

IoT or NoT: Identifying IoT Devices in a Short Time Scale

Anat Bremler-Barr, Haim Levy, Zohar Yakhini
Computer Science Department, Interdisciplinary Center, Herzliya, Israel

Abstract

In recent years the number of IoT devices in home networks has increased dramatically. Whenever a new device connects to the network, it must be quickly managed and secured using the relevant security mechanism or QoS policy. Thus a key challenge is to distinguish between IoT and NoT devices in a matter of minutes. Unfortunately, there is no clear indication of whether a device in a network is an IoT. In this paper, we propose different classifiers that identify a device as IoT or non-IoT, in a short time scale, and with high accuracy.

Our classifiers were constructed using machine learning techniques on a seen (training) dataset and were tested on an unseen (test) dataset. They successfully classified devices that were not in the seen dataset with accuracy above 95%. The first classifier is a logistic regression classifier based on traffic features. The second classifier is based on features we retrieve from DHCP packets. Finally, we present a unified classifier that leverages the advantages of the other two classifiers.

I. INTRODUCTION

The number of IoT devices in home network has increased dramatically in recent years. IoT devices are much more vulnerable to attacks than general-purpose endpoint computers and will be insecure in the foreseeable future. In most cases, the IoT device is strong enough to host an attacking zombie but too weak to protect itself from malicious code. Thus, it clearly poses new security challenges. Attacks on IoT have severe implications in both the cyber and physical domains [1], [2], [3], [4]. There are many proposed security and management solutions, with the common practice being a network-based solution that is geared to protecting IoT devices and resides at the home router [5], [6] or in an additional security device designed to protect the home network [7], [8], [9], [10]. The security solution cannot reside in the IoT device itself, due to the low CPU power and memory of IoT.

Whenever a new device connects to the network, it must be managed and secured as quickly as possible using the relevant security policy. Incorrect device classification may lead to incorrect security policies applied to the device.

A key challenge is thus to quickly distinguish between IoT (smart camera, bulbs, speakers and so on) and non-IoT devices (general purpose computers, mobile phones, desktops, tablets and laptops and so on), referred to in our paper as NoT

devices. We assume that the classification is done in a device that observes the LAN traffic, wired and wireless traffic. Our paper focuses on IoT that are actual physical entities connected to the internet. Classification of borderline devices such as smartTVs (that can be used to surf the Internet and to run different applications) is part of our future work.

In this paper, we propose three classifiers for identifying devices as IoT or NoT (see a summary of results in Table I). We use a machine learning methodology, where we trained classifiers on the *seen* dataset of labeled (IoT or NoT) devices, and then analyze the accuracy of our classifiers on an *unseen* dataset of devices. The unseen dataset includes IoT device types that *were not in the seen dataset*. This challenging requirement is due to the huge variety of IoT devices, types and vendors. Moreover, the IoT market is very dynamic, with new vendors and new devices appearing constantly. We therefore seek a general classifier: one that identifies the inherent characteristics of IoT vs NoT devices.

The desired classification approach should have the following properties:

- Universality - the classifier should be generic and work for all IoT device types, including those with encrypted traffic. It should also work on device types it has not yet seen.
- Low classification latency - We consider 1 to 20 minute latency.
- Accuracy - the classifier should be accurate. We use F1-score, recall and precision as measures for accuracy.
- Efficient - the classifier should also be efficient, with low CPU processing and memory requirements, since it needs to process on-line traffic.
- Passiveness - the classifier should passively process the traffic and may not use active probing of devices. Active techniques are usually tailored for specific IoT devices and/or require special permission from the owner. Moreover, in some cases, active probing might unintentionally activate the devices.

The first classifier, presented in Section IV, is a logistic regression classifier using **traffic features**. The resulting classifier is efficient and requires light processing on a few features of the traffic (up to 5). We found the most informative features to be the TCP window size and the number of unique DNS queries. We show that these features were chosen by our ML algorithm because of the very limited number of endpoints

Classifier	Classification Latency	F1-Score	Recall	Precision	Efficiency
Classifier I: Traffic Features	1 min	91.54%	97.38%	87.02%	Counters
	5 min	96.59%	98.72%	94.54%	Counters
	10 min	98.12%	98.91%	97.34%	Counters
	20 min	98.07%	98.71%	97.43%	Counters
Classifier II: DHCP	Long	95.73%	93.75%	97.82%	DPI required
Classifier III: Unified	20 min	99.04%	100%	98.11%	DPI required

TABLE I
EXPERIMENTAL RESULTS OF THE PRESENTED CLASSIFIERS OVER THE UNSEEN DATASET

with which IoT devices communicate, as well as their small TCP buffer size.

The second classifier, presented in Section V, is based on a decision tree on **DHCP** information. The DHCP protocol is very common in home networks, and the resulting decision tree is simple (with a height smaller than 5). The downside of the algorithm is that not all the networks or devices use DHCP. Moreover, it might have a long classification latency, since DHCP appears relatively seldom in the traffic.

We then present in Section VI a **unified classifier** that leverages the advantages of the other two above classifiers and achieves F-score, precision and recall above 98. The unified classifier can be found in our github repository [11].

The closest work to ours is the recent published work, DeviceMien [12], where the authors also note the blind spot in the literature in categorizing as IoT previously unseen IoT devices. However we achieve better accuracy, and we also provide a clear intuition behind the features and signatures selected by our classifiers (see Section II). Thus, our work sheds light on the unique network characteristics of IoT devices.

II. RELATED WORK

A technique that uses user-agent field information was suggested in [13] to classify IoT devices. A user-agent value is sent during HTTP requests, and it contains a short description of the properties of the requesting device. For NoT devices, this parameter is usually of greater length, since it describes properties relevant only for NoT devices, such as screen size, OS language and browser. However, we discovered that the technique does not meet our requirements. The user-agent parameter cannot be observed in encrypted traffic. In our dataset, only 69.5% of devices transmit this parameter as a plain text, and a similar result was shown in [14]. Moreover classification latency is high - the chance to find an user-agent value sent by an IoT device in a slot of 20 minutes is about 25%. We note that unlike the DHCP client information, the user-agent is information that was sent to the endpoints, and hence it is not stored at the home-router and thus cannot be actively retrieved.

MAC OUI can be helpful in identifying the manufacturer of a device. But is of very limited use as a unique identifier of IoT due to manufacturers that supply both IoT and non-IoT device types (such as Samsung's smartcam and smartphones). The authors of [15] tried to associate MAC ranges of manufacturers

with models, but their technique was often ineffective due to lack of regularization in this field.

Other works address related areas, such as *device* fingerprinting and identification using classes [16], [17], [13], [18], [19], [20], but IoT devices that were not seen before cannot be identified by these techniques.

The authors of [21] propose a proactive request to IoT devices in order to classify the exact IoT vendor/model device. However, this work does not meet the passiveness requirement, since sending packets to IoT requires non-trivial permissions.

OS fingerprinting is addressed in [22], [23], [14]; however, the device type cannot be easily identified from the resulting OS information. The Satori project [22], [23] inspired our use of some of the features we tested in this work, mostly data from the IP-TCP layers.

The closest work to ours is DeviceMien [12], where the authors also note the blind spot in the literature in categorizing as IoT previously unseen IoT devices. Their approach, which uses auto-encoder, a deep-learning technique, cannot provide any intuition regarding the received classifier, in contrast to our approach. We also achieve better accuracy: F1-score of above 95% on ours dataset, as opposed to 76% by the authors of [12] on theirs. Since they did not publish the dataset or their resulting classifier, we cannot compare our techniques on the same dataset. However, they did provide a list of the devices in the dataset, which are very similar to our. We note that they also checked a few borderline IoT devices, such as SmartTVs. We predict, based on the list of devices, that we would achieved an F1-score of 92% on their dataset, assuming misclassification in the borderline IoT devices. We suspect that our superior results due to the ability of machine learning techniques such as the one we used to achieve good results on small datasets. Deep learning techniques, on the other hand, require huge datasets, which are difficult to obtain due to the need to label the devices. Another advantage of our classifiers is that their latency is time-bound. Finally, our implementation is more efficient since they require only a few memory references.

We note that a recent initiative calls for IoT device vendors to provide a *Manufacturer Usage Description (MUD)* for their IoT products [24], which as a by-product identifies the device as an IoT. Only a very few IoT currently provide MUD files. It is moreover questionable whether the majority of the vendors would comply with MUD, since the vendor apathy is one of

the root causes of the IoT security problems.

III. METHODOLOGY

Our dataset is composed of captured network traffic data (pcap files), recorded at the router or at an access point, of labeled devices from various sources: [16], [25], [26], [27], [28] and pcaps collected from our IoT lab. The IoT devices in our dataset are unique by type, model and/or OS version. Overall we had about 46GB of data. Recording time varied greatly, with some devices that were recorded for weeks and some for hours. Overall, our dataset contained 121 devices: 77 IoT and 44 NoT devices.

At first, we arbitrarily split the dataset into two groups: *seen* and *unseen*. Later on, we gained access to more devices, and we added them to the unseen dataset. Our seen group contained 45 devices, 24 IoT and 21 NoT (see Table V in Appendix). Our unseen group contained 76 devices, 53 IoT devices and 23 NoT (see Table VI in Appendix).

As the names indicate, we performed the learning on the seen group and tested our classifiers on the unseen group. We want to emphasize that the samples of the unseen group were not available to us during the learning phase.

In order to test classification performance in various time periods, we divided our dataset into time slots of 1, 5, 10 and 20 minutes. Working with a model of slots, our classifiers process information of a time slot (in the training phase and testing phase). Slots with a small number of packets were also considered, since we observed that they might contain sufficient information for classification. Some devices were characterized by their very rare network usage, such as the Nest Smoke Alarm, which sends only a few packets every 23 hours. Thus, our classifier can classify a device as soon as it sends data.

Our goal is to classify a new device in a network as being IoT or NoT. This goal of dichotomy classification is a choice we made. We could also have used a scoring mechanism, estimating the probability of being in one of the classes, or classification to three categories: IoT or NoT or Undecidable. The dichotomy classification fits well with our need to always decide how to protect and manage the device, as a general purpose computer or as an IoT device.

In our model, an IoT device is considered 'Positive', and a NoT device is considered 'Negative'. We thus defined the following performance metrics: True Positive (TP) - correct classification of an IoT device; False Positive (FP) - misclassification of a non-IoT (NoT) device as IoT; True Negative (TN) - correct classification of a non-IoT (NoT) device; False Negative (FN) - misclassification of an IoT device as non-IoT (NoT).

We measure the accuracy using recall, precision and F1-Score: **Recall** is the probability of an actual IoT to be successfully classified as such, i.e., $\frac{TP}{TP+FN}$. **Precision** is the probability that an IoT-classified device is truly an IoT, i.e., $\frac{TP}{TP+FP}$. **F1-score** is a unified performance index defined as $2 \cdot \frac{\text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}}$.

IV. CLASSIFIER ON TRAFFIC FEATURES

In this section, we propose a logistic regression classifier that operates on traffic features. We start by explaining the two-step learning phase (see Section IV-A), which consists of a feature selection followed by constructing an optimized feature set. We then explain the intuition behind the selected features in Section IV-B. In Section IV-C we present the testing phase results that demonstrate its accuracy on the unseen data set. We then discuss some implementation considerations (in Section IV-D).

A. Learning phase

1) *Feature Selection*: We tested 22 features from standard protocols (Link-Layer, IP, TCP, DNS, HTTP) (see Table II). We tested every feature that we thought might be an indicator. We then *automatically* selected from among them a small set of ten informative features that achieved an F1-score above 0.5 for the seen dataset (see Table III). The feature selection was done as follows: the seen data traffic was first divided into sets of IoT or NoT. For each feature, we analyzed those sets (IoT vs. NoT) using statistical tools (Welch's t-test, ROC curve and AUC calculation [29], [30]) to determine the separation potential for each feature. We narrowed our feature set to the best performing features. Then, we normalized the seen traffic, such that each device had the same number of samples. This helped us deal with the big differences in recording time, where some devices were recorded for weeks and some for hours. In order to represent a wide range of scenarios, we chose representative samples according to bandwidth (low, medium and high).

We calculated the F1-score using 5-fold cross-validation, a common technique in machine learning, to choose features with no over-fitting. Our 5-fold cross-validation method randomly splits the devices in the dataset into 5 independent sets of train (80%) and test (20%). Note that no device appears in more than one test set. Every test was run 5 times, once for each fold, and the results were averaged. We ran this test separately for every classification latency.

For feature information that appears only from time to time in the traffic (e.g., TCP timestamp or user-agent), we filled slots with missing values with the average values of the feature, as learned from the seen dataset. Therefore, a low F1-score may also indicate that this feature does not appear in most slots.

2) *Constructing an Optimized Feature Set*: Our classifier uses sklearn's StandardScaler function [31] in order to standardize feature values and relies on the logistic regression algorithm [32] when applying classification. In order to test incoming traffic against the classifier, we again imputed any missing value from an average we learned for each feature. Later on, we used logistic regression on the standardized values in order to get a classification result.

In constructing a machine learning model, we chose a combination of features that optimizes prediction rates, for each classification latency separately. We applied our learning using 5-fold cross-validation, as proposed in Section IV-A1

Layer	Feature description
Link-Layer	Number of outgoing packets
Link-Layer	Bandwidth (in bytes) of outgoing traffic
Link-Layer	Average (in bytes) of packets length
Link-Layer	Average of interleaving time for outgoing packets
Link-Layer	Standard deviation of interleaving time for outgoing packets
IP	Number of unique interacted endpoints of remote IPs
IP	Average of the TTL value in outgoing IP packets
IP	Average of the header length value in outgoing IP packets
IP	Maximum of the header length value in outgoing IP packets
IP	Minimum of the header length value in outgoing IP packets
IP	Count of unique header length values in outgoing IP packets
IP	Number of unique outgoing ports
IP	Ratio between the number of TCP to UDP packets
IP	Number of unique interacted endpoints of remote End-Points ($IP \times Ports$)
TCP	Maximum TCP window size
TCP	Mean TCP window size
TCP	Minimum TCP window size
TCP	Count of unique TCP window size values
TCP	Linear-least-square error for TCP timestamp value
DNS	Number of unique DNS queries
DNS	Number of DNS queries
HTTP	Average length of user-agent field in http requests

TABLE II
LIST OF RAW FEATURES TESTED.

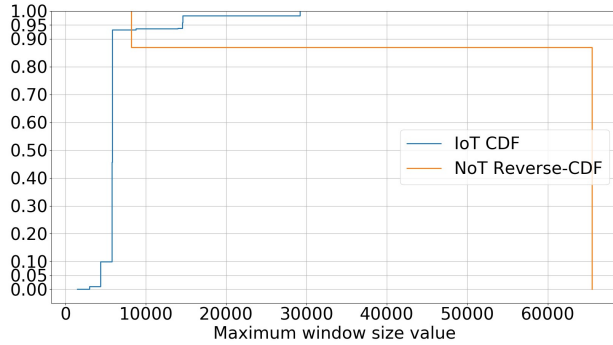


Fig. 1. The values of maximum TCP window size in IoT devices (presented as CDF) and NoT devices (presented as Reverse-CDF), seen dataset, 10-minute time slot.

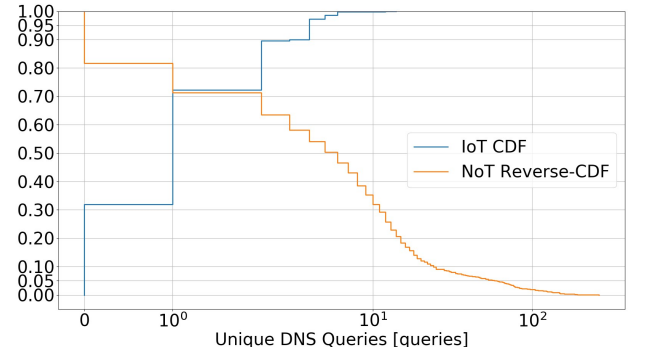


Fig. 2. The number of unique DNS queries in IoT devices (presented as CDF) and NoT devices (presented as Reverse-CDF), seen dataset, 10-minute time slot.

To learn the optimal number of features and the optimal features themselves, we used a greedy algorithm. Our goal was not only to optimize the F1-score for each classification latency but also the number of selected features. We used a parameter α (set to 1%) as a threshold in order to prefer a smaller vector size over a larger one with a tiny performance

gain (less than α). Table IV shows the optimal combinations and their averaged F1-score over the 5-fold cross-validation for each classification latency (on the seen dataset).

B. Intuition

We then tried to understand the reason behind the dominant selected features: window size and number of unique DNS

Feature name	Feature description	F1-score
window size	maximum TCP window size	0.942
# unique DNS reqs	number of unique DNS queries	0.845
# remote IPs	number of unique interacted endpoints of remote IPs	0.829
# dns reqs	number of DNS queries	0.738
# ports	number of unique outgoing ports	0.658
bandwidth	bandwidth (in bytes) of outgoing traffic	0.601
pckt count	number of outgoing packets	0.588
tcp ts deviation	linear-least-square error for TCP timestamp value	0.582
interleaving deviations	standard deviation of interleaving time for outgoing packets	0.576
tcp/udp ratio	ratio between the number of TCP to UDP packets	0.548

TABLE III
LIST OF FEATURES WITH F1-SCORE ABOVE 0.5 (SEEN DATASET, TIME SLOT OF 10 MINUTES).

Classification Latency	F1-Score	Feature set
1 min	89.52%	window size, # unique DNS reqs, tcp/udp ratio, pckt count
5 min	95.41%	window size, # unique DNS reqs, # remote IPs
10 min	96.33%	window size, # unique DNS reqs, # remote IPs, interleaving deviations
20 min	96.48%	window size, # unique DNS reqs, # remote IPs

TABLE IV
BEST FEATURE SETS FOR EACH CLASSIFICATION LATENCY WITH THE F1-SCORE OVER CROSS-VALIDATION DATA (SEEN DATASET)

requests. We noticed that IoT device hardware is not as well equipped as NoT devices hardware, and have small buffer size for TCP stack and therefore commonly has a smaller TCP window size [33]. Figure 1 compares the CDF of the IoT window size values to the reverse-CDF of the NoT window size values. This comparison shows the separability over this feature. This feature is highly available, visible and unencrypted. All of our devices had TCP traffic in their time slots.

In addition, IoT devices connects to limited endpoints (mostly vendor cloud servers), and thus have fewer unique DNS requests, remote IPs and ports (a similar observation was made in [16]). Figure 2 compares the CDF of the number of unique DNS queries of IoT devices to the reverse-CDF for NoT devices. Note that if there is no DNS traffic, this is also data, and the value of the feature for that slot will be zero.

The classifiers that work on time-slots from 5 minutes and above used the number of unique remote IPs in addition to the number of the unique DNS requests. We suspect that these numbers differ since we capture the traffic in some slots, i.e., in the middle of the device operation, we might not capture the DNS queries that resolve the IPs. Thus the number of unique remote IPs adds information.

C. Testing phase

After training our models, we validated the classifiers against the *unseen* dataset (see Table I). Again, we considered all the time-slots and average the results per device. We received a good F1-score (91.54%) for 1 minute time slot and very high F1-scores (above 96.5%) from 5 minutes time slot and above.

In Figure 3 we present the CDF of the classification success rate, defined as the fraction of time-slots, with correct classification of a device. For a given class classification success rate x , the graph shows the fraction of devices with a successful classification rate smaller than or equal to x . Except for one device, all the inconsistently classified devices were classified in more than 82% of the time slots in the same correct way. The most inconsistent devices were NoT devices, such as Apple iPad, Samsung Android, Win 10 and Win 7. Devices are incorrectly classified when the window size is not informative enough, and the NoT device is not very active in that time slot. To improve the results for these cases, we present in the next section, a classifier that works on the DHCP information. This result also motivated us to test a classifier with longer classification latency that uses the majority in a sequence of time slots. This observation is applied in our unified classifier (see Section VI).

D. Implementation Considerations

Performance wise, implementing the classifier on traffic features requires finding the number of distinct elements efficiently (e.g., of the number of unique DNS queries and the number of remote IPs). Note, however, that if the number of distinct elements, denoted by x , is small, the naive solution would be to store the last $x + 1$ unique elements seen. In our retrieved classifier the threshold of the number of unique elements was small (less than 15), and hence this is a practical solution. Another possible implementation is to apply algorithms that approximate the number of distinct elements, as was done in [34], [35], [36].

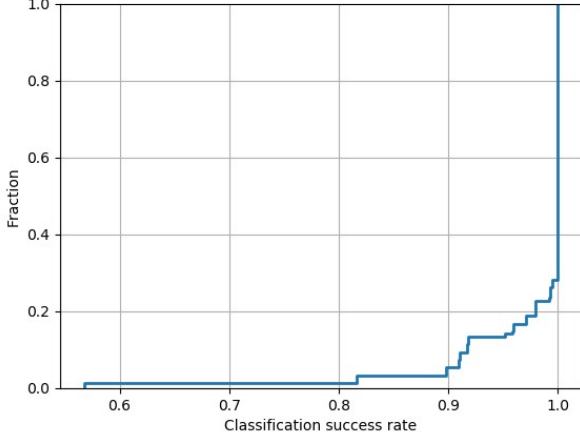


Fig. 3. CDF of classification success rate of devices, classifier on traffic features, 10 minute time-slot, unseen dataset.

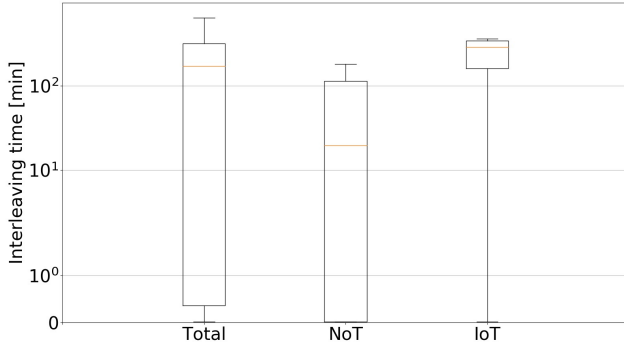


Fig. 4. Interleaving time distribution of DHCP packets on the seen dataset

V. DHCP BASED CLASSIFIER

In this section, we present a decision tree based on DHCP information. For devices that are configured to use the DHCP, IoT and NoT devices use the protocol to notify the router of their existence in the network with some information about the device.

The DHCP classifier has some inherent drawbacks: First, DHCP packets are not available in certain network configurations such as static-IP and in some IPv6 networks due to SLAAC (stateless address auto-configuration [37]). Second, classification requires the use of DPI, which is costly in terms of CPU. Third, DHCP packets are sparsely available, since DHCP traffic occurs when a device connects to the network or renews its IP. Figure 4 represents the interleaving time distribution. The median between consecutive DHCP packets is about 3 hours in our dataset.

Nonetheless, on DHCP enabled networks, this can be overcome and DHCP packets can be retrieved instantly, if active requests can be taken. We can actively disconnect all devices in the network (for example, by running the aireplay-

ng tool[38]). This action causes every device to renegotiate and triggers DHCP traffic, and hence we can retrieve the DHCP information in less than three seconds. With the appropriate credentials, it might be possible to use tr-69 [39] (the common protocol that is used by ISPs to manage and operate the home-routers) to retrieve the required values from the router.¹

In the next sections, we explain the learning phase of the decision tree on DHCP information, the intuition behind the retrieved tree, the testing results on the unseen data, and some implementation considerations.

A. Learning Phase

In order to construct the classifier automatically, we collected all possible information from DHCP packets, obtained from five fields: hostname, vendor-class ID (vci), parameter-request-list (prl), maximum-dhcp-size and message-types. We created a list of labels (words/values/numeric values) using the following algorithm: For the hostname and vci fields, which contain concatenations of words, we extract labels by splitting those values into labels separated by delimiters (such as `./_-+`), while filtering numbers. For the parameter-request-list, which is a list of identifiers, we add the identifiers as labels. For maximum-dhcp-size and message-type, which have numerical values, we add the numerical values as labels. We then construct a binary vector according to those labels for each device. Every i -th bit in a vector represents the fact that the i -th label exists.

We trained a decision tree model [40] using vectors we constructed according to the seen dataset. We obtained a simple decision tree (see Figure 5). Fortunately, due to the plurality of devices in the seen dataset, this tree was generic and did not receive any specific vendor IoT information.

B. Intuition

The chosen tree uses parameter-request-list (prl) information, which is a list of parameters the device requests from the DHCP server. We saw two dominant values: the first is the *hostname* value (tag number 12), which indicates that the device wants its hostname to be assigned by the DHCP server. In our dataset, only IoT devices request it. The second is the *domain name* value (15), which is used primarily to support easy access to other LAN entities using domain names instead of IP addresses. This value is mainly relevant for NoT devices.

The decision tree also uses vendor-class ID (vci) information. The vci field mostly contains an information about the type of DHCP client of the device. Some of our IoT devices use vci values of SOC (system on chip). We observed a number of other prominent values with high potential, such as 'udhcp' - a lightweight dhcp client (common for IoT), as opposed to 'dhcpcd' - a featureful dhcp client. However, the prl values were more prominent and were chosen by the algorithm.

We note that the *hostname* field is the device's name and it usually contains a value configured by the vendor (but can

¹Hostname and vci are mandatory in TR-69, other DHCP values are optional.

be changed by the user). Note that the decision tree did not choose to use this information, we suspect since there are too many possible IoT vendors and NoT vendors.

C. Testing Phase

We tested our model on the unseen dataset and achieved an F1-score of 95.7% on devices that use DHCP (see Table I). The incorrectly classified devices were Harmon Kardon Invok, RENPHO Humidifier, Ubuntu PC and Homepod of Apple. We note that the NoT devices from [25] were configured with a static IP, and thus the technique cannot be applied to them. These devices comprised 10% of our unseen dataset. We classified the devices according to their DHCP packet (one packet is enough), regardless of the classification latency (i.e., without division to time-slots). Thus, the classification latency might be long, if no active request to the router is allowed. We tried to utilize also a random forest algorithm and achieved only slightly better results.

D. Implementation Consideration

Implementing the DHCP classifier required light deep packet inspection, since the required information is in very specific locations, only in the DHCP packets, so that analyzing one such packet is sufficient. Thus, the DPI would require only very few memory references using known DPI algorithms [41].

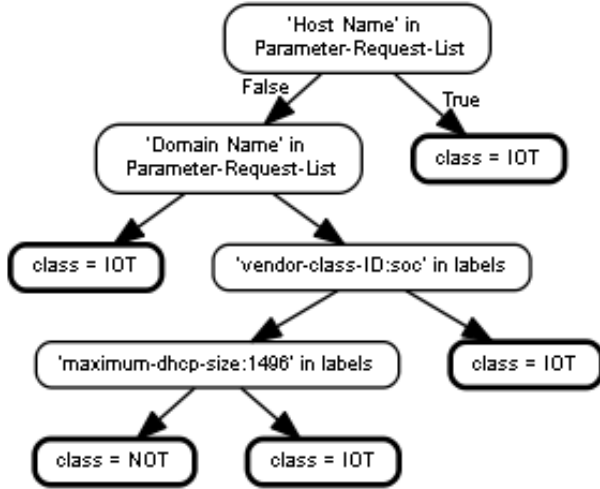


Fig. 5. Decision tree visualization for DHCP -based classifier.

VI. UNIFIED CLASSIFIER

As mentioned, the DHCP information is not always available, but the traffic features classifier is inaccurate for some of the NoT devices in time-slots where the devices are not very active, and when tcp window size information is not indicative enough. Hence, we created a unified classifier using the traffic features classifier on different time slots and the DHCP classifier to improve accuracy.

The unified classifier was heuristically created. We focused on a classification time of 20 minutes. For 20 minutes we checked four 5-minute traffic feature classifiers, two 10-minute

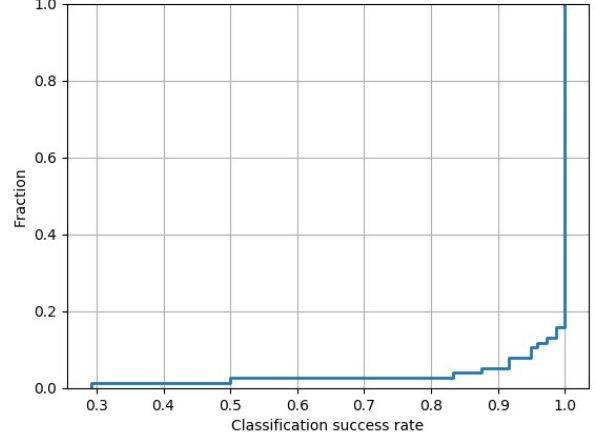


Fig. 6. CDF of a classification success rate of devices, unified classifier, 20 minute time-slot, unseen dataset.

classifiers, and one 20-minute classifier. Then we combined the classifying results based on traffic features with the DHCP classifier result, and weighted the result as two classifiers if DHCP information exists (as a tie breaker). We classified according to the majority (of nine classifiers). The unified approach slightly improves the accuracy, achieving an F1-score of 99.04; see Table I for comparison. We present in Figure 6 the analysis of the classification success rate per device.

VII. CONCLUSION

In this paper, we show that it is possible to classify devices as IoT or NoT with short classification latency using simple and efficient classifiers. Understanding whether a device is IoT or NoT is crucial for visibility and security. Our classifiers are able to classify devices that were not seen in the learning phase. This is an important property of our classifier, since there are no datasets that can cover the huge variety of devices, especially IoT devices. A key benefit of our classifiers is that we can explain the intuition behind the learned classifiers. The unified classifier code was published in our github repository [11], for use and comparative study by the community.

A limitation of our research is that we did not focus on borderline IoT devices (such as Android TV and Echo Show). A further study should be performed on the ability to identify this borderline category.

VIII. APPENDIX

Category	Device Name
NoT	Samsung Galaxy Tab
	Android Phone
	Windows Laptop
	MacBook
	Android Phone
	IPhone
	MacBook/Iphone
	Lg Smartphone
	Nexus 5x Smartphone
	Apple Macbook
	Xiaomi Smartphone
	Laptop win10
	Macbook
	Macbook
	Samsung s7
IoT	Xiaomi mi5
	Galaxy-A7-2017 Smartphone
	Lenovo Win10 laptop
	LG G3 Smartphone
	Asus ZenPad Tablet
	Dell Win7 laptop
	Smart Things
	Netatmo Welcome
	TP-Link Day Night camera
	Samsung SmartCam
	Dropcam
	Insteon Camera, wifi
	Insteon Camera, wired
	Withings Smart Monitor
	Belkin Wemo switch
	TP-Link Smart plug
	iHome
	Belkin wemo motion sensor
	NEST Protect smoke alarm
	Netatmo weather station
	Withings Smart scale
	Blipcare Blood Pressure meter
	Withings Aura smart sensor
	LiFX Smart Bulb
	Tribby Speaker Smart Speaker
	PIX-STAR Photo-frame
	HP Printer
	Nest Dropcam
	Chromecast Streamer
	Yeelink Smart Light Bulb

TABLE V
SEEN DATASET

REFERENCES

- [1] "DDoS attack halts heating in Finland amidst winter." [Online]. Available: <https://metropolitan.fi/entry/ddos-attack-halts-heating-in-finland-amidst-winter>
- [2] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the Internet-of-Things," *Proceedings of the 14th ACM Workshop on Hot Topics in Networks - HotNets-XIV*, 2015.
- [3] W. Wei, "Casino Gets Hacked Through Its Internet-Connected Fish Tank Thermometer," 2018.
- [4] C. Koliass, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the iot: Mirai and other botnets," *Computer*, vol. 49, no. 7, pp. 80–84, 2017.
- [5] "McAfee: Built-in Protection for Your Connected Devices." [Online]. Available: <https://securehomeplatform.mcafee.com/>
- [6] "Trend Micro partners with Asus to beef up IoT security in homes." [Online]. Available: <https://internetofbusiness.com/trend-micro-asus-iot-security/>

Category	Device Name
NoT	Samsung s5 Smartphone
	Android samsung
	Apple iPhone
	Windows Laptop
	Ubuntu PC
	Apple ipad
	xiaomi A2
	Dell Laptop win 10
	Mac laptop
	Xiaomi smartphone
	4× VM Win: 7 64B & Win 8.1 Pro 64B
	VM Win 10 pro 32B
	2× VM Ubuntu 16.4 32B & 64B
	2× VM Ubuntu 14.4 32B & 642B
	Macbook
IoT	iPad
	Iphone
	Android Tablet
	2× Amazone Echo
	Apple HomePod
	August Doorbell Cam
	Belkin Netcam
	Belkin WeMo Link
	Bezeq smarhome
	Bose SoundTouch 10
	Canary
	Caseta Wireless Hub
	Chamberlain myQ Garage Opener
	Chinese Webcam
	D-Link DCS-5009L Camera
	Foscam
	Google Home
	3× Google Home Mini
	Google OnHub
	Harmon Kardon Invoke
	Insteon Hub
	iRobot
	Koogeek Lightbulb
	lifiLab
	Logitech Harmony Hub
	Logitech Logi Circle
	MiCasaVerde VeraLite
	2× Motorola Hubble
	NestCam
	NestDetector
	Nest Camera
	Nest Cam IQ
	Nest Guard
	Netgear Arlo Camera
	Philips HUE Hub
	Piper NV
	Provision ISR
	RENPHO Humidifier
	Ring Doorbell
	Roku 4
	Roomba
	Samsung smart home camera
	Securifi Almond
	smartHub
	Sonos
	TP-Link Smart WiFi LED Bulb
	2× TP-Link WiFi Plug
	WeMo Crockpot
	Wink 2 Hub
	Withings Home
	Wyze Cam

TABLE VI
UNSEEN DATASET

- [7] "RATtrap." [Online]. Available: <https://www.myrattrap.com/>
- [8] "CUJO." [Online]. Available: <https://www.getcujo.com/>
- [9] "Fing-Box." [Online]. Available: <https://www.fing.com/products/fingbox>
- [10] "Bitdefender BOX." [Online]. Available: <https://www.bitdefender.com/box>
- [11] "IoT-or-NoT-Execution-code." [Online]. Available: <https://github.com/haimlevvy/IoT-or-NoT-Execution-code>
- [12] J. Ortiz, C. Crawford, and F. Le, "DeviceMien: Network device behavior modeling for identifying unknown IoT devices," in *IoTDI 2019 - Proceedings of the 2019 Internet of Things Design and Implementation*, 2019.
- [13] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. Ole Tippenhauer, and Y. Elovici, "ProfilIoT: A Machine Learning Approach for IoT Device Identification Based on Network Traffic Analysis," *SAC 2017: The 32nd ACM Symposium On Applied Computing*, 2017.
- [14] M. Lastovicka, T. Jirsik, P. Celeda, S. Spacek, and D. Filakovsky, "Passive os fingerprinting methods in the jungle of wireless networks," in *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018*, 2018.
- [15] J. Martin, E. Rye, and R. Beverly, "Decomposition of MAC address structure for granular device inference," *Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC '16*, pp. 78–88, 2016.
- [16] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Characterizing and classifying IoT traffic in smart cities and campuses," in *2017 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2017*, 2017.
- [17] H. Guo and J. Heidemann, "IP-Based IoT Device Detection," *IoT S&P'18*, pp. 36–42, 2016.
- [18] M. Msadek, R. Soua, T. E. T. I. W. C. And, and U. 2019, "IoT Device Fingerprinting: Machine Learning based Encrypted Traffic Analysis," in *The IEEE Wireless Communications and Networking Conference (WCNC)*, 2019. [Online]. Available: <http://orbilu.uni.lu/handle/10993/39318>
- [19] T. J. OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves, and A.-R. Sadeghi, "Homesnitch: behavior transparency and control for smart home iot devices," in *WiSec '19*, 2019.
- [20] 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 preprint arXiv:1709.04647*, 2017.
- [21] X. Feng, Q. Li, H. Wang, and L. Sun, "Acquisitional Rule-based Engine for Discovering Internet-of-Things Devices," in *Usenix Security*, 2018, pp. 327–341.
- [22] E. Kollmann, "Chatter on the wire: A look at dhcp traffic," 2007. [Online]. Available: <http://myweb.cableone.net/xnih/download/chatter-dhcp.pdf>
- [23] —, "GitHub - xnih/satori." [Online]. Available: <https://github.com/xnih/satori?files=1>
- [24] E. Lear, R. Droms, and D. Romascanu, "RFC 8520: Manufacturer Usage Description Specification," Internet Engineering Task Force, March 2019. [Online]. Available: <https://datatracker.ietf.org/doc/rfc8520/>
- [25] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *4th International Conference on Information Systems Security and Privacy*, 2018.
- [26] A. Shiravi, H. Shiravi, M. Tavallae, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers and Security*, 2012.
- [27] "IoT devices' First-Time Bootup Traces, PREDICT ID: USC-LANDER/IoT_Bootup_Traces-20181107. Provided by the USC/LANDER project <http://www.isi.edu/ant/lander/>." [Online]. Available: https://ant.isi.edu/datasets/readmes/IoT_Bootup_Traces-20181107.README.txt
- [28] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "SoK: Security Evaluation of Home-Based IoT Deployments," *IEEE Symposium on Security and Privacy*, 2019.
- [29] A. B. L. Welch, "The Generalization of 'Student's' Problem when Several Different Population Variances are Involved Published by : Biometrika Trust Stable URL : <http://www.jstor.org/stable/2332510>," *Biometrika*, 2008.
- [30] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, 2006.
- [31] "scikit-learn StandardScaler python documentation." [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>
- [32] L. Regression, "Logit Models for Binary Data," *Bernoulli*, 1978.
- [33] M. C. Gomez, J. Crowcroft, Scharf, "TCP Usage Guidance in the Internet of Things (IoT)," *IETF Internet Draft*, 2018.
- [34] S. Venkataraman, D. Song, P. B. Gibbons, and A. Blum, "New streaming algorithms for fast detection of superspreaders," in *Proc. Network and Distributed System Security Symposium (NDSS)*, 2005.
- [35] N. Bandi, D. Agrawal, and A. El Abbadi, "Fast algorithms for heavy distinct hitters using associative memories," in *International Conference on Distributed Computing Systems (ICDCS)*, 2007.
- [36] T. Locher, "Finding heavy distinct hitters in data streams," in *SPAA*. ACM, 2011.
- [37] D. T. Narten, T. Jinmei, and D. S. Thomson, "IPv6 Stateless Address Autoconfiguration," RFC 4862, Sep. 2007. [Online]. Available: <https://rfc-editor.org/rfc/rfc4862.txt>
- [38] "aireplay-ng - Linux man page." [Online]. Available: <https://linux.die.net/man/1/aireplay-ng>
- [39] The broadband forum, "TR-069 - CPE WAN Management Protocol." [Online]. Available: <https://www.broadband-forum.org/technical/download/TR-069.pdf>
- [40] "scikit-learn DecisionTreeClassifier python documentation." [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>
- [41] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Commun. ACM*, vol. 18, no. 6, pp. 333–340, Jun. 1975. [Online]. Available: <http://doi.acm.org/10.1145/360825.360855>