IoTDevID: A Behavior-Based Device Identification Method for the IoT

Kahraman Kostas, Mike Just, and Michael A. Lones

arXiv:2102.08866v2 [cs.CR] 20 Oct 2021

***Abstract*—Device identification is one way to secure a network of IoT devices, whereby devices identified as suspicious can subsequently be isolated from a network. In this study, we present a machine learning-based method, *IoTDevID*, that recognizes devices through characteristics of their network packets. As a result of using a rigorous feature analysis and selection process, our study offers a generalizable and realistic approach to modelling device behavior, achieving high predictive accuracy across two public datasets. The model's underlying feature set is shown to be more predictive than existing feature sets used for device identification, and is shown to generalize to data unseen during the feature selection process. Unlike most existing approaches to IoT device identification, IoTDevID is able to detect devices using non-IP and low-energy protocols.**

***Index Terms*—IoT security, fingerprinting, machine learning**

I. Introduction

The Internet of Things (IoT) is a collection of objects with embedded systems that can communicate with each other over a wired or wireless network. The "things" can be any physical items that we have now or will use in the future [1]. IoT contributes to human life in many critical areas such as smart homes/cities, retail, healthcare, transportation, agriculture, military, and manufacturing [2]. By 2026, the number of IoT devices in the world is expected to reach 80 billion [3], with a total market of 1.1 trillion USD [4].

Securing IoT devices with traditional security solutions can be challenging due to limited device resources, such as processor, battery and bandwidth [5]. Further challenges arise due to device heterogeneity [1], [6]. For example, a device can have many sensors (temperature, humidity, motion, light, etc.), and the channels it uses to communicate with other devices may need very different requirements. Although some research focuses on specialist areas such as home appliances [7], [8] or smart cities [9], [10], many devices have very different characteristics even under this classification. For example, baby monitors and smart kettles are both classified under home appliances [7], [8]. Further, IoT security has the potential to be neglected by users, primarily due to the lack of a familiar interface [8]. However, since today's IoT devices are widespread and diverse, users can not be expected to recognise and understand the security implications of all of them. Hence, automatic identification of devices in the network is vital in dealing with potential security problems. A successful device identification (DI) system can detect devices in a network, so necessary measures such as updating, limiting, or isolating devices with security vulnerabilities can be taken. In this way, potential security problems caused by IoT devices on the network can be prevented before they arise.

In this paper, we introduce a new IoT DI method that models the behavior of the network packets communicated by the devices. It classifies device behaviors at the individual packet level using generalizable features. In this way, it can detect all kinds of devices, while it offers high detection success with the aggregation method it uses. This work offers the following contributions and differences from previous work:

1) Our method does not strictly depend on identifying features such as the MAC and IP address in the feature extraction step. This gives it wider applicability than previous approaches that depended on this information, such as [11]–[13]. Notably, it allows identification of devices that use non-IP and low-energy protocols such as Bluetooth, ZigBee, or ZWave.
2) Our DI model uses a feature set that summarizes the network behavior of IoT devices. We carried out a systematic study of packet-level features, using various feature selection techniques to build a feature set that improves on previous approaches. Unlike previous studies [12], [14], [15], we were careful to identify and remove features (such as session IDs and port numbers) which do not generalise between data sets.
3) Our approach was developed using a rigorous experimental process. We took steps to prevent information leaking from the test set into training, and used a second, independent, dataset to demonstrate the generality of our approach. Consequently, we believe that our results are more realistic and generalizable than much of the previously published work in this area. Our datasets and scripts are publicly available[1].

The paper is organised as follows. Section II reviews related work. Section III describes the methods we use to model network packets, and the data sets used for evaluation. Section IV gives an overview of our model selection approach, Section V evaluates the selected model, and Section VI compares it against other DI approaches. Limitations are discussed in Section VII, and conclusions are presented in Section VIII.

II. Related Work

This section reviews previous studies that used fingerprinting to classify IoT devices. Fingerprints are feature sets derived from network packets or statistics to reflect the behavior patterns of devices. There are many studies in the literature on DI using fingerprints, but their applicability to IoT devices

All authors are with the Department of Computer Science, Heriot-Watt University, Edinburgh EH14 4AS, UK, e-mail: {kk97, m.just, m.lones}@hw.ac.uk

Supported by Republic of Turkey - Ministry of National Education

[1]Source code available at:github.com/kahramankostas/IoTDevIDv2

is controversial, as these often focus on the physical layer or application layer where IoT has wide protocol variety [13]. Hence, we focus here on research that is based on network packet behavior, including relevant information from each of the data link, network, and transport layers.

One of the first studies to use network packet features in a fingerprint method to identify IoT devices is *IoT Sentinel* [11]. This study used a behavior-based fingerprint to identify vulnerable devices and isolate them from the network, using data from Aalto University IoT device captures. The fingerprints specific to each device were created based on 23 features extracted from each of the first 12 packets for each device, resulting in a fingerprint with 276 values. These 12 packets do not exactly represent flow; they are sequential packets from the same MAC address. The features contain information about source and destination, packet properties, and protocols used. However, since it performs packet merging from MAC addresses, it cannot identify non-IP devices. Also, the *IP address count* feature used in this study is unlikely to generalize beyond the Aalto dataset, since it refers to the number of devices each device communicates with, which is environment-dependent. In this study, 17 of 27 device types were detected with an identification accuracy of above 95%, and 10 with an accuracy of around 50% using random forest (RF). Two other notable studies also used data from the Aalto dataset. In the first [12], a fingerprint consisting of 67 statistical features was created using information extracted from the headers of Ethernet, IP, UDP, and TCP packets of 20-21 consecutive packets. Unlike the others, this study used network statistics in addition to features obtained from the packets. If a device used in one network is moved to a different network, the features extracted from the network packet will not change, but the network statistics will change. Since the network statistics are based not only on devices but also on their interrelationships, it is unlikely that they will generalize well. This study classified the devices with 90.3% accuracy using various machine learning (ML) methods: RF, latent dirichlet allocation, k-nearest neighbours (kNN), gradient boosting (GB), decision tree (DT), naïve Bayes (NB), support vector machines (SVM), and adaptive boosting. The second study [14] obtained 95% accuracy using 33 features selected by a genetic algorithm (GA) from 212 features extracted from the headers of individual packets. They used DT, decision table, one rule, and PART as their ML algorithms. However, this study selectively presented its results with only part of the dataset (23 out of 27 device classes) and it used features that are unlikely to generalise (e.g., IP ID, TCP acknowledgment, TCP sequence, source and destination port numbers).

IoTSense [13] used selected features of IoT Sentinel based on their own design assessment, namely 17 protocol-based features which reflect device behavior. They also added three payload-related features, notably payload length and entropy. This feature list was applied to five packets for each device to produce a 100-member fingerprint, as an average of five packets were found to make up a session in this study. The packets are presumably merged according to common MAC address, though this is not explicitly stated in the paper. As a result of this study, per device recall of 93–100% and an average accuracy of 99% were achieved using GB, DT and kNN. While some comparisons are made with the work of IoT Sentinel, the evaluation of IoTSense used a much smaller number of devices (i.e., 10 vs. 31). In addition, the IoTSense experiment set began with 14 devices, though only 10 devices were used for the evaluation as four devices did not produce sufficient data for the analysis approach that was used.

Sivanathan et al. [15] used the data obtained from a reasonably large variety of 28 IoT devices (such as cameras, lights, plugs, motion sensors, appliances and health-monitors) during six months to classify IoT devices. Eight features were used for the classification: flow volume, flow duration, average flow rate, device sleep time, server port numbers, domain name server (DNS) queries, network time protocol (NTP) queries and cipher suites. NB was used in the first step of the two-step classification system and RF was used in the second step. As a result, 28 IoT devices were classified with an accuracy of 99.88%. However, four devices from the dataset were not used, and some elements of the feature set were too specific, thereby not focusing on device behavior, e.g., port numbers, DNS queries, and cipher suites.

III. Materials and Methods

A. IoT Data Selection

We used two public datasets to measure and compare the performance of our DI approach. Both of these contain real device data. Since our approach is designed for benign networks, the datasets do not contain attack data.

The first, the Aalto University IoT Devices Captures dataset [11] (referred to as *Aalto dataset*) has 31 devices and contains only the device installation data, but this installation process was repeated 20 times for each device to increase the amount of data [11]. Whilst there are a total of 31 devices in this dataset, four of these devices are in pairs (see supplementary material (SM) Table S1). For example, there are two WeMoSwitches with different MAC / IP addresses. Our purpose is to detect according to device behavior, so these device pairs are considered as a single device, not as two separate devices. Therefore, although the dataset contains 31 devices, it contains 27 classes. Another characteristic of the dataset is that two devices using low energy protocols connect to the gateway where the data is collected via other devices. Therefore, these two devices (D-LinkDoorSensor and HueSwitch) do not have identifying features such as their own MAC and IP address. They use the IP and MAC addresses of the devices they communicate through (D-LinkHomeHub and HueBridge respectively). We refer to this as "the transfer problem" in this paper (related device MAC addresses and labels are shared in SM-Table S1). Notably, when using an individual packet-based approach (which is independent of MAC and IP addresses), our method is able to identify devices that are suffering from the transfer problem.

The second, UNSW IoT Traffic Traces [15] (referred to as *UNSW dataset*) contains the day-to-day network logs of 28 IoT devices, each recorded over a period of 26 weeks. However, only a 60-day portion of this data is publicly available. This limited data does not include the four devices (Ring Door Bell,

Hello Barbie, August Doorbell Camera, Belkin Camera) from the work of Sivanathan et al. [15]. Therefore, we extracted data for these devices from the benign data of another study [16] by the same institution. The dataset we use contains a total of 32 IoT devices and 7 non-IoT devices. We gathered non-IoT devices under one class, as the other study [15] working with this dataset did, resulting in 33 total labels.

Since the size of the UNSW dataset is quite large, we used the smaller Aalto dataset to formulate our modelling approach and the UNSW dataset to measure its broader generality.

B. Individual, Aggregated and Mixed Method Packets

Creating fingerprints by combining multiple packets is a common strategy [11]–[13], [15], where identifying features such as a MAC or IP address are used to combine multiple packets. These identifying features uniquely identify the device but do not provide information about the device type or behavior. However, they give us the source of the data. Assuming that packets from the same address come from the same device, larger fingerprints can be constructed by combining these packets. However, assuming that "packets from the same address come from the same device" is not always correct. In cases where there is a *transfer problem* (e.g., multiple devices connecting through the same gateway), a MAC and IP address can represent more than one device. Consequently, the packet merging logic will not work, as devices with the same MAC address but different behavior will be considered as a single device. Further, there is no standard for the size of this merging process (number of packets, duration, etc.). The use of individual packets is also beneficial in terms of speed and efficient use of limited resources [17]. For these reasons, unlike many previous studies [11]–[13], [15], we chose to use *individual packets*, not merged ones as the basis for discrimination.

However, when viewed individually, some individual packets may be ambiguous and match the behavioral profile of more than one device. Therefore, the success rate from discrimination based on single packets is limited. We address this by introducing an alternative to individual packets, called *aggregated method* that groups packets based on identifier attribute (MAC address) and machine learning (ML) labels assigned at the individual packet level. Our aggregation algorithm (see SM-Algorithm 1 for pseudocode) takes the device MAC addresses $M = \{m_1, m_2, \ldots, m_n\}$ and the result of the ML algorithm (initial labels) $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n\}$ as input. It creates groups of size g from the labels grouped according to MAC addresses. The labels originating from the same MAC address are collected under the same group, and then the algorithm re-evaluates these groups. During this re-evaluation process, the mode (the most frequent element) of each group is assigned to the entire group as a new value. The algorithm gives the modified results as outputs (new labels) $\hat{Y}' = \{\hat{y}'_1, \hat{y}'_2, \ldots, \hat{y}'_n\}$. While this works well for DI with benign data, care should be taken in networks with malicious data. Depending on the tolerance of the aggregation algorithm, malicious packets that impersonate an IP/MAC address and are close to the behaviour of the benign packets with the same IP/MAC address may be grouped together.

In this way, we can increase the success rate by ignoring the mislabelled packets. The difference of this approach from previous work which combined packets is that the behavior analysis in our aggregation is performed *before* the merging process. In other words, we combine not the packets, but the labels assigned to these packets by the ML algorithm. Unfortunately, this method does not work for MAC addresses suffering from the transfer problem. We have added an exception to our evaluation phase and followed a *mixed method* to deal with this issue. If the results of the aggregation algorithm show more than one behavior at a MAC address, if there is not a single dominant behavior label, this MAC address is added to the exception list. Only individual packet evaluation results are used for packets from this MAC address, while aggregated results are used for the remaining devices.

C. Feature Extraction

We performed feature extraction from the packet headers of `pcap` (packet capture) files. Before this, we separated the `pcap` files into test and training sets to completely isolate these sets. The Aalto dataset has 20 sessions (`pcap` files) per device. We divided them in two parts: 16 parts training and four parts test data (80%:20%). The UNSW dataset consists of a different `pcap` file for each day's records. To isolate, we constructed the training and test data from data collected on different days (`pcap` files) (see the SM-Table S2 for distribution). Section IV-A explains our feature selection process in detail.

In addition to the 111 features extracted from the network packet headers, we included payload entropy [13], protocol (from TCP-IP layers), source and destination port class [11], which are all features found to be useful in previous studies. While payload entropy gives clues about the characteristic of the payload, the protocol and port class features provide summary information about source and destination ports. Source-destination port class features gather port numbers under 14 classes: No port, 0-Reserved, 53-DNS, 67-BOOTP server, 68-BOOTP client, 80-HTTP, 123-NTP, 443-HTTPS, 1900-SSDP, 5353-mDNS, 49153-ANTLR, 0:1023-well-known ports, 1023:49151-registered ports, and 49151:65535-dynamic (private) ports [11]. The port classification above is ordered so that, for example, a packet with port 53 is not also classified as a well-known port. Section IV-A explains further how port number classes are created.

MAC and IP addresses are source-destination based identifying features. Although they uniquely identify source and destination devices, they do not provide information about device behaviors. For example, two devices with the same behavior can have different MAC and IP addresses. Because they are not predictable and generalizable, we do not use them in our feature set (though they are used in our aggregation method).

IV. Model Selection

A. Feature Selection

With our initial features extracted, we used a feature-importance-based voting method via the Xverse[2] package to eliminate unnecessary features. This method calculates the importance scores of all features for each device using six different techniques, and then uses voting to decide whether to include them (see Fig. 1 for features and their vote rates). The six scoring techniques used by this method are information value using the weight of evidence, variable importance using RF, recursive feature elimination, variable importance using extra trees classifier, chi-square best variables, and L1-based feature selection. We removed from our feature pool the 26 features that failed to receive votes on any device from any of these six techniques.

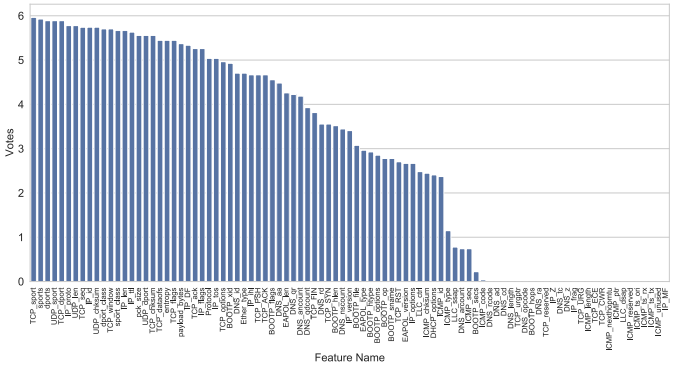

Fig. 1: The features and the average votes they received (Aalto dataset).

A number of the remaining features contain information that is potentially specific to a particular session, and therefore may not be useful for identifying devices more generally. For some features, this is clear, e.g., the initial values of the IP ID, TCP sequence and acknowledgment numbers are randomly assigned, and the next values are consecutive numbers following this initial value; hence, they they do not contain information that is useful for DI beyond their current session. For other features, such as TCP port numbers, their generality is less clear. So, to measure whether or not each of these features is beneficial, we excluded all of them from the feature set and trained an ML model (DT) to determine a baseline level of performance. In turn, we then added each of the features to the baseline set and observed whether this improved or impaired the performance.

The results are shown in Fig. 2. First of all, it is notable that the results are strongly dependent on whether the models are evaluated using cross-validation or using isolated training and test sets. This is presumably because, when using cross-validation, it is more likely that the training and test data will be sampled from the same session — meaning that session-specific patterns learned in the training set will also generalise to the test set. This may explain why session-specific features were chosen during feature selection, and highlights the danger of using off-the-shelf feature selection algorithms on data of this kind. Using isolated training and test sets, by comparison,

[2]xverse package available at: pypi.org/project/xverse/

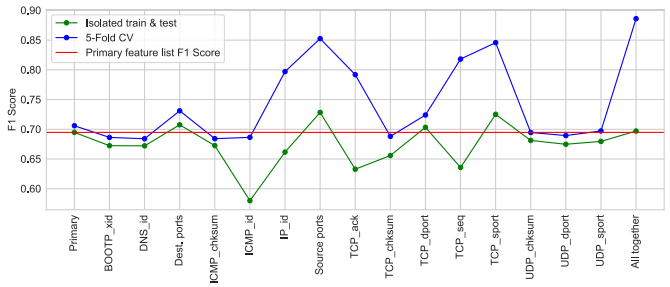

Fig. 2: Effect of identifying features on generality, as measured by isolated test sets and cross-validation.

it is clear that session-specific features such as the IP ID cause over-fitting and impair the generality of models.

On the other hand, Fig. 2 shows that port-based features can be useful for building generalisable models. For example, consider the port numbers used by one of the devices in the Aalto dataset which contains multiple device instances. Fig. 3 shows a word cloud built from the data from the two Edimax camera instances. Although a few port numbers (representing the protocols BOOTP, HTTP, SSDP and mDNS) generalise between the devices, the remaining ports are presumably session-based and do not generalise. To address this, we do not use the raw feature values for port-based features, but rather map them to discrete values representing port classes and protocols (see Section III-C) — in effect, pruning the instance-specific information whilst preserving the behavioral information they provide.

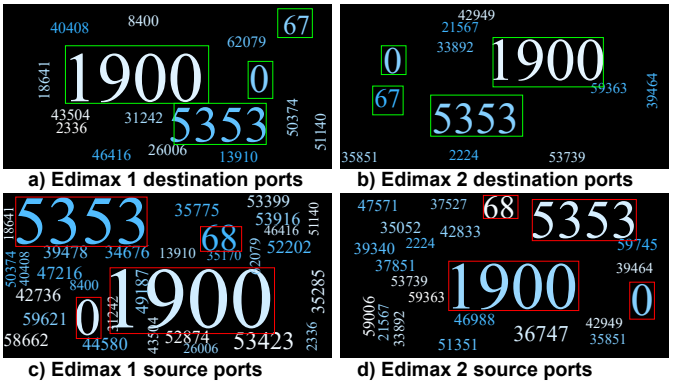

Fig. 3: Representation of the port number distributions of two devices of the same type using word clouds. The red and green boxes show port numbers specific to protocols such as SSDP, mDNS, and BOOTP. The size of the number shows how often it appears in the data.

Finally, after eliminating identifying and redundant features, we used a GA to decide on the most appropriate feature set from the remaining 52-member feature pool. The GA uses a wrapper method and thus tests the usefulness of feature sets in the context of a particular classifier (DT). In this way, we have created a feature subset that detects with higher performance, while decreasing the model complexity by reducing the number of features. Table I shows the final list of features the GA selected.

BOOTP_chaddr	dports	ICMP_nexthopmtu	DNS_ra	payload_bytes	DNS_aa
BOOTP_ciaddr	ICMP_chksum	ICMP_ts_ori	DNS_rcode	EAPOL_version	DNS_ancount
BOOTP_giaddr	ICMP_id	ICMP_ts_rx	DNS_tc	TCP_ACK	DNS_arcount
BOOTP_siaddr	IP_id	ICMP_ts_tx	DNS_z	TCP_dataofs	DNS_nscount
BOOTP_yiaddr	sports	ICMP_unused	ICMP_length	TCP_FIN	EAPOL_len
DNS_an	sport23	IP_frag	dport_class	TCP_window	ICMP_seq
DNS_ar	TCP_ack	IP_MF	EAPOL_type	BOOTP_file	ICMP_type
DNS_ns	TCP_chksum	IP_Z	pck_size	BOOTP_flags	IP_proto
DNS_qd	TCP_dport	LLC_dsap	UDP_len	BOOTP_hlen	IP_version
Ether_dst	TCP_seq	TCP_CWR	Ether_type	BOOTP_options	LLC_ssap
Ether_src	TCP_sport	TCP_ECE	ICMP_code	BOOTP_sname	Protocol
ICMP_addr_mask	UDP_chksum	TCP_reserved	IP_DF	DHCP_options	sport_class
ICMP_gw	UDP_dport	TCP_URG	IP_flags	DNS_qdcount	TCP_flags
IP_chksum	UDP_sport	TCP_urgptr	IP_ihl	DNS_qr	TCP_options
IP_dst	BOOTP_xid	BOOTP_hops	IP_len	DNS_rd	TCP_PSH
IP_src	DNS_id	DNS_ad	IP_options	Payload entropy	TCP_RST
MAC_dst	dport23	DNS_cd	IP_tos	BOOTP_htype	TCP_SYN
MAC_src	ICMP_ptr	DNS_length	IP_ttl	BOOTP_op	
ts (timestamp)	ICMP_reserved	DNS_opcode	LLC_ctrl	BOOTP_secs	

Source-destination based identifying features
Session based identifying features
Features removed as a result of the voting process
Features selected by the genetic algorithm

TABLE I: List of features used in this study. The green part is the final feature set.

B. Algorithm Selection

Based on earlier approaches (see Section II) and surveys of related work [1], [18] which sought to train an ML algorithm to correctly predict a device's type from extracted features, we considered the following six ML algorithms: random forest (RF), k-nearest neighbours (kNN), gradient boosting (GB), decision tree (DT), naïve Bayes (NB), and support vector machine (SVM). We used random search with nested cross-validation (as implemented by scikit-learn[3]) to find suitable hyperparameters for each algorithm. Using these algorithms, we trained multi-class classifiers to discriminate the 27 classes from the Aalto dataset, training each model 100 times to measure its stability. See Table II for the results.

TABLE II: Comparison of ML algorithms with average and standard deviation (SD) of 100 repeats on the Aalto dataset. t is time, in seconds.

ML	Accuracy	Precision	Recall	F1 score	Train-t	Test-t
DT	0.705±0.001	0.774±0.003	0.706±0.001	0.727±0.001	0.128	0.004
GB	0.699±0.001	0.789±0.003	0.693±0.001	0.725±0.001	918.3	8.312
kNN	0.705±0.000	0.752±0.000	0.705±0.000	0.718±0.000	0.005	20.20
NB	0.617±0.000	0.584±0.000	0.629±0.000	0.559±0.000	0.433	0.032
RF	0.708±0.001	0.768±0.004	0.708±0.001	0.727±0.002	3.742	0.333
SVM	0.680±0.000	0.697±0.000	0.634±0.000	0.649±0.000	101.3	64.80

It can be seen that RF, DT and GB are the best performing algorithms, in terms of DI. The fastest in terms of inference time are DT and NB, though the accuracy of NB is very low. Although kNN and GB perform relatively well in terms of accuracy, they are not practical to use due to their slowness. The SVM algorithm is also not a reasonable option in terms of speed and accuracy. kNN, GB, and SVM are particularly disadvantageous due to their very large inference times. In a real-time device detection system that will operate in a millisecond-level processing environment such as network traffic, inference time in seconds are not acceptable. Based on these observations, we use DT in the remainder of this work, since it offers the best balance between speed and accuracy.

V. Performance Evaluation

Based on our model selection we present our results here for the three variations of our method: individual, aggregated, and mixed. First, we looked at the relationship between group size and performance within the context of the aggregation algorithm. From Fig. 4, there is a positive relationship, showing that larger group sizes are generally more effective. However, many of the IoT devices communicate infrequently, so large group sizes would be impractical in some cases. For this reason, we selected a group size of 13, which is approximately the point where performance starts to plateau.

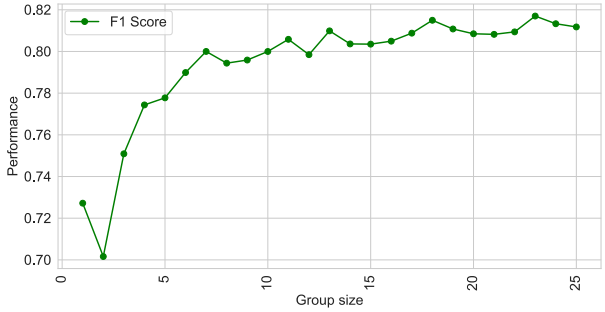

Fig. 4: The effect of group size on model success.

Table III shows overall results for the two datasets. Using the individual packet approach, an F1 score of 73% was achieved in the Aalto dataset and 83% in the UNSW dataset. However, a significant increase in the ability to correctly identify devices of both datasets was observed by using aggregation. The overall F1 score increased to 81% on the Aalto dataset and to approximately 94% on the UNSW dataset. Since feature selection was done using only the Aalto dataset, it is very encouraging to see such a high level of discrimination on the UNSW dataset. This is a good indication that the selected feature set will generalise well to other IoT environments.

TABLE III: Results and their SD obtained using Individual, Aggregated and Mixed approaches on Aalto and UNSW datasets. t is time in seconds. Alg-t indicates the running time of the aggregation algorithm.

Method	Dataset	Accuracy	F1 score	Test-t	Alg-t
Individual	Aalto	0.705±0.001	0.727±0.001	0.004	0.000
	UNSW	0.853±0.010	0.834±0.012	0.008	0.000
Aggregated	Aalto	0.745±0.011	0.809±0.005	0.007	0.164
	UNSW	0.943±0.012	0.937±0.017	0.017	0.425
Mixed	Aalto	0.833±0.002	0.861±0.004	0.008	0.216
	UNSW	0.941±0.012	0.935±0.017	0.022	0.479

Table IV shows the average discrimination performance of DT models at the device level for the Aalto Dataset. As Table IV also shows, the dataset is highly imbalanced, meaning that we need to use a metric that is relatively insensitive to class size imbalance in order to give a meaningful picture of the model's performance at the device level – hence our use of the F1 score. Other works have used accuracy for this, which is an inappropriate metric for imbalanced datasets.

When the device-based results of the Aalto dataset are examined, we see that the aggregation algorithm contributes positively to almost all devices, while negatively affecting only four devices. This is because the pairs that make up these four devices suffer from the transfer problem (they share the same MAC address, see SM-Table S1). To deal with this, we used

[3]scikit-learn package available at: scikit-learn.org/stable/

TABLE IV: Device proportions over the entire dataset, F1 scores per device according to different approaches (Aalto Dataset, average of 100 repetitions).

	Packet statistics		Packet discrimination		
Device name	Packets	Percent	Individual	Aggregated	Mixed
Aria	441	0.420	0.932	1.000	1.000
D-LinkCam	6244	5.940	0.891	1.000	0.988
D-LinkDayCam	1063	1.010	0.864	1.000	1.000
D-LinkDoorSensor	1892	1.800	0.762	0.057	0.788
D-LinkHomeHub	8595	8.180	0.681	0.797	0.777
D-LinkSensor	6549	6.230	0.382	0.644	0.626
D-LinkSiren	6186	5.890	0.367	0.640	0.631
D-LinkSwitch	6519	6.200	0.665	0.969	0.964
D-LinkWaterSensor	6435	6.120	0.392	0.624	0.622
EdimaxCam	831	0.790	0.872	1.000	1.000
EdimaxPlug1101W	1160	1.100	0.601	0.827	0.824
EdimaxPlug2101W	1010	0.960	0.453	0.708	0.699
EdnetCam	408	0.390	0.833	1.000	1.000
EdnetGateway	683	0.650	0.908	1.000	1.000
HomeMaticPlug	611	0.580	1.000	1.000	1.000
HueBridge	13936	13.260	0.810	0.217	0.815
HueSwitch	18448	17.560	0.891	0.720	0.891
IKettle2	145	0.140	0.727	1.000	1.000
Lightify	4149	3.950	0.977	1.000	1.000
MAXGateway	567	0.540	0.964	1.000	1.000
SmarterCoffee	149	0.140	0.727	0.983	0.981
TP-LinkPlugHS100	667	0.630	0.693	0.748	0.746
TP-LinkPlugHS110	636	0.610	0.429	0.336	0.328
WeMoInsightSwitch	5962	5.670	0.667	0.874	0.866
WeMoLink	6625	6.300	0.638	0.929	0.924
WeMoSwitch	4477	4.260	0.511	0.769	0.768
Withings	688	0.650	1.000	1.000	1.000

TABLE V: Device proportions over the entire dataset, F1 scores per device according to different approaches (UNSW Dataset,average 100 repetitions).

	Packet statistics		Packet discrimination		
Device name	Packets	Percent	Individual	Aggregated	Mixed
Amazon Echo	10000	2.831	0.882	0.985	0.983
AugustDoorbellCam	10000	2.831	0.701	0.960	0.955
AwairAirQualityMon	10000	2.831	0.952	0.993	0.994
Belkin Camera	10000	2.831	0.890	1.000	1.000
BelkinWeMoSwitch	10000	2.831	0.803	0.988	0.988
BelkinWeMoSensor	10000	2.831	0.772	0.987	0.988
BlipcareBPMeter	119	0.034	0.851	1.000	1.000
Canary Camera	10000	2.831	0.761	0.928	0.911
Dropcam	10000	2.831	0.998	1.000	1.000
GoogleChromecast	10000	2.831	0.448	0.347	0.341
HPPrinter	10000	2.831	0.468	0.710	0.707
HelloBarbie	164	0.046	0.677	1.000	1.000
InsteonCam	10000	2.831	0.953	0.997	0.997
LightLiFXSmartBulb	10000	2.831	0.979	1.000	1.000
NEST-PSmokeAlarm	7063	1.999	0.935	1.000	1.000
Nest Dropcam	10000	2.831	0.966	1.000	1.000
NetatmoWelcome	10000	2.831	0.922	0.998	0.994
NetatmoWeatherSt	10000	2.831	0.896	1.000	1.000
Non-IoT	67295	19.05	0.872	0.929	0.929
PIX-STAR-Pframe	10000	2.831	0.587	0.928	0.885
PhillipHLightbulb	10000	2.831	0.971	1.000	1.000
RingDoorBell	10000	2.831	0.856	0.999	0.995
SamsungSCam	10000	2.831	0.678	0.892	0.896
SmartThings	10000	2.831	0.995	1.000	1.000
TP-LinkCloudCam	10000	2.831	0.979	1.000	1.000
TP-Link Smart plug	10000	2.831	0.875	1.000	1.000
TP-LinkRouter	10000	2.831	0.908	0.971	0.979
TribySpeaker	10000	2.831	0.908	0.999	0.999
WithingsSlpSensor	10000	2.831	0.437	0.404	0.411
WithingsBabyMon	10000	2.831	0.984	1.000	1.000
WithingsSScale	8279	2.344	0.973	1.000	1.000
iHome	10000	2.831	0.958	1.000	1.000
unknownIoT	340	0.096	0.676	0.894	0.891

our mixed method by adding an exception to our aggregation algorithm. Table III shows that with the application of the mixed approach, the overall F1 score, which was 81% in the Aalto dataset, increased to 86%. Since there is no transfer problem in the UNSW dataset, this approach does not impact its results significantly — see Table V.

It is notable that the performance on the Aalto dataset is significantly lower than the UNSW dataset. This appears to be due to certain device subgroups. A confusion matrix containing only low-performance devices is given in Fig. 5. In this case, the devices in a subgroup have some similarities: they are either similar purpose devices manufactured by the same companies (e.g., Groups 1, 3 and 5 in Fig. 5) or different models of the same device (e.g, Groups 2 and 4 in Fig. 5). It does not seem possible to perfectly separate these devices according to their behavior, at least when observed at the network level. However, it is likely that these devices use very similar hardware and software, and so exhibit similar behavior, as well as similarities in vulnerabilities and their prevention [11]. Consequently, from a device-identification perspective, it is plausible to consider these devices under a single label. When doing this, the accuracy for the Aalto dataset increases from 73% to 88% for the individual method, and from 86% to 97% for the mixed method.

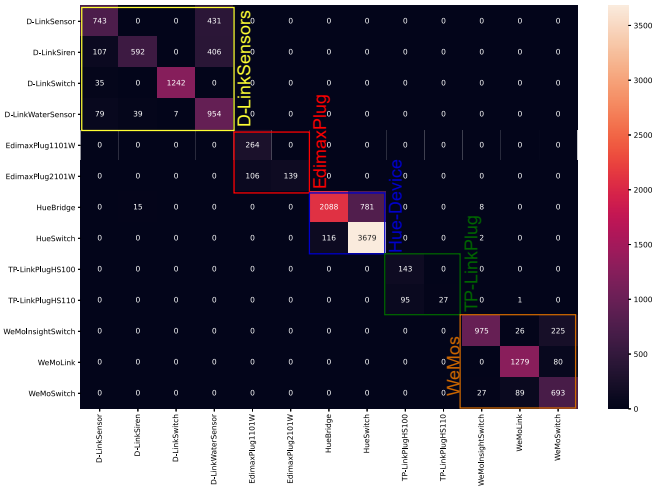

Fig. 5: The confusion matrix of low-performance devices. Groups 1: yellow, 2: red, 3: blue, 4: green, 5: orange.

VI. Comparison with previous work

In Table VI, we compare the overall results of our study with previously published results. While, at first glance, our study does not appear to yield higher numeric results than existing work, there are a number of reasons why this is not a fair comparison. First, many approaches use overly specific features that are unlikely to generalize to unseen datasets. As our analysis in Section IV-A showed, this may be a result of information leaking from the test set into the training set, something that is easy to do unintentionally. Second, there is widespread use of accuracy figures with imbalanced datasets; since IoT device datasets inherently suffer from unbalanced distributions, using accuracy as the sole evaluation criterion can be highly misleading. Third, there is inconsis-

tency amongst the datasets used, with some studies reporting results for only partial datasets. We summarize these points in Table VI. In our work, by comparison, we were careful to filter out dataset-specific features, to prevent information leakage, to use appropriate metrics, and to report full results. We also explicitly used a second dataset to measure the generality of the approach. Consequently, we believe that our results are unbiased and more likely to indicate likely success on unseen data than those reported in previous studies.

TABLE VI: Comparing IoTDevID with former studies

Study	Dataset	Result	Metric	Key limitations
[11]	Aalto	81.50%	Accuracy	Transfer problem.
[12]	Aalto	90.30%	F1 score	Transfer problem. Overly specific features.
[14]	Aalto	82%	Accuracy	Used partial dataset. Overly specific features.
[13]	Private	99%	Accuracy	Used partial dataset. Unshared dataset.
[15]	UNSW	99.98%	Accuracy	Overly specific features.
Our Study	Aalto	83.30%	Accuracy	See Section VII
	UNSW	94.30%		
	Aalto	86.10%	F1 score	See Section VII
	UNSW	93.70%		

A key contribution of our work is the determination of a feature set that provides good discrimination across datasets, and is therefore likely to provide a good basis for DI more generally. To give more insight into this, and to offer a more meaningful comparison against previous approaches, we compared this feature set against the feature sets used in IoTSense [19] and IoT Sentinel [11]. Note we did not include the other methods listed in Table VI in this comparison, either because they were flow-based [12], [15] or their feature set was not shared [13]. We applied all three of our approaches (individual, aggregated, and mixed) using all three feature sets and both datasets. Table VII shows that, in all cases, our feature set resulted in significantly better device discrimination than the previously published feature sets. It is also notable that the performance metrics we observed when using the feature set from IoTSense are significantly worse than that method's published figure of 99% accuracy, which again highlights the problem of basing comparisons on published figures alone.

VII. Limitations

In our study, we used all the IoT DI datasets that were publicly available at the time (see Section III-A). However, IoT technologies are a rapidly developing field and new data is likely to have been released even as we continued our work. We encourage other researchers to use other datasets to further test the generalizability of our results and we provide scripts to make this possible. We used a broad selection of commonly-used ML algorithms (see Section IV-B), though this was not exhaustive, and there may be other algorithms that might perform better. However, we have done some preliminary research using deep neural networks (not reported here), and so far our results suggest that more traditional ML techniques work better on this particular problem.

TABLE VII: Comparison of feature sets on the Aalto and UNSW datasets.

Data		Method	Accuracy	F1 score	Test-t	Alg-t
Individual	Aalto	IoTDevID	0.705±0.001	0.727±0.071	0.004	N/A
		IoTSense	0.639±0.000	0.561±0.001	0.006	N/A
		IoTSentinel	0.700±0.001	0.602±0.002	0.008	N/A
	UNSW	IoTDevID	0.853±0.010	0.834±0.012	0.022	N/A
		IoTSense	0.710±0.010	0.697±0.012	0.023	N/A
		IoTSentinel	0.526±0.010	0.510±0.010	0.027	N/A
Aggregated	Aalto	IoTDevID	0.745±0.011	0.809±0.005	0.007	0.164
		IoTSense	0.671±0.003	0.657±0.004	0.007	0.142
		IoTSentinel	0.671±0.003	0.639±0.005	0.007	0.136
	UNSW	IoTDevID	0.943±0.012	0.937±0.017	0.023	0.425
		IoTSense	0.840±0.011	0.834±0.014	0.024	0.450
		IoTSentinel	0.710±0.015	0.689±0.016	0.026	0.420
Mixed	Aalto	IoTDevID	0.833±0.002	0.861±0.004	0.008	0.216
		IoTSense	0.748±0.004	0.701±0.005	0.007	0.174
		IoTSentinel	0.778±0.010	0.691±0.008	0.006	0.165
	UNSW	IoTDevID	0.941±0.012	0.935±0.017	0.022	0.479
		IoTSense	0.844±0.010	0.835±0.012	0.024	0.487
		IoTSentinel	0.700±0.017	0.679±0.018	0.027	0.528

Although IoTDevID identifies devices well, it does not provide solutions for detecting vulnerable devices and taking necessary precautions about these devices. A Software-Defined Networking (SDN) based network management solution may be useful for this process. Furthermore, IoTDevID is currently intended only for use in benign networks, meaning that there is still a need for an intrusion detection system to protect against attacks. A further consideration is how a DI model would be deployed, trained and maintained in a particular network. Whilst we focus on the performance of multi-class classifiers, i.e., a single model for discriminating between all devices on the network, this may not be optimal in situations where devices are frequently added and removed. A more practical architecture in this case may be an ensemble of single-class classifiers, since this would not require retraining of the entire model. However, our initial results (not reported here) suggest there may be a trade-off against performance, with multi-class models offering better discrimination.

VIII. Conclusions

In this study, we present an ML-based method for identifying IoT devices. Identifying the IoT devices with a network is important, both for finding and removing rogue devices, and for understanding the security vulnerabilities of a network more generally. A number of previous studies have attempted to use ML in this process. However, we have identified various factors that may limit the generality of their findings, including the use of session-based identifying features, and the use of inappropriate metrics. In this study, we have attempted to go about this process in a rigorous and transparent manner, with the aim of developing a robust method of DI that reliably generalises beyond the data on which it was trained.

A key contribution of this work is the use of a multi-stage feature selection process to determine a set of generalizable packet-level features that can be used as a basis for building robust models. Notably, we compared this feature set against those used in previous studies, and showed that it supports the

training of significantly better ML models. We also demonstrate that the feature set generalises well to a data set that was unseen during feature selection, giving us confidence that it provides a meaningful basis for the broader discrimination of IoT devices.

We also address the problem of detecting non-IP devices. A limitation of earlier approaches is that they use IP or MAC addresses to merge packets prior to applying an ML model, which means they are not applicable in situations where a device does not have an IP or MAC address. Instead, we apply ML at the packet level, and then use an aggregation algorithm that takes into account both the packet-level classification and, if available, the IP or MAC address. If the latter is not available, it is still possible to carry out identification based upon individual packets alone.

From an ML-perspective, we found decision trees to offer the best trade-off between predictive performance and inference time, the latter being an important consideration for deploying a model to monitor network traffic in real time.

In future work, we plan to develop an IDS that will work with IoTDevID and detect attacks against the network, and an SDN-based network management system that evaluates both device identification and intrusion detection outputs. Thus, the aim is to have an IoT security system that avoids risk of potential vulnerabilities, prevents attacks and can be used in a real world setting.

REFERENCES

[1] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in iot security: Current solutions and future challenges," *IEEE Com. Surveys & Tutorials*, vol. 22, no. 3, pp. 1686–1721, 2020.

[2] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef, "Internet of things security: A top-down survey," *Computer Networks*, 2018.

[3] M. Rosen, "Driving the digital agenda requires strategic architecture," accessed: 2020-04-07. [Online]. Available: https://idc-cema.com/dwn/SF_177701

[4] "Internet of things market size, growth IoT industry report 2026," 2019, accessed: 2020-04-07. [Online]. Available: https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307

[5] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[6] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Communications Surveys & Tutorials*, 2019.

[7] M. Nobakht, V. Sivaraman, and R. Boreli, "A host-based intrusion detection and mitigation framework for smart home IoT using openflow," in *2016 11th Int. Conf. on ARES*. IEEE, 2016, pp. 147–156.

[8] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "Sok: Security evaluation of home-based IoT deployments," in *Symp. on Security and Privacy*. IEEE, 2019, pp. 1362–1380.

[9] M. Esmalifalak, L. Liu, N. Nguyen, R. Zheng, and Z. Han, "Detecting stealthy false data injection using machine learning in smart grid," *IEEE Systems Journal*, vol. 11, no. 3, pp. 1644–1652, 2014.

[10] M. Ozay, I. Esnaola, F. T. Y. Vural, S. R. Kulkarni, and H. V. Poor, "Machine learning methods for attack detection in the smart grid," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 8, 2015.

[11] 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 *37th Int. Conf. DCS*. IEEE, 2017.

[12] S. A. Hamad, W. E. Zhang, Q. Z. Sheng, and S. Nepal, "IoT device identification via network-flow based fingerprinting and learning," in *18th IEEE TrustCom*. IEEE, 2019, pp. 103–111.

[13] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Behavioral fingerprinting of iot devices," in *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Sec.*, 2018, pp. 41–50.

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

[15] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying IoT devices in smart environments using network traffic characteristics," *IEEE-TMC*, 2018.

[16] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, "Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity," in *2019 ACM Symp. on SDN Research*, 2019, pp. 36–48.

[17] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang, "An LSTM-based deep learning approach for classifying malicious traffic at the packet level," *Applied Sciences*, vol. 9, no. 16, p. 3414, 2019.

[18] M. A. Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (IoT) security," *IEEE Communications Surveys & Tutorials*, 2020.

[19] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Iotsense: Behavioral fingerprinting of IoT devices," *arXiv preprint arXiv:1804.03852*, 2018.

Kahraman Kostas received the MSc degree in Computer Networks and Security from the University of Essex, Colchester, U.K., in 2018. He is a PhD candidate in Computer Science at Heriot-Watt University, Edinburgh, U.K. His research focuses on the security of computer networks and Internet of Things. You can find more information at https://kahramankostas.github.io/aboutme/.

Mike Just earned his Ph.D. in Computer Science from Carleton University in 1998 and is currently an Associate Professor at Heriot-Watt University. He is primarily interested in computer security, and in applying human-computer interaction and machine learning techniques to solve computer security problems. You can find more information at http://www.justmikejust.co.uk/.

Michael A. Lones (M'01—SM'10) is an Associate Professor of Computer Science at Heriot-Watt University. He received both MEng and PhD degrees from the University of York. He carries out research in the areas of machine learning and optimisation, where he has a particular interest in biologically-inspired approaches. Application areas of his work include medicine, robotics and security. You can find more information at http://www.macs.hw.ac.uk/∼ml355.

S1. SUPPLEMENTARY MATERIAL

TABLE S1: Device MAC addresses and labels in Aalto University dataset.
MAC addresses in bold correspond to more than one label, and labels in bold correspond to more than one MAC address.

MAC Address	Label	MAC Address	Label
94:10:3e:35:01:c1	**WeMoSwitch**	3c:49:37:03:17:db	**EdnetCam**
94:10:3e:34:0c:b5	**WeMoSwitch**	3c:49:37:03:17:f0	**EdnetCam**
1c:5f:2b:aa:fd:4e	D-LinkDoorSensor	**00:17:88:24:76:ff**	HueBridge
1c:5f:2b:aa:fd:4e	D-LinkHomeHub	**00:17:88:24:76:ff**	HueSwitch
94:10:3e:41:c2:05	**WeMoInsightSwitch**	74:da:38:80:7a:08	**EdimaxCam**
94:10:3e:42:80:69	**WeMoInsightSwitch**	74:da:38:80:79:fc	**EdimaxCam**

Algorithm 1 Aggregation Algorithm

$devices_{ij} \leftarrow [][]$ ▷ create an empty two-dimensional array
$seen \leftarrow \emptyset$
$g \leftarrow 12$ ▷ specifying the group size
for each $m \in M$ **do**
  if $m \notin seen$ **then** ▷ detect first-time m is seen
    $seen \leftarrow seen \cup m$
    $i \leftarrow |seen| - 1, j \leftarrow 0$
    $devices_{ij} \leftarrow m$ ▷ create new row in device and assign m as the first element
for $j \leftarrow 0$, **length**$(\hat{Y})$ **do**
  for $i \leftarrow 0$, $|seen|$ **do**
    if $device_{i0} == m_j$ **then**
      $device_i \leftarrow device_i \cup j$ ▷ assign the index of m_j to $device_i$
for $i \leftarrow 0$, $|seen|$ **do**
  $C \leftarrow []$ ▷ will hold the indices of ms divided into chunks
  for $j \leftarrow 1$, **length**$(device_i)-g$, **Step** $= g$ **do**
    $C \leftarrow C \cup device_{i[j:j+g]}$ ▷ divide $device_i$ into chunks of g
  for each $c \in C$ **do**
    $g\_list \leftarrow []$
    for each $j \in c$ **do**3
      $g\_list \leftarrow g\_list \cup \hat{y}_j$ ▷ assign $\hat{y}$ sharing same index with m
    $mode \leftarrow$ **mode**(g_list)
    for j **in** c **do**
      $\hat{y}'_j \leftarrow mode$

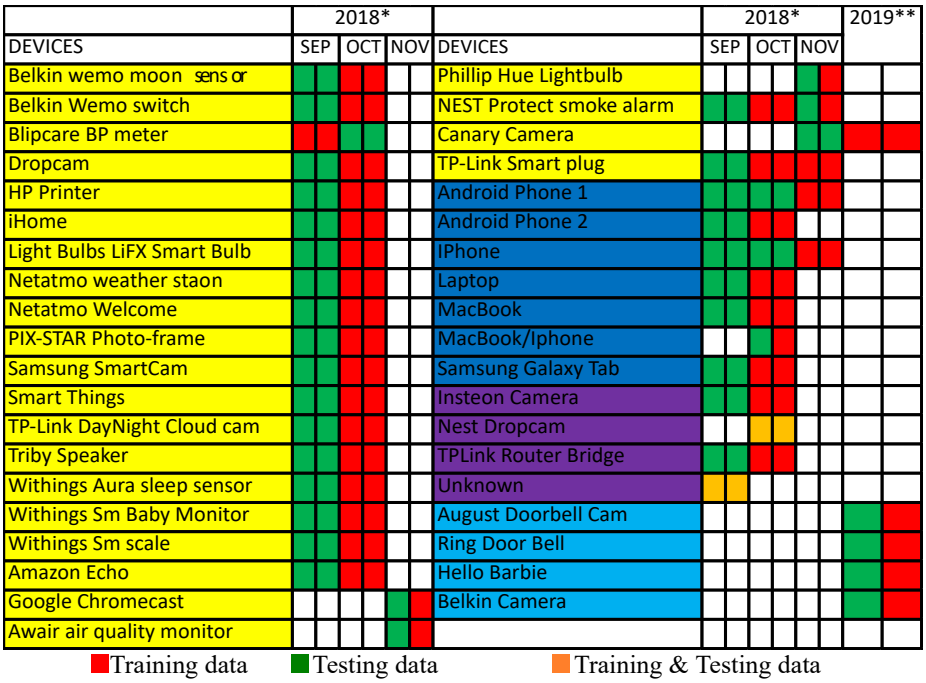

TABLE S2: Which data in the UNSW (*IEEE TMC 2018, **ACM SOSR 2019) dataset were taken from which date range.