IDS-MA: Intrusion Detection System for IoT MQTT Attacks Using Centralized and Federated Learning

Adebayo Omotosho
University of Central Lancashire
Preston, United Kingdom
aomotosho@uclan.ac.uk

Yaman Qendah
University of Passau
Passau, Germany
qendah01@ads.uni-passau.de

Christian Hammer
University of Passau
Passau, Germany
christian.hammer@uni-passau.de

***Abstract*—Yearly, the number of connected Internet of Things (IoT) devices is growing. The attack surface is also increasing because IoT is generally functionality-centric and security is usually an after-thought. Therefore, memory corruption attacks, man-in-the-middle attacks, and distributed denial of service attacks are a few of the attacks that have been widely exploited on these devices communicating via Message Queue Telemetry Transport (MQTT), which is the most commonly used messaging protocol in IoT. However, much of the research on MQTT intrusion detection has either covered a smaller number of attacks, completely ignored memory attacks, or used inadequate classification evaluation metrics (e.g., only accuracy). In this paper, we design and simulate an MQTT IoT network and present IDS-MA, an intrusion detection system for MQTT attacks by training both centralized and federated learning models. Seven different MQTT attacks were implemented with the models evaluated with metrics such as accuracy, precision, and recall. Our evaluation results show high detection scores on MQTT attacks (including memory attacks). We also obtain an average model detection accuracy of over 80% on 2,210,797 real attacks from the MQTT-IoT-IDS2020 benchmark for both centralized and federated models.**

***Index Terms*—Message Queue Telemetry Transport protocol (MQTT), Internet of Things (IoT), intrusion detection, federated learning, centralized learning**

I. Introduction

Nowadays, Internet of Things devices are ubiquitous and have tremendously impacted businesses and our daily lifestyles. For example, in 2020 alone, close to 1 billion IoT home devices were shipped worldwide [1] and the international data corporation forecasted that there will be about 56 billion connected IoT devices by 2025 [2].

IoT devices are generally resource-constrained [3], [4] when compared to conventional PCs, therefore traditional security tools, techniques, and software cannot be directly applied to them [5]. Initially, security features were not usually the priority of IoT product design considerations, which conversely focused on device functionalities [6], [7]. The change in the landscape of attackers that have continually targeted these devices and their networks has called for the prioritization of security. In addition, because attackers and attacks are also evolving, it is increasingly difficult to have a reliable attack mitigation solution [8]. For example, distributed denial of service, man-in-the-middle, and memory corruption attacks are among the top leading threats to IoT devices and networks [9], [10]. These attacks have been exploited on a large scale in the IoT because they are difficult to detect especially when a rogue device masquerades as a benign device. Successful attacks could result in a breach of privacy that could affect both the users and the devices [11].

The Message Queue Telemetry Transport protocol (MQTT) is one of the most popular and widely used data communication protocols by IoT devices in homes and business networks because it is designed to conserve bandwidth and transmits a relatively low number of messages [10], [12], [13]. As a *de facto* standard for IoT device communication, there have been numerous attacks targeting devices using MQTT [9]. Additionally, the protocol itself has design flaws that allow an attacker to send variable length data and exploit memory corruption attacks [10]. This also supports the argument that security concerns were an afterthought in the IoT world. However, IoT devices generate an enormous amount of data while interacting either in device-to-device or device-to-human mode and machine learning has been used successfully to develop solutions in several fields where there is continuous generation of data [14], [15]. Machine learning has likewise shown some promises in general intrusion detection [8], [16]. Hence, machine learning could serve as a go-to alternative for detecting malicious intrusion of IoT devices because it can recognize patterns from previous and similar attack data. The scope of this paper does not cover redesigning MQTT but improving, implementing, and validating machine learning techniques to detect multiple categories of attacks on MQTT.

A. Motivation

The Internet of Things network is one of the most vulnerable networks to external threats resulting from, e.g., a brute force attack, aggressive scan, user datagram protocol (UDP) scan, Sparta SSH attack, denial of service (DoS), man in the middle (MitM), or buffer overflow attack because it allows communication between devices over the Internet [17]. MQTT is a simple, lightweight, message-based, publish/subscribe protocol for transmitting data between IoT devices. It is a client-

server messaging protocol that delivers messages with minimized transport overhead [18]. Being lightweight makes the protocol portable while sacrificing security-related overhead, for example, it disables data encryption by default [9], [18], [19]. Even though there are alternative messaging protocols to MQTT in terms of security, MQTT is preferred because it has the benefits of excelling when there is high latency, and low bandwidth [20]. Technically, the major weakness of the protocol is that it has extraneous fields that are rarely used during communication and this serves as an attractive attack surface, e.g., CVE-2018-8531, CVE-2021-41036, CVE-2020-10071, and CVE-2020-10070 led to buffer overflow exploitation due to MQTT's support for optional and varying packet length [10]. A more renowned real-life example of the attack on connected devices is the Mirai bot which exploited port scanning and authentication attacks to hijack and convert thousands of IoT devices running Linux into attack bots [19]. Since 2016, there have been several variants of Mirai that could exploit different kinds of vulnerabilities in MQTT brokers [9], [21]–[23] such as brokers with no authentication. It is important to develop a mechanism to make the IoT MQTT network more secure, fortunately, IoT security is an open research area currently being addressed by researchers. Nevertheless, whatever the security mechanism that is being adopted, because IoT devices do not receive regular security updates [24], IoT devices' security should be proactive and guaranteed, without having to modify the device. Therefore in this research, the development of intrusion detection systems for detecting different MQTT-related attacks using centralized machine learning and federated learning is investigated. The contributions of our paper are:

- We perform a detailed experiment using data collected from our simulated network and introduced three additional MQTT attacks (MitM, DoS, Buffer Overflow) to the four attacks contained in MQTT-IoT-IDS2020 to develop more robust models for MQTT attack detection.
- We show the capability of binary and multi-class classifiers to detect MQTT attacks with precision and recall scores of about 80%. Unlike many of the existing MQTT attack papers which report only general model accuracy (e.g., [25], [26]), we further show the accuracy, precision, and recall for detecting individual attacks. Our model performance is significant in both the majority and minority classes.
- The results we obtained from both federated and centralized models' performance on the MQTT-IoT-IDS2020 benchmark dataset show that our models learn and can generalize on a different dataset. On the MQTT-IoT-IDS2020 dataset, we obtained precision and recall scores up to 85% and 87% respectively in federated learning, and the highest score for the centralized models is 100%.
- Our model evaluation result using 10-fold cross-validation showed that *buffer overflow memory corruption attacks* in IoT MQTT communication could be detected with precision and recall of about 87%. Existing works using network traffic data have average detection precision and recall of 50% and 22%, respectively for this category of attacks [27], [28].
- The comparison of federated machine learning (using random forest) with federated deep learning showed that both can perform well in detecting MQTT attacks albeit federated deep learning converges faster, has lower overhead, and is capable of continuous learning.

TABLE I
COMPARSION WITH EXISTING MQTT WORKS (A)

Attacks	*this paper*	[29]	[30]	[31]	[32]	[17]	[33]
Bruteforce	✓	✗	✗	✗	✗	✓	✓
Aggressive scan	✓	✗	✗	✗	✓	✓	✓
UDP scan	✓	✗	✗	✗	✓	✓	✓
Sparta ssh scan	✓	✗	✗	✗	✓	✓	✓
DoS	✓	✗	✓	✓	✗	✗	✗
MitM	✓	✓	✗	✗	✗	✗	✗
Bufferoverflow	✓	✗	✗	✗	✗	✗	✗

TABLE II
COMPARSION WITH EXISTING MQTT WORKS (B)

Author	Evaluation score	ML Method	Training set	Benchmark set
this paper	individual	federated and centralized learning	Experiment	MQTT-IoT-IDS2020
[29]	cumulative	centralized learning	Experiment	-
[30]	cumulative	centralized learning	Experiment	-
[31]	cumulative	-	Experiment	-
[32]	cumulative	centralized learning	MQTT-IoT-IDS2020	MQTT-IoT-IDS2020
[17]	cumulative	centralized learning	MQTT-IoT-IDS2020	MQTT-IoT-IDS2020
[33]	cumulative	federated learning	MQTT-IoT-IDS2020	MQTT-IoT-IDS2020

Threat model: Our system is targeted at detecting different MQTT attacks using network traffic data. It is assumed that at least one of the MQTT publisher-subscriber network devices such as the broker or subscriber has exploitable weaknesses like opened ports, weak authentication, or vulnerable C language codebase. When the federated model converges, all the devices have an updated shared model. Then, if one of the devices that has access to *MQTT topic(s)* becomes malicious and is masquerading as a legitimate publisher or subscriber to compromise other clients (e.g., by sending a memory bug payload), the network traffic data fed into our model can automatically detect the attack on-device. This results in classifying a communication as either an attack (1) or benign (0). Because learning is continuous, knowledge of the threat is updated on the devices and shared globally for up-to-date network protection.

II. RELATED WORK

This section briefly summarizes the state of the art concerning MQTT attacks.Table I and II summarize how our work is different from existing works.

Communication Protocols: The MQTT protocol is the most used communication protocol in resource-constrained devices [31] because it has superior energy efficiency, resource usage, and Quality of Service (QoS) when compared to alternative IoT communication protocols like Hypertext Transfer Protocol, Constrained Application Protocol, Extensi-

ble Messaging and Presence Protocol, and Advanced Message Queuing Protocol [18], [20], [34], [35]. However, in the last 8 years, at least 81 MQTT protocol-related vulnerabilities relating to memory attack and DoS, were reported in the National Vulnerability Database and Common Vulnerabilities and Exposures database [10]. Therefore in this paper, the focus is on attack detection in IoT interactions using the MQTT protocol.

Attack Detection on MQTT: The DoS attack detection mechanism for MQTT protocol was proposed in [31] but our IDS-MA contains seven different categories of attacks. To mitigate attacks, [10] addresses MQTT protocol modification, our approach does not involve making changes to how MQTT processes information but uses artificial intelligence algorithms to learn and detect attacks without a protocol modification. [18] identified only the MQTT attack scenario, our work includes the presentation of the detection of attacks and not just identification. [32] used MQTT-IoT-IDS2020 to test their approach without any simulation or data collection, in our work, we created an experimental testbed and use data from communicating devices to develop our models. We only use the MQTT-IoT-IDS2020 dataset for validating our result. [30] also developed an intrusion detection system called ARTEMIS for MQTT attacks by comparing six machine learning algorithms. The evaluation was only based on accuracy metrics on a smaller dataset. Many machine learning models are built on the assumption that classification data is balanced but in reality, classes often imbalance as attack traffic could be less than normal traffic. Therefore an exaggerated high accuracy score based on the majority class would lead to a poor classification of other classes. For example, a classifier with an accuracy of 99% may not perform better than a purely random classifier. In addition, there was no information on the kind of attacks detected. In our work, we evaluated our classifier not just on accuracy but the precision and recall metrics which reflect the true capabilities of the classifiers. We also clearly showed the detection scores of *individual* attacks that were detected by our models.

An approach similar to ours, but only focused on deep learning, was by [17] which provided general detection scores of different models on four attacks without providing the details on the detection of individual attacks. No experiment or simulation of the MQTT network was done but only an evaluation of models built on the existing MQTT-IoT-IDS2020 dataset. In our work, we performed experiments on our simulated IoT network and collected data in the process which were used to develop centralized and federated learning models. [33] uses federated learning but presented only simple results from the MQTT-IoT-IDS2020 evaluation. Our paper presents centralized and federated models (federated ML and federated DL), attacks, and detailed performance of the model as well as MQTT-IoT-IDS2020 validation scores.

One of the challenges that we highlighted earlier was the misleading performance of the existing model which does not usually generalize well on all attacks used in training, resulting in the minority classes either poorly detected or undetected [25], [36], [37]. In our work, we showed the precision, recall, and accuracy of our model on each of the seven attacks we considered.

Centralized vs. Federated Learning: The centralized machine learning approach is the costly traditional approach where data from multiple IoT devices is aggregated and then used to train a single model once [38]. Because the volume of the aggregated data is typically large, model training usually takes more time, and dedicated storage/servers might be required to reduce the interval between the training time and the release of an updated model. Another challenge of this approach is privacy because confidential data could be moved between devices. Data could be exposed to different kinds of attacks in transit, especially when using the cloud [39]. In federated learning, training is decentralized and is done on the IoT devices which collaboratively learn a shared prediction model while each IoT device keeps its training data [40]. Typically, each device continually trains a model and sends small incremental updates or parameters (instead of huge data) to the server for aggregation which is then used to create a shared consolidated model. Federated learning has the advantage of having a realistically up-to-date model while maintaining data privacy [41]. In this paper, we employed the federated learning approach to train a decentralized model for detecting *multiple attacks* on the MQTT protocol.

III. Planning and Execution Method

This section presents the development of the MQTT protocol intrusion detection system (IDS), the attack techniques, data processing and model development, and the evaluation strategy.

Fig. 1 is a simplified workflow of our IDS-MA that depicts the different stages of our methodology, e.g., the IoT network, different forms of packets, packets processing, model development, deployments, and evaluation.

To collect the data needed for building our IDS, we constructed the IoT communication network described in Fig. 2 consisting of a broker, seven sets of sensors/clients (publishers and subscribers) where one of the clients is malicious. An MQTT client act as either a subscriber (receiver) or a publisher (sender). The broker is a central server that handles the published and subscribed messages. Normally, each subscribed or published message has to be under a specific topic and every subscriber or publisher has to either subscribe or publish to a specific topic to be able to receive or send messages to the MQTT broker. The IoT devices were simulated using virtual machines running Linux operating systems. The MQTT broker device is based on Mosquitto v3.1.1 with activated authentication, which means that whenever a client wants to connect to the broker, a username and password are required.

In a real-life scenario, data generated by sensors is sent from the publisher to the subscriber(s). To satisfy this in our simulation, the client devices continuously exchange data received from temperature and humidity sensors[1] , smartphone

[1]https://www.kaggle.com/datasets/edotfs/dht11-temperature-and-humidity-sensor-1-day

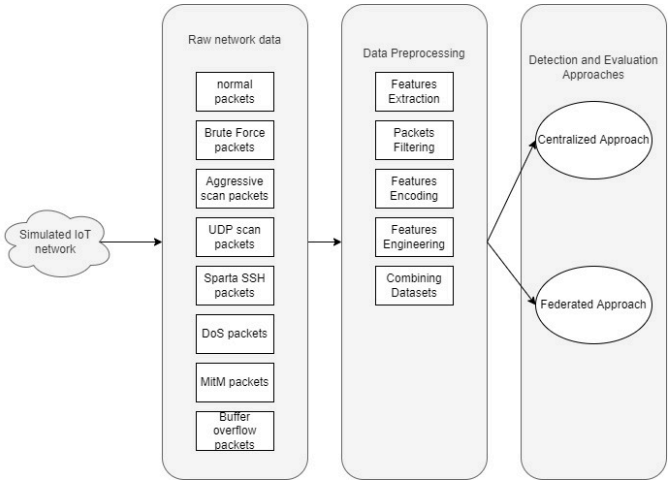


Fig. 1. IDS-MA workflow

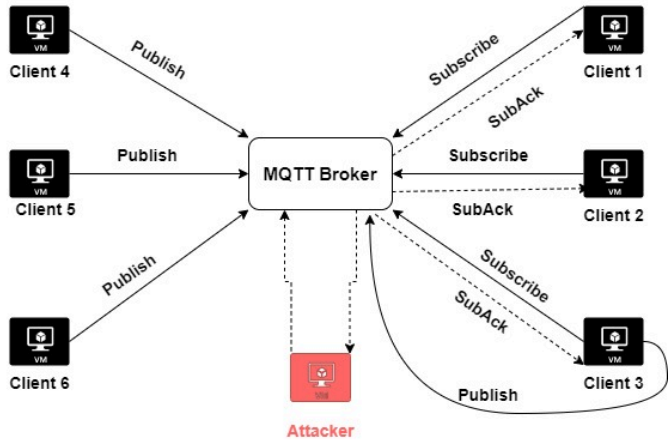


Fig. 2. Our IoT MQTT Network

motion sensors[2] , and the temperature and light sensors[3] . The malicious device injects seven different classes of attacks into the network to compromise communication. These attacks are listed in Table III alongside their affected security goals.

A. Attacks description

A brief description of the attacks used in our study is as follows:

MQTT Brute Force: MQTT protocol uses a centralized broker to communicate between clients. Those brokers can define a basic authentication mechanism in the form of a username/password pair. The authentication in our simulated broker is active and the Static Program Analysis for Reliable Trusted Apps `sparta` tool was used to perform credential brute forcing on the broker e.g., via a set of compromised usernames and passwords.

Aggressive Scan: Network Mapper (Nmap) tools can be used for network discovery security auditing. It has a special flag to activate aggressive discovery, namely -A. Aggressive mode enables OS discovery (-O), version detection (-sV), script scanning (-sC), and traceroute (–traceroute). This attack generates only few packets, so a bash script was written to execute the attack multiple and random times to get more packets for the IDS. The command used to execute attack:

`nmap` *–A victim-ip-address*

User Datagram Protocol (UDP) scan: This attack works by sending UDP packets to the victim's port to check if the service is running. This was accomplished using the Nmap command with the flag -U activated. Also, this attack generates few packets, so a bash script is written, to execute this attack multiple times. The command used is as follows:

`nmap` *-U victim-ip-address*

Sparta SSH brute-force: It is also possible that brokers have the SSH service activated (e.g., with an open port 22) to allow remote access. An attacker could exploit this and `sparta` is one of the tools that could be used to scan, check, brute force and attack the broker if the SSH service is running.

Man in the Middle: Eavesdropping was achieved using *Ettercap* tool which is capable of sniffing live connections and content filtering on the fly. It supports active and passive dissection of many protocols and includes many features for network and host analysis. The command used to execute attack:

`sudo timeout 15 ettercap -T -i ens33 -M arp:remote` */first-victim// /secondvictim//*

Denial of Service (DoS): DoS attacks aim to send a huge number of packets to the victim, which cause the victim's device to get overwhelmed by received packets while preventing the victim from being able to deliver its intended service correctly

Distributed Denial of Service (DDoS): This attack is very similar to the DoS attack, but instead of using one machine to send the packets, the attacker could use multiple machines. In this work, the Dos attack was executed using multiple fake IP addresses.

Syn flooding DoS: This is a type of DoS attack, that aims to make a server unavailable to legitimate traffic by consuming all available server resources. By repeatedly sending initial connection request (SYN) packets, the attacker can overwhelm

TABLE III
ATTACKS SECURITY GOALS

Attack	Affected security goals
MQTT Brute Force	Confidentiality
Aggressive scan	Confidentiality
UDP scan	Confidentiality
Sparta SSH brute-force	Confidentiality
Man in the middle	Confidentiality, Integrity and Availability
Denial of Service (DoS), Distributed Denial of Service (DDoS) and Syn flooding DoS	Availability
Buffer overflow	Confidentiality, Integrity and Availability

[2]https://www.kaggle.com/datasets/malekzadeh/motionsense-dataset
[3]https://www.kaggle.com/datasets/bjoernjostein/temperature-and-light-data

TABLE IV
TARGET FEATURES CATEGORIES

Target Feature	Classification	Categories
is_attack	Binary	Not an attack= 0, An attack = 1
attack_type	Multi	Normal=0, MQTT brute force=1, Agressive scan=2, UDP scan=3, SPARTA scan=4, DoS=5, MitM=6

all available ports on a targeted server machine, causing the targeted device to respond to legitimate traffic sluggishly or not at all. DoS, DDoS, and Syn flooding are all implemented but since the base behavior is similar, the data for the three are combined into one category as *DoS* attack. `hping3` tool was used to execute the attack as follows:

`hping3` -d 50 -p 1883 -S –`flood` victim-ip-address

Buffer overflow: IoT devices' firmware is largely written in the C language and this attack exploits the inherent weakness of the language. As stated in the motivation section, MQTT is highly susceptible to this kind of attack but existing works on MQTT excluded detection of this kind of attack. By sending an input that writes beyond the boundary of a variable and then hijacking the control flow, an attacker could send new commands to devices in the network or run a shell script and can even gain root access to the broker or subscriber. Examples of exploits to maliciously call a *vulnerable_function* function in the broker or subscriber from a rogue device is:

(`python3` -c "`print` ('B'*88+'\x1e\x55\x55\x55\x55\x55 \x00\x00')") | *./vulnerable_function*

where **\x1e \x55 \x55 \x55 \x55 \x55 \x00 \x00** is the address of the new instructions an exploited device is redirected to.

In another scenario, the rogue device corrupts the memory to run a shell script to gain root access e.g.,

(`python3` -c "`print`('B'*63+'\x31\xc0\x50\x68\x2f\x2f \x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x89\xe2\x53\x89 \xe1\xb0\x0b\xcd\x80'+ '\x1e\x55\x55\x55\x55\x55\x00\x00') ") | *./vulnerable_function*

Each of these attacks was executed at different times from the rogue device, and the packets generated were collected, filtered, and saved in a CSV file using the Wireshark[4] network protocol analyzer. The network traffic is observed during normal communication and the traffic data is collected. When each attack is triggered the corresponding traffic data was equally collected and analyzed. The Wireshark network tool was used to gather the packets communicate in the network.

B. Data Pre-Processing

The packet data associated with the described attacks was collected using the Wireshark network tool while the different components of the system communicate. The network traffic is observed live during normal communication and the traffic data is collected. When each attack is triggered the corresponding traffic data were equally collected and analyzed. The collected packet features were preprocessed and engineered to remove redundant fields and select relevant features e.g., features such as Time stamp *timestamp*, Source IP *src_ip*, and Destination IP *dst_ip* were dropped because these features vary from one attack to another and we do not want our model to be tailored for a fixed scenario. The packet features were encoded, as shown in Table VI. Two additional target features were added to our dataset, one is used for the binary classification and labeled *is_attack*, and the other is used for the multi-classification and labeled *attack_type*. The categories of these features are presented in Table IV.

[4]https://www.wireshark.org

TABLE V
SIZE OF THE DATA COLLECTED FOR EACH ATTACK

Attack	Datasize
MQTT Brute Force attack	1,048,563
Aggressive scan attack	502,927
UDP scan attack	22,359
Sparta SSH brute-force attack	32,405
MitM attack	7,000
DoS	288,657
Buffer Overflow attack	9,235

The shapes of the data collected within the normal communication and attack are 1,704,747 * 30 and 1,911,146 * 30 respectively. Table V shows the extracted data shape for each attack type that was collected from our IoT MQTT network that we want to correctly classify. After feature engineering, we obtained a dataset with the shape 3,319,571 * 30.

C. Model validation dataset

There is a variety of datasets that have been proposed for a intrusion detection system. A number of them, such as the popular KDD-99, and NSL-KDD[5], are for general computer networks and do not target a specific IoT or network protocol, whereas MQTT-IoT-IDS-2020 and MQTTset [26] are examples of datasets that represent specific types of specialized devices, networks, or protocols that are not usually available in the general network. In this paper, we validated our models using the MQTT Internet of things intrusion detection dataset (MQTT-IoT-IDS2020), which is the first and the most detailed special purpose dataset for MQTT attacks [17], [32], [42]. This benchmark data set was collected from an IoT MQTT network consisting of 1 broker, 12 sensors, a camera, and a rogue device [32]. It contains four categories of attacks such as *MQTT brute force, aggressive scan, UDP scan, Sparta ssh Brute force*. This paper collects and includes datasets from three additional attacks. For the validation of *DoS and MitM* attacks on MQTT protocol we merged another public MQTT DoS/MitM [6] dataset. Our MQTT-IoT-IDS2020 dataset also has over 2 million records (shape - 2,210,797 * 30) of MQTT attacks and our validation is only based on the packet features abstraction level in the dataset.

Our captured packet features were also structured based on the MQTT-IoT-IDS2020. The different attack categories and how they were exploited have been discussed earlier in Section III-A.

[5]https://www.unb.ca/cic/datasets/index.html
[6]https://joseaveleira.es/dataset

TABLE VI
FEATURES ENCODING

Feature	Encoding	Categories
timestamp	float	-
scr_ip	category	-
dst_ip	category	-
protocol	category	*TCP = 1, MQTT = 2, MP2T = 3, DATA = 4, UDP = 5, DN = 6 ,MPEG_PMT = 7, MPEG_PAT = 8, DVB_SDT = 9, SSH = 10, RADIUS = 11, MDNS = 12, PORTMAP = 13, Portmap = 13, NFS = 14, ECHO = 15, NAT − PMP = 16, NTP = 17, NBNS = 18, SNMP = 19, SRVLOC = 20, CLDAP = 21, ISAKMP = 22, DTLS = 23, RIP = 24, XDMCP = 25, ARP = 26, None = −1*
ttl	category	-
ip_len	float	-
ip_flag_df	category	*Notset = 0, Set = 1, None = −1*
ip_flag_mf	category	*Notset = 0, Set = 1, None = −1*
ip_flag_rb	category	*Notset = 0, Set = 1, None = −1*
src_port	category	-
dst_port	category	-
tcp_flag_res	category	*NotSet = 0, Set = 1, None = −1*
tcp_flag_ns	category	*NotSet = 0, Set = 1, None = −1*
tcp_flag_cwr	category	*NotSet = 0, Set = 1, None = −1*
tcp_flag_ecn	category	*NotSet = 0, Set = 1, None = −1*
tcp_flag_urg	category	*NotSet = 0, Set = 1, None = −1*
tcp_flag_ack	category	*NotSet = 0, Set = 1, None = −1*
tcp_flag_push	category	*NotSet = 0, Set = 1, None = −1*
tcp_flag_reset	category	*NotSet = 0, Set = 1, None = −1*
tcp_flag_syn	category	*NotSet = 0, Set = 1, None = −1*
tcp_flag_fin	category	*NotSet = 0, Set = 1, None = −1*
mqtt_messagetype	category	*0 = 0, ConnectCommand = 1, ConnectAck = 2, PublishMessage = 3, SubscribeRequest = 12, SubscribeAck = 12, DisconnectRequest = 13, PingRequest = 14, PingResponse = 14, None = −1*
mqtt_messagelength	float	-
mqtt_flag_uname	category	*NotSet = 0, Set = 1, None = −1*
mqtt_flag_passwd	category	*NotSet = 0, Set = 1, None = −1*
mqtt_flag_retain	category	*NotSet = 0, Set = 1, None = −1*
mqtt_flag_qos	category	*NotSet = 0, Atmostoncedelivery = 1, Atleastoncedelivery = 2, Exactlyoncedelivery = 3*
mqtt_flag_willflag	category	*NotSet = 0, Set = 1, None = −1*
mqtt_flag_clean	category	*NotSet = 0, Set = 1, None = −1*
mqtt_flag_reserved	category	*NotSet = 0, Set = 1, None = −1*

D. Model development

The primary languages used for our experiments are Python, C, and bash. Model development was made using several libraries such as Pandas (v1.3.5), NumPy (v1.19.5), TensorFlow (v2.5.3), Tensorflow_federated (v0.19.0), Keras (v2.8.0), Sklearn (v1.0.2), and Tensorflow_decision_forests (v1.0.1). We implemented and evaluated both the centralized and decentralized (using federated learning) approach for detecting MQTT attacks. Five conventional machine learning algorithms and one deep learning algorithm were evaluated. These are Decision Tree (DT), Random Forest (RF), Logistic Regression (LR), Linear Discriminant Analysis (LDA), Gaussian Naive Bayes (NB), and Deep Neural Network (DNN).

1) Centralized approach: In this approach, each model is trained once and then deployed. We developed two variants of each centralized model, in one form as a binary classifier and the other as a multi-class classifier. For binary classification, each target feature classifies the packet as an attack or not. The target feature is `is_attack`. In the multi-class classification, the models are trained to detect each attack individually, and the encoding for each target feature is presented in Table IV. Because the classes in our dataset were not balanced, and to reduce model bias, we used the *stratified shuffle split* resampling technique to ensure that each attack type is equally present in each of our sampling corpora. We use 10-fold cross-validation, that is, 10% of the data is used for testing, which is a standard procedure to ensure the validity of the learned classifiers.

2) Decentralized approach: In this approach, the model is trained incrementally on each client and then consolidated into a shared global model. This was developed using *TensorFlow Federated* [7] TFF framework in *Google Collab*[8] .

In decentralized learning using federated learning, we want to minimize the number of round trips for exchanging model updates between our client and a central server. Therefore, we implemented the federated averaging *FedAvg* algorithm that minimizes the objective function $\mathrm{f}(w)$, such that:

$$\mathrm{f}(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad \mathrm{F_k}(w) = \frac{1}{n_k} \sum_{i \in P_k}^{m} f_i(w) \quad (1)$$

w is the weight of model updates or parameters, K is the number of clients or devices over which the data is partitioned, k is the index of a device, F_k is the local objective function for the $k_t h$ device, P_k is the set of indexes of datapoints on device k, and n_k is the number of data samples that are available during training of device k. Although there are other techniques such *FedACNN* [27], *Per-FedAvg* [43] for federated learning convergence and minimizing the objective function, *FedAvg* is the most popular [44].

We also implemented TFF for binary and multi-class classification. A different number of clients were sampled for collaborative learning and for this purpose we experimented with 10 to 40 clients in the federated learning model. However, due to the limited resources (e.g., RAM and thread) supported by Google Collab[9], the training was done in multiple phases, each training phase trained by datasets from 5 clients, which is the maximum number supported by Google Colab.

Algorithm 1 simplifies and formalizes the training and prediction procedure in our method that results in centralized and decentralized models m_c and m_g respectively. The models are trained with different machine learning algorithms $train_\alpha$ using our IDS-MA dataset β_1 and validated with the MQTT-IoT-IDS2020 dataset β_2. The p_c and p_g are the predictions for different attack classes in the centralized and decentralized model respectively.

[7]https://www.tensorflow.org/federated
[8]https://research.google.com/colaboratory/faq.html
[9]https://colab.research.google.com/

Algorithm 1: IDS-MA Algorithm

Input: Train and test data of IDS-MA MQTT data
Output: MQTT attack classification
Data: IDS-MA(β_1), MQTT-IoT-IDS2020(β_2)

$\sigma = \delta_1 \ldots \delta_{40}$ /* devices' data */
$\alpha = \alpha_1 \ldots \alpha_6$ /* machine learning algorithms */
def idsma_training(m_g, α, σ):
  while *!convergence* **do**
    $send\_to\_devices(m_g)$
    for δ_i *in* σ **do**
      $m_i = train_\alpha(\delta_i)$
      $send\_to\_central\_server(m_i)$
    /* update server model, m_g */
    /* by averaging parameters of models on all devices */
    $m_g = FedAvg(m_i, \ldots, m_{40})$
  $m_c = train_\alpha(\sigma)$
  return m_g, m_c
def predict_attacks(m_g, m_c, $class$):
  $p_g = m_g(\beta_1, \beta_2, class)$
  $p_c = m_c(\beta_1, \beta_2, class)$
  return p_g, p_c

E. Evaluation metrics

Our models' performance evaluations were based on the following research questions.

RQ1: At what accuracy, precision, and recall can centralized learning and federated learning detect intrusion on the MQTT protocol?
RQ2: How well can the models perform over different attack categories on the MQTT protocol?
RQ3: Can we improve the detection of memory corruption attacks (e.g., buffer overflows) on MQTT protocol using packet data?

To answer the first research question, our classifiers' performance was measured using standard machine learning classification metrics such as accuracy, precision, recall, and F1-score. In the metrics definition TP, FP, TN, and FN refer to true positive, false positive, true negative, and false negative respectively.

Accuracy represents the overall rate of the positive and negative predictions made by the classifier. Accuracy is defined by the equation:

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \tag{2}$$

Precision measures the accuracy of the positive predictions when classifying the data instances. The precision metric is defined by:

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

Recall is also known as true positive rate. it is the ratio of positive instances that are correctly predicted by the classifier.

TABLE VII
CENTRALIZED MODELS (AS BINARY CLASSIFIERS) PERFORMANCE

Metrics	DT	RF	LDA	NB	LR	DL
Accuracy	86.87	87.25	76.59	75.84	78.14	86.23
Precision	87.00	87.28	77.23	79.16	78.08	80.16
Recall	87.07	87.38	76.97	74.95	78.13	85.29
F1-score	86.87	87.24	76.57	77.00	78.09	82.65

TABLE VIII
CENTRALIZED MODELS (AS MULTI-CLASS CLASSIFIERS) PERFORMANCE

Attacks	Metrics	DT	RF	LDA	NB	LR	DL
Brute Force	Accuracy	88.29	89.22	84.42	73.38	84.79	81.91
	Precision	88.75	89.27	83.02	75.49	83.15	80.95
	Recall	88.49	88.21	81.62	70.43	82.64	80.00
	F1-score	88.62	88.74	82.32	72.87	82.90	80.47
Aggressive Scan	Accuracy	100.00	100.00	97.61	68.51	97.86	88.61
	Precision	100.00	100.0	98.30	67.387	98.64	80.00
	Recall	100.00	100.00	94.93	72.81	95.32	83.57
	F1-score	100.00	100.00	96.49	69.99	96.68	81.74
UDP Scan	Accuracy	100.00	99.99	99.96	56.51	99.96	98.72
	Precision	100.00	99.99	99.97	51.40	99.98	89.36
	Recall	100.00	99.95	98.72	77.53	98.72	90.00
	F1-score	100.00	99.97	99.03	38.62	99.34	89.67
Sparta ssh scan	Accuracy	100.00	100.00	98.88	45.59	98.90	98.12
	Precision	100.00	100.00	99.43	51.66	99.19	89.60
	Recall	100.00	100.00	90.31	72.27	90.95	90.00
	F1-score	100.00	100.00	94.60	34.04	94.21	89.82
DoS	Accuracy	99.62	99.60	95.10	54.40	95.45	85.42
	Precision	99.77	99.72	97.99	62.10	89.87	82.71
	Recall	98.71	98.69	95.05	73.31	92.61	84.35
	F1-score	99.23	99.19	91.02	51.30	91.17	83.52
MitM	Accuracy	100.00	100.00	100.00	100.00	100.00	100.00
	Precision	100.00	100.00	100.00	100.00	100.00	100.00
	Recall	100.00	100.00	100.00	100.00	100.00	100.00
	F1-score	100.00	100.00	100.00	100.00	100.0	100.00
Buffer Overflow	Accuracy	89.99	89.99	88.00	47.78	89.67	89.68
	Precision	89.99	89.99	82.43	50.16	89.83	83.59
	Recall	89.90	89.90	83.82	52.23	89.48	81.89
	F1-score	89.95	89.95	83.85	46.15	89.91	82.55

The higher the recall, the more positive samples detected. The recall is defined by the equation:

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

F1-score metric measures the harmonic mean of precision and recall. F1-Score is defined by the equation:

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5}$$

The second research question is answered by comparing the evaluation metric for different categories of attacks to see if our models could generalize on the considered of attacks. For the third research question, we answered by measuring how well our model can correctly classify memory corruption attacks using precision and recall scores. This category of attacks has proven difficult to detect in much of existing works relying network packet features [28], [36], [45].

IV. DISCUSSION OF RESULTS

RQ1: *At what accuracy, precision, and recall can centralized learning and federated learning detect intrusion on the MQTT protocol?*
The goal is to determine, using classification metrics, the performance of the models. Table VII and Table VIII depict the performance scores of our *centralized models* trained as binary and multi-class classifiers based on the performance evaluation

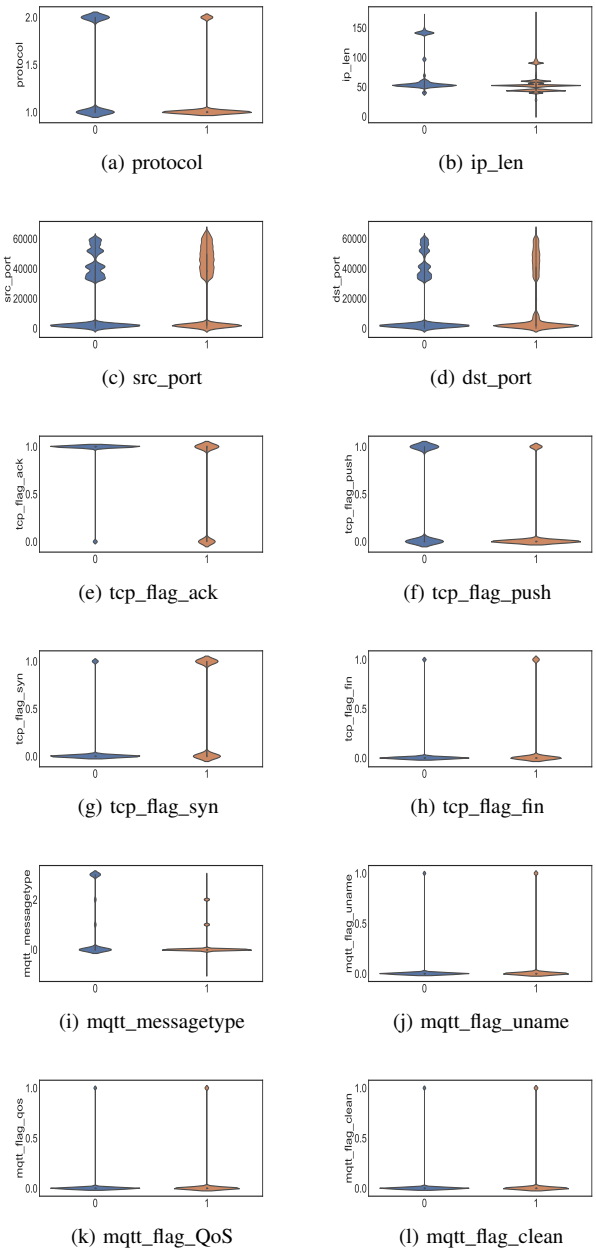
TABLE IX
CENTRALIZED MODELS (AS MULTI-CLASS CLASSIFIERS) VALIDATION ON MQTT-IOT-IDS2020

Attacks	Metrics	DT	RF	LDA	NB	LR	DL
Brute Force	Accuracy	76.00	75.77	76.64	71.83	71.91	74.24
	Precision	76.94	76.97	76.67	78.30	72.13	73.55
	Recall	76.01	75.79	76.64	71.87	71.92	74.19
	F1-score	76.47	76.38	76.59	74.95	71.74	70.68
Aggressive Scan	Accuracy	92.48	93.90	95.20	76.67	98.12	88.13
	Precision	94.70	85.08	88.43	79.98	99.04	80.00
	Recall	85.58	95.64	93.39	79.88	94.16	89.04
	F1-score	89.82	88.71	90.57	74.78	92.77	84.28
UDP Scan	Accuracy	93.79	93.79	99.12	54.55	94.36	97.90
	Precision	92.61	92.61	85.22	52.20	93.55	88.95
	Recall	96.82	96.82	97.11	76.78	92.84	90.00
	F1-score	91.51	91.51	91.10	39.10	91.84	89.47
Sparta ssh scan	Accuracy	96.96	97.04	91.20	60.38	97.25	99.69
	Precision	90.05	94.69	90.72	50.05	92.52	89.84
	Recall	94.30	98.51	88.89	54.30	91.92	90.00
	F1-score	87.99	97.82	89.23	37.99	92.00	89.92
DoS	Accuracy	99.69	99.58	95.06	54.30	95.39	85.51
	Precision	99.81	99.74	87.91	61.86	89.811	82.75
	Recall	98.93	98.59	94.77	72.99	92.25	89.84
	F1-score	99.37	99.15	90.87	51.11	90.97	86.09
MitM	Accuracy	99.99	99.96	99.63	99.63	99.63	91.01
	Precision	99.94	99.98	100.0	100.0	100.0	89.80
	Recall	99.99	95.16	100.0	100.0	100.0	85.67
	F1-score	99.97	97.44	100.0	100.0	100.0	87.64

metrics. Our models' performance is based on 10-fold cross-validation. In Table VII the machine learning model with the highest precision and recall is the Random Forest (RF) with about 87% precision and recall scores respectively, while the deep learning model has precision and recall of about 80% and 85% respectively. The models' performance metrics were better in multi-class classification as shown in Table VIII. For the seven categories of attacks, our model performance in terms of precision and recall scores is high. Except for NB, model scores significantly improve, for example, it was possible to obtain 100% precision and recall for MitM, Sparta ssh scan, UDP Scan, and Aggressive Scan. To validate the centralized model and check if it generalizes well, we evaluated the performance using the MQTT-IoT-IDS2020 containing over 2 million attack records dataset and we also recorded high-performance metrics scores shown in Table IX. This shows that our models generalize well on the different attacks.

The distribution of some of the classification features used by our models to distinguish the normal and attack communication is shown in Fig. 3 by violin plots. They are a combination of TCP and MQTT flags because MQTT runs on top of TCP/IP for reliable message delivery. The values 0 and 1 indicate feature distribution in the normal and attack communication respectively. Fig. 3a to Fig. 3l are the distributions for protocol, packet length, source port, destination port, TCP acknowledgement flag, TCP push flag, TCP synchronization flag, TCP finish flag, MQTT message type, MQTT user name flag, MQTT QoS level flag, and MQTT clean session flag respectively.

In the decentralized approach using federated learning, we tested our federated learning models (binary classification) with multiple clients and we obtained model performance scores in Table X. The federated learning model's precision and recall were both about 83% and this performance was not



(a) protocol (b) ip_len (c) src_port (d) dst_port (e) tcp_flag_ack (f) tcp_flag_push (g) tcp_flag_syn (h) tcp_flag_fin (i) mqtt_messagetype (j) mqtt_flag_uname (k) mqtt_flag_QoS (l) mqtt_flag_clean

Fig. 3. Violin plots of the features for the normal (0) and attack (1) communications

significantly affected as the number of clients grows from 10 to 40. This was also equally validated using MQTT-IoT-IDS2020 although the *performance slightly dropped* we still obtained precision and recall as high as 80% and 81% respectively.

Training federated deep learning takes less time than federated random forest, e.g., in Fig. 4, which plots the logarithmic training time, federated random forest algorithm took over 42 minutes to train with just 10 clients whereas just 37 seconds with federated deep learning. Federated deep learning also takes a less total training time than the centralized deep learning model as shown in Fig. 5. In addition, during detection, federated random forest shows an additional 24% classification

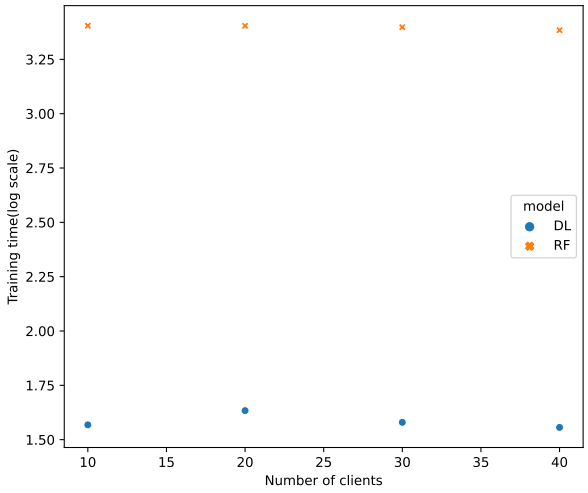


Fig. 4. Training time of federated random forest vs. federated deep learning

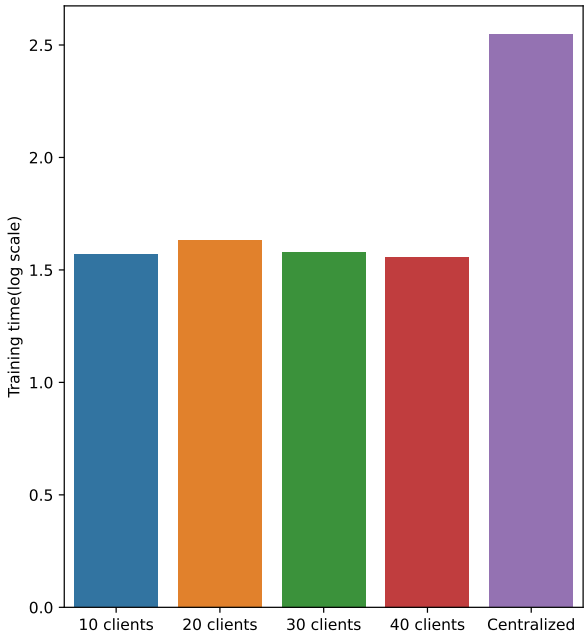


Fig. 5. Training time of federated deep learning vs. centralized deep learning

time overhead compared to federated deep learning on the MQTT-IoT-IDS2020 dataset.

RQ2: *How well can the models perform over different attack categories on the MQTT protocol?*

Existing works on MQTT intrusion detection generally do not provide individual attack detection scores which makes it difficult to know how well the classifier generalizes on different attacks. From the result of our models' cross-validation, both random forest and decision tree outperform the other models' detection scores for all the seven attacks considered. The multi-class results in Table VIII show that on average both RF and DT can detect *brute force*, *aggressive scan*, *UDP scan*, *sparta ssh scan*, *DoS*, *MitM*, and *buffer overflow* with approximate precision between 89 - 100%. To confirm that these numbers were not exaggerated, on the MQTT-IoT-IDS2020 dataset *brute force*, *aggressive scan*, *UDP scan*, *sparta ssh scan*, *DoS*, and *MitM* were detected with approximate precisions between 76-100%. These results are significantly close to the actual model performance on the different attack categories.

TABLE X
FEDERATED MODELS (AS BINARY CLASSIFIERS) PERFORMANCE AND VALIDATION ON MQTT-IOT-IDS2020

Evaluation dataset	Clients	Metrics	DL	RF
Model performance	10	Accuracy	82.62	85.20
		Precision	82.63	85.02
		Recall	89.97	84.82
	20	Accuracy	82.65	84.29
		Precision	82.68	84.67
		Recall	89.89	84.22
	30	Accuracy	82.59	86.00
		Precision	82.59	85.89
		Recall	90.02	86.04
	40	Accuracy	82.81	86.22
		Precision	82.81	86.76
		Recall	88.24	86.69
MQTT-IoT-IDS2020 dataset	10	Accuracy	80.79	78.20
		Precision	76.57	78.01
		Recall	76.57	78.66
	20	Accuracy	80.95	78.76
		Precision	78.57	78.52
		Recall	78.57	78.40
	30	Accuracy	80.00	79.23
		Precision	81.57	79.10
		Recall	80.57	79.00
	40	Accuracy	80.94	79.26
		Precision	79.57	79.24
		Recall	80.25	79.03

RQ3:*Can we improve the detection of memory corruption attacks (e.g., buffer overflows) on MQTT protocol using traffic data?*

Except for the Naive Bayes classifier, our models have average precision and recall of about 87% which implies they can correctly detect positive memory attacks with an accuracy of 87% and that the predictions are right 87% of the time. Furthermore, as of when this study was carried out, we are not aware of any MQTT dataset that has memory attack data probably because of the difficulty of implementing it even though it is commonly exploited [10]. However, our detection score for memory attacks is significant because some of the past works that have used general network datasets for IoT intrusion detection (e.g., NSL-KDD) have recorded as low as 0% precision and recall despite an accuracy of 99.95% [28]. This also justifies the incompleteness of accuracy metrics in assessing classifiers. Despite being one of two minority classes in Table V, it appears that MQTT traffic data is more efficient at detecting this category of attack in IoT networks.

TABLE XI
EVALUATION RESULT COMPARSION WITH EXISTING MQTT WORKS

Author	#Attack	Accuracy	Precision	Recall	#ML
this paper	7	0.86	0.84	0.85	6
[29]	1	-	-	-	1
[30]	4	0.84	-	-	6
[31]	1	-	0.86	0.81	-
[32]	3	0.75	0.72	0.75	6
[17]	4	0.91	0.89	0.82	1
[33]	4	0.94	0.93	0.93	1

In Table XI we compare the average accuracy, precision, and recall over all the six algorithms used for federated and centralized learning with results obtained from other recent MQTT intrusion research. If we have reported only the average of our best algorithms such as *RF* and *DT*, of course, our overall performance scores would be over 90%. Nonetheless, this comparison shows that our MQTT network design closely reflects the behavior of a typical MQTT network based on the performance of our models on the benchmark dataset. Our models detect more attack types and were not trained and benchmarked on the same dataset. In addition, buffer overflow attack detection scores were excluded from the reported average because this attack, although detected with high precision and recall, was not in any MQTT benchmark dataset. Generally, our average models' detection scores do not significantly deviate from our MQTT-IoT-IDS2020 validation scores hence, the cross-validation and test set scores could well be used to justify our models' performance for this attack as done in works without separate model validation dataset [17], [29]–[33].

1) Threats to Validity: We identified the following potential threats to validity:

- the results presented in this work are based on simulation of IoT devices and therefore the traffic patterns recorded may differ from actual IoT devices. In fact, an IoT network will usually contain more than 40 clients that we simulated for our federated learning. However, the benchmarking of our network with the existing MQTT validation dataset shows our simulation is reasonably similar and acceptable.
- This work is limited to attack detection on MQTT protocol. Of course, there are other protocols that IoT devices could use for communication. However, in practice, MQTT is one of the most common messaging protocols used by IoT devices and we aim to detect anomalies over communication using this protocol.
- Lastly, we do not focus on the details of the operation or modification of the MQTT protocol stack but rather on using artificial intelligence to detect and learn attack patterns from communicated packet data over MQTT in an IoT network. Our results show that our approach is feasible and practical.

V. CONCLUSION

In this paper, a robust intrusion detection system for some important MQTT attacks is presented. An IoT MQTT network consisting of subscribers, publishers, and a broker (alongside a rogue device) communicating over the MQTT protocol is simulated. Then, the network traffic data is used to develop and evaluate centralized learning and federated learning models to classify different MQTT attacks. Our models demonstrate good performance evaluation scores over seven categories of attacks that include attacks such as man-in-the-middle attacks, denial of service attacks, and memory corruption attacks. The classification results of our model validation on the MQTT-IoT-IDS2020 benchmark dataset are significant. Furthermore, this paper differs from existing works with the improved classification of MQTT memory attacks which has been mostly overlooked in the majority of MQTT intrusion detection studies. The results show that both centralized learning and federated learning are capable of detecting MQTT attacks with similar classification scores, and federated random forest has more classification time overhead than federated deep learning. Overall, our findings support federated learning practical application in IoT intrusion detection, together with the added advantage of privacy because instead of training data, model parameters are usually exchanged.

REFERENCES

[1] D. Harkin, M. Mann, and I. Warren, "Consumer iot and its under-regulation: Findings from an australian study," *Policy & Internet*, vol. 14, no. 1, pp. 96–113, 2022.

[2] A. Ahmed, S. U. Din, E. Alhanaee, and R. Thomas, "State-of-the-art in iot forensic challenges," in *2022 8th International Conference on Information Technology Trends (ITT)*. IEEE, 2022, pp. 115–118.

[3] M. A. U. Rehman, R. Ullah, C.-W. Park, B. S. Kim *et al.*, "Towards network lifetime enhancement of resource constrained iot devices in heterogeneous wireless sensor networks," *Sensors*, vol. 20, no. 15, p. 4156, 2020.

[4] M. Salimitari, M. Chatterjee, and Y. P. Fallah, "A survey on consensus methods in blockchain for resource-constrained iot networks," *Internet of Things*, vol. 11, p. 100212, 2020.

[5] J. Pacheco and S. Hariri, "Iot security framework for smart cyber infrastructures," in *1st International Workshops on Foundations and Applications of Self-Systems, FAS-W 2016*. Institute of Electrical and Electronics Engineers Inc., 2016, pp. 242–247.

[6] H. Bauer, O. Burkacky, and C. Knochenhauer, "Security in the internet of things," *Semiconductor, McKinsey & Company: New York, NY, USA*, 2017.

[7] S. Rizvi, A. Kurtz, J. Pfeffer, and M. Rizvi, "Securing the internet of things (iot): A security taxonomy for iot," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 163–168.

[8] G. Singh and N. Khare, "A survey of intrusion detection from the perspective of intrusion datasets and machine learning techniques," *International Journal of Computers and Applications*, vol. 44, no. 7, pp. 659–669, 2022.

[9] F. Chen, Y. Huo, J. Zhu, and D. Fan, "A review on the study on mqtt security challenge," in *2020 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 2020, pp. 128–133.

[10] M. Husnain, K. Hayat, E. Cambiaso, U. U. Fayyaz, M. Mongelli, H. Akram, S. Ghazanfar Abbas, and G. A. Shah, "Preventing mqtt vulnerabilities using iot-enabled intrusion detection system," *Sensors*, vol. 22, no. 2, p. 567, 2022.

[11] A. Rostami, M. Vigren, S. Raza, and B. Brown, "Being hacked: Understanding victims' experiences of {IoT} hacking," in *Eighteenth Symposium on Usable Privacy and Security (SOUPS 2022)*, 2022, pp. 613–631.

[12] R. A Light, "Mosquitto: server and client implementation of the mqtt protocol," *The Journal of Open Source Software*, vol. 2, no. 13, p. 265, 2017.

[13] M. Nasir, K. Muhammad, A. Ullah, J. Ahmad, S. W. Baik, and M. Sajjad, "Enabling automation and edge intelligence over resource constraint iot devices for smart home," *Neurocomputing*, vol. 491, pp. 494–506, 2022.

[14] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: Opportunities and challenges," *Neurocomputing*, vol. 237, pp. 350–361, 2017.

[15] I. K. Nti, J. A. Quarcoo, J. Aning, and G. K. Fosu, "A mini-review of machine learning in big data analytics: Applications, challenges, and prospects," *Big Data Mining and Analytics*, vol. 5, no. 2, pp. 81–97, 2022.

[16] R. A. Disha and S. Waheed, "Performance analysis of machine learning models for intrusion detection system using gini impurity-based weighted random forest (giwrf) feature selection technique," *Cybersecurity*, vol. 5, no. 1, pp. 1–22, 2022.

[17] M. A. Khan, M. A. Khan, S. U. Jan, J. Ahmad, S. S. Jamal, A. A. Shah, N. Pitropakis, and W. J. Buchanan, "A deep learning-based intrusion detection system for mqtt enabled iot," *Sensors*, vol. 21, no. 21, p. 7016, 2021.

[18] S. Andy, B. Rahardjo, and B. Hanindhito, "Attack scenarios and security analysis of mqtt communication protocol in iot system," in *2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*. IEEE, 2017, pp. 1–6.

[19] G. Perrone, M. Vecchio, R. Pecori, R. Giaffreda *et al.*, "The day after mirai: A survey on mqtt security solutions after the largest cyber-attack carried out through an army of iot devices." in *IoTBDS*, 2017, pp. 246–253.

[20] B. H. Çorak, F. Y. Okay, M. Güzel, Ş. Murt, and S. Ozdemir, "Comparative analysis of iot communication protocols," in *2018 International symposium on networks, computers and communications (ISNCC)*. IEEE, 2018, pp. 1–6.

[21] S. N. Firdous, Z. Baig, C. Valli, and A. Ibrahim, "Modelling and evaluation of malicious attacks against the iot mqtt protocol," in *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2017, pp. 748–755.

[22] S. M. Almtrafi, B. A. Alkhudadi, G. Sami, and W. Alhakami, "Security threats and attacks in internet of things (iots)," *International Journal of Computer Science & Network Security*, vol. 21, no. 1, pp. 107–118, 2021.

[23] H. Wong and T. Luo, "Man-in-the-middle attacks on mqtt-based iot using bert based adversarial message generation," in *KDD 2020 AIoT Workshop*, 2020, pp. 1–7.

[24] A. Omotosho, G. B. Welearegai, and C. Hammer, "Detecting return-oriented programming on firmware-only embedded devices using hardware performance counters," in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, 2022, pp. 510–519.

[25] H. Alaiz-Moreton, J. Aveleira-Mata, J. Ondicol-Garcia, A. L. Muñoz-Castañeda, I. García, and C. Benavides, "Multiclass classification procedure for detecting attacks on mqtt-iot protocol," *Complexity*, vol. 2019, 2019.

[26] 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, 2020.

[27] D. Man, F. Zeng, W. Yang, M. Yu, J. Lv, and Y. Wang, "Intelligent intrusion detection based on federated learning for edge-assisted internet of things," *Security and Communication Networks*, vol. 2021, pp. 1–11.

[28] N. A. A.-A. Al-Marri, B. S. Ciftler, and M. M. Abdallah, "Federated mimic learning for privacy preserving intrusion detection," in *2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. IEEE, 2020, pp. 1–6.

[29] E. Jove, J. Aveleira-Mata, H. Alaiz-Moretón, J.-L. Casteleiro-Roca, D. Y. Marcos del Blanco, F. Zayas-Gato, H. Quintián, and J. L. Calvo-Rolle, "Intelligent one-class classifiers for the development of an intrusion detection system: The mqtt case study," *Electronics*, vol. 11, no. 3, p. 422, 2022.

[30] E. Ciklabakkal, A. Donmez, M. Erdemir, E. Suren, M. K. Yilmaz, and P. Angin, "Artemis: An intrusion detection system for mqtt attacks in internet of things," in *2019 38th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2019, pp. 369–3692.

[31] H. AP *et al.*, "Secure-mqtt: an efficient fuzzy logic-based approach to detect dos attack in mqtt protocol for internet of things," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–15, 2019.

[32] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, "Machine learning based iot intrusion detection system: An mqtt case study (mqtt-iot-ids2020 dataset)," in *International Networking Conference*. Springer, 2020, pp. 73–84.

[33] D. C. Attota, V. Mothukuri, R. M. Parizi, and S. Pouriyeh, "An ensemble multi-view federated learning intrusion detection for iot," *IEEE Access*, vol. 9, pp. 117 734–117 745, 2021.

[34] J. Sidna, B. Amine, N. Abdallah, and H. El Alami, "Analysis and evaluation of communication protocols for iot applications," in *Proceedings of the 13th international conference on intelligent systems: theories and applications*, 2020, pp. 1–6.

[35] A. Niruntasukrat, C. Issariyapat, P. Pongpaibool, K. Meesublak, P. Aiumsupucgul, and A. Panya, "Authorization mechanism for mqtt-based internet of things," in *2016 IEEE International Conference on Communications Workshops (ICC)*. IEEE, 2016, pp. 290–295.

[36] J. Li, L. Lyu, X. Liu, X. Zhang, and X. Lyu, "Fleam: A federated learning empowered architecture to mitigate ddos in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4059–4068, 2021.

[37] G. Genovese, G. Singh, C. Campolo, and A. Molinaro, "Enabling edge-based federated learning through mqtt and oma lightweight-m2m," in *2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring)*. IEEE, 2022, pp. 1–5.

[38] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 739–753.

[39] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning," *IEEE Access*, vol. 10, pp. 40 281–40 306, 2022.

[40] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, pp. 1–11, 2021.

[41] S. AbdulRahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5476–5497, 2020.

[42] M. B. Gorzałczany and F. Rudziński, "Intrusion detection in internet of things with mqtt protocol–an accurate and interpretable genetic-fuzzy rule-based solution," *IEEE Internet of Things Journal*, pp. 1–13, 2022.

[43] C. T Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with moreau envelopes," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 394–21 405, 2020.

[44] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Proceedings of the second workshop on distributed infrastructures for deep learning*, 2018, pp. 1–8.

[45] D. Breitenbacher, I. Homoliak, Y. L. Aung, N. O. Tippenhauer, and Y. Elovici, "Hades-iot: A practical host-based anomaly detection system for iot devices," in *Proceedings of the 2019 ACM Asia conference on computer and communications security*, 2019, pp. 479–484.