International Journal on Advanced Science Engineering Information Technology

Vol.12 (2022) No. 1
ISSN: 2088-5334

Early Generation and Detection of Efficient IoT Device Fingerprints Using Machine Learning

Vian Adnan Ferman [a,*], Mohammed Ali Tawfeeq [a]

[a] *Computer Engineering Department, Mustansiriyah University, Baghdad,10052, Iraq*
*Corresponding author: *egma018@uomustansiriyah.edu.iq*

Abstract— **The proliferation of Internet of Things (IoT) markets in the last decade introduces new challenges for network traffic analysis, and processing packet flows to identify IoT devices. This type of device suffers from scarcity, making them vulnerable to spoofing operations. In such circumstances, the device can be recognized by identifying its fingerprint. In this paper, a novel idea to elicit Device FingerPrint (DFP) is presented by extracting 30 features from the collected traffic packets of 19 IoT devices during setup and startup operations. Raspberry Pi 3 Model B+ is configured as an access point to collect and analyze the traffic of seven networked IoT devices using Wireshark Network Protocol Analyzer. Moreover, the rest of IoT devices traffic is taken from the publicly available network traffic dataset. Each IoT device's feature extraction process starts from getting Extensible Authentication Protocol over LAN (EAPOL) protocol, continuing with the other flowed protocols until the first session of Transmission Control Protocol (TCP) related to that device is closed. Depending on some produced variation of device traffic features, 20 fingerprints for each device are created. The probability theorem of Gaussian Naive Bayes (GNB) supervised machine learning is utilized to identify fingerprints of individual known devices and isolate the unknown ones. The performance evaluation for the proposed technique was calculated based on two measures, F1-score and identification accuracy. The average F1 score was around 0.99, while the overall identification accuracy rate was 98.35%.**

Keywords— **EAPOL protocol; gaussian naive bayes; IoT device fingerprint; network traffic analysis; Raspberry Pi.**

Manuscript received 29 Jan. 2021; revised 6 Apr. 2021; accepted 25 May 2021. Date of publication 28 Feb. 2022.

I. INTRODUCTION

The widespread availability with the huge diversity of Internet of Things (IoT) devices over the past decade has given attackers a great chance to hack IoT networks, bearing in mind that many of these devices are characterized by their limited resources and cheapness, which made them as open gates to attackers [1]–[3]. The Experts revealed that IoT devices' number may reach 75.4 billion by 2025, with a fivefold rise in ten years [4]. Recently, hackers provided Mirai botnets for renting over 400,000 IoT devices which was the largest amount of attack traffic ever [5], [6]. Besides, more than 600,000 worldwide devices were infected by Mirai, and these devices were mainly used to launch distributed denial-of-service attacks; therefore, the security of IoT networks has become crucial [7], [8].

The first defensive line of IoT network is: identifying the connected devices to isolate unauthorized devices. Network administrators can use software like packet sniffing tools to extract features and information from received packets of devices on a network and know which devices are connected or installed. However, it is time-consuming and is less accurate when the number of IoT devices is increased or installed new devices related to the same manufacturer (e.g., same prefixes of MAC addresses) of the existing devices [9], [10]. Furthermore, IoT devices of the same vendor, especially with the same functionality, have similar behaviors that become so hard to distinguish unauthorized devices. Moreover, it is so easy to spoof MAC addresses and change the DHCP features of some devices such as laptops. These reasons lead to a search for rapid and more precise techniques to identify the connected devices in the IoT network depending on their behaviors.

Device FingerPrint (DFP) uses a device-specific signature or packet feature set necessary for network communication [11]. The effective fingerprint of the device must confirm two characteristics (i) extracting features that are difficult to forge, (ii) fingerprint stability guaranteed despite the network environment change [12]. Using the device fingerprint technique, authorized and unauthorized devices can be identified without relying on the knowledge of the device's network or other assigned identities like Media Access

Control (MAC), Internet Protocol (IP), and International Mobile Equipment Identity (IMEI).

This paper proposed an approach for passive DFPs generation of 19 IoT devices (seven of them are lab experimental devices while the remaining are from the free online dataset: captures-Sentinel [13], [14]) during installation and startup operations depending on a list of protocols and some features extracted from these protocols. After collecting, analyzing, and preprocessing packets, different fingerprints are created for each device. Gaussian Naive Bayes (GNB) machine learning algorithm is used for identification purposes. Utilizing this approach, unknown devices of unknown manufacturers were perfectly distinguished. Meanwhile, the known devices were identified even if they were from the same manufacturer and almost even with a similar operation.

II. Materials and Method

This section is separated into five sub-sections. The related works are summarized in sub-section A. A brief overview of IoT devices and testbed devices is presented in sub-section B. The proposed method based on collecting and analyzing the network traffic is presented in sub-section C, feature extraction and data preprocessing to create device fingerprints are explained in sub-section D, and then the method of identifying the IoT devices is presented in sub-sections E.

A. Related Works

The tremendous expansion in the uses of IoT devices made them vulnerable to targeting, which made the trends of many researchers focus on identifying these devices. Meidan *et al.* [15] have used the Random Forest algorithm to identify IoT devices by training a multi-stage meta classifier. Throughout the first stage of the process, the classifier can differentiate between traffic produced by IoT and non-IoT devices. The overall classification model's accuracy is about 99%. Lin and Wang [16] tried to improve this work and reach a similar accuracy using the decision tree model. Machine learning techniques first identified 23 binary features from TCP/IP packets, then investigated their correlation with device types, device models, and device manufacturers.

Miettinen *et al.* [12] have presented an IoT SENTINEL system capable of automatically recognizing the type of 27 devices connected to an Access point. The identification step is done using a device fingerprint technique based on passively observed network traffic. To create a fingerprint for each IoT device during its setup phase, 23 features are collected from successive 12 packets. The final dimension of the fingerprint vector was 276. A Random Forest classification algorithm is used to build the model. The obtained average accuracy is 81.5%.

Shahid *et al.* [17] have applied six distinct machine learning classifiers to identify the type of four IoT devices connected to a smart home network by analyzing streams of sent and received packets. A set of features are selected from the generated network traffic data, such as the size of the first N packets sent and received and their corresponding inter-arrival times (IAT). Random Forest achieves the highest accuracy. Also, Hamad *et al.* [18] have applied a set of machine learning algorithms to automatically identify white-listed device types and individual device instances connected to a network. A fingerprint is created for each IoT device using network flow data and extract unique flow-based features from packets.

Kotak and Elovici [19] have automatically used deep learning to recognize known and unknown IoT devices. The communication behavior of IoT devices is represented using small-size images constructed from the network traffic payloads of IoT devices. Also, Bai *et al.* [9] have used deep learning to recognize the semantic type of a device automatically. The recoded accuracy is 74.8%.

Bezawada *et al.* [20] have proposed a technique to perform device behavioral fingerprinting using two available features from the network packets: packet header features and payload-based features. The technique is based on choosing five packets as the number of sessions packets whose features correspond to the device's fingerprint, and for each of the five packets, 20 features are extracted. A machine learning model is used to identify similar device types.

Salman *et al.* [21] have presented a framework for identifying IoT devices and malicious traffic detection. The proposed framework extracts packet size, IAT, the direction, and the transport protocol per every 16 packets of a flow to identify the source, the type of the generated traffic, and to detect network attacks. Distinct machine learning algorithms are applied; however, Random Forest results achieved the best accuracy.

Deng *et al.* [22] have designed IoTSpot to determine the IoT devices using their network traffic data. IoTSpot first extracts 19 traffic measurement metrics from TCP flows, then finds 11 important features out of them using Principal Component Analysis algorithm, after that leverages Random Forest modeling to construct a customized network traffic model for each IoT device.

Cheng *et al.* [23] have proposed a real-time method for IoT devices' auto-detection and classification system. This approach is accomplished by using passive listening to collect messages received from various IoT devices, then using a multi-classification recognition method to identify these devices based on the differences in the header fields of various devices.

Bao *et al.* [24] have proposed a hybrid supervised and unsupervised machine learning framework that uses network traffic to detect anomalies to secure IoT networks against unauthorized device access. This approach combines deep neural networks with clustering to classify 10 different IoT devices into known and unknown device categories and employs the Autoencoder technique to reduce the dimensionality of the dataset.

Most of the published research did not address the identification of IoT devices during the setup process, while some focused on identifying the genre of IoT devices or did not record high accuracy in identifying devices that have the same model. Furthermore, the identification of DFP, created during startup time requires high computational time due to their high dimension. Besides, IAT is considered in some research and uses statistical operations to increase its weight. IAT is an improper feature because it may conflict with other devices (especially devices within the same model or manufacturers) when extending the IoT system. Sometimes, IAT is an inaccurate feature within environments that suffer from inferior Internet service. Moreover, in some research, multiple features are extracted and generated, then a specific model for features dimensionality reduction selects the important one, increasing computational operations and time.

The main contribution of this study is to create a strong fingerprint for each IoT device utilized in this work during the setup stage. This fingerprint is suitable in dimension and stable even when the network environment changes. At the same time, the ability to detect and identify the individual device with high accuracy rates based on the probability theorem of Gaussian Naive Bayes.

B. *Utilized Devices*

IoT device refers to any device with an IP address used to connect to a network (e.g., smart switch, a smart plug, camera, home appliance, etc.) or any device that does not necessarily have an IP address. However, it can connect and send information to another device provided that the other device can connect to the internet (e.g., a smartwatch connected with the smartphone via Bluetooth). So connections of IoT devices may be through Wi-Fi based on IEEE 802.11n, Bluetooth, Zigbee, NFC., etc.

Seven IoT test lab devices with Wi-Fi connection types have been used in this work. The dataset is also enriched with the traffic of 12 IoT captures-Sentinel datasets (nine known and authorized devices, while the rest are considered unauthorized devices). A list with some information about all of these devices is shown in Table.1

To collect the IoT test lab devices traffics, Raspberry-Pi Model B+ has been used, a lightweight, single-board computer that supports 802.11.b/g/n/ac wireless LAN. Raspbian OS is installed on Raspberry Pi then configured as a routed wireless access point. The requirements for using a Raspberry Pi as an access point are not too high: the device needs to be connected via an Ethernet cable, and some configurations have to be set up after updating the operating system. A script is written, which can be run on the device without anything special installed. The configuration steps include network management services, defining the wireless interface IP, enabling routing and IP masquerading, configuring the services of DHCP and DNS [25].

After these steps, IoT devices can connect to Raspberry pi and take IP addresses. Wireshark Network Protocol Analyzer is used to read, collect, and analyze device traffic.

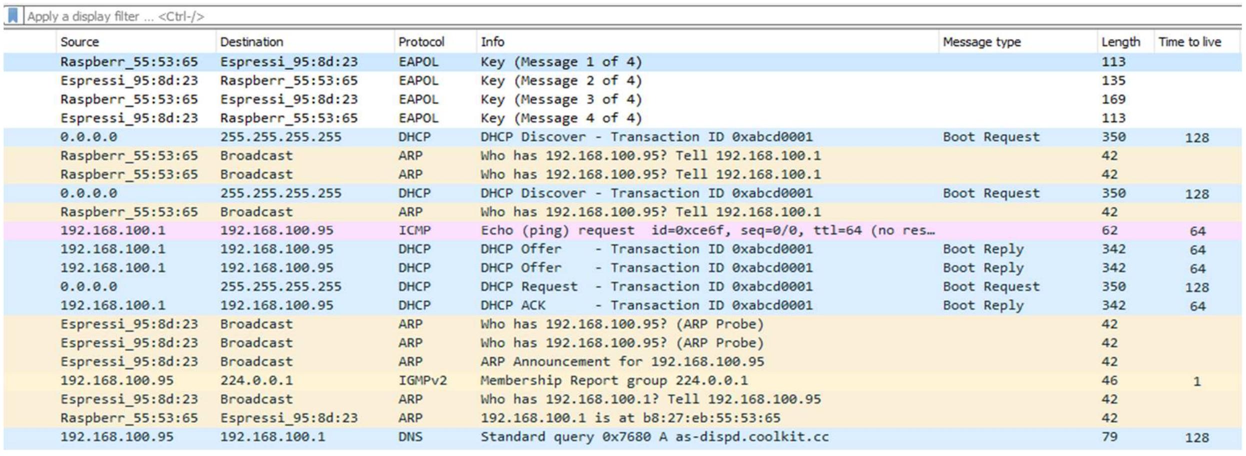
TABLE I
LIST OF USED IOT DEVICES

		Manufacturer	Device Name
1	IoT test Lab devices	SonoFF	SonoFF_Power_Strip
2			SonoFF_Power_Plug
3			SonoFF_Smart_Light_Bulb
4			SonoFF_Smart_Switch with Temperature Sensor
5		Google Assistance	Google_Home_Mini
6		Aswar	Aswar_Camera
7		TEKIN	TEKIN-Plug
8	IoT-Sentinel devises traffic	D-Link	D-LinkCam
9			D-LinkSensor
10			D-LinkSwitch
11			D-LinkWaterSensor
12		Edimax	EdimaxPlug1101W
13		Ednet	EdnetGateway
14		TP-Link	TP-LinkPlugHS100
15			TP-LinkPlugHS110
16		WeMo	WeMoSwitch
17		*iKettle	iKettle2
18		*SmarterCoffee	SmarterCoffee
19		*Withings	Withings

Where * refers to unauthorized devices.

C. *Network traffic collection and analysis*

After installing the IoT devices, the produced traffic from each device is collected in the form of a Packet Capture (PCAP) file; a sample of such packets is shown in Fig. 1. After that, the packets are analyzed. It is found that the number of packets per time rate differs from one device to another but all with high rates during installing or startup operation.



```
Source             Destination        Protocol  Info                                                     Message type    Length  Time to live
Raspberr_55:53:65  Espressi_95:8d:23  EAPOL     Key (Message 1 of 4)                                                     113
Espressi_95:8d:23  Raspberr_55:53:65  EAPOL     Key (Message 2 of 4)                                                     135
Raspberr_55:53:65  Espressi_95:8d:23  EAPOL     Key (Message 3 of 4)                                                     169
Espressi_95:8d:23  Raspberr_55:53:65  EAPOL     Key (Message 4 of 4)                                                     113
0.0.0.0            255.255.255.255    DHCP      DHCP Discover - Transaction ID 0xabcd0001                Boot Request    350     128
Raspberr_55:53:65  Broadcast          ARP       Who has 192.168.100.95? Tell 192.168.100.1                               42
Raspberr_55:53:65  Broadcast          ARP       Who has 192.168.100.95? Tell 192.168.100.1                               42
0.0.0.0            255.255.255.255    DHCP      DHCP Discover - Transaction ID 0xabcd0001                Boot Request    350     128
Raspberr_55:53:65  Broadcast          ARP       Who has 192.168.100.95? Tell 192.168.100.1                               42
192.168.100.1      192.168.100.95     ICMP      Echo (ping) request  id=0xce6f, seq=0/0, ttl=64 (no res…                 62      64
192.168.100.1      192.168.100.95     DHCP      DHCP Offer    - Transaction ID 0xabcd0001                Boot Reply      342     64
192.168.100.1      192.168.100.95     DHCP      DHCP Offer    - Transaction ID 0xabcd0001                Boot Reply      342     64
0.0.0.0            255.255.255.255    DHCP      DHCP Request  - Transaction ID 0xabcd0001                Boot Request    350     128
192.168.100.1      192.168.100.95     DHCP      DHCP ACK      - Transaction ID 0xabcd0001                Boot Reply      342     64
Espressi_95:8d:23  Broadcast          ARP       Who has 192.168.100.95? (ARP Probe)                                      42
Espressi_95:8d:23  Broadcast          ARP       Who has 192.168.100.95? (ARP Probe)                                      42
Espressi_95:8d:23  Broadcast          ARP       ARP Announcement for 192.168.100.95                                      42
192.168.100.95     224.0.0.1          IGMPv2    Membership Report group 224.0.0.1                                        46      1
Espressi_95:8d:23  Broadcast          ARP       Who has 192.168.100.1? Tell 192.168.100.95                               42
Raspberr_55:53:65  Espressi_95:8d:23  ARP       192.168.100.1 is at b8:27:eb:55:53:65                                    42
192.168.100.95     192.168.100.1      DNS       Standard query 0x7680 A as-dispd.coolkit.cc                              79      128
```

Fig. 1 Sample of captured packets during the startup operation

Furthermore, the packet rate of some devices becomes exceptionally low after the startup process is completed depending on the nature of the functionality of each IoT device. Smart switches, smart bulbs, smart power strips, smart plugs, and even IoT sensors are low data rate devices compared with Google Home Mini and cameras. Thus, it may be difficult to create good fingerprints for low-traffic rate devices. Moreover, extracting the device's fingerprint in the startup stage gives a better chance of detecting intrusive or hacked devices early. Fig.2 shows the traffic of the startup time for some of the lab IoT devices within 250 seconds.

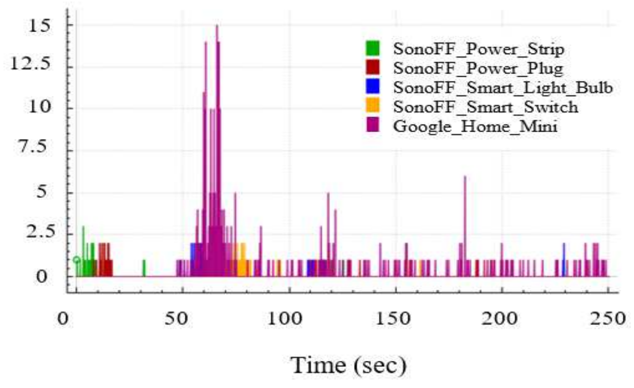


Fig. 2 Traffic of some IoT devices during 250 sec in startup time.

During the connection of any device to an access point, Extensible Authentication Protocol over LAN (EAPOL) protocol starts sending packets from the authenticator (access point) to the supplicant (device) to generate some encryption keys which can be used to encrypt actual data transmitted over a wireless medium so that 4-way handshake will be accrued. Creating DFP is begun by getting EAPOL packet and continued with the following packets that carry important protocols and features to complete connection with the internet. Besides the lab IoT devices, 12 of the 31 devices collected traffics of captures_IoT-Sentinel are taken (the remaining public traffics datasets are not taken since they are either without EAPOL packets or contain a few packets).

While traffics are analyzed, some properties are seen as important features and should be considered to create a strong DFP like:

- First TCP session properties as the number of packets in the session, packet length, segment length, window size, and protocol type used to depend on ports type (Well-known port like 443 for HTTPS and 80 for HTTP, or TCP with Dynamic ports).
- In some devices, UDP data packets are transmitted between the first TCP session, so their properties are also important, as the number of packets carrying UDP data with the same port numbers as the first UDP data packet and data length.

In addition to the previous, some of the first TCP sessions of the captures_IoT-Sentinel dataset (WeMoSwitch, TP-LinkPlugHS100, and TP-LinkPlugHS110) are with a large number of packets or may continue with no packet carrying FIN flag. Only the first 20 TCP packets' details are taken in these cases. Also, the first TCP session of D-LinkCam is found with unstable protocol because it may be HTTPS or HTTP. Fig. 3 depicts the number of authorized IoT devices for each port type of the first TCP session.

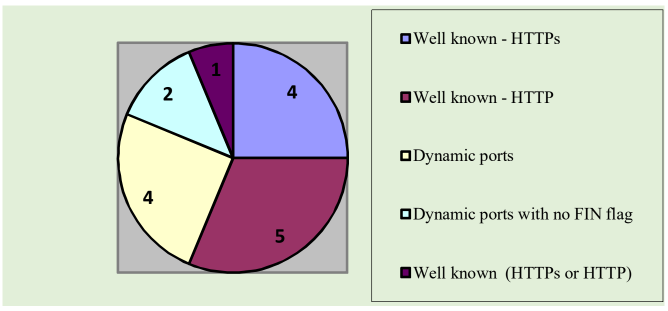


Fig. 3 Number of IoT devices for each port type of First TCP session.

Meanwhile, Fig.4 shows the number of packets transmitted within the session, excluding TP-LinkPlugHS100, TP-LinkPlugHS110, and D-LinkCam.

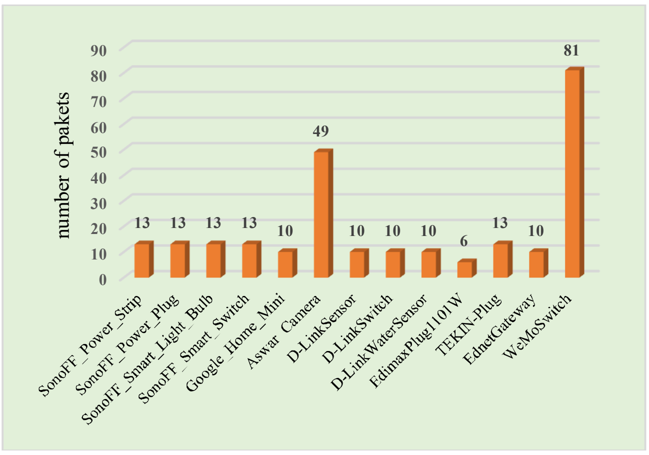


Fig. 4 Number of packets in the first TCP session of each device.

There are differences in port types and the number of packets in the first TCP session. On the other hand, the similarity is found especially between devices of the same manufacturer, so other features are considered like segment length, window size, time to live, DNS Query Name, and DHCP details. In this work, 25 features are extracted from each IoT device's traffic during the initial step, as shown in Table.2

TABLE II
THE INITIAL LIST OF FEATURES

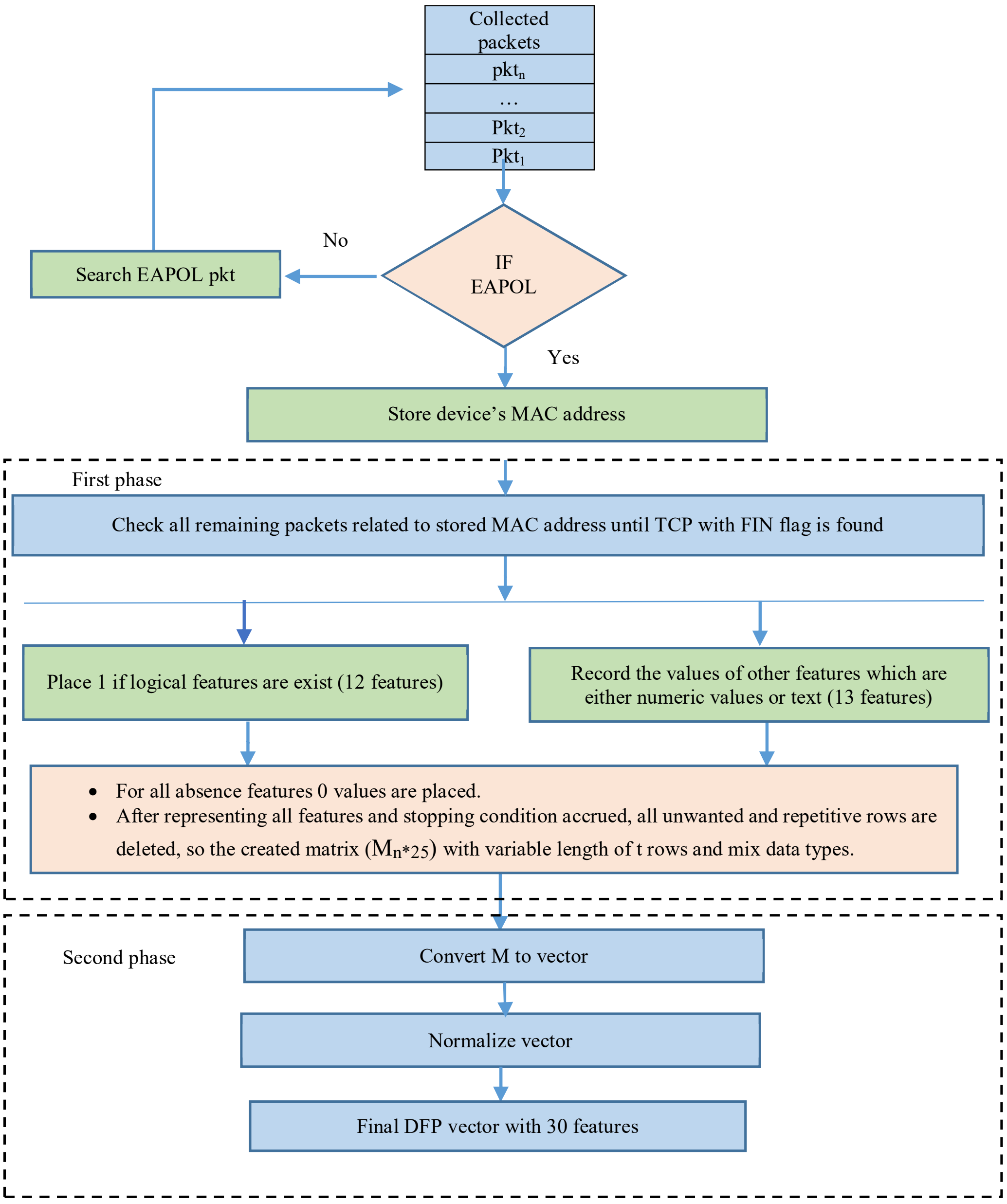
Feature Details	No. of Features
Data Link layer: Source and destination MAC addresses, ARP protocol, packet length if TCP.	4
Network layer: Source and destination IP addresses, EAPOL, ICMPV6, IGMPv2, IGMPv3.	6
Transport layer: UDP data, UDP data length, TCP segment length, TCP window size.	4
Application layer protocols: DHCP, DNS (or MDNS).	2
IP: Time To Live (TTL).	1
TCP: TCP with HTTPS protocol, TCP with HTTP protocol, TCP with Dynamic source and destination ports	3
DHCP: Length of DHCP Parameter Request List (LDHCPPRL), Maximum DHCP Message Size (MDHCPMS), Vendor class identifier (VCI), and Host Name (H).	4
DNS or MDNS: Query Name.	1

D. DFP Generation

The proposed technique for generating the fingerprint of each IoT device relies on a two-phase completion (using python scapy): Feature extraction and data preprocessing. The DFP generation procedure is clarified in Fig.5.

1) Feature Extraction: Feature extraction of each device is begun from getting the EAPOL protocol packet. Then, the new MAC address is saved, and other packets related to that address are checked until the first TCP session is closed with FIN flag. If there is no packet with FIN flag or long TCP session, only 20 TCP packets are taken. Furthermore, if UDP data packets were coming in between TCP session or there are no TCP packets and all packets are UDP data packets such as in some cases of a camera, so source and destination port numbers are stored and collected all UDP data coming or sent from that device with the same stored port numbers.

Initially, 2D matrix (M) with 25 columns and variable length of rows is created. Each row represents packet (pkt) with 25 features (F = {f_1, f_2, f_3, …, f_{25}}) which are listed in Table 2. In this phase, place 1 for all protocols in the features list and also for TCP packets with HTTPS protocol, HTTP protocol, or Dynamic source and destination ports, UDP data, and Vendor class identifier (these protocols and features are encoded as logical features). The values of other features are recorded either as numeric values like TCP packet length, segment size, window size, UDP data length, TTL, Length of DHCP Parameter Request List, and Maximum DHCP Message Size or as text like MAC addresses, IP addresses, DHCP Host Name, and DNS (or MDNS) Query name.



Collected packets
pkt$_n$
…
Pkt$_2$
Pkt$_1$
IF EAPOL
No
Search EAPOL pkt
Yes
Store device's MAC address
First phase
Check all remaining packets related to stored MAC address until TCP with FIN flag is found
Place 1 if logical features are exist (12 features)
Record the values of other features which are either numeric values or text (13 features)
For all absence features 0 values are placed.
After representing all features and stopping condition accrued, all unwanted and repetitive rows are deleted, so the created matrix (M_{n*25}) with variable length of t rows and mix data types.
Second phase
Convert M to vector
Normalize vector
Final DFP vector with 30 features

Fig. 5 DFP generation phases

All unwanted and repetitive rows are deleted after representing all features and stopping conditions accrued. The created matrix (M_{n*25}) has mixed data types and different row sizes depending on the device type and even within the same device type since some protocol packets appear or are absent each time the traffics is collected. Eq. (1) represents the created matrix in this phase, where n represents the number of packets approved for each device.

$$M = \begin{array}{c} pkt_1 \\ pkt_2 \\ \vdots \\ pkt_n \end{array} \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,25} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,25} \\ \vdots & \vdots & \vdots & \vdots \\ f_{n,1} & f_{n,2} & \cdots & f_{n,25} \end{bmatrix} \quad (1)$$

2) Data Preprocessing: This phase involves performing some statistical calculations on the features extracted in the previous phase to generate additional features. After that, all the extracted features for each device in both phases are converted to a single vector which represents the DFP of that device. The features added in this phase are prepared and calculated as follows:

- Adding all elements of each column that contain logical value 1,0 so the count of 12 features (columns) are gotten.
- The columns of Maximum DHCP message size and length of the DHCP parameter request list contain (0) elements except that one cell in each column contains value rather than 0; therefore, those values are taken.
- Min, max, and average are computed from values in the TCP packet length, window size, and TTL column columns. Also, min and max of UDP data length are found. An average segment length column is obtained.
- DHCP Host Name is converted to ASCII code and found min (MinH), max (MaxH), average (AvgH), and length (LH) of the result.
- Merge columns of source and destination MAC addresses, remove repetitive addresses, and then find the remaining addresses count. The same procedure is done to the source and destination IP address columns.
- Number of rows (n) is added.
- Since the protocol ICMPv6 is noticeable in some devices and not on others bearing the same brand, it may be considered an important feature that should be exploited to strengthen the proposed DFP. The locations (row indices of matrix M) of presented ICMPv6 are taken and concatenated to represent one number. If there are more than three ICMPv6 packets, only the first three locations are taken then divided by the max location for early normalization purposes and appended to the fingerprint vector (e.g., if the first three ICMPv6 packets appear in locations i, j, and k respectively, so they are converted to ijk/k).
- DNS query names may be unique for some devices with the same manufacturer and function, but some devices like cameras have more DNS query packets requests to resolve DNS query names to their relevant IP address. A lookup table is created that contains all query names giving a specific number to each of them, and if there is a new query name, 0 value is placed, and also DHCP features are cleared to distinguish unknown devices in a predication phase.
- Another field associated with the DNS query name is added to distinguish a new device with the same known brand. It is found by computing the number of DNS packets, but if one of DNS query names is not found in the lookup table, 0 value is placed.
- To reduce the similarities between DFPs, the values of DHCP features are reduced to one value by taking the average value (Average (LDHCPPRL, MDHCPMS, VCI, MinH, MaxH, AvgH, LH)).
- The final DFP vector is of 30 features ($V = [f_1, f_2, f_3, \ldots, f_{30}]$) that contains a combination of small and high values, so normalization is required. MinMaxScaler and StandardScaler are applied to compare the accuracy resulting from each of them.

E. IoT Device Identification

In this work, the Naive Bayes method has been adopted to identify the IoT devices based on their fingerprint. Naive Bayes *is* a supervised machine learning algorithm used for classifying binary and multiclass classification problems based on the probability's principle. The computed probabilities are saved in a list for a learned model, including classes' probabilities and conditional probabilities for each feature value given each class value.
So the posterior probabilities are calculated using this list as shown in Eq. (2).

$$p(C_n \mid V) = \frac{p(V \mid C_n) \times p(C_n)}{P(V)} \qquad (2)$$

where C_n is the class name or the device name, and (V) is the features' vector ($P(V|C_n) = P(f_1|C_n) \times P(f_2|C_n) \times P(f_m|C_n) \times \ldots \times P(f_{30}|C_n)$).

Gaussian Naive Bayes is used for multiclass classification since the proposed features' values are real values. Normal distribution is needed to estimate mean (μ) and standard deviation (σ) for each input feature in the training data for each class. Eq. (3) represents Gaussian Probability Density Function (Gaussian PDF) for each feature, where x is the input value for the input feature.

$$\mathrm{PDF}(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \times e^{-\frac{(x-\mu)^2}{\sigma^2}} \qquad (3)$$

After calculating the posterior probabilities for different classes, the class with the maximum probability is selected according to Eq. (4), where P (V) is dropped as it is constant and only used for normalization purposes [26],[27].

$$\mathrm{MAP}(C_n) = \mathrm{Max}\,(p(V \mid C_n) \times p(C_n)) \qquad (4)$$

Gaussian Naive Bayes is implemented using scikit-learn library. Gaussian noise (with μ = 0 and σ =1) is added randomly for building a robust model.

To assess the performance of the proposed approach, the following metrics were used [21]:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \qquad (5)$$

$$\text{Precision} = \frac{TP}{TP+FP} \qquad (6)$$

$$\text{Recall} = \frac{TP}{TP+FN} \qquad (7)$$

$$FI - score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \qquad (8)$$

TP, FP, TN, and FN are truly positive, false positive, true negative, and false negative. A block diagram representing the proposed IoT device identification system is shown in Fig. 6.

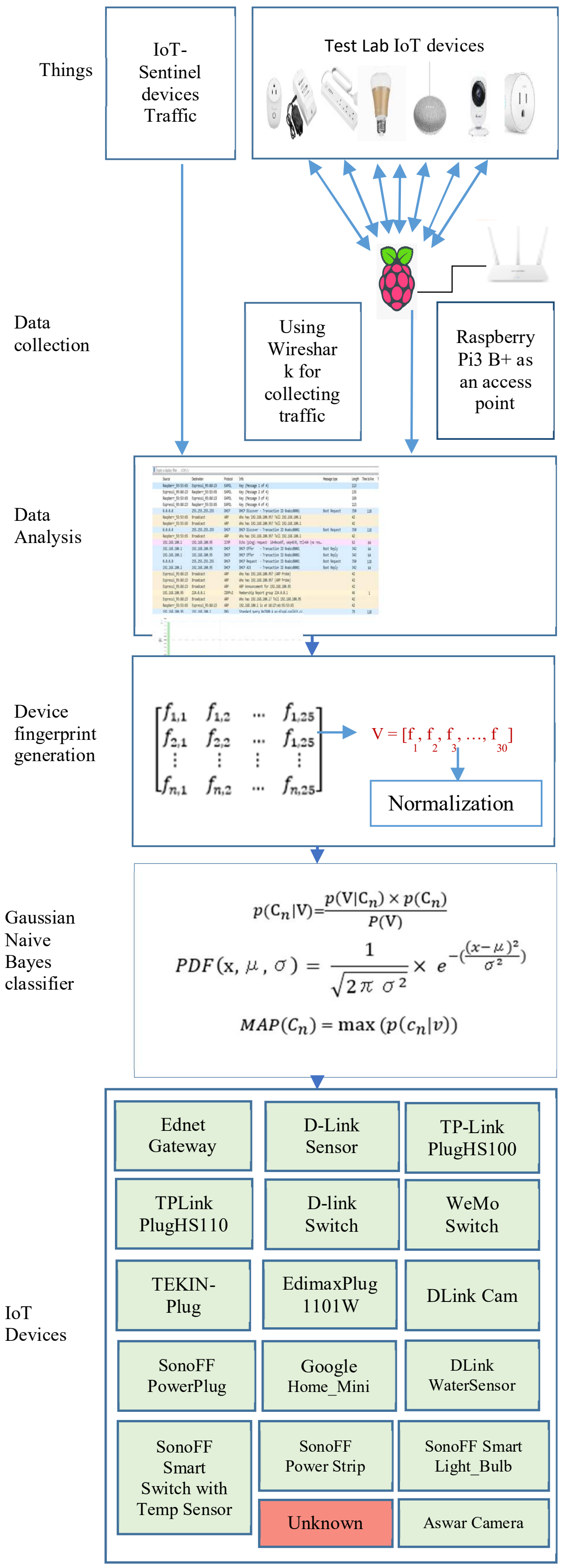

Things
IoT-Sentinel devices Traffic
Test Lab IoT devices
Data collection
Using Wireshark for collecting traffic
Raspberry Pi3 B+ as an access point
Data Analysis
Device fingerprint generation
$\begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,25} \\ f_{2,1} & f_{2,2} & \cdots & f_{1,25} \\ \vdots & \vdots & \vdots & \vdots \\ f_{n,1} & f_{n,2} & \cdots & f_{n,25} \end{bmatrix}$
$V = [f_1, f_2, f_3, \ldots, f_{30}]$
Normalization
Gaussian Naive Bayes classifier
$p(C_n|V) = \frac{p(V|C_n) \times p(C_n)}{P(V)}$
$PDF(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\ \sigma^2}} \times e^{-(\frac{(x-\mu)^2}{\sigma^2})}$
$MAP(C_n) = \max(p(c_n|v))$
IoT Devices
Ednet Gateway
D-Link Sensor
TP-Link PlugHS100
TPLink PlugHS110
D-link Switch
WeMo Switch
TEKIN-Plug
EdimaxPlug 1101W
DLink Cam
SonoFF PowerPlug
Google Home_Mini
DLink WaterSensor
SonoFF Smart Switch with Temp Sensor
SonoFF Power Strip
SonoFF Smart Light_Bulb
Unknown
Aswar Camera

Fig. 6 Proposed system of IoT device identification

III. Results and Discussion

The generated DFPs data are splitted into 75% for training data and 25% for testing data. The training process is done in two modes: before and after adding Gaussian noise. The obtained accuracy before adding noise is about 96% on average after data normalization with MinMaxScaler and StandardScaler, while it is about 97.5% with MinMaxScaler, and 98.35% with StandardScaler after noise addition. The purpose of adding Gaussian noise in this approach is to raise the model accuracy rate and make a robust model that can predict devices' fingerprints if features' values are changed when dealing with wireless data traffic. Furthermore, noise addition and normalizing features during preprocessing lead to decrease misclassifications of devices with the same model as shown in Table 3, in which the F1-score and accuracy of each device are computed as an average for six times of the model execution.

TABLE III
Identification Result of GNB Model

Device Name	**Results before noise addition**		**Results after noise addition**	
	F1-score	**Accuracy**	**F1-score**	**Accuracy**
Aswar_Camera	1	1	0.99	0.99
D-LinkCam	0.98	0.99	1	1
D-LinkSensor	0.96	0.99	1	1
D-LinkSwitch	0.95	0.99	1	1
D-LinkWaterSensor	0.95	0.99	1	1
EdimaxPlug1101W	1	1	1	1
EdnetGateway	1	1	1	1
Google_Home_Mini	0.98	1	1	1
SonoFF_Power_Plug	0.90	0.99	0.98	0.99
SonoFF_Power_Strip	0.94	0.99	0.95	0.99
SonoFF_Smart_Light_Bulb	0.94	0.98	0.95	0.99
SonoFF_Smart_Switch	0.97	0.99	0.99	1
TEKIN-Plug	0.99	1	1	1
TP-LinkPlugHS100	0.93	0.99	0.95	0.99
TP-LinkPlugHS110	0.97	0.99	0.96	0.99
WeMoSwitch	0.98	0.99	0.99	0.99
Unknown	1	1	1	1

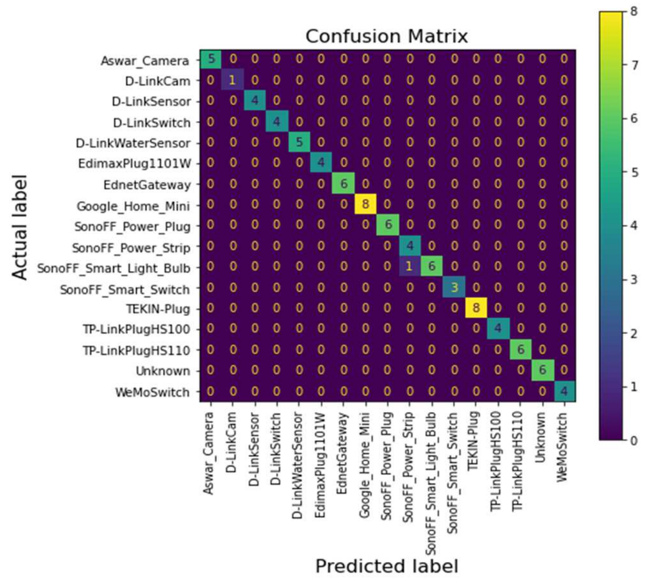


Fig. 7 Confusion matrix of GNB model after adding Gaussian noise.

While testing data and both with and without noise, sometimes only one of WeMoSwitch DFP is identified as TP-LinkPlugHS100 DFP or Aswar_Camera DFP. Furthermore, misclassification accrued between devices with the same model as SonoFF or TP-Link devices even after noise addition, but still, they are given more acceptable accuracy. To differentiate the authorized devices from unauthorized, DFPs are created for three IoT devices traffic (iKettle2, SmarterCoffee, and Withings) taken from captures_IoT-Sentinel. During the preprocessing step, DNS Query Names of these devices traffic is not found in the lookup table file, so 0 value is placed, and there is no need to check DHCP features of unknown devices. This way, unauthorized devices are identified with a 100% F1- score. Fig.7 shows the confusion matrix of this work after noise addition and applying StandardScaler. Only one identification error relevant to the SonoFF model has occurred.

IV. Conclusion

In this study, real-time IoT device fingerprints are created during the setup and startup phases, then using Gaussian Naive Bayes machine learning for identification purposes. This work can identify the fingerprints of individual IoT devices with 98.35% as an average accuracy. The proposed model improves the previous work [13] by extracting the features as soon as getting the EAPOL packet. At the same time, it does not rely on all features, but instead, it generates and uses new features. The creation of a lookup table for DNS Query Names makes a good feature to distinguish devices' models and isolate the unknown devices while reducing DHCP features to be one feature, and the addition of Gaussian noise and normalization during data preprocessing increased the contrast between devices of the same model. The feature aggregation method of converting a matrix to be a 30-dimensional vector decreases the computational operations and time and increases the effect of some features that may lose their importance by a high dimensionality vector. As a suggestion for future work, the creation of DFP can be extended to cover the transmitted packets during the device's running time and the installation or setup time.

Acknowledgment

The authors are grateful to Mustansiriyah University for supporting this work. Also, we appreciate the effort made to make the captures_IoT-Sentinel pcap dataset available, which enriches our dataset.

References

[1] A. Aksoy and M. H. Gunes, "Automated IoT device identification using network traffic," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[2] S. Zeadally and M. Tsikerdekis, "Securing Internet of Things (IoT) with machine learning," *Int. J. Commun. Syst.*, vol. 33, no. 1, p. e4169, 2020.

[3] A. Sivanathan *et al.*, "Classifying IoT devices in smart environments using network traffic characteristics," *IEEE Trans. Mob. Comput.*, vol. 18, no. 8, pp. 1745–1759, 2018.

[4] T. Alam, "A reliable communication framework and its use in internet of things (IoT)," *CSEIT1835111| Receiv.*, vol. 10, pp. 450–456, 2018.

[5] B. Charyyev and M. H. Gunes, "IoT Traffic Flow Identification using Locality Sensitive Hashes," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[6] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer (Long. Beach. Calif).*, vol. 50, no. 7, pp. 80–84, 2017.

[7] M. Antonakakis *et al.*, "Understanding the mirai botnet," in *26th {USENIX} security symposium ({USENIX} Security 17)*, 2017, pp. 1093–1110.

[8] M. M. Salim, S. Rathore, and J. H. Park, "Distributed denial of service attacks and its defenses in IoT: a survey," *J. Supercomput.*, pp. 1–44, 2019.

[9] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Z. Yang, "Automatic device classification from network traffic streams of internet of things," in *2018 IEEE 43rd conference on local computer networks (LCN)*, 2018, pp. 1–9.

[10] A. Sivanathan *et al.*, "Characterizing and classifying IoT traffic in smart cities and campuses," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2017, pp. 559–564.

[11] S. Aneja, N. Aneja, and M. S. Islam, "IoT device fingerprint using deep learning," in *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, 2018, pp. 174–179.

[12] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 1, pp. 94–104, 2015.

[13] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT sentinel: Automated device-type identification for security enforcement in IoT," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 2177–2184.

[14] "The kaggle website," 2021.

[15] Y. Meidan *et al.*, "ProfilIoT: a machine learning approach for IoT device identification based on network traffic analysis," in *Proceedings of the symposium on applied computing*, 2017, pp. 506–509.

[16] Y. C. Lin and F. Wang, "Machine Learning Techniques for Recognizing IoT Devices," in *International Computer Symposium*, 2018, pp. 673–680.

[17] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, "Iot devices recognition through network traffic analysis," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 5187–5192.

[18] S. A. Hamad, W. E. Zhang, Q. Z. Sheng, and S. Nepal, "IoT device Identification via network-flow based fingerprinting and learning," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2019, pp. 103–111.

[19] J. Kotak and Y. Elovici, "IoT device identification using deep learning," in *Conference on Complex, Intelligent, and Software Intensive Systems*, 2020, pp. 76–86.

[20] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Iotsense: Behavioral fingerprinting of IoT devices," *arXiv Prepr. arXiv1804.03852*, 2018.

[21] O. Salman, I. H. Elhajj, A. Chehab, and A. Kayssi, "A machine learning-based framework for IoT device identification and abnormal traffic detection," *Trans. Emerg. Telecommun. Technol.*, p. e3743, 2019.

[22] L. Deng, Y. Feng, D. Chen, and N. Rishe, "IoTspot: Identifying the IoT devices using their anonymous network traffic data," in *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*, 2019, pp. 1–6.

[23] W. Cheng, Z. Ding, C. Xu, X. Wu, Y. Xia, and J. Mao, "RAFM: A Real-time Auto Detecting and Fingerprinting Method for IoT devices," in *Journal of Physics: Conference Series*, 2020, vol. 1518, no. 1, p. 12043.

[24] J. Bao, B. Hamdaoui, and W.-K. Wong, "IoT device type identification using hybrid deep learning approach for increased IoT security," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*, 2020, pp. 565–570.

[25] L. Nagy and A. Coleşa, "Router-based IoT Security using Raspberry Pi," in *2019 18th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, 2019, pp. 1–6.

[26] J. Brownlee, *"Naive Bayes," in Master Machine Learning Algorithms: discover how they work and implement them from scratch*. 2016.

[27] F.-J. Yang, "An implementation of naive Bayes classifier," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2018, pp. 301–306.