Received: 21 May 2019 | Revised: 14 July 2019 | Accepted: 11 August 2019
DOI: 10.1002/ett.3743

SPECIAL ISSUE ARTICLE

WILEY

A machine learning based framework for IoT device identification and abnormal traffic detection

Ola Salman | **Imad H. Elhajj** | **Ali Chehab** | **Ayman Kayssi**

Department Electrical and Computer Engineering, American University of Beirut, Beirut, Lebanon

Correspondence
Ola Salman, Department Electrical and Computer Engineering, American University of Beirut, Beirut 1107 2020, Lebanon.
Email: oms15@mail.aub.edu

Funding information
AUB University Research Board; Lebanese National Council for Scientific Research; TELUS Corp., Canada

Abstract

Network security is a key challenge for the deployment of Internet of Things (IoT). New attacks have been developed to exploit the vulnerabilities of IoT devices. Moreover, IoT immense scale will amplify traditional network attacks. Machine learning has been extensively applied for traffic classification and intrusion detection. In this paper, we propose a framework, specifically for IoT devices identification and malicious traffic detection. Pushing the intelligence to the network edge, this framework extracts features per network flow to identify the source, the type of the generated traffic, and to detect network attacks. Different machine learning algorithms are compared with random forest, which gives the best results: Up to 94.5% accuracy for device-type identification, up to 93.5% accuracy for traffic-type classification, and up to 97% accuracy for abnormal traffic detection.

1 | INTRODUCTION

A lot has been said in the literature about Internet of Things (IoT) and its characteristics. What is of interest to us is the change in security constraints and requirements. IoT does not only bring new vulnerabilities of its own but also provides a very rich platform to launch attacks. The same properties that IoT is famous for are the ones that give rise to the security challenges: heterogeneity of the connected devices, their constraints in terms of power and processing resources, and scale. They all contribute to making the security of IoT unique and challenging.

In this context, identifying IoT devices is important not only for applying security policies but also for quality-of-service (QoS) differentiation. The challenge is the lack of a unique authenticated identifier in this heterogenous network, where most of the identifiers (MAC addresses, IP addresses, Bluetooth ID, Zigbee ID, etc) can be spoofed. The alternative is to try and identify IoT devices using characteristics that reflect their behavior. In this paper, we propose to identify IoT devices by extracting statistical features from their generated network traffic.

In fact, traffic classification has been applied for QoS and security management. Knowing the traffic type and type of connected devices, many network functions can be enabled like prioritization of critical time applications, intrusion detection, trends analysis, firewalling, and network planning. Different methods have been proposed including port-based methods but today's applications are using dynamic port then came deep packet inspection (DPI) to identify applications and attacks' signatures by inspecting the header and packet contents. However, DPI introduces high overhead and fails when the traffic is encrypted. Recently, machine leaning–based methods have been proposed for traffic classification and abnormal traffic detection. In this context, a new research direction appeared with the IoT emergence, which is the device identification through their generated traffic. Being able to identify the devices along with the type of their generated traffic permits abnormal traffic detection. A matching between the device type and the type of generated traffic at the network edge can serve this aim.

Actually, network traffic characteristics depend on device specifications in terms of operating system, applications, memory capacity, supported network protocols, and processing power. Therefore, IoT devices and attacks are theoretically distinguishable from traditional ones. Accordingly, machine learning-based intrusion detection systems (IDSs) have been proposed to automate the attack detection process.

However, practically, deploying new traffic control functionalities on top of the traditional network is not a straightforward task, and given the large scale within IoT networks, there is a need for a scalable management solution.[1] To this end, the framework that we propose, which is an extension of our previous work,[2] will extract only simple features per packet for any 16 consecutive packets of a flow to identify the IoT device type and detect attack traffic. The extracted features are the packets' direction, size, timestamp, and transport protocol. Thus, the proposed solution does not present any significant processing, communication, or memory overheads. In addition, considering only 16 packets allows the proposed framework to operate in real-time mode.

Moreover, the deployment of such a framework at the network edge is crucial, given that the core network is highly congested. Pushing the intelligence to the network edge comes with the benefits of having more contextual information about the connected devices.[3,4] Hence, the detection of abnormality can be more accurate.[5] For this aim, the use of machine learning–based methods at the network edge has been proposed for security and privacy protection for IoT.[6] More specifically, abnormal traffic detection has attracted the attention due to the severity of security attacks in the IoT domain.[7,8]

This paper is organized as follows: Section 2 presents the related work. In Section 3, we present our proposed framework. Section 4 details the experimental setup. The results evaluation is included in Section 5. Finally, we conclude in Section 6.

2 | LITERATURE REVIEW

Applying machine learning in the IoT domain for device identification and attack traffic detection is a recent research topic. The related work, mostly published within the last few years, is reviewed in this section. First, we present a work related to IoT device identification and in the second subsection, we review a work related to IoT intrusion detection.

2.1 | IoT devices identification

Miettinen et al propose IoT SENTINEL, a framework that aims at extracting an IoT device signature.[9] The proposed method consists of identifying the device during the setup process by extracting 23 features. These features are extracted from the flow of packets generated when the device is connected to the network for the first time. The data was collected from 27 IoT devices. The random forest classifier was used for classifying the preprocessed data into 27 classes representing the device names/models. The results show that the considered devices can be identified with an accuracy of 81.5%. However, an essential limitation of the proposed method is the one-time identification at the initial stage of device setup that could be bypassed. Furthermore, the authors present a security framework that restricts access of certain devices based on their known vulnerabilities. However, device firmware might be updated, and thus, new vulnerabilities may exist before being known, and thus, there is a need to detect the abnormal activity dynamically. In a more recent work,[10,11] the authors tried to overcome the limitation of the one-time identification by proposing a self-learning identification approach. Their approach consists of mapping a device traffic to a certain device type without knowing the exact device model. In the abnormality detection part, the authors represent a packet flow by a symbol containing the following information: direction, local port, remote port, packet length, Transmission Control Protocol (TCP) flags, protocols, and interarrival time.

Meidan et al[12] present a technique to identify unauthorized IoT devices. The data was collected from 17 IoT devices. Random forest was applied with 334 extracted features per network flow. The results show that the considered IoT devices can be identified with an accuracy of 96%. However, the considerable number of extracted features is computationally expensive and limits the generalization of the proposed method. Moreover, the used features contain application layer information, and thus, they require packet header inspection (up to the application layer). The work done by Siby et al[13] presents a different approach to detect IoT devices. The authors consider the physical layer communication pattern to identify the devices connected to a certain network. The proposed traffic analysis method aims to detect unauthorized devices trying to connect unexpectedly to the considered network. In the work of Kawai et al,[14] the identification of nine IoT devices was presented by means of support vector machine (SVM)–based method. A total of 180 traffic features were used, 30-quantile of packet size and interarrival time for 3 types of directions (client to server, server to client, and bidirectional). Sivanathan et al[15] propose a multistage classification method. At the first stage, the bag of words method is employed to classify the IoT devices based on the port numbers, the domain names, and the cipher suites. A random forest classifier is used at the second stage using statistical features, including the flow volume, the flow rate, the flow duration, the sleep time, the DNS interval, and the NTP interval. In the work of Bai et al,[16] an automatic classification of IoT devices is proposed using the network traffic flows. Long short-term memory with convolutional neural network (LSTM-CNN) cascade model is applied on the collected and preprocessed flows. Moreover, in the work of Robyns et al,[17] the physical channel information, including the signal power, the attenuation, and the interference, is used for IoT devices identification. A deep learning–based classification is proposed in the work Das et al[18] for IoT devices authentication. LSTM was applied to detect imperfections in the transmitter's signals of low-power devices. The differentiation is performed between legitimate and illegitimate devices, including the high-power devices that try to impersonate the low power legitimate devices. In the works of Acar et al[19] and Copos et al,[20] IoT device network activity detection in smart home is proposed to monitor and profile the user activity. In the work of Sciancalepore et al,[21] detection of drones' existence, state, and movement is proposed using their generated network traffic. Neural networks are used for extracting devices fingerprints in the work of Yang et al.[22] The considered features are extracted from the network, transport, and application layers. A large number of devices with different types and from different vendors are considered and the results show the effectiveness of the proposed method in extracting distinguishing devices fingerprints. In the aim of authenticating the IoT devices, a framework was proposed in the work of Dabbagh and Saad.[23] This framework consists of authenticating the devices based on their generated traffic. In fact, the authors rely on the fact that the identification of the device type and firmware can help in mitigating their known vulnerabilities. Another framework was proposed to identify the IoT devices based on their communication patterns.[24] The conducted experiments using the traffic generated by different network cameras show the effectiveness of the proposed identification scheme.

2.2 | Attack traffic detection

Nobakht et al presented IoT-IDM, a host-based intrusion detection and mitigation framework.[25] IoT-IDM consists of a machine learning–based application deployed on top of a software-defined network (SDN) controller to identify potential IoT security threats. The normal traffic was collected from an installed Hue lighting system and the attack traffic was collected by developing a script employing the vulnerabilities of the Hue light bulb. The extracted features are as follows: (i) the number of bytes in request packets, (ii) the number of bytes in acknowledgement packets, and (iii) the interpacket time interval. For classification, SVM was compared to logistic regression. SVM gave the best results with up to 98.5% detection precision. In the work of Bhunia and Gurusamy,[26] another SDN-based security framework was proposed for IoT. This framework, called sofThings consists of implementing intelligence through an SDN controller to detect abnormal behavior and attacks. The conducted experiments show that the proposed framework was able to detect attacks on IoT devices with a precision up to 98% using SVM. However, in the case where we have more than two classes, SVM presents complexity in building a multilabel classifier.

Hafeez et al[27] developed IoT-GUARD, a semisupervised classification system aiming at detecting different kinds of attack traffic: port sweep attack, port scan attack, authentication-based attack, worm activity, botnet activity, and spying activity. Fuzzy C-mean clustering was used with 39 discrete and continuous features extracted for each network flow. The evaluation results show that a 98.6% prediction score can be achieved. Tama and Rhee[28] apply deep learning, more precisely a deep belief network to classify attack traffic. They considered several data sets, namely, UNSW-NB15, CIDDS-001, and GPRS. However, these data sets are not IoT specific.

The Mirai attack is one of the famous IoT-specific attacks. Recent works have installed it to collect attack traffic from vulnerable IoT devices. In the work of McDermott et ,[29] Two Sricam AP009 IP cameras running busybox utilities were used as Mirai bots to attack a Raspberry Pi. Long-short term memory recurrent neural network (LSTM-RNN) and bi-directional LSTM-RNN (BLSTM-RNN) were compared with the aim of classifying traffic into four attack classes: Mirai, User Datagram Protocol (UDP) flooding, TCP-ACK flooding, and domain name system (DNS) flooding. Word embedding was used to transform the packet details (length, protocol, and info) into vectors or real numbers used for training the LSTM- RNN network. The experiments show that BLSTM-RNN presents the best results (Mirai, UDP, and DNS with 99%, 98%, 98% as validation accuracy and 0.000809, 0.125630, 0.116453 as validation loss, respectively).

In the work of Doshi et al,[30] different machine learning methods were compared for detection of abnormal IoT traffic. The normal traffic is collected from three devices: YI home camera, Belkin WeMo smart switch, and WiThings blood pressure monitor. The abnormal traffic was collected from a Mirai infected Kali Linux virtual machine attacking a Raspberry Pi 2 server running an Apache web server, both connected to a Raspberry Pi 3 gateway. Three types of attacks were chosen: TCP-SYN flooding, UDP flooding, and Hypertext Transfer Protocol (HTTP) flooding. The extracted features are classified into two types: stateful and stateless. The stateful features are bandwidth and IP destination address cardinality and novelty. The stateless features are packet size, interpacket interval, and protocol. These features are extracted from the collected data per packet, and then, they are aggregated per flow.

In the work of Meidan et al,[31] behavioral profile is extracted per flow considering the different hosts that are contacted. The normal traffic was collected from nine different IoT devices (Ennio Doorbell, Ecobee Thermostat, Philips B120N/10 Baby Monitor, Provision PT-737E Security Camera, Provision PT-838 Security Camera, Simple Home XCS7–1002-WHT Security Camera, Simple Home XCS7–1003-WHT Security Camera, Samsung SNH 1011 N Webcam, and Danmini Doorbell). Mirai and BASHLITE's attacks were deployed to collect abnormal traffic. In total, 115 statistical features were extracted. An autoencoder-based deep learning method was proposed and the results show that the proposed method outperforms local outlier factor (LOF), one-class SVM, and isolation forest.

In our previous work,[2] we present a framework for traffic identification and abnormal traffic detection. This framework consists of three components, a classifier, a security orchestrator, and an intrusion detection module. These modules can be implemented at the network controller. After receiving 16 packets of a flow, the edge node sends to the controller the following information: the packets sizes, interarrival times, along with the flow identifiers which are the source IP address, destination IP address, the source port number, and the destination port number. Computing a set of statistical features, the controller is able to classify the traffic based on the generating device and traffic types. Then, if there is no match between device and traffic types, the flow features are passed to the IDS for abnormal traffic detection. To do so, the IDS checks if the traffic pattern conforms to one of the known attacks. In this work, we present a lightweight feature representation. In addition, we consider the comparison between different machine learning methods including deep learning. Attack traffic for IoT devices is collected and the identification of the attack type is considered. Moreover, in this paper, we propose to combine device identification, traffic-type classification and intrusion detection while considering different types of IoT and non-IoT attacks. Much of the previous work considers only one aspect of security management (ie, attack traffic detection) and considers only IoT specific attacks (ie, Mirai). However, future networks will connect IoT and non-IoT devices, and therefore, any IDS must detect attacks from/to IoT devices that might not be IoT specific. Furthermore, in some of the previous work, many features are extracted, some of which are data dependent. This leads to overfitting and hinders the model generalization. Moreover, the used features are mostly hand designed. In our proposed method, we choose to extract only four features per packet for any 16 consecutive packets of a network flow. The extracted features do not require packet inspection and, yet, they are application dependent.

3 | PROPOSED FRAMEWORK

3.1 | Attacker model

Device identification in IoT is very valuable for security context knowledge. For example, file transfer from a scanner is not unusual unless the destination is identified as a camera. Furthermore, due to the differentiation between IoT devices and traditional devices, we aim at differentiating between the attacks from/to the different types of devices. Considering a home network, we define four attack vectors, as shown in Figure 1: (1) from an outsider to the traditional devices, (2) from an outsider to the IoT devices, (3) from the traditional devices to an outsider, and (4) from the IoT devices to an outsider. Recent IoT attacks are based on a remote exploitation. In this case, the responsiveness of the devices to this exploitation depends on their vulnerabilities. Being at the edge, the IDS must monitor the incoming and the outgoing traffic. In this context, the in/out traffic presents different patterns that help in detecting attacks. As such, we aim at identifying the devices by their model and detecting abnormal traffic from/to the IoT and non-IoT devices.

3.2 | Framework description

Our proposed framework, shown in Figure 2, consists of four components: *features extractor*, *IoT device identification, traffic-type identification*, and *intrusion detection*.

The features, namely, packet size, timestamp, direction, and transport protocol, are extracted for each network flow. A network flow is determined by the five-tuple: source IP address, source port number, destination IP address, destination port number, and transport protocol. The ***features extractor*** keeps an updated list of the active flows. When a packet arrives at the ***features extractor*** component, and if it belongs to an active flow, its features are recorded. If there is no active flow for this packet, a new flow is added, and the packet features are recorded. After receiving 16 packets of the same flow, the flow features are sent to the classifier.

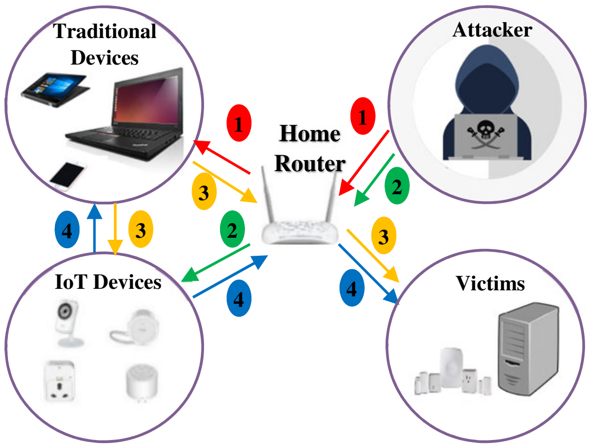

FIGURE 1 Attack model. IoT, Internet of Things

```
Classification Framework ()
if packet is received:
        if packet is in flow:
                if packet count < 16:
                        packet count ++
                else:
                        packet count = 0
        else:
                add flow
                add packet to flow

        if packet count == 16:
                classify device type
                classify traffic type

        if match (traffic type, device type):
                detect normal traffic
        else:
                send flow to IDS
```

FIGURE 2 Classification framework. IDS, intrusion detection system

The ***IoT device identification*** component is responsible for classifying the devices based on their network traffic flow statistical features. The ***traffic-type identification*** component aims at classifying the generated traffic based on the type. If there is mismatch between the expected traffic-type from a certain device and the generated traffic, an abnormal activity can be detected. The ***intrusion detection*** component has the role of profiling the normal device behavior and is able to detect abnormal activity (ie, attack traffic).

The proposed approach relies on machine learning to define fingerprints of the IoT devices by examining their generated traffic. In addition, each device is associated with a certain traffic type. These two characteristics can be used for a multifactor authentication. In case the devices are exploited by malicious attacks, their generated traffic might differ from their normal traffic type, and thus, the authentication process will fail. Moreover, the device type can be linked with known vulnerabilities to be prevented by taking the convenient countermeasures at the gateway.

This intelligent framework can detect the attacks without DPI. In addition, monitoring the traffic, this framework can help in detecting abnormal activities when the traffic statistics are different from the normal ones. However, in this case, the definition of abnormality is not a straightforward task, having many factors that influence the traffic generated by these devices. For example, studying the traffic size as a function of the time of day might give basic information about the average of normal traffic at a certain time of the day. Hence, the dynamic nature of the traffic makes this task unpredictable. However, in this case, having contextual information of the considered network can alleviate this issue and decrease the number of false alarms.

Thus, in a hierarchical type of classification, the proposed framework consists of a cascade of classifiers as shown in Figure 3. At a first stage, the intrusion detection component classifies the traffic into IoT and non-IoT. At a second stage,

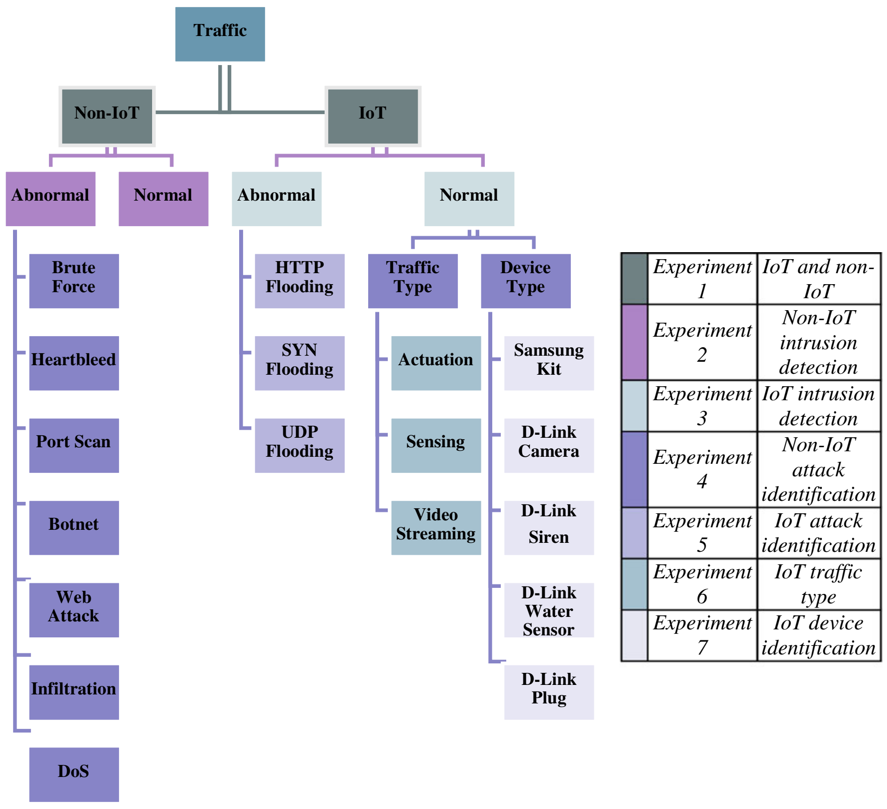

FIGURE 3 Hierarchical classification framework. IoT, Internet of Things; UDP, User Datagram Protocol

the classification between normal and abnormal traffic is performed. Finally, for abnormal traffic, the intrusion detection classifier tries to identify the attack type. If an IoT normal traffic is detected, the flow is passed to the traffic-type identification and the device identification components to check if the traffic type and the generating device-type match.

In our case, the feature extraction can be performed at the edge of an IoT network, where the edge nodes are controlled by a logically central controller.[32,33] Thus, traffic classification can be thought of as a service running at an aggregation point (real or virtual router). This means that the traffic can be of any type: clear text, encapsulated (HTTPS), tunneled (VPN), or anonymized (TOR). This requires that the features which can be hidden by encapsulation, tunneling, or anonymization not be included as part of the scheme.

4 | EXPERIMENTAL SETUP

The experimental setup goals can be summarized by the following points: collecting normal and attack traffic from different IoT devices, preprocessing the collected data to extract statistical features per network flow, analyzing the features correlation with the class labels, and comparing different machine learning methods based on the performance metrics results.

Our performed experiments are listed in Figure 3. In total, we performed seven experiments. In *Experiment 1*, the aim is to differentiate between IoT and non-IoT traffic. *Experiment 2* and *Experiment 3* are designed for intrusion detection in IoT and non-IoT domains, respectively. *Experiment 4* and *Experiment 5* have the aim of identifying the type of IoT and non-IoT attacks, respectively. Considering the IoT traffic, in *Experiment 6* and *Experiment 7*, the goal is to detect the attacks that generate normal traffic by maliciously employing vulnerable IoT devices. In *Experiment 6*, the IoT collected traffic is classified based on the traffic type such as video, sensing, and actuation. *Experiment 7* is composed of two parts: In part 1, the aim was to identify the device model, even if the devices run behind an IoT hub. Thus, the considered traffic was labeled based on the generating device model. In part 2, the traffic coming from a hub is considered to form one class.

4.1 | Data collection

To collect the traffic from IoT devices, we consider the set of seven devices shown in Table 1. To collect normal traffic from the considered devices, the devices were installed in a private home network. There are two types of devices:

TABLE 1 Set of Internet of Things (IoT) devices

	Device model
1	*D-Link HD 180-Degree Wi-Fi Camera DCS-8200LH*
2	*D-Link Wi-Fi Smart Plug DSP-W215*
3	*D-Link Wi-Fi Siren DCH-S220*
4	*D-Link Wi-Fi Water Sensor DCH-S160*
5	*Samsung SmartThings Home Monitoring Kit- Motion Sensor*
6	*Samsung SmartThings Home Monitoring Kit- Multipurpose Sensor*
7	*Samsung SmartThings Home Monitoring Kit- Smart Plug*

Wi-Fi–enabled devices (D-Link devices), and z-wave–enabled devices connected to a hub (Samsung Home Kit). To collect data from Wi-Fi–enabled devices, we configured a laptop as a Wi-Fi access point, and we connected the devices to the configured network. To collect traffic from the Samsung Home Kit devices, a laptop was configured as a router connected by an Ethernet cable to the Samsung Kit hub and connected wirelessly to the home network. Using Wireshark, the traffic generated by these devices for 5 days was collected (by running it on the laptop configured as router) and saved as packet capture (PCAP) files in the *pcap* format.

To collect IoT attack traffic, we chose to perform some attacks employed by Mirai from an ESP8266 module, which is the basic component of many IoT devices. ESP8266 is a Wi-Fi System on Chip (SoC) supporting the full TCP/IP stack and able to send and receive data. Three types of attacks, exploited by Mirai, are coded on the ESP8266 module: HTTP flooding, SYN flooding, and UDP flooding. To deploy these attacks, we created an isolated network where we configure a laptop to act as a DNS server, a DHCP server, a web server using Apache listening at port 80, and a Wi-Fi access point. The ESP8266 module is programmed using Arduino IDE through the serial interface. To collect TCP SYN attack traffic, the ESP8266 module is configured to connect continuously to the Wi-Fi access point with a random source port number. In the UDP case, UDP packets are sent continuously to the Wi-Fi access point with a random destination port number. To perform the HTTP attack, a TCP connection is established first with the HTTP server. Then, a GET request is continuously sent from the ESP8266 module to the web server. The attack traffic is captured at the access point using Wireshark. One hour of traffic is captured for each attack.

In addition, we considered the CICIDS2017 data set containing non-IoT devices normal and abnormal traffic.[17] This data set consists of 5 days of collected traffic, from which one day traffic is normal, and the remaining days collected traffic includes different attacks: web based, brute force, DoS, DDoS, infiltration, heartbleed, botnet, and port scan. The provided PCAPs are not labeled; however, for each traffic type, the time slot and the source and destination IP addresses are provided. Thus, filtering the provided PCAPs, each type of traffic can be obtained.

4.2 | Data preprocessing and features importance

Before preprocessing the obtained PCAP files, we filtered them to have only the TCP and UDP sessions. Using the *dpkt* python library,[34] the flows are extracted per network flow. For each 16 packets of each flow, the features (packet size, the interarrival time, the direction, and the transport protocol) are extracted to form a $4 \times 4 \times 4$ vector. To do so, all the flow packets features are recorded in a flow list. The extracted features are normalized within the range [0, 1]. The interarrival time is computed by subtracting the packet timestamp from the previous packet timestamp. If the computed time delta is greater than 1 second, the interarrival time is set to 1, otherwise, the time delta is divided by 1000 ms.

Similarly, the packet size is checked and if it is greater than 1500 bytes, the packet size is set to 1. Otherwise, the packet size is normalized by dividing it by 1500. The packet direction is set to 0 or 1 based on the first flow packet direction determined by the source IP and destination IP addresses. The protocol is set to 0 if it is a UDP packet and to 1 if it is a TCP packet. The flows extraction algorithm is sketched in Figure 4.

After the normalization, the flows are subdivided into M subflows of 16 packets each, as shown in Figure 5. The initial flow is composed of n packets $[p_1, p_2, \ldots, p_n]$. Each packet P_i is represented by four features $[S_i, T_i, D_i, P_i]$, where S_i stands for the packet size, T_i stands for interarrival time, D_i stands for direction, and P_i stands for transport protocol. This resulted in a $4 \times 4 \times 4$ vector for each subflow. At the same time, we saved each feature values independently (ie, packet size for each 16 packets of the flow, interarrival time for each 16 packets of the flow, the protocol for each 16 packets of the flow, and the direction for each 16 packets of the flow). In this case, the obtained vectors are 4×4 for each subflow.

To investigate the chosen features importance, we visualize the extracted subflow data for the interarrival time, the packet size, and the direction for the different experiments. To do so, the *matplotlib*[35] and *pandas*[36] python libraries were

```
Flows extraction():

for each packet in PCAP file:
    for each flow in flows:
        if (src_ip = flow.src_ip || src_ip = flow.dst_ip) \\
        && (dst_ip = flow_src_ip || dst_ip = flow_dst_ip) \\
        && (src_port = flow_src_port || src_port = flow_dst_port) \\
        && (dst_port = flow_src_port || dst_port = flow_dst_port) \\
        && (protocol = flow_protocol):
            current_packet ← new Packet ()
            delta_time ← packet_timestamp – flow[length-1]_packet_timestamp
            if delta_time > 1000:
                current_packet_interarrival_time ← 1
            else:
                current_packet _interraval_time ← delta_time/ 1000
            if packet_size > 1500:
                current_packet.size ← 1
            else:
                current_packet.size ← packet_size/1500
            if src_ip = flow_src_ip:
                direction ← 0
            else:
                packet.direction ← 1
            current_packet_protocol ← flow_protocol
            flow.add_packet(current_packet)
```

FIGURE 4 Flow extraction algorithm. PCAP, packet capture

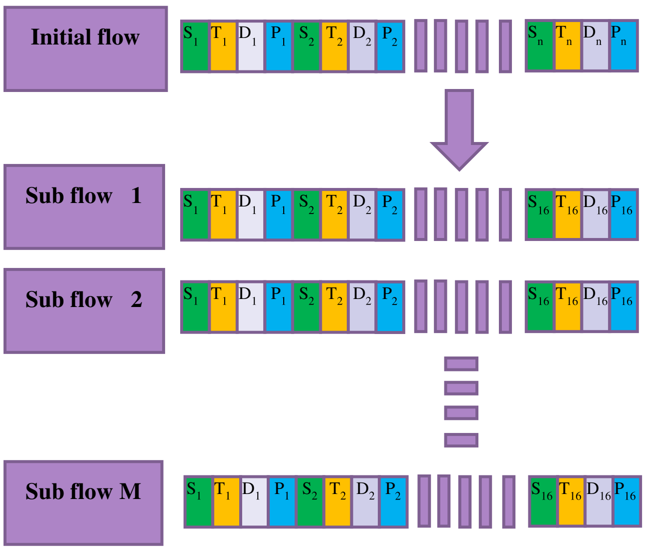

FIGURE 5 Flow equalizer

used. The subflows feature vectors are grouped by class, and for each class, the mean of the feature values is calculated for the ith packet, where i is in the range.[1,16] As shown in Figure 6, the plotted features (for experiment 3) present different patterns for the different classes and this shows that the differentiation is possible among the different classes, even if the patterns are not linearly separable. The other experiments showed similar properties for the features but are not included for brevity. In fact, the chosen features are the basic features from which any set of statistical features can be extracted. In this work, we aimed at investigating the power of these features in differentiating between different traffic classes. The packets sizes of normal traffic, shown in Figure 6, present different distribution from those of the abnormal one. The same applies for the packets interarrival times and directions. In fact, for the considered types of attacks, which are based on flooding of SYN, ACK, and UDP packets, it is expected to have distinguishable packet sizes and small interarrival times distributions and this appeared clearly in Figure 6. However, not all the attacks are that simple, some attacks might try to mimic the normal traffic behavior in generating packet sizes and interarrival times similar to the normal one's distributions. In this case, the obtained accuracy might drop and methods relying on unsupervised learning must be considered.

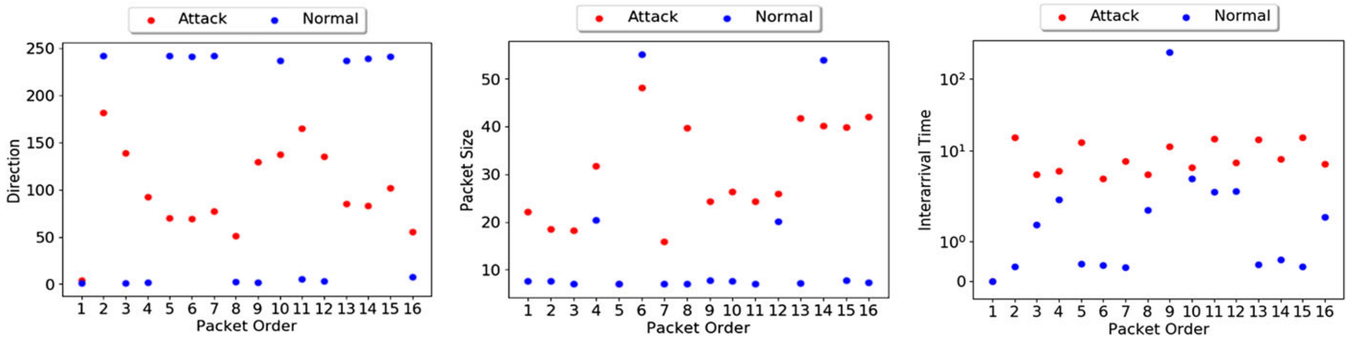

FIGURE 6 Features visualization for Experiment 3 collected traffic

On the other hand, for comparison with a state-of the-art set of features, we considered a subset of the comprehensive list of features proposed by Moore et al.[37] The list of features, which we consider for comparison, is included in Table 2. Similarly, these features were extracted for the first 16 packets of each flow.

4.3 | Classification algorithm

For classification, we compared decision tree–based methods (decision tree and random forest) and deep learning–based methods, including recurrent neural network (RNN), residual neural network (ResNet), and convolutional neural network (ConvNet). A decision tree consists of sorting data based on a sequence of conditions until reaching the leaf that represents the class label. Each node represents a certain feature condition. The sequence of features is chosen based on some measures, such as information gain and Gini index, that reflect the features correlation with the output labels. Random forest is an aggregation ensemble method. It consists of multiple decision trees trained on subsets of the data and the features. At the end, the results of the different decision trees are aggregated by averaging or voting. This method helps in reducing the model bias and variance. Decision trees and random forest have shown good results in traffic classification.[38] On the other hand, Deep learning is a branch of machine learning consisting of neural networks with several layers. Deep learning has been applied mainly to complex tasks, like image recognition and natural language processing (NLP). Deep learning includes different architectures, from which we choose three for our performance testing: recurrent, residual and ConvNet. ConvNet architectures are mainly designed for image-based classification. A main layer in any ConvNet architecture is the convolution layer, where filters are applied to the input in order to extract the classification features. Residual neural network is a type of Convolution Neural Network architectures with "skip connections" in order to avoid the vanishing gradients issue. This is done by reutilizing the weights of the previous layers. Finally, the RNN is applied in case the input comes in sequence. As the authors in a previous work[38] compared decision tree to neural network for Internet traffic classification, in this paper, our aim is to compare deep learning to decision tree–based methods for abnormality detection and device identification.

Decision tree and random forest were implemented using sickit-learn.[39] Grid search was used to tune the classifiers parameters. For decision tree, the max depth of the tree was in the range of [0-1000] and for random forest, the number of estimators was in the range of [0-10 000]. For the deep learning-based classifiers, *Tflearn*[40] was used, which is a high-level deep learning python library based on *tensorflow*.[41]

We balanced our data using random oversampling. This method consists of randomly adding instances to the class with a smaller number of samples, by copying the existing ones. Stratified cross-validation was applied with four folds. To evaluate the performance, four metrics were calculated as follows:

$$\begin{aligned}
Accuracy &= \frac{TP+TN}{TP+FP+TN+FN} * 100\\
Precision &= \frac{TP}{TP+FP}\\
Recall &= \frac{TP}{TP+FN}\\
f1 &= \frac{2 * Precision * Recall}{Precision + Recall},
\end{aligned}$$

TABLE 2 Feature set for comparison

Feature	Description
1- total_fwd_pkt	Total packets in the forward direction
2- total_fwd_bytes	Total bytes in the forward direction
3- total_bck_pkt	Total packets in the backward direction
4- total_bck_bytes	Total bytes in the backward direction
5- min_pckt_size_fwd	The min packet size in the forward direction
6- mean_pckt_size_fwd	The mean packet size in the forward direction
7- max_pckt_size_fwd	The max packet size in the forward direction
8- std_pckt_size_fwd	The standard deviation packet size in the forward direction
9- min_pckt_size_bck	The min packet size in the backward direction
10- mean_pckt_size_bck	The mean size of packets in the backward direction
11- max_pckt_size_bck	The max packet size in the backward direction
12- std_pckt_size_bck	The standard deviation packet size in the backward direction
13- min_iat_fwd	The minimum interarrival time in the forward direction
14- mean_iat_fwd	The mean interarrival time in the forward direction
15- max_iat_fwd	The maximum interarrival time in the forward direction
16- std_iat_fwd	The standard deviation interarrival time in the forward direction
17- min_iat_bck	The minimum interarrival time in the backward direction
18- mean_iat_bck	The mean interarrival time in the backward direction
19- max_iat_bck	The maximum interarrival time in the backward direction
20- std_iat_bck	The standard deviation interarrival time in the backward direction
21- total_time	The duration of the flow
22- min_act_time	The minimum active time
23- mean_act_time	The mean active time that the flow
24- max_act_time	The maximum active time
25- std_act_time	The standard deviation active time
26- min_idle_time	The minimum idle time
27- mean_idle_time	The mean idle time
28- max_idle_time	The maximum idle time
29- std_idle_time	The standard deviation of idle time
30- avg_pckt_fwd	The average number of packets in the forward direction
31- avg_bytes_fwd	The average number of bytes in the forward direction
32- avg_pckt_bck	The average number of packets in the backward direction
33- avg_bytes_bck	The average number of bytes in the backward direction
34- PSH_fwd	The number of PSH flags in the forward direction (0 for UDP)
35- PSH_bck	The number of PSH flags in the backward direction (0 for UDP)
36- URG_fwd	The number of URG flag in the forward direction (0 for UDP)
37- URG_bck	The number of URG flag in the backward direction (0 for UDP)
38- total_bytes_hdr_fwd	The total bytes used for headers in the forward direction
39- total_bytes_hdr_bck	The total bytes used for headers in the backward direction

Abbreviation: UDP, User Datagram Protocol.

where TP (true positive) is the number of instances correctly classified as belonging to class C, TN is the number of instances correctly classified as not belonging to class C, FP is the number of instances wrongly classified as belonging to class C, and FN is the number of instances wrongly classified as not belonging to class C. These metrics are as computed over the four folds, and the mean is considered as the final performance evaluation.

5 | EVALUATION AND ANALYSIS OF RESULTS

The graphs shown in Figure 7 present the accuracy results obtained by applying the considered classification algorithms on different sets of features. In the Figure, A stands for the set of all extracted features, B stands for the interarrival time, C stands for the packet size, and D stands for the packet direction. For each experiment, four sets of features are compared: combination of all features, packet size, interarrival time, and direction. The results show that considering the combination of all the chosen features (packet size, interarrival time, direction, and protocol) gives the highest accuracy with all the considered classification methods. However, the individual features are shown to be highly correlated with

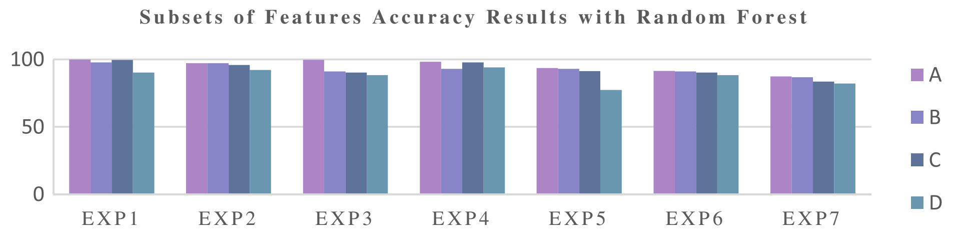

FIGURE 7 Accuracy results for the different sets of features

the different classes in each experiment. The obtained accuracy for each feature is promising and, more specifically, the one obtained by employing the packet interarrival time (B) shows great promise.

Tables 3 to 10 show complete results that include all the performance metrics for the combination of all the extracted features. Differentiating between IoT and non-IoT traffic is shown to be feasible, where a ***99.93%*** accuracy and ***0.9985*** f1-score are achieved with random forest in ***Experiment 1***. In ***Experiment 2***, where we classify traffic into attack and normal non-IoT traffic classes, the achieved f1-score decreases noticeably. This can be explained by the fact that some of the considered attacks might behave like normal traffic. However, when differentiating between normal and attack IoT traffic in ***Experiment 3***, the achieved accuracy and f1-score increase to ***99.97%*** and ***0.9997,*** respectively with random forest. In this case, the differentiation between the IoT normal and attack traffic is an easy task, given that the considered attacks do not present similar behavior to the normal one. ***Experiment 4*** aims at classifying the type of the non-IoT attack traffic into seven classes: brute force, heartbleed, port scan, botnet, web attack, infiltration, and DoS. In this experiment, the achieved accuracy (***98.28%***) is very promising, also a high f1-score (***0.8705***) is achieved with the random forest classifier. In ***Experiment 5***, the IoT attack traffic is classified into 3 classes: HTTP flooding, UDP flooding, and SYN flooding. In this experiment, the highest achieved accuracy is ***93.51%*** and the highest achieved f1-score is ***0.7576***. This can be explained by the fact that the considered IoT attacks present similar behavior. When we consider the traffic

TABLE 3 Experiment 1

	Accuracy	Precision	Recall	F1-score
RF	***99.93%***	***0.999***	***0.999***	***0.9985***
DT	*99.85%*	*0.9977*	*0.9977*	*0.9977*
RNN	*99.77%*	*0.9965*	*0.9966*	*0.9965*
ConvNet	*99.78%*	*0.9971*	*0.9962*	*0.9966*
ResNet	*0.9978*	*0.997*	*0.9965*	*0.9967*

Abbreviations: ConvNet, convolutional neural network; ResNet, residual neural network; RNN, recurrent neural network.

TABLE 4 Experiment 2

	Accuracy	Precision	Recall	F1-score
RF	***97.14%***	***0.8581***	***0.8628***	***0.8601***
DT	*96.28%*	*0.8299*	*0.8034*	*0.8157*
RNN	*95.35%*	*0.7925*	*0.9459*	*0.8469*
ConvNet	*95.16%*	*0.7872*	*0.9394*	*0.8410*
ResNet	*94.77%*	*0.7777*	*0.9533*	*0.8363*

Abbreviations: ConvNet, convolutional neural network; ResNet, residual neural network; RNN, recurrent neural network.

TABLE 5 Experiment 3

	Accuracy	Precision	Recall	F1-score
RF	***99.97%***	***0.9997***	***0.9997***	***0.9997***
DT	99.63%	0.9963	0.9963	0.9963
RNN	99.91%	0.9991	0.9991	0.9991
ConvNet	99.61%	0.9962	0.9961	0.9961
ResNet	99.37%	0.9939	0.9936	0.9937

Abbreviations: ConvNet, convolutional neural network; ResNet, residual neural network; RNN, recurrent neural network.

TABLE 6 Experiment 4

	Accuracy	Precision	Recall	F1-score
RF	***98.28%***	***0.9027***	0.8722	***0.8705***
DT	97.54%	0.8498	0.8349	0.8399
RNN	97.61%	0.8223	0.8747	0.8412
ConvNet	97.39%	0.8161	***0.8798***	0.8372
ResNet	90.98%	0.7632	0.8585	0.7888

Abbreviations: ConvNet, convolutional neural network; ResNet, residual neural network; RNN, recurrent neural network.

TABLE 7 Experiment 5

	Accuracy	Precision	Recall	F1-score
RF	***93.51%***	***0.7788***	***0.7567***	***0.7576***
DT	92.52%	0.6928	0.7124	0.6989
RNN	92.69%	0.6479	0.7258	0.6548
ConvNet	93.12%	0.6628	0.7171	0.6624
ResNet	93.03%	0.6248	0.7029	0.6327

Abbreviations: ConvNet, convolutional neural network; ResNet, residual neural network; RNN, recurrent neural network.

TABLE 8 Experiment 6

	Accuracy	Precision	Recall	F1-score
RF	***91.42%***	*0.8455*	*0.8455*	***0.8443***
DT	89.65%	0.8129	0.8129	0.8129
RNN	91.00%	0.8502	0.8433	0.8385
ConvNet	91.39%	**0.8625**	**0.8484**	0.8435
ResNet	***91.42%***	*0.8455*	*0.8455*	***0.8443***

Abbreviations: ConvNet, convolutional neural network; ResNet, residual neural network; RNN, recurrent neural network.

TABLE 9 Experiment 7 - Part 1

	Accuracy	Precision	Recall	F1-score
RF	***87.40%***	***0.6962***	***0.7074***	***0.6946***
DT	85.20%	0.6565	0.6553	0.6558
RNN	86.40%	0.6732	0.6979	0.6543
ConvNet	86.31%	0.6703	0.6948	0.6508
ResNet	85.01%	0.6493	0.6740	0.6208

Abbreviations: ConvNet, convolutional neural network; ResNet, residual neural network; RNN, recurrent neural network.

TABLE 10 Experiment 7 - Part 2

	Accuracy	Precision	Recall	F1-score
RF	**94.47%**	**0.8159**	**0.8151**	**0.8142**
DT	93.76%	0.8023	0.8020	0.8022
RNN	93.24%	0.7850	0.7938	0.7713
ConvNet	93.79%	0.8065	0.8062	0.8057
ResNet	91.91%	0.7298	0.7869	0.7408

Abbreviations: ConvNet, convolutional neural network; ResNet, residual neural network; RNN, recurrent neural network.

type and classify the device traffic into 3 classes: video, actuation, and sensing as in ***Experiment 6***, the accuracy reaches ***91.41%*** and the f1-score reaches ***0.8443.*** For ***Experiment 7 - Part 1***, where we have seven classes representing the different devices including the Samsung kit devices connected to a hub, the highest recorded accuracy is ***87.40%*** and the highest recorded f1-score is ***0.69***, which means that some classes present similar features. Note that when the classification task is more complex, the f1-score is more significant because it considers both the false positive and false negative rates. Furthermore, the confusion matrix (presented in Figure 8A) shows that the Samsung Kit devices are not easily discernable. However, when their traffic is combined into one class (see Figure 8B), in ***Experiment 7 - Part 2***, the accuracy increases to ***94.47%,*** and the f1-score increases to ***0.8142***. Finally, it is noticeable that random forest gives overall the best accuracy

Class	1	2	3	4	5	6	7
1	1818	0	347	6	2	0	219
2	0	1208	0	7	73	1346	0
3	992	0	957	3	2	0	1294
4	0	11	0	31080	30	5	8
5	100	92	0	14	2898	88	2
6	0	921	0	5	85	2399	0
7	43	0	587	3	3	0	2583

(A) *Experiment 7 - Part 1*

1	***Samsung plug***
2	***D-Link water sensor***
3	***Samsung motion sensor***
4	***D-Link Camera***
5	***D-Link plug***
6	***D-Link siren***
7	***Samsung multi-purpose sensor***

Class	1	2	3	4	5
1	8879	0	7	0	1
2	0	2403	1	928	90
3	6	5	31101	6	25
4	1	1354	7	1134	89
5	15	76	7	86	2763

(B) *Experiment 7 - Part 2*

1	***Samsung Devices***
2	***D-Link water sensor***
3	***D-Link Camera***
4	***D-Link plug***
5	***D-Link siren***

FIGURE 8 Confusion matrix

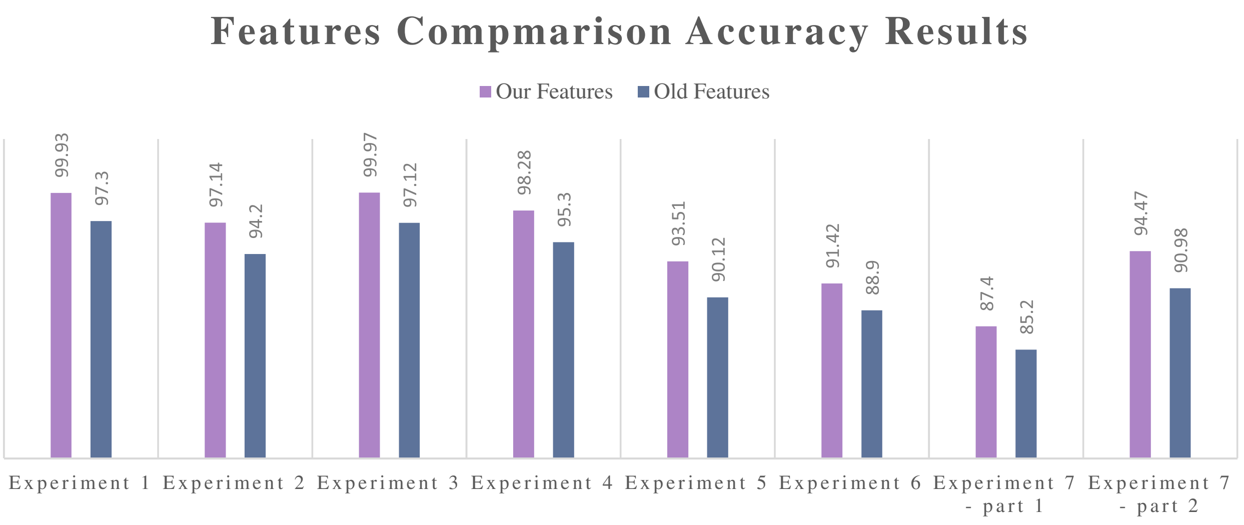

FIGURE 9 Features comparison accuracy results

results in all the experiments**.** In addition, the comparison with a previous work set of features shows that our features present better results for all the experiments, as shown in Figure 9. This can be explained by the fact that our features, being the base of any features set, present high differentiation power.

As a conclusion, the proposed feature extraction method presents promising results for intrusion detection, traffic-type classification, and device-type classification. Most of the previous works in this domain propose hand-crafted features that are data dependent. However, we show, in this paper, that primitive packets features are enough for differentiating between different types of traffic. In fact, our proposed features represent the main statistical characteristics of a flow. In a previous work,[42] we considered the application of deep learning on these raw features, given that deep learning has the representation learning power. However, in this work, we compared random forest to deep learning methods with these features in order to investigate their differentiating power with statistical-based methods such as random forest. As shown in the results, random forest produced the best results. However, if we derive different statistical features based on our proposed ones and we applied random forest, the obtained results show that our features give better results. This is an indication that adding features does not improve performance yet might increase complexity.

6 | CONCLUSION

The Internet traffic is in a continuous shift regarding the type of connected devices, the type of generated traffic, and the type and number of attacks. In this context, rule-based methods for QoS and security management are not sufficient. Applying machine intelligence has been proposed to overcome network security and QoS management complexity. In this paper, we present a framework for identifying IoT devices, traffic types, and attacks. Normal and attack traffic were collected from different IoT devices. A comprehensive set of experiments was conducted with the aim to compare different machine learning methods using a set of selected features. The proposed data representation is shown to be efficient for classifying traffic using only statistical features without inspecting the packet content. Only four features for each 16 packets of a flow are employed to detect the device type, the traffic type, and the attack type. The performance metrics results show that random forest gives the best results overall, which reveals the robustness of the proposed method.

While the results present low false positive and false negative rates, the detection of unknown traffic is key to detecting new types of devices, new types of traffic, and new types of attacks. Unsupervised classification methods have to be considered for this aim. Moreover, the robustness of the chosen features toward confusion or morphism needs to be tested. In addition, tunneling and anonymization can affect the classification accuracy.

ACKNOWLEDGMENTS

This research was funded by the AUB University Research Board, the Lebanese National Council for Scientific Research, and TELUS Corp., Canada.

ORCID

Ola Salman https://orcid.org/0000-0002-1011-8665

REFERENCES

1. Salman O, Elhajj I, Chehab A, Kayssi A. IoT survey: an SDN and fog computing perspective. *Computer Networks*. 2018;143:221-246.
2. Salman O, Chaddad L, Elhajj IH, Chehab A, Kayssi A. Pushing intelligence to the network edge. Paper presented at: Fifth International Conference on Software Defined Systems (SDS); 2018; Barcelona, Spain.
3. Al Ridhawi I, Aloqaily M, Kotb Y, Al Ridhawi Y, Jararweh Y. A collaborative mobile edge computing and user solution for service composition in 5G systems. *Trans Emerg Telecommun Technol*. 2019;29(11):e3446.
4. Aloqaily M, Otoum S, Al Ridhawi I, Jararweh Y. An intrusion detection system for connected vehicles in smart cities. *Ad Hoc Netw*. 2019;90:101842.
5. Yaseen Q, AlBalas F, Jararweh Y, Al-Ayyoub M. A fog computing based system for selective forwarding detection in mobile wireless sensor networks. Paper presented at: IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W); 2016; Augsburg, Germany.
6. Moh M, Raju R. Using machine learning for protecting the security and privacy of Internet of Things (IoT) systems. In: *Fog and Edge Computing*. Hoboken, NJ: John Wiley & Sons; 2019:223-257.
7. Ariyaluran Habeeb RA, Nasaruddin F, Gani A, et al. Clustering-based real-time anomaly detection—a breakthrough in big data technologies. *Trans Emerg Telecommun Technol*. 2019:e3647.
8. Adeel A, Ali M, Khan AN, et al. A multi-attack resilient lightweight IoT authentication scheme. *Trans Emerg Telecommun Technol*. 2019:e3676.
9. Miettinen M, Marchal S, Hafeez I, Asokan N, Sadeghi A, Tarkoma S. IoT SENTINEL: automated device-type identification for security enforcement in IoT. Paper presented at: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS); 2017; Atlanta, GA.
10. Nguyen TD, Marchal S, Miettinen M, Dang MH, Asokan N, Sadeghi A-R. D\" IoT: a crowdsourced self-learning approach for detecting compromised IoT devices. 2018. arXiv preprint arXiv:1804.07474.
11. Marchal S, Miettinen M, Nguyen TD, Sadeghi A-R, Asokan N. AuDI: toward autonomous IoT device-type identification using periodic communication. *IEEE J Sel Areas Commun*. 2019;37(6):1402-1412.
12. Meidan Y, Bohadana M, Shabtai A, et al. Detection of unauthorized IoT devices using machine learning techniques. 2017. arXiv preprint arXiv:1709.04647.
13. Siby S, Maiti RR, Tippenhauer N. IoTScanner: detecting and classifying privacy threats in IoT neighborhoods. 2017. arXiv preprint arXiv:1701.05007.
14. Kawai H, Ata S, Nakamura N, Oka I. Identification of communication devices from analysis of traffic patterns. Paper presented at: 2017 13th International Conference on Network and Service Management (CNSM); 2017; Tokyo, Japan.
15. Sivanathan A, Habibi Gharakheili H, Loi F, et al. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Trans Mob Comput*. 2019;18(8):1745-1759.

16. Bai L, Yao L, Kanhere SS, Wang X, Yang Z. Automatic device classification from network traffic streams of Internet of Things. 2018. arXiv preprint arXiv:1812.09882.
17. Robyns P, Marin E, Lamotte W, Quax P, Singelée D, Preneel B. Physical-layer fingerprinting of LoRa devices using supervised and zero-shot learning. In: Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks; 2017; Boston, MA.
18. Das R, Gadre A, Zhang S, Kumar S, Moura JMF. A deep learning approach to IoT authentication. Paper presented at: IEEE International Conference on Communications (ICC); 2018; Kansas City, MO.
19. Acar A, Fereidooni H, Abera T, et al. Peek-a-Boo: I see your smart home activities, even encrypted! CoRR. 2018. arXiv preprint arXiv:1808.02741.
20. Copos B, Levitt K, Bishop M, Rowe J. Is anybody home? Inferring activity from smart home network traffic. Paper presented at: IEEE Security and Privacy Workshops (SPW); 2016; San Jose, CA.
21. Sciancalepore S, Ibrahim OA, Oligeri G, Di Pietro RD. Picking a needle in a haystack: detecting drones via network traffic analysis. CoRR. 2019. arXiv preprint arXiv:1901.03535.
22. Yang K, Li Q, Sun L. Towards automatic fingerprinting of IoT devices in the cyberspace. *Computer Networks*. 2019;148:318-327.
23. Dabbagh YS, Saad W. Authentication of wireless devices in the Internet of Things: learning and environmental effects. *IEEE Internet Things J*. 2019;6(4):6692-6705.
24. Noguchi H, Kataoka M, Yamato Y. Device identification based on communication analysis for the Internet of Things. *IEEE Access*. 2019;7:52903-52912.
25. Nobakht M, Sivaraman V, Boreli R. A host-based intrusion detection and mitigation framework for smart home IoT using OpenFlow. Paper presented at: 2016 11th International Conference on Availability, Reliability and Security (ARES); 2016; Salzburg, Austria.
26. Bhunia SS, Gurusamy M. Dynamic attack detection and mitigation in IoT using SDN. Paper presented at: 2017 27th International Telecommunication Networks and Applications Conference (ITNAC); 2017; Melbourne, Australia.
27. Hafeez I, Ding AY, Antikainen M, Tarkoma S. Toward secure edge networks taming device-to-device (D2D) communication in IoT. 2017. arXiv preprint arXiv:1712.05958.
28. Tama BA, Rhee KH. Attack classification analysis of IoT network via deep learning approach. *Res Briefs Inf Commun Technol Evol*. 2017;3(15):1-9.
29. McDermott CD, Majdani F, Petrovski AV. Botnet detection in the Internet of Things using deep learning approaches. Paper presented at: 2018 International Joint Conference on Neural Networks (IJCNN); 2018; Rio de Janeiro, Brazil.
30. Doshi R, Apthorpe N, Feamster N. Machine learning DDoS detection for consumer Internet of Things devices. 2018. arXiv preprint arXiv:1804.04159.
31. Meidan Y, Bohadana M, Mathov Y, et al. N-BaIoT—network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Pervasive Comput*. 2018;17(3):12-22.
32. Salman O, Elhajj I, Kayssi A, Chehab A. Edge computing enabling the Internet of Things. Paper presented at: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT); 2015; Milan, Italy.
33. Salman O, Elhajj I, Kayssi A, Chehab A. An architecture for the Internet of Things with decentralized data and centralized control. Paper presented at: 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA); 2015; Marrakech, Morocco.
34. dpkt. https://dpkt.readthedocs.io/en/latest/
35. matplotlib. https://matplotlib.org/
36. pandas. https://pandas.pydata.org/
37. Moore A, Zuev D, Crogan M. Discriminators for use in flow-based classification. 2005.
38. Michael AKJ, Valla E, Neggatu NS, Moore AW. *Network Traffic Classification via Neural Networks*. Technical Report. No. UCAM-CL-TR-912. Cambridge, UK: University of Cambridge; 2017.
39. scikit-learn. https://scikit-learn.org/
40. TFLearn. http://tflearn.org/
41. TensorFlow. https://www.tensorflow.org/
42. Salman O, Elhajj IH, Chehab A, Kayssi A. A multi-level internet traffic classifier using deep learning. Paper presented at: 9th International Conference on the network of the future (NOF); 2018; Poznań, Poland.

How to cite this article: Salman O, Elhajj IH, Chehab A, Kayssi A. A machine learning based framework for IoT device identification and abnormal traffic detection. *Trans Emerging Tel Tech*. 2019;e3743. https://doi.org/10.1002/ett.3743