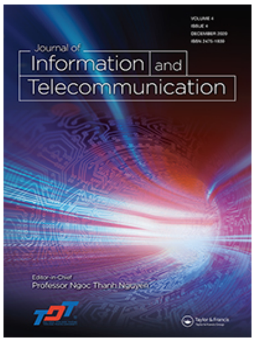

Taylor & Francis
Taylor & Francis Group

Journal of Information and Telecommunication

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tjit20

Denial of service attack detection through machine learning for the IoT

Naeem Firdous Syed, Zubair Baig, Ahmed Ibrahim & Craig Valli

To cite this article: Naeem Firdous Syed, Zubair Baig, Ahmed Ibrahim & Craig Valli (2020) Denial of service attack detection through machine learning for the IoT, Journal of Information and Telecommunication, 4:4, 482-503, DOI: 10.1080/24751839.2020.1767484

To link to this article: https://doi.org/10.1080/24751839.2020.1767484

JOURNAL OF INFORMATION AND TELECOMMUNICATION
2020, VOL. 4, NO. 4, 482–503
https://doi.org/10.1080/24751839.2020.1767484

OPEN ACCESS Check for updates

Denial of service attack detection through machine learning for the IoT

Naeem Firdous Syed[a], Zubair Baig[b], Ahmed Ibrahim[a] and Craig Valli[a]

[a]Security Research Institute, School of Science, Edith Cowan University, Perth, Australia; [b]School of Information Technology, Deakin University, Geelong, Australia

ABSTRACT

Sustained Internet of Things (IoT) deployment and functioning are heavily reliant on the use of effective data communication protocols. In the IoT landscape, the publish/subscribe-based Message Queuing Telemetry Transport (MQTT) protocol is popular. Cyber security threats against the MQTT protocol are anticipated to increase at par with its increasing use by IoT manufacturers. In particular, IoT is vulnerable to protocol-based Application layer Denial of Service (DoS) attacks, which have been known to cause widespread service disruption in legacy systems. In this paper, we propose an Application layer DoS attack detection framework for the MQTT protocol and test the scheme on legitimate and protocol compliant DoS attack scenarios. To protect the MQTT message brokers from such attacks, we propose a machine learning-based detection framework developed for the MQTT protocol. Through experiments, we demonstrate the impact of such attacks on various MQTT brokers and evaluate the effectiveness of the proposed framework to detect these malicious attacks. The results obtained indicate that the attackers can overwhelm the server resources even when legitimate access was denied to MQTT brokers and resources have been restricted. In addition, the MQTT features we have identified showed high attack detection accuracy. The field size and length-based features drastically reduced the false-positive rates and are suitable in detecting IoT based attacks.

ARTICLE HISTORY
Received 16 April 2020
Accepted 7 May 2020

KEYWORDS
IoT; security; network security; MQTT; DoS

1. Introduction

Critical infrastructures (CIs) are increasingly aiming to improve their efficiencies to deliver services to their stakeholders, with the Internet of Things (IoT) promising to provide significant opportunities for improving various CI processes for industries and consumers alike. The role of IoT in building future smart cities is indispensable and is already playing an important role in transforming the energy, transportation and communications sectors (Zanella et al., 2014). The increasing role of IoT devices in such infrastructures, is exposing their vulnerabilities to adversarial cyber threats (Bekara, 2014). It is anticipated that the attack surfaces will rapidly evolve in the near future, exposing CIs to a range of cyber

CONTACT Naeem Firdous Syed n.syed@ecu.edu.au

security challenges (Asplund & Nadjm-Tehrani, 2016; Sadeghi et al., 2015). The attack surface for a cyber-physical IoT system of a typical CI spans across its layers of operation; Physical to Application (Sadeghi et al., 2015). Moreover, the variety of communication patterns and formats adopted by the heterogeneous IoT protocols of a CI encumber the design of a robust cyber security solution (Heer et al., 2011).

MQTT is a publish/subscribe-based Application layer protocol suitable for Machine to Machine (M2M) communication pattern, where a central message broker routes the published messages to subscribers based on their topics of interest. On a publish/subscribe platform, the publishers would publish messages to a topic and subscribers would subscribe to their respective topics of interest. The message broker, acting as an intermediary, then forwards the published messages to subscribers based on the subscribed topics. Figure 1 shows the message publish/subscribe process as implemented by the MQTT protocol.

The message broker plays an important role in MQTT as it decouples the sensors and actuators or monitoring IoT devices in both space and time. This is achieved by a process known as filtering. The publish/subscribe messaging pattern employs two common forms of filtering: content-based and topic-based. In content-based filtering, the subscribers receive only those messages that contain or match the attributes defined by the subscribers, whereas for topic-based filtering, subscribers receive only a subset of published messages that match the message topics on logical channels subscribed by them. The MQTT protocol does topic-based filtering to route messages to interested subscribers. The protocol specifies various control packets to facilitate message exchange between its endpoints. Some of the common control packets are enumerated in Table 1.

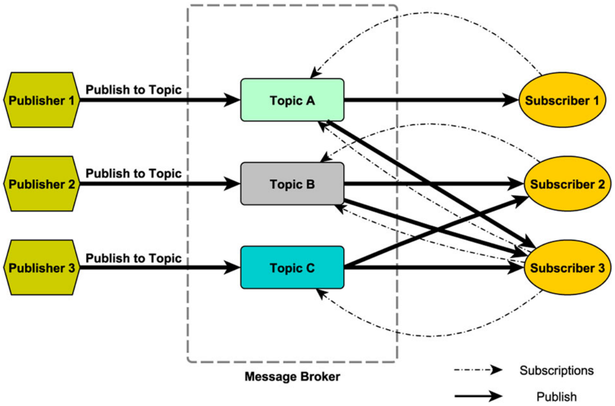

Figure 1. Publish/subscribe process in MQTT.

Table 1. MQTT control packets.

Control packet	Description
CONNECT	First packet sent by a client to broker to initiate a MQTT connection
CONNACK	Acknowledgement for a CONNECT packet sent by broker to a client
PUBLISH	Message sent by the publisher (client to broker or broker to client) to publish a message to a Topic
SUBSCRIBE	Message sent by the subscriber client to the broker to subscribe to a Topic
DISCONNECT	Message sent by the client to broker to disconnect the MQTT connection

The messages between publishers and subscribers are communicated using various control packets which contains a fixed two byte protocol header. This ensures a small message overhead suitable for constrained devices operating in unreliable communication networks. These features have made MQTT protocol a favourable choice for M2M communication in IoT. As the MQTT adoption rate increases for IoT data communication, it is very important to assess and design its security.

A range of security challenges exist for IoT, but Denial of Service (DoS) attacks that target the data communication system pose a particularly significant challenge to IoT deployments (Heer et al., 2011). Such attacks can also pose as a major challenge in cyber-physical CIs, which rely on real-time inter-device communications (Ten et al., 2010). A consequent delay in message delivery due to DoS attacks would disrupt smooth operations of cyber-physical systems of CI. Most of the work as found in the IoT literature either presents DoS attack detection techniques (Alanazi et al., 2015; Kasinathan et al., 2013) or merely refers to DoS attacks as one of the challenges to the IoT ecosystem (Borgohain et al., 2015; Roman et al., 2013; Shaker & Zarrabi, 2017). Furthermore, the MQTT OASIS specification (Cohn, 2014) lists DoS attacks as one of the security threats to the MQTT protocol. Hence, a security solution to protect MQTT-based IoT infrastructures against such attacks is essential.

In this paper, we propose a machine learning-based MQTT DoS attack detection framework for the IoT platform, based on a custom made DoS attack model. The key contributions of this paper are:

- Identification of network traffic features to represent the MQTT protocol meta-data,
- Modelling of DoS attacks for MQTT brokers,
- Design of an effective detection framework to detect MQTT DoS attacks, and
- Testing of the attack model and the attack detection framework on a physical IoT deployment.

The rest of the paper is organized as follows: in Section 2, we present the related work. Section 3 details the DoS attack model. Section 4 presents attack detection framework. Section 5 presents the experimental results and analysis thereof. The conclusion is presented in Section 6.

2. Literature review

The most common types of DoS attacks aim to exhaust the network bandwidth, CPU cycles or memory on the target system to make services unavailable for legitimate users (Durcekova et al., 2012). To maximize the impact of DoS attacks against a target system, findings reported in the literature indicate that Application layer attacks are being increasingly perpetrated by adversaries (Brenner, 2010; Mantas et al., 2015). Such attacks aim to consume the target resource by sending carefully crafted legitimate requests towards the victim. The most common DoS attacks against Application layer protocols exploit vulnerabilities in the initial connection-establishment message exchanges, as listed in Table 2.

Most of the works related to MQTT protocol available in the literature focus on performance evaluation (Fehrenbach, 2017; Gündoğan et al., 2018; Lee et al., 2013; Luzuriaga et al.,

Table 2. DoS attack strategies adopting protocol-specific legitimate requests for various Application layer protocols.

Protocol	Request type	Attack packet sending methods
HTTP[a]	HTTP-GET, HTTP-POST	• Flooding (exponentially increasing attacks, Flash attacks, constant high-rate attacks) • Low rate: Periodic (square wave DoS stream – low average packet rate[b]) , Slowloris
SIP[c]	SIP-INVITE, SIP-REGISTER	Flooding
DNS[d]	Name resolution query	Flooding
SMTP[e]	Email flooding	Flooding

[a]Singh et al. (2017), Adi et al. (2016) and Ranjan et al. (2009).[b]Shan et al. (2017).[c]Rafique et al. (2009) and Luo et al. (2008).[d]Ballani and Francis (2008).[e]Bencsath and Ronai (2007).

2015; Scalagent, 2015; Thangavel et al., 2014; Yokotani & Sasaki, 2016), proposing security enhancements to the existing protocol (Mektoubi et al., 2016; Shin et al., 2016; Singh et al., 2015), formal modelling (Aziz, 2016; Houimli et al., 2017) and security evaluation (Andy et al., 2017; Firdous et al., 2017; Perrone et al., 2017).

The various performance evaluation methods proposed in Scalagent (2015), Lee et al. (2013), Thangavel et al. (2014), Luzuriaga et al. (2015), Yokotani and Sasaki (2016), Gündoğan et al. (2018) and Fehrenbach (2017) do not evaluate the broker performance during the DoS attacks. In a work done by Fehrenbach (2017), the author conducted tests to simulate a Distributed-DoS (DDoS) attack on the MQTT broker. The main focus of his contribution was to identify the impact of using Transport Layer Security (TLS) for communication and message QoS on the broker resources. One of the drawbacks of his work was that it focused only on the impact of various message QoS levels and TLS connection on the broker performance. Our work is different from the study presented in Fehrenbach (2017) as the focus of this work is on modelling DoS attacks by varying the parameters in control packets and assessing their impact on various MQTT brokers and deployment scenarios.

Perrone et al. (2017) presented a security analysis of MQTT protocol and described the various security requirements for IoT deployments. In another work, Andy et al. Andy et al. (2017) presented some attack scenarios as well as a security analysis of the MQTT protocol. In their work, the authors highlighted the security issues of the MQTT protocol and discussed attack scenarios against brokers with open authentication. Feasibility of such attacks is questionable as most MQTT broker deployments in industrial environment disable open authentication feature as it poses security risk of unauthorized access. In order to understand the security issues in the MQTT protocol, a threat model and the impact of SYN-Flood DoS attack on message brokers was presented in Firdous et al. (2017).

Santiago Hernández Ramos and Lacuesta (2018) proposed a fuzzing approach to test vulnerabilities of an MQTT based application. The proposed approach tested the behaviour of the MQTT based application when fuzzed data was inserted between clients and the broker. The authors used a proxy fuzzing technique along with a non-normative packet variable header data template to asses the behaviour of both broker and clients when presented with unexpected data. Failures were detected in certain versions of the broker software and in client applications. A similar MQTT fuzzer tool known as F-secure

MQTT-FUZZ was developed by Vähä-Sipilä (2015) which uses sniffed raw MQTT control packet payload to launch fuzzed MQTT packets against the broker.

An existing work that attempts to detect IoT based attacks proposed MQTT transaction based features Moustafa et al. (2019). However, the authors used features based on the TCP protocol analysis, which do not provide sufficient information on the MQTT protocol parameters. In contrast, our proposed MQTT features are based on MQTT header and payload meta-data, which can effectively detect and differentiate such attacks. In addition, the main drawback of Moustafa et al. (2019) is that the performance of their attack detection scheme was not presented for MQTT attacks. The primary reason behind this was that no real MQTT attack datasets existed to evaluate the detection techniques. In our work, we first present various vulnerabilities in MQTT and also develop several attack scenarios to generate actual DoS attack traffic. We also evaluate the detection capability of the proposed IoT attack detection framework. In the following section, the DoS attack model used to create the DoS attack scenarios on MQTT protocol is presented.

3. DoS attack model

The main goal of DoS attacks is to overwhelm server resources and to deny access by legitimate clients. According to Little's Law (Little & Graves, 2008), the average number of items in a queuing system can be defined as:

$$L = \lambda * W \tag{1}$$

where λ is the arrival rate of items into the system and W is the average time spent by an item in the system. DoS attacks aim to fill-up the system queue, thus denying service to legitimate clients. DoS attacks can either increase arrival rate of packets or increase the per-packet processing time by forcing complex computing operations at the victim device. In most industrial MQTT applications, authentication and authorization help prevent unauthorized access, and authenticated clients are authorized to either send or receive messages on selected topics only. Access levels available for MQTT clients are one of two:

- Valid Credentials to connect to MQTT broker, or
- Valid Authorisation to Publish/Subscribe to topics.

An attacker without valid credentials can only vary the parameters of a CONNECT packet as clients cannot publish or subscribe without successful connection to the broker. However, after a successful connection using valid credentials but without valid authorization to publish and subscribe to topics, the attacker can vary PUBLISH or SUBSCRIBE control packet parameters. Based on the control packet type and the access level available to an attacker, the flooding attacks can be categorized into:

(1) **Basic CONNECT Flooding (BF1)**: The attacker only sends a large volume of CONNECT packets to the target server to overwhelm the server with the processing of authenticating requests.
(2) **Delayed CONNECT Flooding (BF2)**: The CONNECT packet transmission is delayed by the attacker after the establishment of the TCP session. This will result in a high volume

of half-open TCP sessions at the broker, as it is waiting for the CONNECT request to complete. It also causes the broker to process these invalid credentials, thus leading to an increase in CPU utilization.

(3) **WILL Payload CONNECT Flooding (BF3)**: The CONNECT packet size is increased by the attacker through piggy-backing a WILL Payload on a CONNECT packet. This will lead to the consumption of both the entire bandwidth at the victim server as well as CPU resources, preventing it from processing new connections.

(4) **Invalid Subscription Flooding (IAUTHS)**: With valid credentials but no authorization to access various topics, an attacker can flood the broker with invalid subscriptions or publish requests to the subscriber. This will result in consumption of broker CPU resources in verifying individual request.

4. Attack detection framework

In this section, a DoS attack detection framework is proposed. The framework comprises of a network traffic generator, feature extraction engine and a machine learning-based DoS attack traffic classifier.

4.1. MQTT traffic generation

An IoT-MQTT network typically comprises of a set of IoT sensors that observe environmental phenomena and constantly communicate with each other or with monitoring or control devices through a centralized message broker. The proposed traffic generation component of the detection framework includes two physical servers, one hosting the VerneMQ MQTT broker virtual machine and the second serving as the attacker. Thirty Raspberry Pi (RPI) devices and four WEMOS ESP8266 devices were connected to two wireless routers; equally distributed. Twenty RPI devices and four ESP8266s, each interfaced with physical sensors was configured to publish sensor data periodically. The sensors comprised of: a PIR motion sensor, CCS811-Air Quality Sensor, DS130 RTC clock, DS18B20 temperature sensor and MH-MQ Gas sensor. The remaining 14 RPI devices were configured to periodically send MQTT messages to the broker. The broker was configured with 1000 username/password combinations to authenticate the clients, and 1000 MQTT Access Control-list (ACL) to authorize devices to publish/subscribe to various topics. Anonymous login was disabled to allow only authenticated access to publish and subscribe topics.

Figure 2 illustrates the network traffic generator testbed deployed. Sensors were sending updates to the broker with a varying periodicity with a sleep interval between 4 and 8 s. The reason to keep the sleep intervals below 10 s was to achieve a realistic message publish rate, in alignment with the standard practice (Sivanathan et al., 2018). Sensors were also configured with LAST WILL message generated varying length messages, where a LAST WILL message is transmitted to update the subscribed clients if the publishing client disconnects abruptly.

DoS attack traffic was generated using a custom-built MQTT attack tool based on the Ecliplse-Paho library (Eclipse, 2018). Since these MQTT attacks were generated from a single attack source, a multi-threaded approach was adopted in the attack tool to maximize the impact on victim server's available resources. Each attack was based on

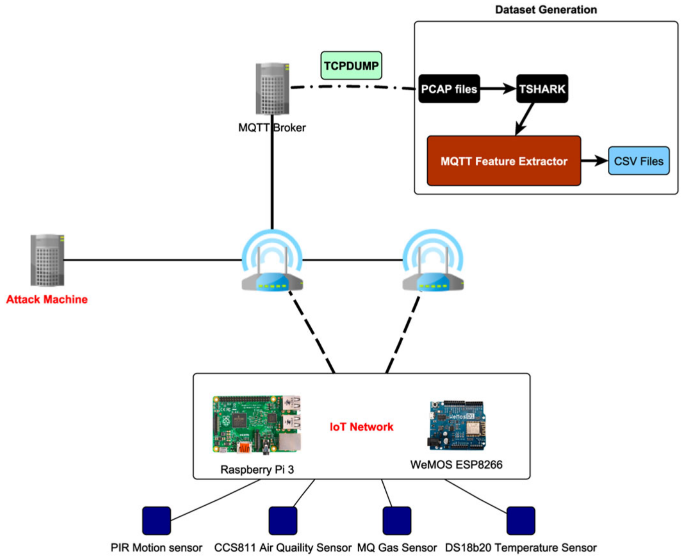

Figure 2. MQTT data set generation testbed and the associated tools for feature generation.

specific MQTT protocol settings. The brute force CONNECT flooding attacks were configured with a random length character comprising a ClientID, username and password, similar to a real client. The Subscribe flooding attack sent 200 subscription requests in each session with six subscriptions per request. The number of subscription requests was randomly selected to maximize the number of SUBSCRIBE packets sent to the broker per connection request. The delayed CONNECT flooding attack (BF2) was launched using 250 threads and the remaining three attacks were launched using three threads to generate maximum number of attack packets targeting the broker. Selection of the number of threads for each attack type was based on the evaluation of number of attack packets received by the broker while incrementing the number of threads and the value that generated maximum number of attack packets was chosen.

Attack traffic was generated from a separate physical server connected to the network based on the various attack scenarios described previously. Network traffic was captured on the victim machine using the TCPDUMP (2019) tool in packet capture (pcap) format separately for normal and individual attacks to ease the labelling process for supervised classification. The TCPDUMP tool was configured to save the captured packets in 30MB chunks so as to reduce the processing load associated with feature extraction. Tshark tool (Wireshark, 2019) was deployed to extract specific packet parameters of the TCP and MQTT protocols for subsequent extraction.

4.2. MQTT feature extraction

In this phase, the custom-built feature generator module was deployed to generate flow-based statistical data. The tool extracted each flow identified through the following network traffic features: Source IP, Destination IP, Source Port and Destination Port, for each pcap file. The tool also extracted various aggregate values of the MQTT parameters. These aggregate/statistical flow features were calculated based on the parameters of

MQTT sessions, such as count, size and field lengths. Specifically, two groups of statistical MQTT flow features namely, session features and MQTT packet and field length features, were generated. Session features were based on the counts of number of packets, number of control packets and number of QoS packets that belonged to the same flow. Packet size and field length features were based on the captured IP packets and the various MQTT field lengths, illustrated in Table 3. Only packet meta-data were utilized to generate the features instead of deep inspection of the payload. Hence, this feature extraction method can also be utilized on flows with encrypted MQTT payloads. In order to measure the accurate number of subscription requests per flow, the feature generation module counted the individual subscription requests and the number of topics in the request as separate request.

4.3. Attack detection module

The attack detection module of the framework is a machine learning (ML) based detection system. Statistical flow features extracted from MQTT network traffic serve as input to the classification system and help differentiate normal from attack flows, as well as inter-attack flow classification. The task of differentiating between the various flooding attacks will enable effective counter measures to be applied to thwart such attacks. The use of legitimate requests in Application layer DoS attacks can pose a significant challenge to the detection framework in differentiating between normal and attack network flows.

Table 3. Proposed MQTT DoS detection features (feature type N: Numeric, B: Binary).

S.No	Feature	Description	Type
		MQTT session statistical features	
1	flow_duration	Duration of Flow	N
2	pkt_in_flow	No. of Packets in a Flow	N
3	Connect_Command	No. of CONNECT packets in Flow	N
4	Publish_Message	No. of PUBLISH packets in the Flow	N
5	Subscribe_Request	No. of SUBSCRIBE packets in the Flow	N
6	Disconnect_Req	No. of DISCONNECT packets in the Flow	N
7	Ping_Request	No. of PING packets in the Flow	N
8	Subs_Qos0	No. of SUBSCRIBE packets with QoS 0 in the Flow	N
9	Subs_Qos1	No. of SUBSCRIBE packets with QoS 1 in the Flow	N
10	Subs_Qos2	No. of SUBSCRIBE packets with QoS 2 in the Flow	N
11	Pub_Qos0	No. of PUBLISH packets with QoP 0 in the Flow	N
12	Pub_Qos1	No. of PUBLISH packets with QoP 1 in the Flow	N
13	Pubs_Qos2	No. of PUBLISH packets with QoP 2 in the Flow	N
14	Will_Qos0	No. of WILL messages with QoS 0 in the Flow	N
15	Will_Qos1	No. of WILL messages with QoS 1 in the Flow	N
16	Will_Qos2	No. of WILL messages with QoS 2 in the Flow	N
17	tcp.time_delta	Time between packets in the flow	N
		MQTT Packet and Field Length Features	
18	frame.len	Avg. Frame Length in the Flow	N
19	tcp.len	Avg. TCP length in the Flow	N
20	mqtt.clientid_len	ClientID length in the Flow	N
21	mqtt.username_len	username length in the Flow	N
22	mqtt.passwd_len	password length in the Flow	N
23	mqtt.willtopic_len	WILL Topic length in the Flow	N
24	mqtt.willmsg_len	WILL Message length in the Flow	N
25	mqtt.len	Avg. MQTT packet length	N
26	mqtt.topic_len	Avg MQTT Topic Length	N
27	mqtt.kalive	MQTT Keep Alive interval	N
28	mqtt.conflag.cleansess	CleanSession Flag Set/Unset	B

Furthermore, broken sessions due to loss of network connectivity can potentially increase the challenge to detect attacks. Hence, the ML algorithms selected in the framework should be sensitive to small variations in feature vector values, to accurately classify network traffic. For example, a normal flow can have the following communication sequence in the IoT-MQTT message exchange: *TCP handshake* + *Connect* + *Publish/Subscribe* + *Disconnect*, whereas, an attack communication sequence can be: *TCP handshake* + *Connect* or *TCP handshake* + *Connect* + *multiple Publish/Subscribe* + *Disconnect*.

In this study, three fundamentally different machine learning approaches namely, average one-dependence estimator (AODE), C4.5 decision trees and artificial neural network (ANN) were integrated into the detection framework. The steps followed in the detection framework to classify MQTT traffic are illustrated in Figure 3 and the three classifiers adopted in MQTT attack detection are discussed below:

AODE Classifier Webb et al. (2005): The AODE classifier is a variant of the Naïve Bayes classifier that estimates the probability of the class of each output variable Y given a set of input features $x_1,\ldots x_n$. It is based on a simple Naïve Bayes classifier which relies on the assumption of independence of attributes. Assuming that all attributes are independent given the class, then Naïve Bayes can be defined as:

$$\arg\max_{c\in C} P(C)\prod_{i=1}^{n} P(a_i|C) \tag{2}$$

Where C is the class label and a is the attribute. In this scenario, the computation cost is reduced; however, the performance of Naïve bayes decreases if the dependency

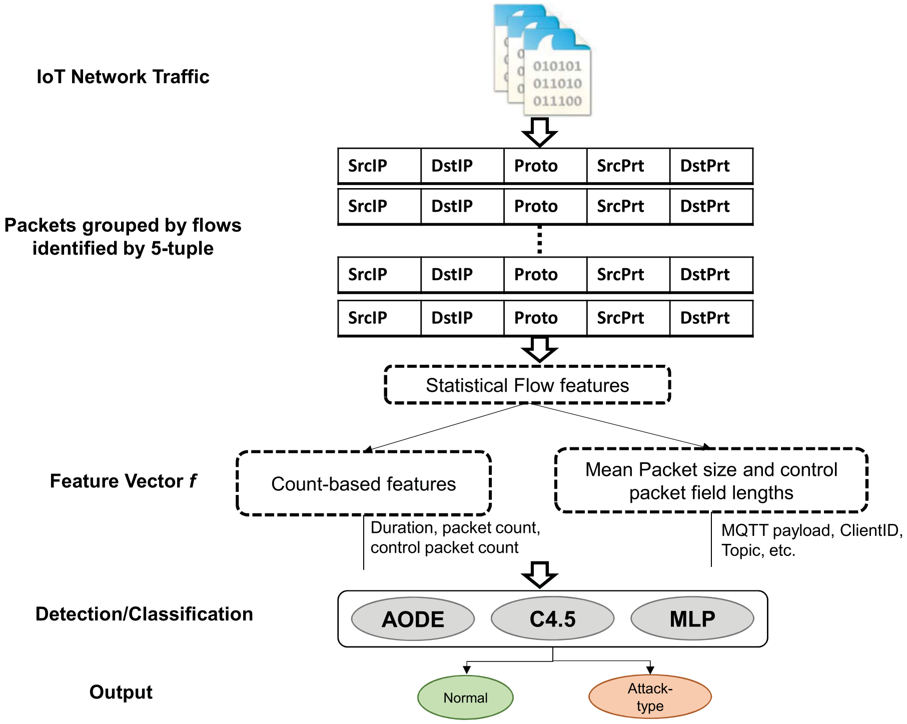
IoT Network Traffic
010101
011010
011100
Packets grouped by flows identified by 5-tuple
SrcIP
DstIP
Proto
SrcPrt
DstPrt
Statistical Flow features
Feature Vector *f*
Count-based features
Mean Packet size and control packet field lengths
Duration, packet count, control packet count
MQTT payload, ClientID, Topic, etc.
Detection/Classification
AODE
C4.5
MLP
Output
Normal
Attack-type

Figure 3. Detection framework work-flow in detecting MQTT attacks.

between the attributes is high (Koc & Carswell, 2015). To counter this effect, the AODE classification technique uses a weaker independence assumption to achieve a higher accuracy rate compared to Naïve Bayes. AODE classifiers are simple to implement and provide high accuracy in classifying data.

Decision Trees (DT): The decision tree-based algorithms build training data sets into the tree structure applying the information entropy principle. Each branch of the tree represents an association between the feature vector and the class label. C4.5 is one of widely used DT method which recursively partitions the training data set by choosing the most effective features to differentiate between the classes. In the first step, C4.5 identifies the best feature that can divide the data instances. In further steps, child nodes are created to divide the instances into subclasses. The attributes selected in each division point in the tree are based on the largest information gain using the best attribute. Entropy is used as a measure of information gain, calculated as follows:

$$H(X) = -\sum_{i=i}^{n} p_i \log_2 (p_i) \tag{3}$$

Multi-Layer Perceptron (MLP): MLP is a type of feed-forward artificial neural network (ANN) that comprises multiple nodes known as artificial neurons, emulating the biological neurons of brain. The nodes in the MLP are grouped as input layers representing the input features, hidden layer and an output layer. In MLP, the nodes of a given layer use activation functions to control the node's output, as well as to serve as an input for the next node. The nodes in MLP are connected by weights which are tuned by using back-propagation algorithms, that adjust the weights to reduce the error between outputs and expected results where, the error is calculated as follows:

$$e_i(n) = t_i(n) - y_i(n) \tag{4}$$

Where $t_i(n)$ is the expected output and $y_i(n)$ produced output value of the instance n and output node i.

5. Experimental results and analysis

5.1. DoS attack assessment

The individual DoS attack scenarios were evaluated on MQTT protocol version 3.1, deployed through three open-source broker software tools deployed using virtual machines. Oracle Virtual Box (Oracle, 2018) running on Windows 10 machine (64GB RAM, Intel Core i7-5820K, 6 physical CPUs, 12vCPU, 3.30GHz) was used to host the three virtual machines to deploy the broker software and each virtual machine was configured with 1 CPU, 8GB RAM and 15GB hard disk. The three open-source MQTT broker implementations deployed in this study were namely: Eclipse Mosquitto (1.4.12) (Mosquitto, 2017), VerneMQ (1.6.2) (Vijay, 2018) and EMQ (3.0) (EMQ, 2018). Mosquitto is a light-weight, portable and single-threaded MQTT broker. In contrast, VerneMQ and EMQ brokers are multi-threaded and scalable MQTT brokers designed using Erlang/Open Telecom Platform(OTP) to achieve high scalability. These brokers were deployed on Ubuntu Server 17.10. A separate physical machine running Ubuntu server 18.04, connected to the broker network using a router, was configured to launch the DoS attacks as shown in Figure 4.

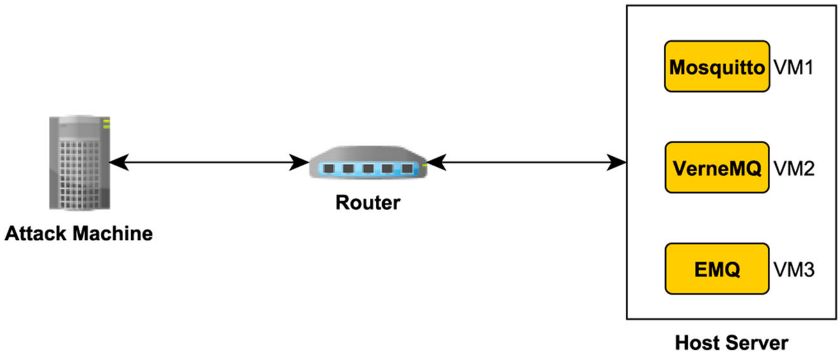

Figure 4. MQTT broker deployment to assess DoS attacks.

5.2. Attack impact analysis

Various attack metrics were measured to assess the impact of DoS attacks against the MQTT brokers. These include: CPU utilization, bandwidth and memory utilization. CPU utilization was measured using *mpstat* Linux command line tool, which provides a break-up of CPU usage by various system tasks. A custom BASH script was run to measure memory and bandwidth utilization at each 1 s interval. All the experiments and measurements were repeated three times to ensure validity and repeatability of results. The metrics measured to evaluate the DoS attack impact were:

- **%usr**: Percentage time spent by CPU executing application related task
- **%sys**: Percentage time spent by CPU executing kernel level task
- **%iowait**: Percentage time spent by CPU executing disk I/O requests
- **%soft**: Percentage time spent by CPU servicing software interrupts
- **%idle**: Percentage time spent by CPU not executing any tasks and no pending I/O requests. A high idle % indicates the CPU is least utilized and a low idle % indicates high CPU utilization.
- **Process CPU (pCPU)**: Measured using bash script fetching the CPU utilization associated with broker process ID using top Linux command
- **Bandwidth**: Total bandwidth consumed during the attack (kbytes)
- **Memory**: Percentage Memory consumed during the attack

The flooding attack results achieved with maximum attack packet rate indicate that the all the three brokers suffered high CPU utilization during the various attack scenarios. The percentage of time the CPU was idle measured during the attack reduced drastically as shown in Figure 5. The VerneMQ and EMQ brokers had the maximum impact as the CPU idle percentage reached zero for more than one attack scenario. However, the idle percentage for Mosquitto broker was close to 20% and reached 0% for invalid subscription attack. The results also show that invalid subscription flooding attack caused the maximum impact on the CPU utilization as all the brokers had CPU idle percentage below 5%.

The CPU utilization break-up in Table 4 shows that during the various attack scenarios the VerneMQ and EMQ brokers spent more time in I/O Wait and application-related processing. In contrast, the Mosquitto broker spent more time in kernel functions and

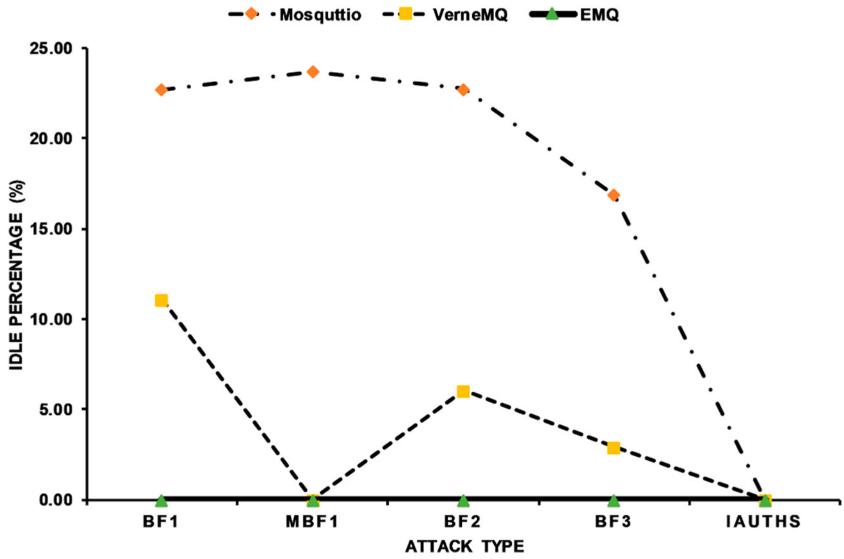

Figure 5. CPU Idle percentage for three MQTT brokers in four attack scenarios along with malformed packet attack.

Table 4. CPU utilization break-up into system tasks during the four DoS attacks along with malformed packet attack (mBF1).

Attack Type	Broker	%usr	%sys	%iowait	%soft	%idle
BF1-RandomID	Mosquitto	7.08	28.42	0.02	41.77	22.69
	VerneMQ	31.95	26.69	0.00	30.25	11.08
	EMQ	20.92	10.12	61.78	7.16	0.00
mBF1	Mosquitto	6.84	26.79	0.01	42.66	23.69
	VerneMQ	36.03	13.56	31.86	18.55	0.00
	EMQ	36.39	7.78	44.10	11.73	0.00
BF2	Mosquitto	7.17	30.88	0.01	39.19	22.75
	VerneMQ	32.79	30.04	0.00	31.13	6.03
	EMQ	21.27	11.81	59.74	7.17	0.00
BF3	Mosquitto	4.63	20.75	0.01	57.75	16.86
	VerneMQ	27.41	24.88	0.01	44.81	2.90
	EMQ	92.78	4.05	2.21	0.95	0.00
	Mosquitto	84.00	0.50	0.00	15.50	0.00
IAUTHS	VerneMQ	82.47	1.33	0.00	16.21	0.00
	EMQ	90.67	0.40	0.00	8.93	0.00

software interrupts. These results show that various MQTT brokers use different techniques to handle connection requests as mentioned in Karagiannis et al. (2015).

The results also indicate that VerneMQ and EMQ brokers are vulnerable to malformed MQTT packets which contain non-ASCII characters. Both the brokers suffered high memory utilization with the non-ASCII characters included in the MQTT fields. The EMQ broker had the worst performance among the brokers as it had high CPU utilization during all the attack scenarios as shown in Figure 6 and higher memory utilization compared to other brokers. Especially, the broker suffered high memory utilization during the bruteforce attack with WILL payload and can be potentially exploited to cause memory-exhaustion attacks to completely incapacitate the broker.

5.3. Attack classification results and analysis

The performance of various classifiers was tested using the Hall et al. (2009) machine learning software. The effect of the selected MQTT features on attack detection was evaluated by comparing the detection framework performance with count-based statistical features and the full feature set including the size and length based features, for both the four-class and seven-class datasets.

5.3.1. Dataset description

The normal and attack packets were captured separately and pre-processed to generate the dataset for testing the detection framework. Three different types of MQTT attacks were generated against the broker to capture the attack dataset namely: MQTT-DoS (based on the attack scenarios described in this work), MQTT-FUZZER (using a MQTT Fuzzing tool Vähä-Sipilä, 2015) and TCP-DOS (using hping3 SYN-Flood tool Sanfilippo, 2006). The MQTT fuzzing tool was configured to send fuzzed packets sniffed from the deployed IoT network. Based on the class labels, two datasets were generated, four-class and seven-class dataset. The class labels in the four-class dataset were: Normal, MQTT-DOS, MQTT-FUZZ and TCP-DOS. In contrast, the seven-class dataset contained four sub-classes of MQTT-DoS attacks presented in this work namely:MQTT-DOS-BF1, MQTT-DOS-BF2, MQTT-DOS-BF3 and MQTT-DOS-IAUTHS. The seven-class dataset was used to evaluate the detection framework performance in detecting the four attack types discussed in this work. The total number of flows in the four-class dataset was 1,012,052 samples and 1,042,500 for the seven-class dataset.

Re-sampling technique was applied to balance the classes to avoid bias in classifier accuracy. The data were under-sampled to produce random sub-samples of the original dataset with following setting in Weka: *biasToUniformClass=1.0, noReplacement=True, sampleSizePercent=40.0*. The break-up of classes in original and re-sampled datasets is presented in Table 5.

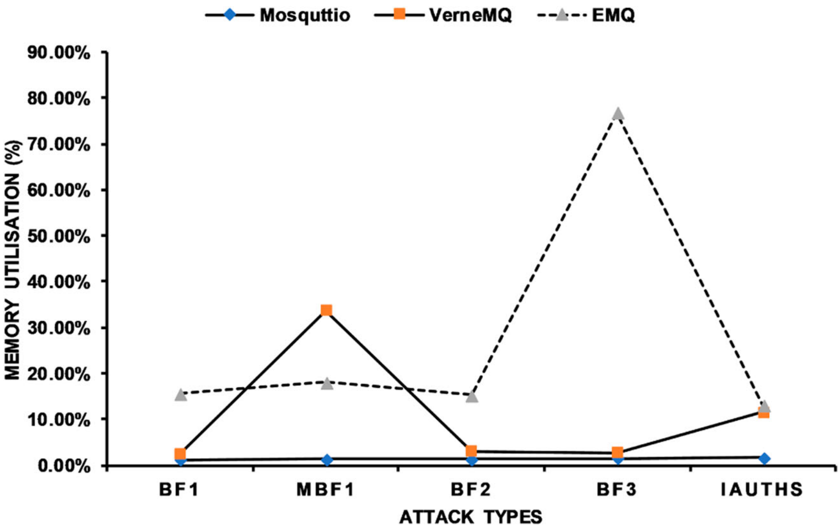

Figure 6. Memory utilization for three brokers during four attack scenarios along with malformed packet attack (mBF1).

Table 5. Class distribution of the two datasets used for training and testing the classifiers.

Class	No. of Instances before balance	No. of Instances after balancing
4-Class MQTT Dataset		
Normal	71217	71217
MQTT-DOS	623246	151807
MQTT-FUZZ	49916	49916
TCP-DOS	267673	151807
7-Class MQTT Dataset		
Normal	73982	59571
MQTT-DOS-BF1	240349	59571
MQTT-DOS-BF2	277454	59571
MQTT-DOS-BF3	90550	59571
MQTT-DOS-IAUTHS	49916	49916
MQTT-FUZZ	42576	42576
TCP-DOS	267673	59571

5.3.2. Detection metrics

The performance of the detection framework was evaluated by conducting several experiments with the two datasets described in the previous section. The individual classifiers performances were measured to assess the effectiveness in detecting anomalous MQTT traffic. The following metrics were adopted to evaluate the performance of detection framework: Detection Rate (DR), Accuracy and False-Positive Rate (FPR). These metrics can be calculated by measuring the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) from the number of correctly and incorrectly classified instances. TP is the number of correctly detected anomalous instances in the dataset. TN is the number of correctly detected legitimate instances. FP is the number of normal records classified as anomalous while the FN is the number of anomalous instances classified as legitimate. Accuracy (ACC) is the percentage of instances correctly classified as either anomalous or legitimate and it is calculated by:

$$\mathrm{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{5}$$

True positive rate (TPR) measures the percentage of instances correctly classified as anomalous and is calculated by:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{6}$$

False-positive rate (FPR) is percentage of instances incorrectly classified as anomalous and is calculated by:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{7}$$

Several experiments were conducted to measure the performance of AODE, DT and MLP classifiers used in the detection framework. A 5-fold cross-validation method was enforced which trained and tested the machine learning models on complementary subset of data to prevent bias and over-fitting issues. Weka settings used for A1DE classifier are: *frequencyLimit=1, subsumptionResolution=False, weight=1.0, weightedAODE=False*. The J48 DT algorithm an implementation of C4.5 classifier in Weka was configured with the following setting: *confidenceFactor=0.25, minNumObj=2, reducedErrorPruning=False, unpruned=False,* *useLaplace=False and useMDLcorrection=True*. The default settings of Weka MLP classifier used in this work was: *batchSize=100, decay=False, hiddenLayers = (attributes + classes) / 2, learningRate=0.3, momentum = 0.2, trainingTime = 500*.

5.3.3. Detection results

The evaluation results of the classifiers used in the MQTT attack detection framework are presented in terms of the accuracy (%), error (%), TPR and FPR and time to build the model. The classifiers were evaluated with both count-based (counts) features (statistical MQTT session features presented in Table 3) and with full features set. The performance of the three classifiers on the four-class and seven-class datasets respectively, is presented in Table 6. Figures 7 and 8 show that the AODE classifier achieved the highest classification accuracy in detecting the attack traffic for both four-class and seven-class datasets.

Table 6. Performance comparison of three classifiers used in the MQTT attack detection framework.

Classifier/Features	ACC	Error	TPR	FPR	Training time
4-Class					
AODE-counts	99.0025	0.9975	0.99	0.005	4.33
C45-Counts	99.0905	0.9095	0.991	0.005	30.19
MLP-Counts	95.9399	4.0601	0.959	0.021	847.41
AODE-full	99.968	0.032	1	0	14.17
C45-Full	99.9546	0.0454	1	0	50.52
MLP-Full	99.4957	0.5043	0.995	0.001	1828.23
7-Class					
AODE-counts	88.2161	11.7839	0.882	0.021	5.51
C45-Counts	85.6236	14.3764	0.856	0.026	31.24
MLP-Counts	65.5542	34.4458	0.656	0.062	1215.42
AODE-full	99.8483	0.1517	0.998	0	10.82
C45-Full	99.8294	0.1706	0.998	0	42.09
MLP-Full	84.221	15.779	0.842	0.028	1962.65

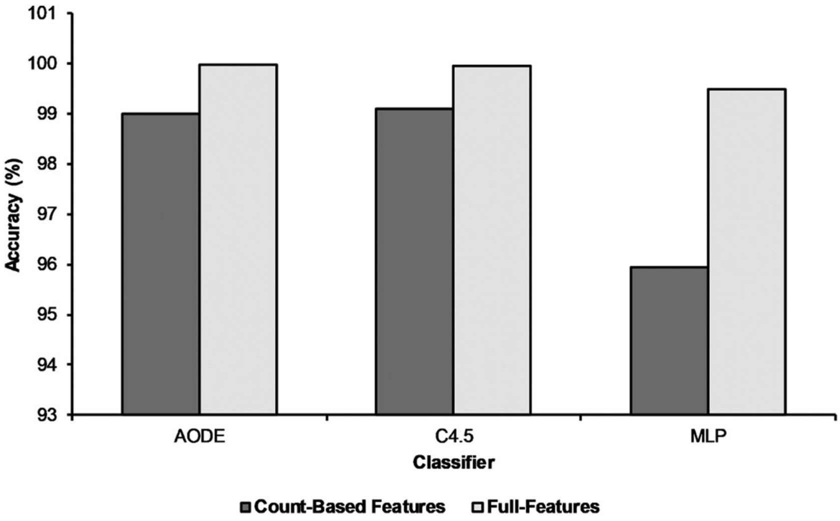

Figure 7. Performance comparison of classifiers with various combination of features on 4-Class dataset.

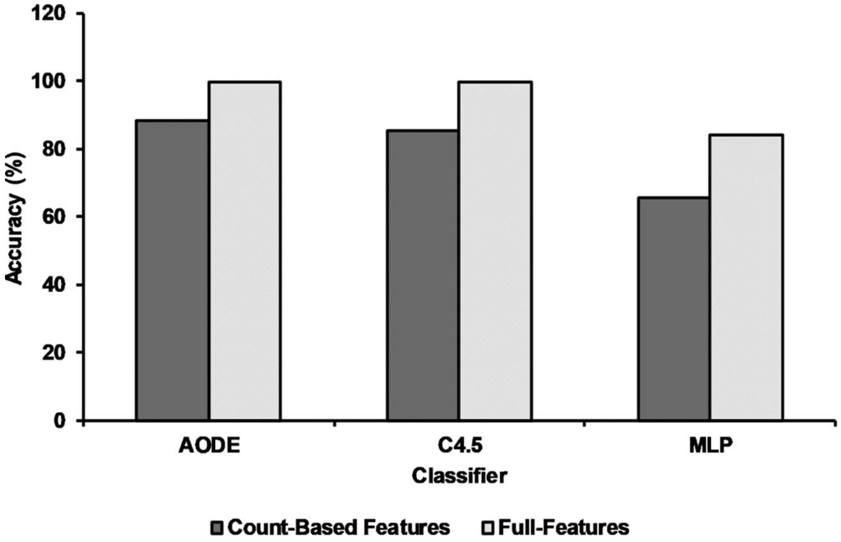

Figure 8. Performance comparison of classifiers with various combination of features on 7-Class dataset.

The results also indicate that the detection accuracy of classifiers increased when packet size and field length based features were considered along with the count-based features, when compared to using only count-based features. The AODE classifier had the lowest and MLP classifier had the highest training times among the selected classifiers. These results indicate that the proposed MQTT features provided good separation between normal and attack traffic resulting in high detection rates and low false positives. However, the MLP classifier only achieved a classification accuracy of 84% for the seven-class dataset, when all the features were used. Hence, the MLP classifier was further evaluated with various optimization parameters to identify the most optimal settings to increase its detection performance. The optimization parameters considered in this study for improving MLP classifier were: activation and solver functions.

In an ANN, an activation function of a neuron maps the input signal to an output signal. Choosing the correct activation function supports the MLP classifier in generating more accurate and complex non-linear mappings between the inputs and outputs, hence improving the classifier accuracy (Karlik & Olgac, 2011). The solver functions refer to algorithms that try to estimate the optimal weights for the hidden and output layers in order to reduce the training errors. These are classified into first and second order methods and vary in computation complexity when they are minimizing or maximizing the loss function. Since Weka does not have options to vary the activation and solver functions, the MLP optimization parameters was tested using Python scikit-learn ML platform (Pedregosa et al., 2011). On this platform, the performance was evaluated with three activation functions: Relu, logistic-sigmoid and tanh. In addition, two solver algorithms: Stochastic Gradient Descent (SGD) and limited-Memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) were compared for their optimization performance in tuning the ANN weights for the seven-class dataset. The SGD algorithm uses learning-rate and momentum to optimize the model by iteratively estimating the training loss using samples from the training dataset. The optimal momentum and learning-rate were identified by iteratively varying the two variables and the settings, selecting the least training loss for evaluating the activation functions. Figures 9 and 10 show the observed training loss of the MLP classifier for various values of momentum and learning rates, respectively. These results show that a momentum of 0.9 and a learning-rate of 0.001 yielded the least training loss. Furthermore, Figure 11 shows that the performance of MLP classifier achieved a higher detection accuracy with relu activation function compared to logistic-sigmoid and tanh activation functions when applied to both SGD and L-BFGS optimization algorithms.

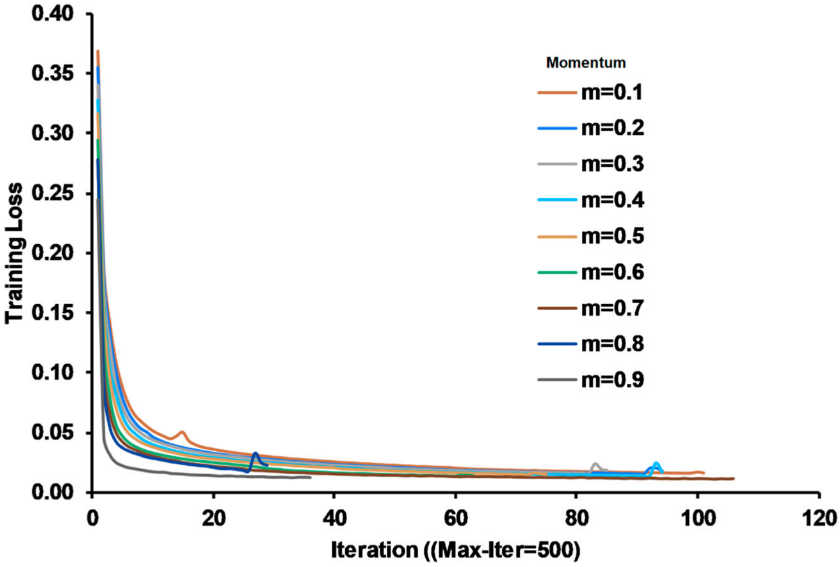

Figure 9. Training loss observed for the MLP classifier for various values of momentum and numbers of iterations on seven-class dataset.

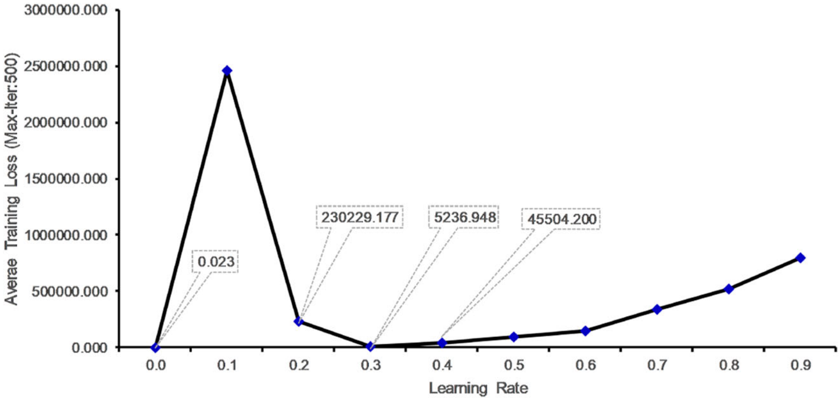

Figure 10. Average training loss for the MLP classifier calculated for various values of learning rate with a maximum of 500 iterations on seven-class dataset.

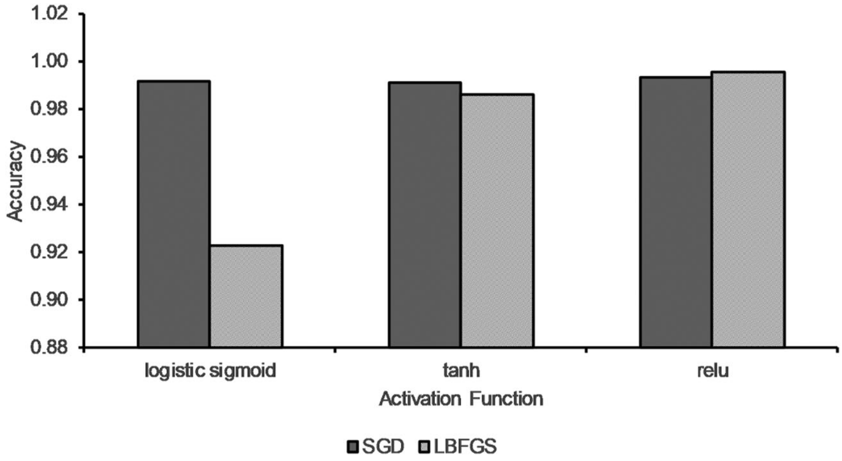

Figure 11. Performance evaluation of the activation function and MLP optimization solver functions on seven-class dataset.

6. Conclusion

In this work, a DoS attack detection framework for MQTT attack detection in IoT environments was proposed and evaluated. The attack detection testbed was designed to capture normal and attack traffic, and count-based statistical flow features. In addition, MQTT control packet field size/length feature sets were evaluated on two datasets. The effectiveness of the proposed feature set was validated using three fundamentally different machine learning algorithms namely, AODE based on Naive Bayes, C4.5 based on Decision Tress and MLP based on ANN. The performance of the classifiers were tested with count-based flow features and field length features, to measure the detection accuracy of normal and attack classes. The MQTT DoS attack modelling results indicate that the adversaries can cause large scale impact with just basic access to the MQTT broker by launching the invalid subscription flooding attack. However, the invalid authentication attacks were found to cause little impact with a single attack source machine, as these attacks depended on a large volume of attack packets. In addition, using a malformed CONNECT request, a high memory utilization on broker machines was witnessed, which could be exploited during a memory-exhaustion attacks. The DoS detection model showed that the proposed MQTT features yielded high detection capabilities, especially when the control packet field size-length based features were selected. Hence, the packet size and field length distribution features can be effectively used in detecting DoS attacks in IoT networks. As future work, we intend to study a real MQTT dataset to assess its features, and how these can be beneficial in clearly de-marking legitimate and malicious MQTT traffic.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Notes on contributors

Syed Naeem Firdous is currently a PhD student at Edith Cowan University, Perth, Australia. He received his Bachelor of Engineering Anna University, Chennai, India in 2005 and MS from King Fahad University of Petroleum and Minerals, Dharan, Saudi Arabia in 2011. He has both academic and industry experience in networking and network security domain. His research interests include IoT, network security, and network forensics.

Zubair Baig is a Senior Lecturer in Cyber Security at the School of Information Technology, Deakin University. He has authored over 65 journal and conference articles and book chapters. He is currently serving as the editor of the *IET Wireless Sensor Systems Journal* and the *PSU - A Review Journal*, *Emerald Publishing House*. He has served on numerous technical program committees of international conferences and has delivered numerous keynote talks on cyber security. His research interests are in the areas of cyber security, artificial intelligence, smart cities and the Internet of Things.

Dr Ahmed Ibrahim is a Lecturer (Computing and Security) at the Edith Cowan University School of Science. Dr Ibrahim's Ph.D. research focused on detecting covertly hidden content in digital objects and electronic communication. He has previously worked as a Post-Doctoral Research Fellow at the ECU Security Research Institute and has held academic positions at the Maldives National University, Villa College, and National Institute of Technology Australia; and worked in the government of Maldives.

Professor Craig Valli has over 30 years experience in the ICT Industry and consults to industry and government on cyber security and digital forensics matters. Along with being Director of the Edith Cowan University Security Research Institute, he is a Professor of Digital Forensics, a Fellow of the Australian Computer Society and Director of the Australian Computer Society Centre of Expertise in Security at ECU. Craig is a member of the High Tech Crime Investigators Association (Australian Chapter) and the INTERPOL Cyber Crime Experts Group. Craig has over 100 peer reviewed academic publications in cyber security and digital forensics. He serves on several cyber security related conference committees and journal editorial boards. Craig is also the founder and current Chair of the Australian Digital Forensics Conference.

References

Adi, E., Baig, Z. A., Hingston, P., & Lam, C.-P. (2016, March). Distributed denial-of-service attacks against http/2 services. *Cluster Computing*, *19*(1), 79–86. doi: https://doi.org/10.1007/s10586-015-0528-7

Alanazi, S., Al-Muhtadi, J., Derhab, A., Saleem, K., AlRomi, A. N., Alholaibah, H. S., & C. Rodrigues., J. J. P. (2015). On resilience of wireless mesh routing protocol against dos attacks in IoT-based ambient assisted living applications. In *2015 17th International Conference on E-health Networking, Application Services (HealthCom)* (pp. 205–210), Boston, USA.

Andy, S., Rahardjo, B., & Hanindhito, B. (2017). 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)* (pp. 1–6), Yogyakarta, Indonesia.

Asplund, M., & Nadjm-Tehrani, S. (2016). Attitudes and perceptions of iot security in critical societal services. *IEEE Access*, *4*, 2130–2138. doi: https://doi.org/10.1109/Access.6287639

Aziz, B. (2016). A formal model and analysis of an iot protocol. *Ad Hoc Networks*, *36*, 49–57. doi: https://doi.org/10.1016/j.adhoc.2015.05.013

Ballani, H., & Francis, P. (2008). Mitigating dns dos attacks. In *Proceedings of the 15th ACM Conference on Computer and Communications Security, CCS '08*, New York, NY, USA (pp. 189–198). ACM. ISBN 978-1-59593-810-7. https://doi.org/10.1145/1455770.1455796

Bekara, C. (2014). Security issues and challenges for the iot-based smart grid. *Procedia Computer Science*, *34*, 532–537. doi: https://doi.org/10.1016/j.procs.2014.07.064ISSN 1877-0509. http://www.sciencedirect.com/science/article/pii/S1877050914009193. The 9th International

Conference on Future Networks and Communications (FNC'14)/The 11th International Conference on Mobile Systems and Pervasive Computing (MobiSPC'14)/Affiliated Workshops.

Bencsath, B., & Ronai, M. A. (2007). Empirical analysis of denial of service attack against smtp servers. In *2007 International Symposium on Collaborative Technologies and Systems* (pp. 72–79), Orlando, FL.

Borgohain, T., Kumar, U., & Sanyal, S. (2015). Survey of security and privacy issues of internet of things. *CoRR*, abs/1501.02211. http://arxiv.org/abs/1501.02211

Brenner, B. (2010). *Layer 7 increasingly under ddos gun*. https://www.csoonline.com/article/2124801/report–layer-7-increasingly-under-ddos-gun.html

Cohn, R. J. (2014). Individual Richard J Coppen, Andrew Banks, and Rahul Gupta. Mqtt version 3.1. https://docs.oasisopen.org/mqtt/mqtt/v3.1.1/csprd02/mqtt-v3.1.1-csprd02.html.

Durcekova, V., Schwartz, L., & Shahmehri, N. (2012, May). Sophisticated denial of service attacks aimed at application layer. In *2012 ELEKTRO* (pp. 55–60), Rajeck Teplice, Slovakia.

Eclipse (2018). *Eclipse paho*. https://www.eclipse.org/paho/.

EMQ (2018). *Emqx*. https://www.emqx.io/.

Fehrenbach, P. (2017). *Messaging queues in the IoT under pressure*. https://blog.it-securityguard.com/wp-content/uploads/2017/10/IOT_Mosquitto_Pfehrenbach.pdf

Firdous, S. N., Baig, Z., Valli, C., & Ibrahim, A. (2017). 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)* (pp. 748–755), Exeter, UK.

Gündoğan, C., Kietzmann, P., Schmidt, T. C., Lenders, M., Petersen, H., & Wählisch, M. (2018). Ndn, coap, and mqtt: A comparative measurement study in the IoT. arXiv preprint arXiv:1806.01444.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The weka data mining software: An update. *ACM SIGKDD Explorations Newsletter*, *11*(1), 10–18. doi: https://doi.org/10.1145/1656274.1656278

Heer, T., Garcia-Morchon, O., Hummen, R., Keoh, S. L., Kumar, S. S., & Wehrle, K. (2011, December). Security challenges in the ip-based internet of things. *Wireless Personal Communications*, *61*(3), 527–542. doi: https://doi.org/10.1007/s11277-011-0385-5

Houimli, M., Kahloul, L., & Benaoun, S. (2017). Formal specification, verification and evaluation of the MQTT protocol in the internet of things. In *2017 International Conference on Mathematics and Information Technology (ICMIT)* (pp. 214–221), Adrar, Algeria. IEEE.

Karagiannis, V., Chatzimisios, P., Vazquez-Gallego, F., & Alonso-Zarate, J. (2015). A survey on application layer protocols for the internet of things. *Transaction on IoT and Cloud computing, 3*(1), 11–17.

Karlik, B., & Olgac, A. V. (2011). Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, *1*(4), 111–122. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.740.9413&rep=rep1&type=pdf.

Kasinathan, P., Pastrone, C., Spirito, M. A., & Vinkovits, M. (2013). Denial-of-service detection in 6lowpan based internet of things. In *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (pp. 600–607), Lyon, France.

Koc, L., & Carswell, A. D. (2015). Application of an aode based classifier to detect dos attacks. *International Journal of Computer Science and Network Security (IJCSNS)*, *15*(2), 24.http://search.ijcsns.org/02_search/02_search_03.php?number=201502005.

Lee, S., Kim, H., Hong, D. k., & Ju, H. (2013, January). Correlation analysis of mqtt loss and delay according to qos level. In *The International Conference on Information Networking 2013 (ICOIN)* (pp. 714–717).

Little, J. D. C., & Graves, S. C. (2008). Little's law. In D. Chhajed & T. J. Lowe (Eds.), *Building intuition* (pp. 81–100). Springer. https://doi.org/10.1007/978-0-387-73699-0_5.

Luo, M., Peng, T., & Leckie, C. (2008). Cpu-based dos attacks against sip servers. In *NOMS 2008–2008 IEEE Network Operations and Management Symposium* (pp. 41–48), Salvador, Bahia, Brazil.

Luzuriaga, J. E., Perez, M., Boronat, P., Cano, J. C., Calafate, C., & Manzoni, P. (2015). A comparative evaluation of amqp and mqtt protocols over unstable and mobile networks. In *2015 12th*

Annual IEEE Consumer Communications and Networking Conference (CCNC) (pp. 931–936), Las Vegas, USA. IEEE.

Mantas, G., Stakhanova, N., Gonzalez, H., Jazi, H. H., & Ghorbani, A. A. (2015). Application-layer denial of service attacks: Taxonomy and survey. *International Journal of Information and Computer Security*, *7*(2-4), 216–239. doi: https://doi.org/10.1504/IJICS.2015.073028

Mektoubi, A., Hassani, H. L., Belhadaoui, H., Rifi, M., & Zakari, A. (2016). New approach for securing communication over MQTT protocol a comparison between rsa and elliptic curve. In *2016 Third International Conference on Systems of Collaboration (SysCo)* (pp. 1–6), Casablanca, Morocco. IEEE.

Mosquitto (2017). *Eclipse mosquitto*. https://mosquitto.org/.

Moustafa, N., Turnbull, B., & Choo, K. R. (2019). An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet of Things Journal*, *6*(3), 4815–4830. doi: https://doi.org/10.1109/JIoT.6488907

Oracle (2018). *Oracle virtual box*. https://www.virtualbox.org/.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. https://dl.acm.org/doi/10.5555/1953048.2078195.

Perrone, G., Vecchio, M., Pecori, R., & Giaffreda, R. (2017). 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* (pp. 246–253), Porto, Portugal.

Rafique, M. Z., Akbar, M. A., & Farooq, M. (2009). Evaluating dos attacks against SIP-based VoIP systems. In *GLOBECOM 2009–2009 IEEE Global Telecommunications Conference* (pp. 1–6), Honolulu, Hawaii.

Ranjan, S., Swaminathan, R., Uysal, M., Nucci, A., & Knightly, E. (2009, February). Ddos-shield: Dos-resilient scheduling to counter application layer attacks. *IEEE/ACM Transactions on Networking*, *17*(1), 26–39. doi: https://doi.org/10.1109/TNET.2008.926503

Roman, R., Zhou, J., & Lopez, J. (2013). On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, *57*(10), 2266–2279. doi: https://doi.org/10.1016/j.comnet.2012.12.018ISSN 1389-1286. http://www.sciencedirect.com/science/article/pii/S1389128613000054. Towards a Science of Cyber Security and Identity Architecture for the Future Internet.

Sadeghi, A., Wachsmann, C., & Waidner, M. (2015, June). Security and privacy challenges in industrial internet of things. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)* (pp. 1–6), San Francisco, CA, USA.

Sanfilippo, S. (2006). *hping*. http://www.hping.org/.

Santiago Hernández Ramos, M. T. V., & Lacuesta, R. (2018). MQTT security: A novel fuzzing approach. *Wireless Communications and Mobile Computing, 2018*, Article ID 8261746. https://doi.org/10.1155/2018/8261746.

Scalagent (2015). *Benchmark of mqtt servers*. http://www.scalagent.com/IMG/pdf/Benchmark_MQTT_servers-v1-1.pdf.

Shaker, V., & Zarrabi, H. (2017). Dos attacks in internet of things. *International Journal of Advanced Research in Computer Science*, *8*(5). https://doi.org/10.26483/ijarcs.v8i5.3217

Shan, H., Wang, Q., & Pu, C. (2017). Tail attacks on web applications. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17* (pp. 1725–1739). ACM. ISBN 978-1-4503-4946-8. http://doi.acm.org/10.1145/3133956.3133968.

Shin, S. H., Kobara, K., Chuang, C.-C., & Huang, W. (2016). A security framework for mqtt. In *2016 IEEE Conference on Communications and Network Security (CNS)* (pp. 432–436), Philadelphia, PA, USA. IEEE.

Singh, M., Rajan, M. A., Shivraj, V. L., & Balamuralidhar, P. (2015). Secure mqtt for internet of things (iot). In *2015 Fifth International Conference on Communication Systems and Network Technologies* (pp. 746–751), Gwalior, India. IEEE.

Singh, K., Singh, P., & Kumar, K. (2017). Application layer http-get flood ddos attacks: Research landscape and challenges. *Computers and Security*, *65*, 344–372. doi: https://doi.org/10.1016/j.cose.2016.10.005

Sivanathan, A., Habibi Gharakheili, H., Loi, F., Radford, A., Wijenayake, C., Vishwanath, A., & Sivaraman, V. (2018). Classifying iot devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, *18*(8), 1745–1759. doi: https://doi.org/10.1109/TMC.7755

TCPDUMP (2019). *Tcpdump*. https://www.tcpdump.org/.

Ten, C., Manimaran, G., & Liu, C. (2010, July). Cybersecurity for critical infrastructures: Attack and defense modeling. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, *40*(4), 853–865. doi: https://doi.org/10.1109/TSMCA.2010.2048028

Thangavel, D., Ma, X., Valera, A., Tan, H.-X., & Tan, C. K.-Y. (2014). Performance evaluation of mqtt and coap via a common middleware. In *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)* (pp. 1–6), Singapore. IEEE.

Vähä-Sipilä, A. (2015). *mqtt_fuzz*. https://github.com/F-Secure/mqtt_fuzz.

VerneMQ (2018). *Vernemq*. https://vernemq.com/.

Webb, G. I., Boughton, J. R., & Wang, Z. (2005, January). Not so naive bayes: Aggregating one-dependence estimators. *Machine Learning*, *58*(1), 5–24. doi: https://doi.org/10.1007/s10994-005-4258-6

Wireshark (2019). *Tshark*. https://www.wireshark.org/docs/man-pages/tshark.html.

Yokotani, T., & Sasaki, Y. (2016). Comparison with http and MQTT on required network resources for IoT. In *2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)* (pp. 1–6), Bandung, Indonesia. IEEE.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014, February). Internet of things for smart cities. *IEEE Internet of Things Journal*, *1*(1), 22–32. doi: https://doi.org/10.1109/JIOT.2014.2306328