2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering

IoT Device Identification via Network-Flow Based Fingerprinting and Learning

Salma Abdalla Hamad*, Wei Emma Zhang*, Quan Z. Sheng*, Surya Nepal†
* *Department of Computing, Faculty of Science and Engineering, Macquarie University, Sydney, Australia*
Email: salma-abdalla-ibrahim-mah.h@students.mq.edu.au, w.zhang@mq.edu.au, michael.sheng@mq.edu.au
†*Data61, CSIRO, Sydney, Australia*
Email: surya.nepal@data61.csiro.au

***Abstract*—Nowadays, increasing number of intelligent devices and smart sensors are connected by Internet of Things (IoT) techniques, and have helped people to manage and improve their lives. However, security issues are emerging in IoT, among which things identification is one of the challenges in that various solutions of different vendors, standards, protocols and communities groups coexist. In this paper, we address the challenge of IoT device identification by analyzing a sequence of packets from its high-level network traffic, i.e., network-flow data and extract unique flow-based features to create a fingerprint for each device. We adopt supervised machine learning techniques for the identification task. The proposed approach can automatically identify white-listed device types and individual device instances connected to a network. Moreover, we propose a security system model design that enables enforcement of rules for constraining the IoT device communications as per their given privileges. Unknown or suspicious devices with abnormal behaviour can be identified, and their communication is restricted for further monitoring. We show that the presented approach is effective in identifying white-listed device types with average accuracy up to 90.3% which is a 9.3% increase compared with the state-of-the-art technique. Individual device instances having the same model and vendor as well as unknown devices are correctly identified with minimal performance overhead.**

***Keywords*-IoT Security, Machine Learning, IoT Identification, Automatic IoT Authentication, Fingerprinting.**

I. Introduction

The Internet of Things (IoT) interconnects the physical world with the digital world with a significant impact on all life aspects. Accordingly, in recent years, IoT is overspreading in both industrial and household sectors. It is predicted that the number of IoT devices to almost reaching 500 devices for each household in 2022 [1]. These heterogeneous smart devices need to be integrated within the common infrastructure. Despite the number of benefits in terms of usability and flexibility, these devices also bring security concerns [2]. Among these concerns are device identification, authentication and privacy. Recently, a large number of attacks that target the security and privacy of IoT systems were performed [3]. For instance, attacks that target faking or change the readings of sensors or maliciously access devices, to capture sensitive data or to change the device into a state useful for the attacker [3]. Security of smart sensors usually, is one of the last, if not the least priorities for IoT companies and developers, which result in a number of successfully exploited device vulnerabilities [4]. With this prompt increase in the number of heterogeneous IoT devices that join the network and the growth of the network complexity and size, appropriate solutions are needed to help network administrators manage the networks. A number of security concerns can be tackled by using well-designed identification and authentication mechanisms. This allows network administrators to manage and enforce security controls for individual device. Automatic identification of devices, not only allow the administrators to ensure devices are connected to the correct zone with the right privileges, but also gives them information such as product vendor, product type, operating system, and software version of a particular device. Such information can help them locate vulnerable devices, unauthorized/rogue devices, or devices that need patches or updates in the network [5].

IoT device identification is challenging due to the existence of various IoT device types, communication protocols, and control interfaces. An untrusted or unknown devices can masquerade as another device from the network's white-list and gain network access. Moreover, rogue or compromised devices might try to gather information or disrupt the normal functions of other devices within the infrastructure. This can be detected by comparing its behavior with its baseline fingerprint.

Traditionally, device identification and/or authentication depend only on the cryptographic authentication techniques. However, IoT devices usually are constrained. Accordingly, these devices use small cryptographic keys to be used for the cryptographic operations. The use of small keys makes IoT devices vulnerable and easier to be compromised. Machine learning identification techniques are used to complement and work in parallel with the traditional cryptographic authentication techniques to compensate for the small key size problem and ensure higher identification/authentication security for such devices.

Machine learning device identification techniques are recently introduced in the literature to automate the identification process. Each device should have a unique fingerprint to be used by the machine learning techniques for efficient automatic identification. A number of methods for device fingerprinting are introduced in the literature. Such techniques include physical, wireless and network traces fingerprinting. Physical fingerprinting is based on the device physical features, such as device hardware Clock Skews [6]. Wireless fingerprinting techniques, focus on certain protocols to identify devices as in [7], [8]. Network traces fingerprinting, extract features from the Internet Protocol (IP) network traces of the device in question, such as in [5], [9], [10]. Existing fingerprinting solutions that target device identifications, either focus on a limited number of protocols, which makes them not suitable with the heterogeneous nature of IoT devices, or recorded low identification rates. Thus, they are not applicable to be used in real-time IoT networks (These solutions will be discussed in more details in section II). Therefore, we present a novel fingerprinting technique that identifies device types and individual device instances with high identification rates.

This work focuses on identifying real-time IoT devices, that are connecting and communicating using IP-Network. We propose an approach that uses a passive behavioural fingerprinting technique to identify devices. The fingerprint is created by extracting features from a sequence, a set of packets ordered by time-stamps. These features are extracted from the network packets header and payload to generate a unique fingerprint for each device. These behavioural and flow-based fingerprints can be used as a baseline to continuously monitor the device behavior while connected to the network. To identify an unseen device connecting to the network, we collect a limited number of sequential packets to create the fingerprint. Then we apply multi-class classifiers on the extracted features to predict the device name. We extensively compare several widely-used classifiers and find Random Forest consistently shows the best performance. Details and analysis of the used features are discussed in Section IV-A.

Contributions. The key contributions of this work are as follows:

- We propose a novel fingerprinting technique that applies supervised machine learning to IP-enabled IoT devices, to recognize multiple known IoT devices types (from white-list). The presented fingerprint study shows that with a sequence of a limited number of packets, it can recognize a device type quite accurately. Moreover, unknown or rogue devices connecting to the network can be identified.
- Our proposed approach can identify individual devices from the same type (same vendor and model), which is an essential feature to manage the network and the access control solutions for each device not only device type.
- We present a security model design that can identify devices and provide each identified device with its Pre-defined privileges. Each device will be allowed to connect to a certain privilege zone as per its identification. Any behavioural changes for any of the communicating IoT devices from the behavioral baseline can be detected by a continuous random moving window fingerprinting of the devices. Hence, the identified rogue devices will be quarantined for further monitoring.

The rest of the of this paper is organized as follows. Section II presents related work, reflecting the differences between previous solutions and this proposed approach. In Section III, we describe a security model design that securely identify devices and then connect them to their communication zone. We then propose a device identification methodology in Section IV. Section V presents the experiments and results. Finally, Section VI concludes the paper.

II. Related Work

IoT device identification and fingerprinting is a recent research topic and is in its early stages. It is evolving with the growth of the IoT industry.

Franklin et al. [11] presented a passive fingerprinting technique that can identify different wireless driver implementations on network-connected devices. Francois et al. in [7] and [8] proposed two fingerprints approaches based on the protocols used, with high detection accuracy. These solutions analyse behavior of certain well-known protocols, specifically, SIP as a protocol. Being limited to certain types of protocols might limit the scalability of solution. Moreover, IoT heterogeneity results in a diverse number of protocols used within IoT devices. Hence, it's difficult to use such techniques in IoT environments. In contrast, we propose an approach that can be used with a variety of connection technologies. Our proposed approach is tested on a dataset with devices that communicates with more than four types of connectivity technologies.

A general purpose of device type identification was described by Radhakrishnan et al. in [9]. The main aim of their solution is to recognize the device types connected to the IP-network. Their solution is based on packets inter-arrival times to extract features related to a particular application. Our work is influenced by their proposed solution. We execute a number of experiments to select the best features. The analysis of these experiments showed that including statistics of inter-arrival times between packets, enhanced the overall accuracy. Siby et al. proposed IoTScanner [10], which identifies devices by visualizing Medium Access Control (MAC) layer traffic. This solution is more concerned with the presence of devices on the network. The proposed technique can be very useful for high-level network mapping. However, it will add a lot of processing loads if used periodically for identifying devices and comparing their fingerprint against the baseline. On the contrary, our approach uses only a limited number of sequential packets to fingerprint any device, with minimal performance overhead.

IoT Sentinel [5] was presented by Miettinen et al., which proposed a passive fingerprinting technique that analyzes the headers of packets for a connecting device. The authors used Random Forest single classifier to identify each device independent from the other 26 devices. Their solution is capable of detecting unknown devices. The authors evaluated their solution with off-the-shelf 27 different IoT device type. The presented solution achieve accuracy average rates of 81% for identifying IoT device types only. A shortcoming of this solution is that it's not capable of identifying individual device instances from the same device type (same vendor and model). Moreover, the prediction accuracy in this solution is lower than other states of the art solutions. But, it has considered a diversity of IoT devices with different protocols. The prediction accuracy of 10 devices from the dataset was around 50% or below. Our approach compliments this solution to achieve higher prediction rates and well as recognize different devices having the same vendor and model. We propose a device fingerprinting technique that captures the patterns in a limited number of sequential packets. Moreover, features from both packet headers and payload, are included, to create a unique fingerprint for each individual device not only device type.

III. System and Threat Model

In this section, we present a security identification, authorization and security enforcement model design. We propose a general approach that can be used by both domestic and industrial sectors, in which IoT devices are connected to the network through a switch or gateway using wireless or wired IP-network. The model identifies IoT devices connecting to an infrastructure to allow them to join the network and set their network privileges. Moreover, IoT devices already connected to the network will be re-checked continuously against a baseline, to ensure the correctness of the identification and detect any malicious or mis-behaving devices.

Threat Model. Adversaries try to compromise IoT devices on the network to either use it as a pivot to compromise other devices connected to the network or to ex-filtrate data from the device [5]. We assume the adversaries can bypass the traditional cryptography authentication of devices. This is because most of the IoT device has low-computational power, leading to the use of limited key sizes. The use of such small key length security algorithms may lead to a more vulnerable and easier to be compromised IoT devices using powerful computers.

The ultimate goal of any security system is to preserve privacy as well as prevent attackers from compromising devices and exploit vulnerabilities on any device in the network. This can be achieved by

- Identifying any device that tries to connect to the infrastructure in real-time.

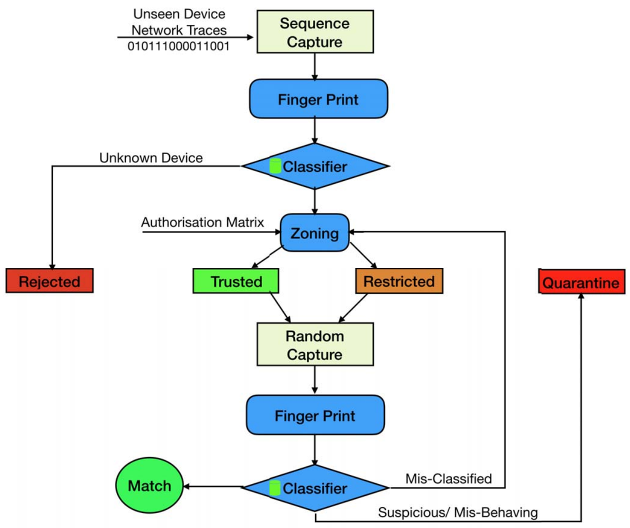

Figure 1. System Model for IoT Device Identification and Security Enforcement.

- Restricting the privileges of each of the connected devices, as per pre-defined rules. These pre-defined rules define where devices should be sited, by separating the devices in different privilege zones according to their communication and data importance.
- Comparing the behavior of current device, with a baseline generated during the normal communication of this device. To help in detecting rogue (miss-classified as authorized) or compromised devices, thus limiting their network resources and privileges for further monitoring.

Enforcement. Fig. 1 presents our proposed security enforcement model. The first stage of the model is used when an unseen device tries to connect to the infrastructure. The model captures a sequence of the network traces for the device in question. It then identifies this unseen device after creating Fingerprint, using a trained classifier, that is trained with white-listed devices. The decision of the classifier can be either an unknown device or a white-listed device with identified device name and type. If the decision is unknown, the network connection between the unknown device and the internal network will be rejected. Otherwise, the identified device name and type will be sent to the zoning stage. This stage takes authorization matrix for each device as a second input. The authorization matrix reflects the trust level of each device as per its known vulnerabilities and latest patches. The zoning stage then allows the device to connect to either the trusted network or to the restricted network as per the pre-defined authorization matrix. The trusted zone devices can communicate with all devices and services within the infrastructure. However, the restricted zone devices will communicate only with restricted devices in the same zone.

The second stage of the model, is during the normal devices communication within the network. A random moving window sequence of packets capture is taken from any device in the two zones. This captured traffic is then fingerprinted, and applied to a trained classifier, to compare the device behavior with a stored baseline. The decision of the classifier can be either device misclassified or misbehaving. Mis-classification can result from the device being wrongly identified as a certain device from Stage 1 classifier. While it is a different device from the same white-list. Accordingly, the new predicted device name and type will be sent to be re-zoned as per its related authorization matrix. Misbehaving of devices, happens if a device is compromised or wrongly identified as a white-listed member. This can be detected, if the fingerprint doesn't match any of the baselines in the database. Hence, the device network communication will be quarantined and monitored for further analysis. This stage can be integrated with security event and information management (SEIM) solutions to take further actions and alerts.

IV. Device Identification Methodology

The proposed device identification approach has two components: fingerprinting and classification. The fingerprint features selection as well as identification problem description are detailed here, leaving the selection of the best classifier to identify devices in Section V-B.

A. Fingerprinting

To create a fingerprint for each device, we extract unique behavioral and flow-based features from the header and payload of (Ethernet, IP and Transmission Control Protocol (TCP)/ User Datagram Protocol (UDP)) from a sequence of packets. The packets sequence is ordered by time stamp. Feature selection is usually used to reduce feature space by selecting features that includes informative unique information. This will increase accuracy of the classifier and reduce computational cost. We focus mainly on packets originating from the device in question. But some features are extracted from bi-directional traffic as well. The source Media Access Control (MAC) address is used to identify the source of packets. We calculate summary statistics (minimum, maximum, first quartile, median, third quartile, mean, variance, and inter-quartile) and Fast Fourier Transforms (FFT) for a number of the used features as suggested by [12], [13].

Capturing long samples from the network packets of the device in questions, usually seize more useful information. However, to achieve real-time performance, smaller length inputs need to be used, which can be insufficient to capture the unique information for each device. Accordingly, the accuracy of the classifier can be influenced. To tackle this conflict, we run a number of experiments to find the optimum trade-off between quality and real-time performance of the classifier as shown in Section V-C. As a result of these experiments, we choose to extract features from a sequence of 20-21 packets.

The fingerprint for each device includes selected behavioural and flow-based features. Behavioural features, usually capture information that reflect the behaviour of the device such as ports used and patterns [14]. Flow-based features include information flow, such as in IP destinations as well as statistical information, such as Inter-arrival-time (IAT) [14]. The fingerprint for each device network traces consists of a set of 67 features. It includes Ethernet packet size, IP packet, and header sizes and TCP payload size features. Some of these features are added as an influence by the authors of [15]. In addition, we include TCP payload data offset. Data offsets give data start information to the upper network layers. We add it due to its significant influence on the uniqueness of the fingerprint during our experiments. Moreover, Time-To-Live (TTL) for TCP and UDP packets is used as a feature. TTL is the number of hops allowed for the packet, which is used to limit the lifespan of packets in any network. TTL feature for TCP is hinted by the authors of [16]. Furthermore, we include IAT summary statistics and 10 highest magnitudes of FFT. IAT measures the delay between the following packets. It was previously used as a feature in [9], [17], [18]. Packet direction is added as an integer to represent the direction of each packet. A count of the number of IP destinations, source and destination ports, within the selected sequence communication, are added as features as well. In addition, we include packet rate, which is a count of transmitted packets within a specific time period (5 secs). Finally, we use the TCP window size feature. TCP window size is the number of packets that can be sent and acknowledged in a single acknowledgment by the receiver. It reflects the device memory as well as processing speed, and was suggested as a feature in [19].

B. Identification

We form the problem of device identification as a multi-class classification problem [19], because we are identifying different device types and different individual device instances. Given a set of authorized devices (white-list), the classification task is to identify the class of the device in question.

A number of machine learning techniques are examined to choose the best model for the selected features. We compare the classification performance using nine classifiers: Adaptive Boosting (ABOOST), Latent Dirichlet Allocation (LDA), K-Nearest Neighbors (KNN), Decision Tree (CART), Nave Bayesian (NB), Support-Vector Machines (SVM), RandomForest with 50 estimators (RFC50), RandomForest with 100 estimators (RFC100), and Gradient Boosting (GBOOST). The settings and results of these experiments are detailed in Section V-B.

V. EXPERIMENTS

In this section, we present our experiments on the identification methodology, which is a key component of our proposed security enforcement model design. We use real-world IoT device data. We describe the settings and dataset, analyze different classifiers, and present the results and our analysis.

A. Setup

We describe the experimental settings in this section, including dataset, evaluation metrics and implementation.

Dataset. The dataset used in this paper is the same dataset created and used by the authors of IoT Sentinel [5]. It consists of traffic traces from 31 off-the-shelf IoT device saved as packet capture (pcap) files. Among these, there are 27 different device types. A number of pcap files for each device is recorded during the initial communication between the device and the infrastructure. The devices used in this dataset has more than 4 types of connection technology. Such as WIFI, ZigBee, Ethernet and Z-Wave.

Metrics. To consider the effect of false positives and false negatives in the identification accuracy, we decide to use F1-Score [20] as the validation metric. It is the "weighted average of the precision and recall" [20].

$$F_1 = 2.\frac{1}{(\frac{1}{recall}) + (\frac{1}{precision}))} = 2.\frac{precision.recall}{precision + recall} \tag{1}$$

The general formula for f_α is shown in (2) [21]. α can be chosen by the deploying organization. The lower the α the stricter the network [16] as the classifier will value precision more than recall. Thus, fewer unauthorized IoT devices will be identified as a white-listed device. However, this also will increase the False Alarm Rate. Moreover, organizations can set a prediction decision threshold tr, to set the network strictness.

$$F_\alpha - Score = (1 + \alpha^2) * \frac{precision.recall}{(\alpha^2.precision) + recall} \tag{2}$$

Our goal is to classify network stream for every device trying to connect to the network, as originating from an authorized white-listed device or unknown device. Accordingly, we apply the classifier on the extracted features. The classifier produce a vector of posterior probabilities of classification P^s for the test input. We then examine this calculated posterior probabilities. If any of the posterior probabilities P_i^s for device d_i is greater than the selected decision threshold tr. Then the device is identified as authorized and the type is as predicted by the classifier. Otherwise, the device is identified as unknown. We set the decision thresholds $tr = 0.6 - 0.85$ through our experiments.

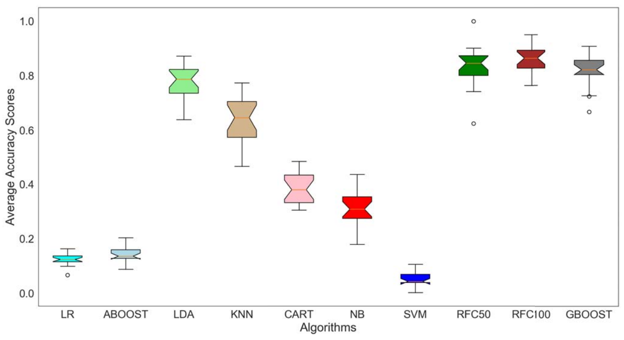

Figure 2. Accuracy Results For Different Classification Algorithms. RFC with 100 n_estimators achieve best identification score with average accuracy of 0.93 % followed by RFC with 50 n_estimators, then GBOOST with average of accuracy of 0.86%. SVM achieve the least identification score with average accuracy of 0.1%

Implementation and Parameters Tuning. In our experiments, we use Scapy [22], to decode network traces. These traces are then applied to the features extractor, to extract the unique selected features from a sequence of decoded packets. The set of selected features with its belonging labels feeds the classifier for training. Thus the trained classifier can distinguish between different White-listed IoT devices.

B. Classification Model Selection

To choose the best classification technique for our problem, we examined a number of machine learning classification techniques and compared their validated accuracy. we use Scikit-learn python library [20] to implement and train the tested models. 10 iterations and 10-fold cross validation are used to generalize the validation results.
Fig. 2 presents a comparison between all tested algorithms accuracy results: ABOOST, LDA, KNN, CART, NB, SVM, RFC50, RFC100, and GBOOST.

We use the default Scikit-learn Library parameters [20] in setting most of the tested algorithms. Table I presents the parameter settings for each of the tested algorithms.

Based on the findings, we choose Random Forest as the base classifier. Random Forest is an ensemble learning technique build on decision tree induction [23]. It has a number of advantages as per the information in the cybersecurity ML methods survey [24]. The advantages include:

- The classifier doesn't need feature selection.
- The algorithm is resistant to overfitting.
- Increasing the number of trees leads to a decrease in variance without adding bias.

After selecting the base classifier, we examined the two multi-class classifiers types, which are One Versus-Rest (One-Vs-Rest) classifier and One Versus One (One-Vs-One) classifier. On the one hand, One-Vs-Rest also called One-Vs-All classifier is a multi-class classifier that trains one classifier per class. It creates N classifiers for an N class problem. In our current solution, we have 27 different classes. Accordingly, a total of 27 classifiers will be trained. For class$_i$ it will assume (i)-labels as positive and the rest as negative, where N reflects total number of classes and class$_i$ represent the ith classifier. This method is efficient and inoperable, as it ensures that each class is served by only one classifier. On the other hand, One-Vs-One classifier trains a separate classifier for each different pair of classes. It creates $(N * (N - 1)/2)$ classifiers [20]. Accordingly, a large number of classifiers will be created. Thus, long training time, especially in solutions with a large number of classes, such as the presented scenario. The average accuracy of both techniques is very close. However, training One-Vs-Rest classifier is faster and more practical. Accordingly, we choose to use it as the solution classifier with Random Forest as the base classifier.

The One-Vs-Rest multi-class classifier is implemented. The base classifier is Random Forest with 100 estimators (decision trees). To train the classifier, the entire dataset is converted into features and labels. Then the features and labels are randomized and divided into training and cross-validation sets. The shuffling process is to ensure that neither the training set not the cross-validation set is biased by any particular device/device type. Finally, for each experiment, we log the training set and validation set accuracy, F-scores, and zero-one loss.

Table I
PARAMETER SETTINGS OF TESTED ALGORITHMS.

Algorithm	**Parameters**

ABOOST	n_estimators= 50
	learning_rate= 1
	base_estimator=Decision Tree (max_depth= 1)

LDA	n_components= 10
	learning_decay= 0.7
	learning_offset= 10

KNN	n$_neighbors = 5$
	leaf_size= 30
	power_parameter(p)= 2

CART	max_depth= 1
	min_samples_split= 2
	min_samples_leaf= 1

NB	var_smoothing= 1e-9

SVM	degree= 3
	decision_function_shape= ovo

RFC50	n_estimators= 50
	min_samples_split= 2
	min_samples_leaf= 2

RFC100	n_estimators= 100
	min_samples_split= 2
	min_samples_leaf= 2

GBOOST	n_estimators= 100
	min_samples_split= 2
	min_samples_leaf= 2

C. Experiments and Results

Using the described model, the network is trained and then evaluated to distinguish between known vendor models, individual devices that have the same type, vendor and model, and unknown devices. Moreover, we investigated the feature importance of each of the selected feature, on the accuracy, weighted-average precision, recall and F1-score of the classifier.

1) Device Type Identification: The extracted features and labels for all the pcap files for all the 27 different device type are randomized. The feature set and belonging labels are divided into training and validation sets. We use, train_test_split module from Scikit-learn library to divide the fingerprints dataset, into 80% for training and 20% for validation. The network is trained with the training set. The validation set is then applied to the trained network. The average recorded weighted F-score for identifying device types is 91% and loss value of 0.09999. The F1-score for each of the device types is shown in Fig. 3.

Device types with low identification rates such as D-LinkSensor (model: D-Link WiFi Motion sensor DCH-S150), D-LinkSiren (model: D-Link Siren DCH-S220) and D-LinkWaterSensor (model: D-link Water sensor DCH-S160), are almost device types that share the same vendor. Thus, the vendor uses mostly, similar technology and configurations, making it difficult to identify these types perfectly. Other, device types identification rates are almost perfect. For instance, Aria (model: Fitbit Aria WiFi-enabled scale) is identified perfectly with identification weighted F1-score of 100%.

2) Unknown Device Identification: We conduct 27 different experiments, as per the number of IoT device types in this dataset. In each experiment, the network is trained with 26 IoT device. Leaving one IoT device as the unauthorized device type. The threshold tr is set as discussed in section V-A to optimized the F-Scores. Features are extracted from all 26 white-listed devices, then accompanied by it's label to create the white-list dataset. this dataset. The classifier is then trained with this created dataset. Then, features are extracted from each of the pcap files for the device in question, which is the device in turn that is not in the white-list. These extracted test device features are applied to the trained classifier. Then the unknown device Identification rate for each of the devices is calculated as per each experiment. Finally, the overall average accuracy in identifying unknown devices is calculated. Table II summarizes the 27 different experiment described above with decision threshold set equals 0.6.

3) Individual Device Identification: The dataset used in this experiment, consists of all the 31 different devices. Among these devices, there are 4 pairs of individual similar device types, which are:

- EdnetCam1 & EdnetCam2

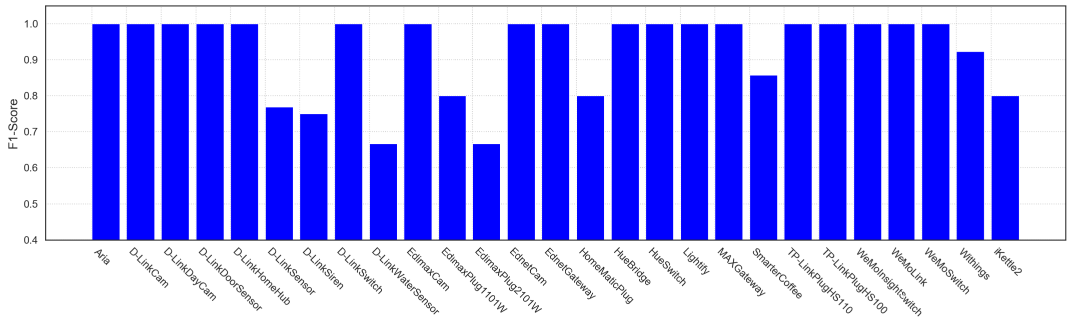

Figure 3. Performance of Device Types Identification.

- EdimaxCam1 & EdimaxCam2
- WeMoInsightSwitch & WeMoInsightSwitch2
- WeMoSwitch & WeMoSwitch2

Each pair shares the same device type, model and vendor. Features are extracted from all the pcap files of all devices. The feature vector for each pcap is then accompanied with it's related label. This creates the overall dataset. This dataset is split into 80% for training and 20% for validation. The trained network is then tested with the validation set to test the accuracy of the classifier. In summary, the F1-score identification rate is between 55%-100% . Fig. 4 shows the comparison of the classification report for all tested devices.

Table II
UNKNOWN DEVICES EXPERIMENTS: #. SAMPLES REPRESENT NUMBER OF TESTED PCAP FILES, IR REPRESENTS UNKNOWN IDENTIFICATION RATE, C. TECHNOLOGY REPRESENTS DEVICE CONNECTION TECHNOLOGY.

Device Name	#. Samples	IR	C. Technology
Aria	20	1	WIFI
HomeMagicPlug	20	1	Proprietary Technology
Withings	19	1	WIFI
MAXGateway	20	1	Ethernet
HueBridge	20	1	ZigBee & Ethernet
HueSwitch	20	1	ZigBee
EdnetGateway	20	1	WIFI
EdnetCam	20	1	WIFI & Ethernet
EdimaxCam	20	0.85	WIFI & Ethernet
Lightify	20	1	WIFI & ZigBee
WeMoInsightSwich	25	1	WIFI
WeMoLink	20	1	WIFI & ZigBee
WeMoSwitch	25	1	WIFI
D-LinkHomeHub	20	1	WIFI & Ethernet & Z-Wave
D-LinkDoorSensor	25	1	Z-Wave
D-LinkDayCam	20	1	WIFI & Ethernet
D-LinkCam	20	1	WIFI
D-LinkSwitch	20	1	WIFI
D-LinkWaterSensor	20	0.5	WIFI
D-LinkSiren	20	0.6	WIFI
D-LinkSensor	20	0.55	WIFI
TP-LinkPlugHS110	20	0.35	WIFI
TP-LinkPlugHS100	20	0.3	WIFI
EdimaxPlug1101W	20	0.1	WIFI
EdimaxPlug2101W	20	0.4	WIFI
SmarterCoffee	20	0.3	WIFI
iKettle2	20	0.1	WIFI

Devices with a similar type, model and vendor are correctly identified in more than 50% of the cases. The low identification of some of these devices is due to the availability of a limited number of pcap files for each of these individual instances. For instance, EdimaxCam2 has only five pcap files available in the dataset. These limited network traces files are split into training and validation datasets. Accordingly, the classifier is not well trained with this device instance.

4) Feature Importance: Besides assessing the overall accuracy level, we inspected the features and their importance. Fig. 5 illustrates each of the 67 features and its importance score. Importance is the total average decrease in node

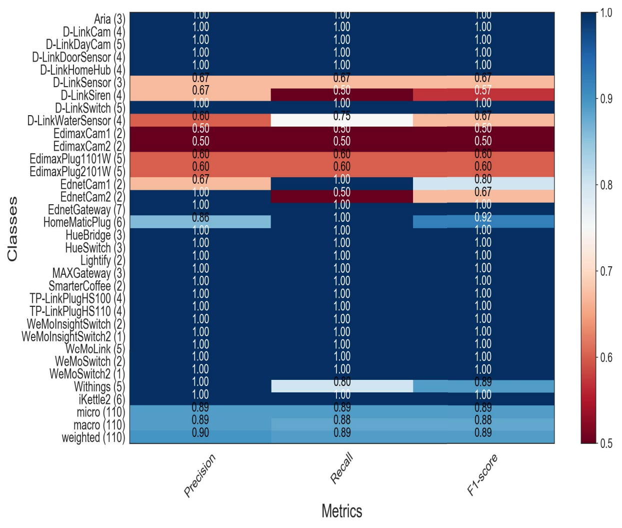

Figure 4. Classification Report For Individual Device Instances.

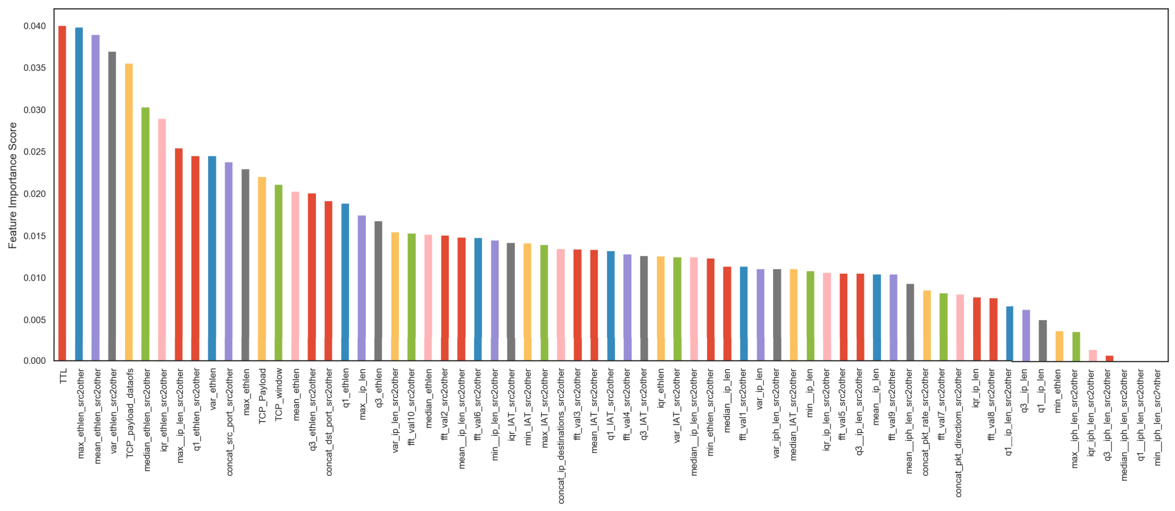

Figure 5. Features Importance Scores.

impurity over all trees of the classifier, that is weighted by the probability of reaching that node [25].

The most important feature is the average TTL. The second, third and fourth important features are all related to summary statistics of Ethernet packet size, which are maximum, mean and variance of Ethernet packet size respectively. Followed by TCP payload data offset feature. Moreover, the three least important features are all related to IP header size, which are median, q1 and minimum IP header size. The values of these three features are almost zero. However, we decide to keep these features, for generalization as they might have a minor effect when using other datasets.

5) Comparison with Existing Approaches: The fact that we are using the same dataset as IoT Sentinel, directed us to compare our results with this state-of-the-art solution. We use F-score for validation, while the authors of IoT sentinel used accuracy metric. Accordingly, we compare here the accuracy metric results we achieved, to be able to compare the two solutions. On the one hand, the authors of IoT sentinel extracted 23 feature from each packet header in pcap file to create a fingerprint for each device. They achieved an identification accuracy of 81%. On the other hand, we use 67 feature from only a sequence of 20-21 packets from each pcap file. Features included both header and payload information, achieving an average identification accuracy of 90.3%. Finally, the proposed solution identified individual devices sharing the same vendor and model. The achieved identification accuracy is 0.89%. However, the IoT Sentinel proposed solution didn't focus on individual device identification and didn't include experiments or results to identify individual devices.

VI. Conclusion

This paper presents a security enforcement model design, that identifies and authorizes IoT devices using behaviour flow-based fingerprints and machine learning. The described model demonstrates a novel IoT passive device fingerprinting technique. The unique fingerprint is created from features selected from the network packets traces, using information in both the packet's headers and payloads. The machine learning network is trained with fingerprints of network white-listed devices. We conduct a number of experiments, to demonstrate the proposed IoT identification approach and further showcase that effectiveness of our proposed security model design. Fingerprints that are created, feed an automated device identification, that uses supervised multi-class One-Vs-Rest machine learning network. The trained One-Vs-rest Random Forest network can distinguish between device types as well as individual device instances from same vendor and model. The identification F1-scores achieved respectively are 91% and 89%. Moreover, the created fingerprints are stored as baselines for each device, to detect rogue or compromised devices. Privileges on an IP-network is assigned to each identified device as per pre-defined rules. Behavioural fingerprint of random capture of each device network communication is created and then compared to a baseline. If a device is detected with behavioural changes, it will be quarantined for further monitoring.

References

[1] Gartner, "Gratner Newsroom," *https://gartner.com/en/newsroom/id/283971*, 2015.

[2] T. Shah and S. Venkatesan, “Authentication of IoT Device and IoT Server Using Secure Vaults,” *IEEE International Conference On Trust, Security And Privacy In Computing And Communications - TrustCom*, pp. 819–824, 2018.

[3] R. Lanotte, M. Merro, R. Muradore, and L. Vigano, “A Formal Approach to Cyber-Physical Attacks,” *IEEE Computer Security Foundations Symposium - CSF*, pp. 436–450, 2017.

[4] Senrio, “Publicly Available IoT Devices Vulnerable to Single Flaw,” *https://blog.senr.io/blog/400000-publicly-available-iot-devices-vulnerable-to-single-flaw*, 2016.

[5] M. Miettinen, S. Marchal, I. Hafeez, T. Frassetto, N. Asokan, A. R. Sadeghi, and S. Tarkoma, “IoT Sentinel Demo: Automated Device-Type Identification for Security Enforcement in IoT,” *IEEE International Conference on Distributed Computing Systems- ICDCS*, pp. 2511–2514, 2017.

[6] T. Kohno, A. Broido, and K. C. Claffy, “Remote Physical Device Fingerprinting,” *IEEE Transactions on Dependable and Secure Computing - TDSC*, vol. 2, no. 2, pp. 93–108, 2005.

[7] J. François, H. Abdelnur, R. State, and O. Festor, “Automated Behavioral Fingerprinting,” *Recent Advances in Intrusion Detection (RAID), Lecture Notes in Computer Science - LNCS*, vol. 5758, pp. 182–201, 2009.

[8] H. Abdelnur, O. Festor, J. François, and R. State, “Machine Learning Techniques for Passive Network Inventory,” *IEEE Transactions on Network and Service Management*, vol. 7, no. 4, pp. 244–257, 2010.

[9] S. V. Radhakrishnan, A. S. Uluagac, and R. Beyah, “GTID: A Technique for Physical Device and Device Type Fingerprinting,” *IEEE Transactions on Dependable and Secure Computing - TDSC*, vol. 12, no. 5, pp. 519–532, 2015.

[10] S. Siby, R. R. Maiti, and N. Tippenhauer, “IoTScanner: Detecting and Classifying Privacy Threats in IoT Neighborhoods,” *arXiv:1701.05007v1*, 2017.

[11] J. Franklin, D. Mccoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker, “Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting,” *USENIX Security Symposium - USENIX-SS*, p. Article No. 12, 2006.

[12] T. T. Nguyen and G. Armitage, “A Survey of Techniques For Internet Traffic Classification Using Machine Learning,” *IEEE Communications Surveys and Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[13] N. Aluthge, “IoT Device Fingerprinting With Sequence-Based Features,” *Thesis, Helsinki University*, 2017.

[14] N. Moustafa, B. Turnbull, and K. K. R. Choo, “An Ensemble Intrusion Detection Technique based on proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things,” *IEEE Internet of Things Journal - IOT-J*, vol. PP, no. c, p. 1, 2018.

[15] A. Moore, D. Zuev, M. Crogan, A. W. Moore, and M. L. Crogan, “Discriminators For Use in Flow-Based Classification Discriminators for use in flow-based classification,” tech. rep., Queen Mary University of London, 2005.

[16] Y. Meidan, M. Bohadana, A. Shabtai, M. Ochoa, N. O. Tippenhauer, J. D. Guarnizo, and Y. Elovici, “Detection of Unauthorized IoT Devices Using Machine Learning Techniques,” *arXiv:1709.04647*, 2017.

[17] C. Rottondi and G. Verticale, “Using Packet Interarrival Times For Internet Traffic Classification,” *IEEE Latin-American Conference on Communications - LATINCOM*, pp. 1–6, 2011.

[18] W. Linlin, L. Peng, M. Su, B. Yang, and X. Zhou, “On the Impact of Packet Inter Arrival Time for Early Stage Traffic Identification,” *IEEE International Conference on Internet of Things - iThings*, pp. 510–515, 2017.

[19] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, “IoTSense: Behavioral Fingerprinting of IoT Devices,” *arXiv:1804.03852v1*, 2018.

[20] Scikit Learn, “scikit-Learn 0.20.2 Documentation,” *https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html*, 2015.

[21] Van Rijsbergen, *Information Retrieval*. Butterworth-Heinemann, 2nd ed., 1979.

[22] Philippe Biondi and the Scapy community., “Scapy,” *https://scapy.net/*, 2018.

[23] E. Scornet, “Random Forests and Kernel Methods,” *IEEE Transactions on Information Theory*, vol. 62, no. 3, pp. 1485–1500, 2016.

[24] A. L. Buczak and E. Guven, “A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection,” *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[25] Ceshine Lee, “Feature Importance Measures for Tree Models,” *The Artificial Impostor, Part I*, 2018.