SN Computer Science (2024) 5:488
https://doi.org/10.1007/s42979-024-02830-4

ORIGINAL RESEARCH

Explainable Anomaly Detection of Synthetic Medical IoT Traffic Using Machine Learning

Lerina Aversano[1] · Mario Luca Bernardi[2] · Marta Cimitile[3] · Debora Montano[4] · Riccardo Pecori[5,6] · Luca Veltri[7]

Received: 21 November 2023 / Accepted: 25 March 2024

Abstract
In the context of the Internet of Things (IoT), particularly within medical facilities, the detection and categorization of Internet traffic remain significant challenges. While conventional methods for IoT traffic analysis can be applied, obtaining suitable medical traffic data is challenging due to the stringent privacy constraints associated with the health domain. To address this, this study proposes a network traffic simulation approach using an open-source tool called IoT Flock, which supports both CoAP and MQTT protocols. The tool is used to create a synthetic dataset, to simulate IoT traffic originating from various smart devices in different hospital rooms. The study shows a complete anomaly detection analysis of IoT-Flock-generated traffic, both normal and malicious, by leveraging and comparing traditional machine learning techniques, deep learning models with multiple hidden layers, and explainable artificial intelligence techniques. The results are very promising. For the binary classification, for example, the obtained accuracy is close to 100% in the case of the CoAP protocol. Good results are also obtained when the multinomial classification is performed, observing that CoAP packets are classified better than MQTT packets, even if the identification of the different MQTT packets reaches very high metrics for the most of the considered algorithms. Moreover, the obtained classification rules are also meaningful in the considered IoT context. The results indicate that IoT-Flock synthetic data can effectively be used to train and test machine and deep learning models for detecting abnormal IoT traffic in medical scenarios. This research attempts also to bridge the gap between IoT security and healthcare, providing useful insights into securing medical IoT networks in general.

Keywords Medical internet of things · Anomaly detection · Intrusion detection systems · Machine learning · Explainable artificial intelligence · Deep neural networks

Introduction

The Internet of Things (IoT) refers to the integration of different devices across various technologies, enabling them to connect and communicate autonomously, without human

This article is part of the topical collection “Recent Trends on Data Science, Technology and Applications” guest edited by Slimane Hammoudi, Alfredo Cuzzocrea and Oleg Gusikhin.

✉ Riccardo Pecori
riccardo.pecori@cnr.it

Lerina Aversano
lerina.aversano@unifg.it

Mario Luca Bernardi
bernardi@unisannio.it

Marta Cimitile
marta.cimitile@unitelmasapienza.it

Luca Veltri
luca.veltri@unipr.it

[1] Department of Agricultural Science, Food, Natural Resources and Engineering, University of Foggia, 71122 Foggia, Italy

[2] Department of Engineering, University of Sannio, 82100 Benevento, Italy

[3] Department of Law and Digital Society, Unitelma Sapienza University, 00161 Rome, Italy

[4] CeRICT Scrl, 82100 Benevento, Italy

[5] SMARTEST Research Centre, eCampus University, 22060 Novedrate, CO, Italy

[6] Institute of Materials for Electronics and Magnetism, National Research Council of Italy, 43124 Parma, Italy

[7] Department of Engineering and Architecture, University of Parma, 43124 Parma, Italy

Published online: 24 April 2024

intervention. These IoT devices not only interact with each other, but also with a wide array of equipment, including industrial machinery, robots, drones, and energy generation systems, making them applicable in numerous sectors, including healthcare. IoT devices, especially when deployed in critical settings like healthcare, raise significant security concerns [1]. Traditional security measures, such as firewalls, intrusion detection systems (IDSs), and intrusion prevention systems (IPSs), are commonly used to safeguard devices and networks against cyber-threats. Presently, many firewalls and IPSs rely on predefined rules to filter out abnormal or malicious traffic. However, certain IDS and IPS systems also leverage Artificial Intelligence (AI) techniques to identify malicious traffic effectively [2]. As a matter of fact, combining AI methods with predefined rules enhances the ability to detect attack traffic compared with using only static rules. AI-based IDS and IPS systems are typically trained and tested using datasets that encompass both normal and attacking traffic. These datasets can be acquired through two approaches: (i) collecting actual traffic data from live systems to capture both malicious and normal traffic, or (ii) utilizing synthetic traffic generators that simulate real-time network traffic patterns. This research aims at a medical IoT environment, where obtaining real IoT traffic data proves challenging due to the stringent privacy constraints associated with patients' health information, even when anonymized. To address this issue and study this critical IoT scenario, we employed a network traffic simulation approach using an open-source tool called "IoT-Flock" [3]. IoT-Flock is capable of generating IoT traffic originating from various smart devices and supports two distinct application layer protocols commonly used in IoT scenarios: CoAP and MQTT. Furthermore, IoT-Flock offers the flexibility to create customized IoT use cases, allowing the inclusion of as many custom IoT devices as needed and the generation of both malicious and normal IoT traffic associated with these scenarios. This approach was chosen to overcome the challenges of acquiring real medical IoT traffic data and facilitate the study of this complex environment.

Major Contributions of the Work

The major contributions of this paper, which is an extension of the preliminary paper presented at DATA 2023 conference [4], include:

- The identification, for the first time as we know, of IoT-Flock-generated CoAP traffic;
- The comparison, for the first time as we know, of both machine and deep learning models on synthetic traffic generated by IoT-Flock;
- A thorough analysis of IoT-Flock-generated traffic by means of a packet-based feature model, thus considering features of single packets, through both a binary and a multinomial classification;
- The application of an inherent explainable machine learning technique, i.e., fuzzy Hoeffding decision tree, to extract useful rules to interpret the classification outcomes and evaluate if the obtained results have a significant meaning in the real-world context.

The final aim is to encourage the large usage of IoT-Flock as a tool to create IoT-based traffic datasets useful to train real-world effective AI-based IDSs in both hospitals and healthcare-related buildings.

Structure of the work

This work is structured as follows:

- Section " Background" offers background information concerning the IoT attacks that are under consideration in this paper and the algorithms used for the analyses of the considered IoT traffic;
- Section "Related Work" discusses related researches on the application of machine and deep learning algorithms for IoT attack categorization;
- Section "IoT-Flock" explains the utilization of IoT-Flock and outlines the considered use-cases;
- Section Experimental Settings" provides a detailed explanation of the regarded attributes, the considered data, the hyper parameters of the machine and deep learning models that we used for the classification analysis, as well as the evaluated performance metrics;
- Section "Results" reports the results of all the analyses we conducted, using both machine and deep learning models, and shows some explainable rules we extracted;
- Section "Discussion and Threats to the Validity" discusses the obtained results, the meaning of the output rules and some threats to the validity of the study;
- Section "Conclusions and Future work" concludes the article by offering guidance and proposals for future research developments.

Background

Attack Description

In this study, we examine the traffic generated by two widely used IoT protocols, namely MQTT (Message Queue Telemetry Transfer) [5] and CoAP (Constrained Application Protocol) [6]. Additionally, we investigate two types of attack for each protocol currently supported by IoT-Flock,[1] an

[1] https://github.com/ThingzDefense/IoT-Flock.

open-source tool, designed for generating synthetic IoT traffic, whose users can construct various IoT use-case scenarios with specific IoT devices and produce both legitimate and malignant IoT packets over a real-time synthetic network. A brief summary of the attacks analyzed in this paper is provided hereinafter:

- *MQTT packet crafting attack*: in this attack, MQTT packets are deliberately crafted to disrupt the MQTT broker. The attacker initiates a connection with the MQTT broker at the transport level and sends a malformed MQTT packet. This malicious packet has the potential to trigger a buffer overflow, causing certain MQTT broker implementations to crash, thereby rendering a Denial of Service (DoS) attack possible [7].
- *MQTT publish flood attack*: in this attack, MQTT publish messages are directed at a broker with high rate, with the intention of potentially initiating a Denial of Service (DoS) attack, by filling all the memory and/or processing capabilities of the broker.
- *CoAP segmentation fault attack*: this attack utilizes a vulnerability of a specific CoAP library, the LibNyoci,[2] for embedded systems that let a malformed Uri-Path option cause a possible segmentation fault, leading to a denial of service attack versus the received CoAP server device [8].
- *CoAP memory leak attack*: like the previous one, also this attack exploits a vulnerability in a CoAP implementation, the Eclipse Wakaama,[3] which gets a crafted packet with invalid options leading to a possible memory leak/waste (of 24 bytes) on the server side. This second-hand attack can be also used to exhaust memory resources leading to a DoS [9].

Employed Machine and Deep Learning Techniques

In this subsection, we summarize the main characteristics of the used machine and deep learning techniques, focusing also on the Fuzzy Hoeffding Decision Tree chosen as explainable AI model.

Machine Learning

Machine Learning (ML) is composed of algorithms and techniques capable of creating systems that learn and enhance their performance over time by leveraging the information inside the data themselves. The quality and nature of the data significantly impact the effectiveness of ML algorithms, as various data representations can either reveal or obscure the underlying explanatory factors driving the variations of the data. In recent years, many studies have been conducted on the classification of IoT traffic using often a supervised ML technique [4, 10].

This study, in particular, employs several algorithms such as:

- *Naïve Bayes* [11], a popular algorithm used for classification and probabilistic modeling. It is based on the Bayes' theorem, which is a probability theory that calculates the likelihood of an event occurring given prior knowledge. Naïve Bayes is particularly well-suited for scenarios with high-dimensional data, where other algorithms might struggle. However, its performance can be sub-optimal when the independence assumption is strongly violated, or when dealing with complex relationships within the data.
- *Support vector machine (SVM)* [12], a powerful and versatile supervised algorithm used for classification, regression, and outlier detection tasks. It works by finding an optimal hyperplane that best separates data-points in different classes. The term "support vector" refers to the data-points that are closest to the decision boundary (the hyperplane), in fact, they play a crucial role in SVM operations. SVM is widely used due to its flexibility and ability to handle both linear and non-linear problems effectively.
- *Logistic regression* [13] is a statistical and ML technique used for binary classification, which is employed to predict one of two possible outcomes based on one or more predictor features. Despite its name, it is not a regression algorithm, but a classification algorithm. Its simplicity, efficiency, and interpretability make it a really valuable classification technique.
- *Decision tree* [14], an algorithm that learns to make decisions by recursively splitting data into subsets based on the most significant features. All the decision rules form a tree-like structure, where each internal node represents a decision or test on an attribute, and each leaf node represents the outcome or class label.

More details about the settings of ML algorithms used in this work are discussed in subsection "Models Settings".

Deep Learning

Deep learning (DL) is a subset of ML, a field of AI that focuses on algorithms and models inspired by the structure and function of the human brain's neural networks. DL algorithms, also known as deep neural networks, are designed to model and understand complex patterns in large datasets. They have gained significant attention and success in various AI applications due to their ability to automatically learn hierarchical representations from data. DL is applied in a

[2] https://github.com/darconeous/libnyoci.

[3] https://www.eclipse.org/wakaama/.

wide range of fields, such as image and speech recognition, autonomous vehicles, recommendation systems, healthcare, finance, and natural language understanding. In recent years, there has been a growing interest in utilizing deep learning techniques also for the classification of IoT traffic [15–17].

In our work, we use a Deep Neural Network (DNN) to classify synthetic IoT traffic. A DNN has multiple layers between the input and output layers. These intermediate layers, the hidden layers, enable the network to learn increasingly abstract and complex representations of the data, making it capable of solving more intricate tasks, and they are a fundamental component of DL. More details about the architecture of our DNN model are reported in subsection “Models Settings”.

Fuzzy Hoeffding Decision Tree

The fuzzified version of the Hoeffding Decision Tree (FHDT) has been also used in this research work. This model entails a fuzzy discretization of the input features, resulting into a linguistic fuzzy partition [18]. FHDT can help in tackling explainability and interpretability issues because (i) it guarantees a balance between classification performance and the complexity of the model, (ii) fuzzy partitions associated with linguistic terms can make the interpretability of the extracted rules easier for a human being [19].

Two details make FHDT different from its traditional version [18]:

- The update process of the statistics in the internal nodes, because in this variant a training instance can reach more than one leaf, encompassing the fuzzy membership degree, the local fuzzy cardinality of the whole node, and the fuzzy cardinalities per class in a node;
- The computation of the fuzzy Information Gain and of the fuzzy Hoeffding bound that exploit the fuzzy cardinality instead of the traditional cardinality.

Moreover, the FHDT model can be explained through linguistic IF-THEN rules extracted from the tree model, by following the paths from the initial node (root) to the terminal nodes (leaves). Each rule is composed of some antecedents, regarding the attributes described through the linguistic terms associated with the fuzzy partitions, and a consequent returning the decided class.

Related Work

The concept of smart health and smart devices has undergone a radical transformation with the rapid growth of the IoT. As IoT-based health monitoring systems continue to evolve and become more intelligent, an increasing number of IoT-based smart health monitoring systems are being introduced and integrated into daily healthcare practices. However, in terms of security of healthcare systems, the IoT is still in its early stages of development. Unfortunately, given that the resources of IoT smart devices are truly limited, and given that the security requirements to be met are increasingly greater, especially in the healthcare sector, it is not possible to use the normal security solutions usually deployed for the protection of network traffic in a scenario dedicated to the medical IoT [20, 21]. Furthermore, IoT intelligent objects record information based on the specific use case, and, therefore, normal security solutions and protocols must be adapted, from time to time, based on the specific use and needs of the IoT scenario to be protected [20]. Therefore, it is necessary to ensure that there are no external attacks on the network of smart health devices, taking into account that IoT solutions applied in the medical field have limited resources. There are many studies regarding the security solutions to be adopted to protect IoT healthcare systems from external attacks. A first proposal in this sense details a technique for identifying replay attacks against battery-powered IoT healthcare equipment [22]. To detect and prevent such attacks, the authors suggest examining battery drain behavior, unique device IDs, and timestamps. The solution proposed by Rathore et al. [21] involves instead the adoption of the semi-supervised Fuzzy C-Means technique, based on Extreme Learning Machine (ELM), so that cyber-attacks can be detected in fog-based IoT systems. The use of Fuzzy C-Means is effective in addressing the difficulties associated with labeling datasets, while ELM techniques are useful for quickly and effectively detecting cyber-attacks. Furthermore, Alradashi et al. [23] propose a strategy capable of identifying malicious devices in a fog-based IoT healthcare scenario; in particular, this is a smart home equipped with a remote patient monitoring system. The authors use a system of online sequential ELMs capable of detecting different types of attacks such as distributed denial of service, man-in-the-middle, and other potential threats to the patient’s health. As already said, IoT security is currently a major concern. Currently, the most used solutions to protect IoT flows are anti-intrusion IDSs and IPSs. In fact, if you use datasets relating to both malicious and normal IoT network traffic it is possible to carry out an evaluation of the traffic. However, this involves the use of a large amount of data relating to IoT traffic to better train IDS and IPS. Multiple recent studies provide collections of real-world IoT data [15–17]; however, in these studies, it is possible to detect one of the main difficulties of the medical IoT scenario, as it is hard to collect adequate data, which follows all the rigorous delivery requirements that all hospitals and healthcare institutions are required to comply with. Furthermore, it is very difficult to collect useful traffic data via IoT systems in the real world, this is due to the very low throughput of some IoT devices and the various difficulties in collecting a large-enough number of instances of both normal and malicious traffic. Furthermore, various datasets are available online, mostly used for training and testing IDSs and IPSs, and they refer to both traditional networks and IoT networks. Indeed, the major studies in the sector in recent years use such data precisely. In particular, some are generated using real-time systems, and can be considered on a par with real datasets, while others are generated through simulation techniques, thus they are labeled as simulated and/or synthetic datasets [24–30]. Concerning the application of explainable models to the anomaly detection of IoT traffic, such a research topic has been investigated more and more in the last years [31]. For example, in [32], explainable deep learning, specifically, autoencoder-based LSTM and DNNs, is applied to an industrial IoT scenario and to a real world GSP system. In [33], an explainable variational autoencoder is applied to detect anomalies in NetFlow traffic. The explanations are provided through a gradient based fingerprinting technique applied to the UGR16 dataset containing anonymized NetFlow traces. The work in [34] is the closest to ours in terms of explainability applied to an IoT scenario: the authors apply a fuzzy decision tree to a real-world dataset containing flow-based features, while we focused on a synthetic dataset with packet-based attributes. In [35], the focus is on explaining the best anomaly detection techniques for a certain IoT dataset, while in [36], there is the claim that explainable neural networks have been applied to an Internet of Vehicles scenario by exploiting K-means clustering for feature scoring and ranking using two public datasets, namely CICIDS2019 and UNSW-NB15. However, explainable results are not discussed, focusing on performance metrics instead. In [37], the focus is again on an industrial scenario, specifically industrial control systems, and the explainability is provided through an LSTM-based autoencoder, with one class SVM and gradient SHAP values applied to a SCADA dataset.

For the work we propose, we have taken inspiration from the Bot-IoT dataset [38], which simulates IoT traffic produced by some normal and attacker virtual machines. Unlike this, however, the traffic we generated uses IoT-Flock as a simulator, which, unlike Bot-IoT, takes the traffic generated by both MQTT and CoAP protocols into account. This allowed us not only to recognize the type of traffic, be it normal or malicious, but also to classify the types of attacks based on the used IoT protocol. Moreover, we applied, for the first time, ante-hoc explainable ML to the synthetic traffic generated by IoT-flock, in order to verify whether the resulting rules have a real-world meaning and thus confirming the feasibility of using IoT-Flock as a tool for training AI models to be tested in real-world scenarios afterwards. We have chosen to use the fuzzy version of the Hoeffding Decision Tree as an explainable ML algorithm, an algorithm usually adopted in stream mining and so perfectly matching the variable nature of IoT traffic flows.

IoT-Flock

In order to generate the data used in this study, we leveraged IoT-Flock [3], an open-source IoT traffic generator that operates in real-time and through which different use cases may be designed, characterizing them with different intelligent devices. IoT-Flock manages to generate both regular and anomalous IoT traffic, a function most commercial and open-source tools lack; more precisely, they do not support the creation of malicious devices in the same use case. This functionality is particularly useful because it allows one to provide more adequate IDSs and IPSs. Furthermore, IoT-Flock provides for the possibility of exporting the designed use cases in XML format and of importing the XML generated using both IoT-Flock itself and other tools. Furthermore, the tool also allows one to generate a series of recent attacks, specifying them based on the protocol in use, MQTT and COAP; this feature is not present in any of the open-source IoT traffic generators.

IoT-Flock can work in two ways, i.e., via a GUI or via a command line. In both modes, to generate a single intelligent device, it is necessary to provide a series of relevant information about it, both functional and non-functional. In particular, the first type of information concerns the working behavior of the device, such as the type of device (normal or malicious), the used protocol (MQTT and/or CoAP), the data profile (type and scope of transmitted data), the time profile (periodic or random), the type of command (Subscribe or Publish in MQTT, Post or Get in CoAP). It is the latest information that distinguishes one IoT device from another one by IP address, device name, number of devices of the same type, etc.

Considered Use Case

In our research, we explored a scenario similar to the one discussed in a previous study [39]. This scenario involves two hospital rooms, each equipped with various devices that are meticulously controlled through smart sensors and actuators connected to the Internet. In both rooms, we can distinguish two categories of devices:

- The devices responsible for monitoring and regulating the environment of the room. These devices use the MQTT protocol to communicate through a dedicated

broker. Their primary role is to autonomously manage and maintain the comfort of the room and environmental conditions.
- The devices designed for monitoring the physical status of the patients occupying the room. These devices employ the CoAP protocol to communicate with a designated server, which can be accessed by healthcare personnel. Their purpose is to supervise and report on the health and well-being of the patients.

This setup illustrates how IoT technology is applied in a healthcare environment, facilitating both patient care and the management of the hospital environmental conditions.

Specifically, the smart devices that use the MQTT protocol establish communication with a broker that functions as an environment control unit. These MQTT-based smart devices encompass nine distinct types and operate at three different Quality-of-Service (QoS) levels. To provide a concise overview, the nine types of MQTT-based smart devices are summarized hereinafter:

- *Light intensity sensor/actuator*: publisher/subscriber type devices. The sensor publishes periodically, namely every second, data on the light detected in the environment on the "Light Intensity" topic. The actuator receives data from both the light publisher device and the movement sensor to fade in or out the light on the basis of the external illumination of the room and the movements in the room itself;
- *Temperature sensor/actuator*: publisher/subscriber type devices. The sensor publishes periodically, every 2 seconds, ambient temperature data on the topic "Temperatures". The actuator receives the temperature values and tries to keep the room temperature at a constant level, i.e., about 20°C;
- *Humidity sensor*: every 1 second it publishes ambient humidity data in the "Humidity" topic. It is a publisher-type device;
- *Motion sensor*: publishes data on movements, occurring in the room, in the "Movement" topic. Unlike the other sensors that publish data periodically at constant time intervals, the sensor of motion publishes data pseudo-randomly in the $1-5$-second interval;
- *CO-GAS sensor*: it receives and publishes data relating to gases detected in the room in the "CO-GAS" topic. The frequency of the publications is random in the range between 1 and 5 seconds;
- *Smoke sensor*: it is a publisher device. On the topic "Smoke", it shares in random intervals, in the range between 1 and 5 seconds, the data relating to the surveys on the presence of smoke;
- *Fan sensor*: it publishes data every 3 seconds related to fan operations in the "Fan" topic;
- *Fan speed controller*: it is an actuator and every second it receives data on the topic "Fan Speed", related to the speed of the fan present in the room. It receives also data from the topics "Smoke", "CO-GAS", "Humidity", "Door Lock", and "Temperatures";
- *Lock*: it publishes data relating to the status of the lock with a random frequency between 1 and 5 seconds in the "Door Lock" topic.

In the case of CoAP devices, each bed in the rooms is equipped with nine smart devices, and there is a CoAP server that works as central control unit. The CoAP server has various duties, including: managing time profiles for patients, regulating the quantity of medication administered to the patient through an infusion pump, initiating alarms for the medical staff based on the patient's health status as monitored by the smart sensors. The nine smart devices that utilize CoAP for communication are the following and are described below:

- *ECG sensor*: it provides information about the heart beat rhythm every 1 second;
- *Infusion pump*: an actuator used to deliver possible nutrients and drugs to the patients, retrieving data from the server every 10 minutes;
- *Pulsoximeter*: a smart sensor furnishing the oxygen saturation in the blood every 1 second;
- *Mouth airflow sensor*: a smart sensor providing the breathing rate of the patient every 1 second;
- *Blood pressure sensor*: a smart sensor conveying information about blood pressure every 2 seconds;
- *Glucometer*: a smart sensor providing information about the glucose in the blood every 10 minutes;
- *Body temperature sensor*: a smart sensor measuring the temperature of the patient every 1 hour;
- *EMG sensor*: a smart sensor measuring the electromiography, i.e., the potential produced by the body muscles, every 5 minutes;
- *GSR sensor*: a smart sensor measuring the galvanic skin response, i.e., the electrical conductance of the skin, every 5 minutes.

In Fig. 1, we illustrate the medical IoT scenario simulated in this study. In this scenario, all devices share a uniform configuration, utilizing the same range of private IP addresses for both MQTT and CoAP devices. The sensor network is confined to a secure and controlled access area, where the devices establish communication with the MQTT broker or the CoAP server. Notably, the simulated network does not include additional components such as firewalls or routers. The traffic within the network is monitored and captured by a Tshark process running in the

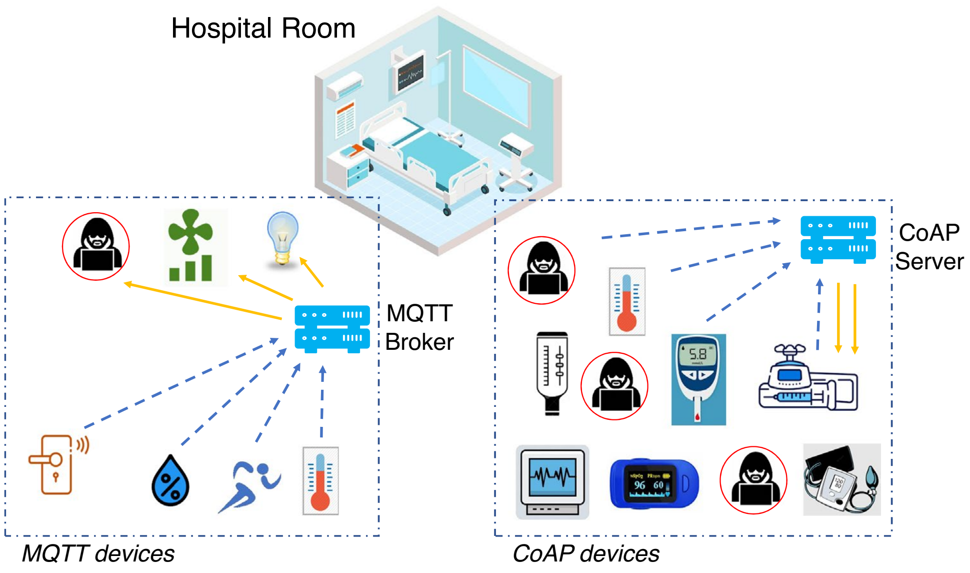


Fig. 1 The smart health scenario we considered: MQTT devices are on the left and monitor the environment, CoAP devices can be seen on the right and control the patient conditions. MQTT publish messages or CoAP POST and GET packets are represented by means of blue dashed lines, while MQTT delivery messages or CoAP response messages are represented through orange solid lines. Malicious nodes are present in both networks

background on the computer that is executing IoT-Flock, facilitating the collection and analysis of IoT traffic data.

Simulated Attacks

For the malicious traffic simulated in this study, we assumed that there is a specific number of malicious devices within the private network, controlled remotely by an attacker, and that this is not adequately protected.

During the simulation of an attack, malicious intelligent devices are directly connected to the MQTT broker or CoAP server to execute one of the considered attacks. In this study, we focused on the analysis of the type of attack and the traffic corresponding to it, while the methodology used to carry out the attacks and/or the way in which the attacker gained control of the node were not discussed as they go beyond the scope of this work.

Experimental Settings

The IoT Traffic Dataset

After the setup of the scenarios, we ran IoT-Flock for the actual generation of IoT packets. While the traffic was being generated, we captured IoT packets, with the help of Tshark,[4] a tool capable of analyzing the traffic generated through IoT-Flock in real-time. The output of running Tshark was a standard.pcap file which we have further processed in order to obtain the considered features and a final.csv file. The capturing phase of IoT packets lasted about 48 hours, 24 hours per considered protocol, on a machine equipped with an Intel Core i7 7^{th} generation CPU, 16GB of RAM, and one NVIDIA GeForce GPU. More precisely, the duration of capturing normal packets was 12 hours, while MQTT and CoAP attack packets have been captured for 6 hours each. When considering the attacks, we supposed the presence of 4 malicious smart devices, in the corresponding private network, that were sending packets following either a periodic (1s) or random $(1-5s$ range) time profile.

Table 1 Description of the considered IoT traffic dataset

Protocol	Number of instances	Type of traffic
CoAP	834,881	Normal traffic
	200,003	Segmentation fault attack
	198,090	Memory leak attack
MQTT	593,363	Normal traffic
	29,406	Publish flood attack
	1532	Packet crafting attack

In order to discriminate between malicious and benign traffic, we have extracted, by using a BASH script outputting the final.csv file, all the features available in Tshark for both the MQTT[5] and the CoAP[6] protocols.

Thereafter, we performed, using a proper Python script, a cleaning phase in order to tear out the features having no values (NaN), the source and destination IP addresses, as well as the non-significant, and to divide features incorporating multiple meaning into different columns.

4 https://tshark.dev/setup/install/.

5 https://www.wireshark.org/docs/dfref/m/mqtt.html.

6 https://www.wireshark.org/docs/dfref/c/coap.html.

Table 2 Considered MQTT features

Name	Type
mqtt.packet_type1,2,3	Categorical
mqtt.hdr_flags1,2,3	Categorical
mqtt.kalive1,2,3	Numeric
mqtt.conflag.cleansess1,2,3	Categorical
mqtt.conack.flags1,2,3	Categorical
mqtt.len1,2,3	Numeric

Table 3 Considered CoAP features

Name	Type
coap.code	Categorical
coap.opt.type1,2,3	Categorical
coap.opt.length1,2,3	Numeric
coap.opt.desc1A,2A,3A	Categorical
coap.opt.desc1B,2B,3B	Categorical
coap.opt.block_size	Categorical
coap.opt.observe	Categorical
coap.opt.end_marker	Categorical
coap.payload_length	Numeric

The final dataset, used for our analyses, consists of 1, 857, 275 instances, each of them corresponding to a packet. Table 1 describes the dataset, in particular, the first column reports the application protocol, the second one shows the number of records per type of traffic, shown in the third column. The type of traffic represents the classification label considered in the following analyses, performed independently for each considered application protocols, i.e., MQTT and CoAP.

Hence, the dataset described in Table 1 has been split in 2 different subdatasets, one for CoAP traffic with 1, 232, 974 packets, and one for MQTT traffic with 624, 301 packets.

Moreover, we used both a binary dataset, with instances/packets assigned only to the "Normal" or "Attack" class, and a multinomial dataset, to classify the four particular types of attacks as described in Subsection "Attack Description" and the two types of normal traffic (MQTT and CoAP). In the binary classification we had 1, 428, 244 packets for the "Normal" class and 429, 031 packets for the general "Attack" class.

The Feature Set

Since Tshark extracts a total of 78 features for MQTT packets and 86 features for CoAP packets, we decided to reduce them by trying to consider only the most significant ones. This was done by means of a thorough analysis campaign leveraging both automated statistical analyses and manual inspections. The outcome of this preliminary phase is shown in Tables 2 and 3, reporting the selected features for MQTT and CoAP, respectively, as well as their inherent type (numeric or categorical).

As concerns MQTT, we considered the following 6 features:

- Packet type (CONNECT, AUTH, PUBLISH, SUBSCRIBE, etc.): this feature and the next one have been extracted from the Tshark feature 'mqtt.hdrflags';
- Header flags, that is the flag bits present in the first byte of the fixed header of an MQTT packet;
- Keep alive interval (measured in seconds), that is the maximum time interval between two MQTT control packets sent by the client;
- Clean session flag of CONNECT message;
- Connect acknowledge flags, that is flag bits of CONNACK packets;
- Packet length.

Since each captured IP packet concerning the MQTT traffic, that corresponds to a TCP segment in turn, may contain more than one MQTT packet, for most of the features mentioned above Tshark provides three different values corresponding to possible three different MQTT packets within the same IP packet (it considers a maximum of three concatenated packets). Therefore, concerning MQTT traffic we considered a total of $3 \times 6 = 18$ features per single IP packet.

As regards CoAP, we considered the following 9 features:

- CoAP code: it discriminates between requests and responses and provides either the type of request (GET, POST, PUT, or DELETE) or response code;
- Option type: when one or more options are present, it indicates the type of each option;
- Option length: the size (in bytes) of the option;
- Option descA: it distinguishes between critical or elective options;
- Option descB: it distinguishes between safe or unsafe options;
- Option block size: the transfer block size specified in the Block1 and Block2 options in case the CoAP blockwise transfer extension is used [40];
- Option observe value: the number used to identify a given notification within a sequence of resource observations, when the CoAP Observe extension is used [41];
- Option end marker: it indicates the end of the options field when a payload is present;
- Payload length: the payload size in bytes.

Table 4 Hyper-parameters of the ML models we considered

Classifier	Hyper-parameters	Description	Used value
Naïve Bayes	Priors	This is the prior probability of the classes. If the value is 'none' the prior probabilities are calculated based on the data	None
	Var_smoothing	It is the portion of the biggest function variance of all features, and shows how it contributed to the others for computation stability	10^{-9}
SVM	Gamma	Kernel coefficient	2
	C	It is a parameter of regularization; in particular, regularization is inversely proportional to the value of C	Squared L2
Logistic	Penalty	The value of the considered penalty	L2 penalty
	Class_weight	The value of the class-specific weights	1 (for all classes)
Decision tree	Max_depth	The maximum depth of the tree	10
	Criterion	The criterion selected to perform splits at each internal node of the tree	Gini impurity

Since more than one option may be present within the same CoAP message, the Tshark tool provides the feature values for the first three options of each packet. This allowed us to consider triples of features for the *option type*, *option length*, *option descA*, and *option descB*, leading to a total of 17 features for each packet.

Models Settings

For the classification tasks, both binary and multinomial, we analyze different ML models and one Deep Neural Network (DNN) architecture. Moreover we have also applied an ante-hoc explainable model, the already mentioned FHDT.

In Table 4, we present the hyper-parameters used for the ML models of the study. The table is structured as follows: the first column shows the ML classifier, the second column lists the name of the hyper-parameter, the third column offers a short description of the hyper-parameter itself and its purpose, while the last column indicates the specific values that have been employed in the analysis. The hyper-parameter settings presented in the table were chosen following a grid-search optimization process. These settings were determined using an 80/20 ratio to divide the dataset into training and test sets, and they represent the configurations that yielded the best performance in the experiments.

On the contrary, the provided DNN model architecture is outlined in Table 5. In particular, the first column designates the hidden layer level, the second column specifies the type of the hidden layer used at that particular level, while the last column details the number of neurons within that specific layer. In this architecture, two different types of layers have been employed, each serving distinct purposes within the DNN model:

- *Dense layer*, a fully connected layer, in which every neuron in the next layer is connected to every other neuron in the previous layer. Its output value becomes the input for the next neuron layer;
- *Dropout layer*, a layer used to set input units to 0 at a chosen rate at each step during training. When the input is not set to 0, it is increased according to the formula $1/(1 - rate)$ so that the total number of input is constant.

Table 5 The DNN model architecture we used

Layer	Type	Neurons
Ia	Dense	1024
Ib	Drop out	–
IIa	Dense	512
IIb	Drop out	–
IIIa	Dense	128
IIIb	Drop out	–
IVa	Dense	32
IVa	Drop out	–
V	Softmax	4

The other components of the DNN models can be detailed as follows:

- As an Activation function we chose the rectified linear unit activation function via *'relu activation'* to all neurons in all considered layers;
- As an Optimizer we applied the *Adam* optimizer, a particular type of stochastic gradient descent. It is useful and widely used because it converges in a short time and, therefore, makes the network less demanding from a computational point of view compared with the classic SDG which converges to 'flat minima' [42];
- Concerning the Dropout rate of the relative layer, we set it to 0.20.

Table 6 Values of the main hyper-parameters for FHDT

Parameter	Value
Split confidence (δ)	10^{-7}
Tie threshold (TT)	2.5
Grace period (GP)	25
Minimum fraction (MF)	0.01

The architectural design of the DNN was implemented using Python, with a specific focus on TensorFlow[7] and Keras[8] libraries:

- TensorFlow is an open-source software framework widely used in the various sub-fields of AI. It is particularly valuable for tasks related to training and inference in DNNs.
- Keras is a Python package that serves as a user-friendly interface to the TensorFlow library. It simplifies the process of building and working with artificial neural networks, making it easier for developers and researchers to create complex models efficiently.

These libraries provide the tools and resources necessary for developing and training DNNs for a variety of applications, including the analysis and classification of IoT traffic in the context of the proposed research. In this study, all the neural network models were trained for a total of 100 epochs. The training data have been divided into three sets using a 60/20/20 splitting ratio for the training, validation, and test sets, respectively. The specific hyper-parameters, including the dropout rate, number of epochs, and batch size, were meticulously selected after a process of grid-search optimization, involving the evaluation of various combinations to determine the most effective configuration for the neural network models. The chosen settings reflect the combination that yielded the best performance and results for the objectives of the study.

As concerns the FHDT, only triangular, strong, and uniform fuzzy partitions have been used, considering 3, 5, and 7 triangles in the $[0-1]$ range. Each partition triangle has been associated with a particular linguistic term, e.g., Low, Medium, and High in case of three triangles, or Very Low, Low, Medium, High, and Very High in case of five triangles. As for the categorical features, we took care to assign fixed numerical values in the considered range $[0-1]$, corresponding to the upper vertex of a triangle, while numeric features were simply normalized accordingly. In Table 6, we detail the main hyper-parameters of the FHDT model we considered as well as their corresponding considered values, obtained after a grid search optimization procedure. Also in the case of FHDT, we used as a training set 80% of the whole dataset and 20% for testing.

[7] https://www.tensorflow.org/.

[8] https://keras.io/.

Evaluation Metrics

In order to evaluate the performance, we used several specific metrics to gain a comprehensive understanding. In this work, the following evaluation metrics, in addition to confusion matrices, have been taken into account:

- *Accuracy*, which provides an overall measure of how often the model correctly classifies items in the dataset concerning the total number of instances. It is a useful general indicator of the performance of a model.
- *Weighted precision*, which is calculated by dividing the number of true positives by the total number of instances marked as belonging to a particular class. It provides insight into the precision of the classifier, taking into account class imbalances.
- *Weighted recall*, which is defined as the number of true positives divided by the total number of instances that genuinely belong to a specific class. It assesses the ability of the classifier to correctly identify instances of a particular class while considering the class distribution.
- *Weighted F1-score*, which is the harmonic mean between precision and recall. It balances the trade-off between precision and recall, offering a more comprehensive measure of the performance of a model, especially in cases of class imbalance.

The considered performance metrics, in combination with confusion matrices, provide a well-rounded view of how well each classifier is performing, especially in our case study with unbalanced datasets wherein it is essential to account for the differences in class distribution.

Beside the aforementioned metrics, when considering FHDT, we have also regarded the total number of nodes in the tree and the number of leaves as metrics useful to assess the complexity of the model and so of its interpretability: less nodes and leaves allow for better interpretability of the corresponding rules. Moreover, the criterion we used to perform inference in the FHDT, given that a testing instance could reach more than one leaf, was the majority voting strategy.

Results

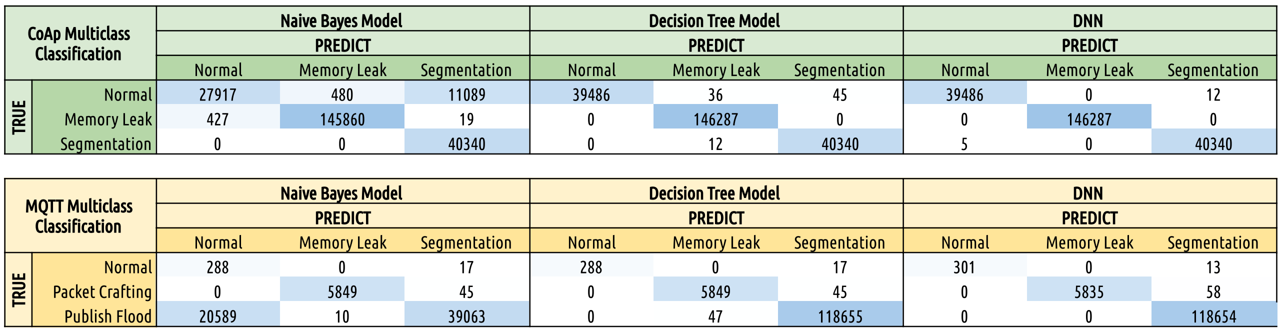
Hereunder we presents the results obtained in the analyses carried out for this research. The evaluation of all models is carried out separately for both MQTT and CoAP protocols. To enhance the readability of the presented results, this section is divided into two subsections. First, we present the

Table 7 Results of the binary classification

AI model	CoAP				MQTT			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
Naïve Bayes	0.915	0.966	0.937	0.939	0.917	0.866	0.942	0.905
SVM	0.991	0.990	0.988	0.989	0.975	0.979	0.975	0.977
Logistic reg.	0.969	0.977	0.981	0.979	0.981	0.988	0.984	0.986
Decision tree	0.988	0.989	0.989	0.989	0.979	0.979	0.965	0.977
DNN	0.993	0.992	0.995	0.994	0.989	0.990	0.989	0.990
FHDT-3FS	0.864	0.983	0.981	0.981	0.984	0.997	0.997	0.997
FHDT-5FS	0.885	0.993	0.991	0.991	0.994	0.998	0.998	0.998
FHDT-7FS	0.893	0.994	0.992	0.992	0.995	0.998	0.997	0.998

Table 8 Results of the multinomial classification

AI model	CoAP				MQTT			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
Naïve Bayes	0.948	0.936	0.929	0.948	0.835	0.670	0.921	0.642
SVM	0.971	0.982	0.970	0.983	0.928	0.929	0.948	0.949
Logistic reg.	0.966	0.945	0.956	0.966	0.929	0.918	0.969	0.939
Decision tree	0.991	0.992	0.996	0.997	0.999	0.999	0.979	0.989
DNN	0.999	0.999	1.00	0.999	0.999	0.993	0.994	0.993
FHDT-3FS	0.866	0.981	0.983	0.983	0.983	0.961	0.997	0.961
FHDT-5FS	0.871	0.99	0.989	0.989	0.984	0.971	0.996	0.972
FHDT-7FS	0.895	0.991	0.99	0.99	0.985	0.973	0.995	0.982



CoAp Multiclass Classification		Naive Bayes Model			Decision Tree Model			DNN		
		PREDICT			PREDICT			PREDICT		
		Normal	Memory Leak	Segmentation	Normal	Memory Leak	Segmentation	Normal	Memory Leak	Segmentation
TRUE	Normal	27917	480	11089	39486	36	45	39486	0	12
	Memory Leak	427	145860	19	0	146287	0	0	146287	0
	Segmentation	0	0	40340	0	12	40340	5	0	40340

MQTT Multiclass Classification		Naive Bayes Model			Decision Tree Model			DNN		
		PREDICT			PREDICT			PREDICT		
		Normal	Memory Leak	Segmentation	Normal	Memory Leak	Segmentation	Normal	Memory Leak	Segmentation
TRUE	Normal	288	0	17	288	0	17	301	0	13
	Packet Crafting	0	5849	45	0	5849	45	0	5835	58
	Publish Flood	20589	10	39063	0	47	118655	0	0	118654

Fig. 2 Confusion matrices of both MQTT and CoAP multinomial classification as regards some considered classification models

results of the binary classification (see subsection "Binary Classification"), followed by the results obtained through the multiclassification (see subsection "Classification of Attacks»). This division facilitates a clear and organized presentation of the outcomes for each classification approach.

Binary Classification

This subsection discusses the results of the binary classification; hence, in this phase, we have considered the packets referring to the studied attacks as a single malicious class.

Table 7 shows the values of the evaluation metrics considering the CoAP and MQTT datasets separately. The analysis shows that the classification accuracy is very high, reaching, in the case of the CoAP protocol, values close to 100% for both some ML models and the DNN, while FHDT performs slightly worse, considering all three types of fuzzy sets (FSs), but, anyway, close to 90% of accuracy and with much more explainable and compact models as described in Subsection "Explainable Rules". Furthermore, all the considered metrics achieve close to optimal results, indicating that the considered algorithms classify the data really well even in cases of unbalanced classes. The best model, if one considers the CoAP protocol and takes a look at all

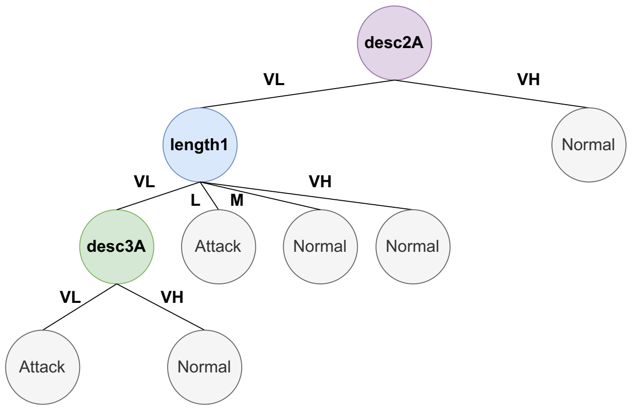


Fig. 3 FHDT resulting after training on the CoAP binary dataset when considering a partition of 5 fuzzy sets

the metrics, is the DNN we have taken into account, but one can see that the difference with SVM or Decision Tree is very small. When considering the MQTT protocol alone, the models that best classify are the DNN and the FHDT, in this latter case with a slight increase in the performance, experienced also in the CoAP case, moving from 3 fuzzy sets (3FS) to seven fuzzy sets (7FS).

Classification of Attacks

This subsection reports the results arising from the multinomial classification analysis for both MQTT and CoAP. The specific results are shown in Table 8. CoAP packets, both normal and malicious, are classified better than MQTT packets. Furthermore, the identification of the different MQTT packets reaches very high figures for all the algorithms we considered, except for the Naïve Bayes classifier, which presents the worst performance, also worse than all versions of FHDT.

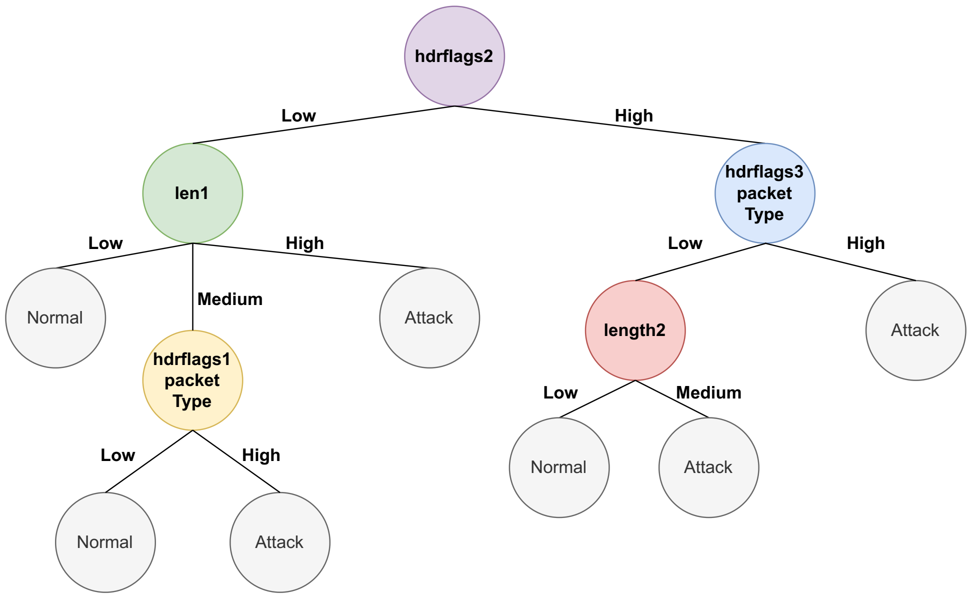
Finally, Fig. 2 shows the confusion matrices for the multinomial classification of the MQTT and CoAP protocols. In particular, we have only reported the results obtained for Naïve Bayes, Decision Tree, and the DNN model, for the sake of brevity. From the figure, it can be inferred that for the CoAP protocol, the only relevant misclassifications occur when using the Naïve Bayes classifier; in particular, the normal packets are those most confused with the segmentation attack packets. In contrast, the decision tree classifier and the DNN architecture we considered are the ones with fewer classification errors: for these algorithms, errors between predicted data and real data are very rare and mainly involve normal traffic when classified with decision trees.

By the same token, when considering MQTT, no model produces a perfect diagonal matrix and when using Naïve Bayes we obtain the worst outcome whose main criticality entails the identification of Publish Flood packets as normal ones. In the decision tree model, the main criticality concerns the confusion of packet crafting packets with publish flood packets or vice versa. This reciprocal misclassifcation of attack packets does not happen in the case of the considered DNN model, which has only a few problems in discriminating packet-crafting packets from publish-flood packets.

Explainable Rules

In this subsection, we report the graphical representation of some of the obtained FHDTs, limiting the analysis to the binary case because the multinomial trees are exactly the same but, obviously, with more than two outcomes. We have

Fig. 4 FHDT resulting after training on the MQTT binary dataset when considering a partition of 3 fuzzy sets



not reported the traditional decision trees, because they were poorly interpretable for their complexity as well as for the lack of associated linguistic terms.

In Fig. 3, we report the binary FHDT when considering 5 fuzzy sets applied to the features of the CoAP protocol, while, in Fig. 4, we show the binary FHDT applied to the features of the MQTT dataset when considering 3 fuzzy sets. The trees we have shown are the most compact and interpretable we obtained among the various combinations of fuzzy sets, although they all have achieved very similar performance results as shown in Tables 7 and 8. Indeed, the 5-fuzzy-set FHDT for CoAP has 9 total nodes and 6 leaves, while the 3-fuzzy-set version has 13 total nodes and 8 leaves, and the 7-fuzzy-set version features 29 total nodes and 24 leaves. As regards MQTT, the 3-fuzzy-set FHDT has 12 total nodes and 7 leaves, while the 5-fuzzy-set version has 21 total nodes, and 16 leaves and the 7-fuzzy-set version has 29 total nodes and 24 leaves.

In the case of CoAP, a very high value of the *desc*2*A* attribute leads to the normal classification, while a very low value requires a test of the *length*1 feature. Attacks take place when *length*1 is low as well as *desc*2*A* is very low, or when *length*1 is very low and *desc*3*A* is very low as well as *desc*2*A*.

In the case of MQTT, the most discriminating feature is *hdrflags*2 and attacks can take place when it is both low or high. In the first case *len*1 has to be high or medium, but with a high value associated to *hdrflags*1*_packet_Type*. In the second case when *hdrflags*3*_packet_Type* is high or it is low, but with a medium value for *length*2.

Two examples of the extracted rules, for both CoAP and MQTT, are the following:

R1: **IF** *desc*2*A is VERY LOW* **AND** *length*1 *is LOW* **THEN** *CoAP Attack*

R2: **IF** *hdrflags*2 *is LOW* **AND** *length*1 *is HIGH* **THEN** *MQTT Attack*

Discussion and Threats to the Validity

When considering both the binary and multinomial classification, we may see that the CoAP protocol is classified better than the MQTT protocol, except when using the FHDT. However, this could arise from the extreme lightness and simplicity implicit in CoAP implementations. As a matter of fact, such a protocol is routinely used on battery-powered devices and in case of limited CPU and RAM. Moreover, it is fair to say that a CoAP packet can be only 4B long, as opposed to a HTTP messages that can have a minimum of 33 bytes for a request and 16B for a response and that are usually significantly longer (hundreds or thousands of bytes). Additionally, during the testing phase of the hyper-parameter configuration for the regarded models, the CoAP protocol returned results with metrics always close to 100%, regardless of the adopted configuration. Therefore, we have been able to notice that packets related to the CoAP protocol can be subject to the so-called “benign overfitting”: this condition occurs when a classifier adapts perfectly to noisy training data, while keeping the error between predicted data and real data very low [43]. As a corroborating information, in our study, the loss function related to CoAP always tends to 0 when using the DNN model.

Contrary to what occurred for the CoAP protocol, the MQTT protocol is classified with a higher error between predicted data and observed data. However, we have to remember that the MQTT protocol involves both a connection, being encapsulated in TCP, and three distinct entities (the publisher, the broker, and the subscriber), compared with the only two involved in CoAP (client and server); this makes it a much more complex protocol to identify.

As regards the obtained rules, for CoAP traffic we see that a deciding aspect appears to be whether the packet contains some options (at least two), and they belong to the ‘critical’ class (that corresponds to the features descA with value very low), that means that the given option needs to be understood and processed by the message destination. This requires at least a processing overhead at the receiver side and may be used for exploiting possible software vulnerabilities in case they are present.

On the other hand, for detecting MQTT attacks, we have that it is important whether the IP packet contains different MQTT messages; this is evident because of the presence of features marked with 1, 2, and 3, related to three different MQTT messages within a single TCP/IP packet. This conditions may be more probable in case of flooding/DoS attacks. In addition, also the type of MQTT packet is relevant, since some methods like UNSUBSCRIBE or DISCONNECT, corresponding to the ‘packet_type’ feature with high values, are mainly used in some attacks.

As for the threats to the validity of our study, we discuss construct, internal, and external validity threats in the following.

Regarding construct validity threats, they usually entail how well a set of indicators represents or reflects a concept that is not directly measurable. In our study, we have focused on the main features of the packets of both MQTT and CoAP created during various simulation sessions with different random seeds and we have accurately removed statistically non-significant features.

When focusing on the internal validity, we exclude any labeling issues because the traffic was synthetically created through IoT-Flock in a controlled environment.

Concerning the threats to external validity, which affect the generalization of the discussed outcomes, we still have to test whether training on the synthetic traffic generated through IoT-Flock and testing on real-world data could lead to similar very good results. However, the rules extracted from the FHDT confirm a real-world significant meaning in the identification of the CoAP and MQTT attacks and thus we can envisage its useful application also to a real-world scenario.

Conclusions and Future work

In this paper, we have applied a complete and explainable anomaly detection analysis, performed via machine and deep learning techniques, to the synthetic traffic produced trough IoT-Flock when considering a smart health scenario. The research has encompassed both a binary and a multinomial classification; moreover, we have considered both MQTT and CoAP messages, i.e., the most used application protocols in the IoT scenario, and both normal and malicious traffic, trying to identify four different attacks by using application-layer packet features. Furthermore we have also applied an explainable machine learning technique in order to extract potential classification rules to be applied in a real-world scenario. The results we obtained have demonstrated the full feasibility in using synthetic traffic produced by IoT-Flock as a base for the anomaly detection of IoT medical traffic, providing also meaningful classification rules.

As for future research directions, we are committed to train the models on the synthetic traffic coming from IoT-Flock and perform the testing phase on real labeled medical IoT traffic. Moreover, we intend also to perform a per-flow analysis considering features related to a certain traffic flow rather than to the single packets, and to enrich the overall study with a feature selection analysis, in order to verify if, even with a reduction of the considered features, we could output similar outcomes.

Acknowledgements Riccardo Pecori is a member of the INdAM GNCS research group. The authors would like to thank Mr. Antonio Enrico Buonocore, for carefully proofreading the whole manuscript, and Guido Iannone for creating and testing the IoT-Flock scenarios.

Author Contributions Debora Montano collected the data and performed the analyses using ML and DL techniques. She also contributed in writing the paper. Riccardo Pecori performed the analyses using FHDT and contributed in writing the paper. Luca Veltri helped in analyzing the features and provide an interpretation of the extracted rules, contributing also in writing some parts of the paper. Lerina Aversano, Mario Luca Bernardi, and Marta Cimitile supervised the work.

Funding Open access funding provided by Consiglio Nazionale Delle Ricerche (CNR) within the CRUI-CARE Agreement. The authors declare not to have received any funding for carrying out this research

Data Availability the obtained dataset will be available upon request

Declarations

Conflict of interest The authors declare no competing nor financial conflicts of interest.

Research Involving Human and /or Animals ’Not Applicable’

Informed Consent ’Not Applicable’

References

1. Hossain E, Khan I, Un-Noor F, Sikander SS, Sunny MSH. Application of big data and machine learning in smart grid, and associated security concerns: a review. IEEE Access. 2019;7:13960–88. https://doi.org/10.1109/ACCESS.2019.2894819.
2. Ajagbe SA, Awotunde JB, Florez H. Ensuring intrusion detection for iot services through an improved CNN. SN Comput Sci. 2023;5(1):49. https://doi.org/10.1007/s42979-023-02448-y.
3. Ghazanfar S, Hussain F, Rehman AU, Fayyaz UU, Shahzad F, Shah GA. IoT-Flock: an open-source framework for IoT traffic generation. In: 2020 International Conference on Emerging Trends in Smart Technologies (ICETST), 2020;1–6. https://doi.org/10.1109/ICETST49965.2020.9080732.
4. Aversano L, Bernardi M, Cimitile M, Montano D, Pecori R, Veltri L. anomaly detection of medical IoT traffic using machine learning. In: Proceedings of the 12th International Conference on Data Science, Technology and Applications-DATA, 2023:173–182. SciTePress
5. OASIS Standard: MQTT Version 5.0. OASIS Standard. Version 5. (2019). https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html. Accessed Jan 2023
6. Internet Engineering Task Force (IETF): The Constrained Application Protocol (CoAP). Internet Engineering Task Force (IETF). Updated by: RFC 7959, 8613, 8974, 9175. (2019). https://www.rfc-editor.org/rfc/rfc7252. Accessed Jan 2023
7. CVE-2016-10523, Common Enumeration of Vulnerabilities. https://www.cve.org/CVERecord?id=CVE-2016-10523. Accessed 30 Jan 2023.
8. CVE-2019-12101, Common Enumeration of Vulnerabilities. https://www.cve.org/CVERecord?id=CVE-2019-12101. Accessed 30 Jan 2023.

9. CVE-2019-9004, Common Enumeration of Vulnerabilities. https://www.cve.org/CVERecord?id=CVE-2019-9004. Accessed 30 Jan 2023.
10. Aversano L, Bernardi ML, Cimitile M, Pecori R. A systematic review on Deep Learning approaches for IoT security. Comput Sci Rev. 2021;40: 100389.
11. Rish, I. An empirical study of the naive bayes classifier. In: IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, 2001;3:41–46.
12. Suthaharan, S. Machine learning models and algorithms for big data classification. In: Integrated Series in Information Systems, Springer, 2016;36:1–12. https://doi.org/10.1007/978-1-4899-7641-3
13. Wright, RE. Logistic regression. (1995).
14. Magee JF. Decision Trees for Decision Making. MA, USA: Harvard Business Review Brighton; 1964.
15. Aversano L, Bernardi ML, Cimitile M, Pecori R, Veltri L. effective anomaly detection using deep learning in IoT systems. Wirel Commun Mobile Comput. 2021. https://doi.org/10.1155/2021/9054336.
16. Pecori R, Tayebi A, Vannucci A, Veltri L. IoT Attack detection with deep learning analysis. In: 2020 International Joint Conference on Neural Networks (IJCNN), 2020:1–8. https://doi.org/10.1109/IJCNN48605.2020.9207171.
17. Aversano L, Bernardi ML, Cimitile M, Pecori R. Anomaly detection of actual IoT traffic flows through deep learning. In: 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), 2021:1736–1741. https://doi.org/10.1109/ICMLA52953.2021.00275.
18. Ducange P, Marcelloni F, Pecori R. Fuzzy Hoeffding decision tree for data stream classification. Int J Comput Intell Syst. 2021;14:946–64.
19. Gacto MJ, Alcalá R, Herrera F. Interpretability of linguistic fuzzy rule-based systems: an overview of interpretability measures. Inf Sci. 2011;181(20):4340–60.
20. Pundir S, Wazid M, Singh DP, Das AK, Rodrigues JJ, Park Y. Intrusion detection protocols in wireless sensor networks integrated to Internet of Things deployment: survey and future challenges. IEEE Access. 2019;8:3343–63.
21. Rathore S, Park JH. Semi-supervised learning based distributed attack detection framework for IoT. Appl Soft Comput. 2018;72:79–89.
22. Rughoobur P, Nagowah L. A lightweight replay attack detection framework for battery depended IoT devices designed for healthcare. In: 2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions)(ICTUS), 2017:811–817. IEEE.
23. Alrashdi I, Alqazzaz A, Alharthi R, Aloufi E, Zohdy MA, Ming H. FBAD: fog-based attack detection for IoT healthcare in smart cities. In: 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2019:0515–0522. IEEE.
24. DARPA Intrusion Detection Evaluation Dataset. (1998). https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset. Accessed Jan 2023.
25. KDD Cup 1999 Data. (1998). (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed Jan 2023.
26. NSL-KDD Dataset. (1999). https://www.unb.ca/cic/datasets/nsl.html. Accessed Jan 2023.
27. (2023). https://defcon.org/html/links/dc-ctf.html. Accessed Jan 2023.
28. LBNL/ICSI Enterprise Tracing Project. (2023). (http://www.icir.org/enterprise-tracing/. Accessed Jan 2023.
29. Center for Applied Internet Data Analysis (CAIDA). (2023). https://catalog.caida.org/. Accessed Jan 2023
30. UNIBS: Data Sharing. (2009). http://netweb.ing.unibs.it/~ntw/tools/traces/index.php. Accessed Jan 2023
31. Moustafa N, Koroniotis N, Keshk M, Zomaya AY, Tari Z. Explainable intrusion detection for cyber defences in the internet of things: opportunities and solutions. IEEE Commun Surv Tutor. 2023;3:1775–807. https://doi.org/10.1109/COMST.2023.3280465.
32. Khan IA, Moustafa N, Pi D, Sallam KM, Zomaya AY, Li B. A new explainable deep learning framework for cyber threat discovery in industrial IoT networks. IEEE Internet Things J. 2022;13:11604–13. https://doi.org/10.1109/JIOT.2021.3130156.
33. Nguyen QP, Lim KW, Divakaran DM, Low KH, Chan MC. GEE: A gradient-based explainable variational autoencoder for network anomaly detection. In: 2019 IEEE Conference on Communications and Network Security (CNS), 2019:91–99. https://doi.org/10.1109/CNS.2019.8802833.
34. Fazzolari M, Ducange P, Marcelloni F. An explainable intrusion detection system for IoT networks. In: 2023 IEEE International Conference on Fuzzy Systems (FUZZ), 2023:1–6. https://doi.org/10.1109/FUZZ52849.2023.10309785.
35. Khelifati A, Khayati M, Cudré-Mauroux P, Hänni A, Liu Q, Hauswirth M. VADETIS: an explainable evaluator for anomaly detection techniques. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE), 2021;2661–2664. https://doi.org/10.1109/ICDE51399.2021.00298.
36. Aziz S, Faiz MT, Adeniyi AM, Loo K-H, Hasan KN, Xu L, Irshad M. Anomaly detection in the internet of vehicular networks using explainable neural networks (xNN). Mathematics. 2022. https://doi.org/10.3390/math10081267.
37. Ha DT, Hoang NX, Hoang NV, Du NH, Huong TT, Tran KP. Explainable anomaly detection for industrial control system cybersecurity. In: 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022, IFAC-PapersOnLine 2022;(10):1183–1188. . https://doi.org/10.1016/j.ifacol.2022.09.550
38. Koroniotis N, Moustafa N, Sitnikova E, Turnbull B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset. Future Gener Comput Syst. 2019;100:779–96.
39. Hussain F, Abbas SG, Shah GA, Pires IM, Fayyaz UU, Shahzad F, Garcia NM, Zdravevski E. a framework for malicious traffic detection in IoT healthcare environment. Sensors. 2021;9:3025. https://doi.org/10.3390/s21093025.
40. Bormann C. Block-wise transfers in the constrained application protocol (CoAP). Internet Engineering Task Force (IETF). Internet Engineering Task Force (IETF). Updated by: RFC 8323. (2016). https://www.rfc-editor.org/rfc/rfc7959. Accessed Jan 2023.
41. Hartke K. Observing resources in the constrained application protocol (CoAP). Internet Engineering Task Force (IETF). Internet Engineering Task Force (IETF). Updated by: RFC 8323. (2015). https://www.rfc-editor.org/rfc/rfc7641. Accessed Jan 2023.
42. Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint. (2014). arXiv:1412.6980.
43. Shamir O. the implicit bias of benign overfitting. In: Loh, P.-L., Raginsky, M. (eds.) Proceedings of Thirty Fifth Conference on Learning Theory. Proceedings of Machine Learning Research, vol. 178, pp. 448–478. PMLR, USA 2022.