[63] S. Ahmed, Y. Lee, H. Seung-Ho, and I. Koo, "Unsupervised machine learning-based detection of covert data integrity assault in smart grid networks utilizing isolation forest," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 10, pp. 2765–2777, Mar. 2019.

[64] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "CorrAUC: A malicious bot-IoT traffic detection method in IoT network using machine-learning techniques," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3242–3254, Mar. 2021.

[65] Y. Xu, C. Zhang, Q. Zeng, G. Wang, J. Ren, and Y. Zhang, "Blockchain-enabled accountability mechanism against information leakage in vertical industry services," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1202–1213, Apr. 2021.

[66] C. Zhang, Y. Xu, Y. Hu, J. Wu, J. Ren, and Y. Zhang, "A blockchain-based multi-cloud storage data auditing scheme to locate faults," *IEEE Trans. Cloud Comput.*, early access, Feb. 8, 2021, doi: 10.1109/TCC.2021.3057771.

[67] M. Alshahrani and I. Traore, "Secure mutual authentication and automated access control for IoT smart home using cumulative Keyed-hash chain," *J. Inf. Secur. Appl.*, vol. 45, pp. 156–175, Apr. 2019.

[68] D. Raman, B. De Sutter, B. Coppens, S. Volckaert, K. De Bosschere, P. Danhieux, and E. Van Buggenhout, "DNS tunneling for network penetration," in *Proc. Int. Conf. Inf. Secur. Cryptol.* Berlin, Germany: Springer, 2012, pp. 65–77.

[69] K. M. S. Soyjaudah, P. C. Catherine, and I. Coonjah, "Evaluation of UDP tunnel for data replication in data centers and cloud environment," in *Proc. Int. Conf. Comput., Commun. Autom. (ICCCA)*, Apr. 2016, pp. 1217–1221.

[70] G. Son, C. C. Tse, and D. Fedotenko, "System and method for enabling VPN tunnel status checking," U.S. Patent 8 458 248, Jun. 4, 2013.

[71] A. Merlo, G. Papaleo, S. Veneziano, and M. Aiello, "A comparative performance evaluation of DNS tunneling tools," in *Computational Intelligence in Security for Information Systems*. Berlin, Germany: Springer, 2011, pp. 84–91.

[72] A. Kim, J. Oh, J. Ryu, and K. Lee, "A review of insider threat detection approaches with IoT perspective," *IEEE Access*, vol. 8, pp. 78847–78867, 2020.

[73] Y. Liu, C. Corbett, K. Chiang, R. Archibald, B. Mukherjee, and D. Ghosal, "SIDD: A framework for detecting sensitive data exfiltration by an insider attack," in *Proc. 42nd Hawaii Int. Conf. Syst. Sci.*, Jan. 2009, pp. 1–10.

[74] R. K. Kodali and S. Soratkal, "MQTT based home automation system using ESP8266," in *Proc. IEEE Region Humanitarian Technol. Conf. (R-HTC)*, Dec. 2016, pp. 1–5.

[75] Y. Wu and X. Zhou, "Research on the IPv6 performance analysis based on dual-protocol stack and tunnel transition," in *Proc. 6th Int. Conf. Comput. Sci. Educ. (ICCSE)*, Aug. 2011, pp. 1091–1093.

[76] F. T. Al-Dhief, N. Sabri, N. M. A. Latiff, N. N. N. A. Malik, M. Abbas, A. Albader, M. A. Mohammed, R. N. Al-Haddad, Y. D. Salman, M. Khanapi, and O. I. O. A. Ghani, "Performance comparison between TCP and UDP protocols in different simulation scenarios," *Int. J. Eng. Technol.*, vol. 7, no. 4.36, pp. 172–176, 2018.

[77] I. Coonjah, P. C. Catherine, and K. M. S. Soyjaudah, "Experimental performance comparison between TCP vs UDP tunnel using OpenVPN," in *Proc. Int. Conf. Comput., Commun. Secur. (ICCCS)*, Dec. 2015, pp. 1–5.

[78] I. S. Helland, "On the interpretation and use of R^2 in regression analysis," *Biometrics*, vol. 43, no. 1, pp. 61–69, Mar. 1987.

[79] I. Vaccari, M. Aiello, F. Pastorino, and E. Cambiaso, "Protecting the ESP8266 module from replay attacks," in *Proc. Int. Conf. Commun., Comput., Cybersecurity, Informat. (CCCI)*, Nov. 2020, pp. 1–6.

[80] F. Farivar, M. S. Haghighi, A. Jolfaei, and M. Alazab, "Artificial intelligence for detection, estimation, and compensation of malicious attacks in nonlinear cyber-physical systems and industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2716–2725, Apr. 2020.

[81] H. Karimipour, A. Dehghantanha, R. M. Parizi, K.-K. R. Choo, and H. Leung, "A deep and scalable unsupervised machine learning system for cyber-attack detection in large-scale smart grids," *IEEE Access*, vol. 7, pp. 80778–80788, 2019.

[82] C. Catal and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," *Inf. Sci.*, vol. 179, no. 8, pp. 1040–1058, 2009.

[83] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, and E. Cambiaso, "MQTTset, a new dataset for machine learning techniques on MQTT," *Sensors*, vol. 20, no. 22, p. 6578, Nov. 2020.

[84] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[85] S. K. Wagh, V. K. Pachghare, and S. R. Kolhe, "Survey on intrusion detection system using machine learning techniques," *Int. J. Comput. Appl.*, vol. 78, no. 16, pp. 30–37, Sep. 2013.

[86] J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes, "Improving efficiency and accuracy in multilingual entity extraction," in *Proc. 9th Int. Conf. Semantic Syst. (I-SEMANTICS)*, 2013, pp. 121–124.

[87] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. M. A. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," *Internet Things*, vol. 7, Sep. 2019, Art. no. 100059.

[88] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[89] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2623–2631.



IVAN VACCARI received the Computer Engineering degree (Hons.) and the Ph.D. degree in computer science from the University of Genoa, in 2017 and 2021, respectively. During his research activities, he worked in different European projects focused on security in healthcare data, the IoT, and financial infrastructures. He is currently a Research Fellow with the Consiglio Nazionale delle Ricerche, IEIIT Institute, working on the IoT and network security focused on identification of vulnerabilities and developed of innovative cyber threats. Regarding detection and mitigation systems, he is working on machine learning and artificial intelligence approaches.



SARA NARTENI received the M.Sc. degree in bioengineering from the University of Genoa, in March 2020, with a focus on pleural line ultrasound videos analysis for computer aided diagnosis in acute pulmonary failure. She is currently a Research Fellow with the IEIIT Institute of Consiglio Nazionale delle Ricerche. She works on data analytics and machine learning topics from different fields, such as industry, healthcare, and automotive. Moreover, her research interests include computer security topics, including covert channels and the Internet of Things.



MAURIZIO AIELLO received the degree in 1994. He worked as a free-lance consultant both for universities and research centre and for private industries. Since August 2001, he has been responsible of CNR network infrastructure. He is currently a Teacher with the University of Genoa and the University College of Dublin. He is a student coordinator, fellowships, and EU projects in the computer security field. His research interests include network security and protocols.



MAURIZIO MONGELLI (Member, IEEE) received the Ph.D. degree in electronics and computer engineering from the University of Genoa (UNIGE), in 2004. The doctorate was funded by Selex Communications S.p.A. (Selex). He worked for both Selex and the Italian Telecommunications Consortium (CNIT), from 2001 to 2010. During his doctorate and in the following years, he worked on the quality of service for military networks with Selex. He was the CNIT Technical Coordinator of a research project concerning satellite emulation systems, funded by the European Space Agency. He spent three months working on the project at the German Aerospace Center, Munich. He is currently a Researcher with the Institute of Electronics, Computer and Telecommunication Engineering (IEIIT) of the National Research Council (CNR), where he deals with machine learning applied to bioinformatics and cyber-physical systems. He is a coauthor of over 100 international scientific articles and two patents.



ENRICO CAMBIASO received the Ph.D. degree in computer science from the University of Genoa. He worked at Ansaldo STS and Selex ES, both companies are part of the Finmeccanica Group. He has a strong background as a computer scientist and he is currently employed as a Technologist with the IEIIT Institute of Consiglio Nazionale delle Ricerche, where he is working on cyber-security topics and focusing on the design of last generation threats and related protection.