Behavioral Fingerprinting of IoT Devices

Conference Paper · October 2018

DOI: 10.1145/3266444.3266452

6 authors, including:

Bruhadeshwar Bezawada
Independent
62 PUBLICATIONS **1,320** CITATIONS
SEE PROFILE

Jordan Peterson
Colorado State University
3 PUBLICATIONS **272** CITATIONS
SEE PROFILE

Hossein Shirazi
Colorado State University
36 PUBLICATIONS **851** CITATIONS
SEE PROFILE

Indrakshi Ray
Colorado State University
326 PUBLICATIONS **6,053** CITATIONS
SEE PROFILE

Behavioral Fingerprinting of IoT Devices

Bruhadeshwar Bezawada, Maalvika Bachani,
Jordan Peterson, Hossein Shirazi, Indrakshi Ray
Computer Science Department, Colorado State University
Fort Collins, Colorado, USA
{bru.bezawada,maalvika.bachani,jordantp,shirazi,
indrakshi.ray}@colostate.edu

Indrajit Ray*
Computer Science Department,
Colorado State University
Fort Collins, Colorado, USA
indrajit.ray@colostate.edu

ABSTRACT

The Internet-of-Things (IoT) has brought in new challenges in *device identification* –what the device is, and *authentication* –is the device the one it claims to be. Traditionally, the authentication problem is solved by means of a cryptographic protocol. However, the computational complexity of cryptographic protocols and/or problems related to key management, render almost all cryptography based authentication protocols impractical for IoT. The problem of device identification is, on the other hand, sadly neglected. Almost always an artificially created identity is softly associated with the device. We believe that device fingerprinting can be used to solve both these problems effectively. In this work, we present a methodology to perform IoT device behavioral fingerprinting that can be employed to undertake strong device identification. A device behavior is approximated using features extracted from the network traffic of the device. These features are used to train a machine learning model that can be used to detect similar device-types. We validate our approach using five-fold cross validation; we report a identification rate of 93-100% and a mean accuracy of 99%, across all our experiments. Furthermore, we show preliminary results for fingerprinting device categories, *i.e.*, identifying different devices having similar functionality.

KEYWORDS

IoT Devices, IoT Network Security, Device Behavior, Device-type Fingerprinting, Machine Learning, Network Traffic Features

ACM Reference Format:
Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray and Indrajit Ray. 2018. Behavioral Fingerprinting of IoT Devices. In *The Second Workshop on Attacks and Solutions in Hardware Security (ASHES'18), October 19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3266444.3266452

1 INTRODUCTION

The Internet-of-Things (IoT) devices industry is rapidly growing [10] with an ever-increasing list of manufacturers offering a myriad of smart devices targeted to enhance the end-users' experience. Unfortunately, security is often an after-thought with manufacturers preferring features and functionality over security. This results in vulnerabilities [24] that can be successfully exploited to launch large scale attacks, best highlighted in the notorious incident of the Mirai botnet [14]. Many security problems, however, can be mitigated through strong identification (and, authentication) of devices. For example, if a malware in a light bulb causes it to behave differently mimicking a thermostat, strong identification would allow an administrator to enforce appropriate security controls, such as, isolate the device from the rest of the network. In this work, we develop a framework to build robust identities of IoT devices using their observed behavioral characteristics. As devices are plugged-in on an IoT network, we establish a behavioral baseline to identify the type of these devices.

Fingerprinting IoT devices is challenging due to the large variety of devices, protocols, and control interfaces across the gamut of devices. An IoT device might respond to queries about its identity and type, which is often a standard way of remotely learning about the device when security is not a major concern. However, when security is indeed a concern, such fingerprint should not be based on a soft identity that is easily cloned, masqueraded or bypassed by advertising relevant false information by an attacker. Examples of such soft identities are device name, device-type, manufacturer information, serial number, network address (even layer 2 address) and so on. More importantly, an untrusted or compromised IoT device might behave contrary to its baseline behavior, *e.g.* by connecting to other devices to disrupt their functioning or to gather network information.

Our hypothesis in this work is that every IoT device possesses a unique "fingerprint" that the IoT device "reveals" when it sends messages over a network. This fingerprint can be remotely established by an automaton that is able to monitor the network traffic. Moreover, this fingerprint cannot be very easily cloned by a malicious adversary. Several researches have shown that many computing devices may possess such fingerprints [1–3, 23, 26].

1.1 Problem Description

An IoT device can be fingerprinted at varying levels of granularity, from a category to a specific instance, as shown in the sample ontology in Figure 1. A device category corresponds to a general grouping of devices having similar functionality, say, *e.g.*, "Light Bulb". This category can have further sub-divisions, like "Monochrome" and "Hue Light". A device-type is a specific device model within a general device category. For instance, in Figure 1, a device-type is: "Light Bulb| Monochrome| TCP Light| A21" or simply "TCP Light | A21". Finally, a device instance is a single instantiation of a device-type. For example, the device-type "TCP| A21", has two different bulbs with serial numbers "A21| S.No: 1" and "A21| S.No: 2". Each of these is a device instance. Ideally, a security administrator

*Also with National Science Foundation.

ASHES'18, October 19, 2018, Toronto, ON, Canada
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5996-2/18/10…$15.00
https://doi.org/10.1145/3266444.3266452

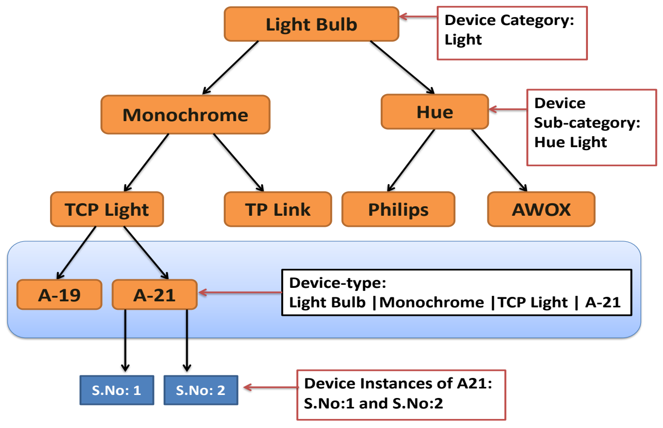

Light Bulb
Device Category: Light
Monochrome
Hue
Device Sub-category: Hue Light
TCP Light
TP Link
Philips
AWOX
A-19
A-21
Device-type: Light Bulb |Monochrome |TCP Light | A-21
S.No: 1
S.No: 2
Device Instances of A21: S.No:1 and S.No:2

Figure 1: Device Category, Type and Instance

would like the capability of establishing strong identities that can differentiate between two device instances, say, "A21| S.No: 1" and "A21| S.No: 2", which are of the same device-type. Towards this goal, device-type identification is a critical first step that we address in this work.

Problem Statement. We define the problem of fingerprinting an IoT device-type as that of identifying the device-type from a sample network activity of the device. We refer to the sample network activity, a, of a device, D_i, as its *fingerprint*, F^i_a. We denote the collection of all F^i_as of a device by $\mathbb{B}_i$, *i.e.*, $\mathbb{B}_i = \{F^i_1, F^i_2, \ldots, F^i_m\}$ where m is the number of observed distinct fingerprints. The collection of all possible network activities of D_i constitutes the behavioral profile, $\langle \mathbb{B}_i, D_i \rangle$, of the device. Then, the problem we address is stated as: Given a collection of previously recorded behavioral profiles:

$$\mathbf{B} = \{\langle \mathbb{B}_1,\ D_1 \rangle,\ \langle \mathbb{B}_2,\ D_2 \rangle,\ \cdots,\ \langle \mathbb{B}_n,\ D_n \rangle\},$$

of n devices and the available fingerprint F_t of a target device D_t, to correctly predict $\langle \mathbb{B}_t,\ D_t \rangle$ where $\mathbb{B}_t \ni F_t$ is the corresponding behavioral profile of D_t.

Key Contributions. (a) For the first time, we describe a practical approach for behavioral fingerprinting of IoT devices using machine learning. (b) Our fingerprinting analysis shows that with a small number of packets we can fingerprint a device-type very accurately. (c) We demonstrate that certain features like TCP window size, entropy and payload lengths are very specific to device-types and are statistically significant in fingerprinting. (d) We demonstrate the machine learning model robustness using three different experiments on 14 different device-types and 7 different device categories. The first experiment examines the identification of device-types based on a five-fold cross-validation and, we report a mean identification rate of 93-99%, and a mean accuracy of 99%. Similarly, the second experiment examined the identification of device-type into its device category and we show a mean identification rate of 91-99%. This result is the first such success reported in this problem domain and shows that we can create behavioral profiles for classes of devices. The final experiment examined identification of different device instances of same device-type, and achieved an excellent mean identification rate of 99.7-100%. (e) Finally, with our existing device set, we show preliminary results that our approach is successful even when the device uses encryption for some of the communication.

2 SYSTEM MODEL AND ASSUMPTIONS

Network Model. Our network model consists of set of IoT devices that are connected through a single gateway router. The devices may be wired or wireless. We assume the capability of capturing all kinds of traffic such as device-to-device, a device to the Internet and from the Internet to a device. We do not make any assumptions on nature of the payload, which may be encrypted, compressed, binary or plain-text. For this work, we focused on home IoT devices.

Threat Model. Our threat model considers the following scenarios:

1. A device might be replaced with another device that is malicious.

2. A compromised device has the ability to spoof soft identities and/or IP/MAC layer addresses and any other information specific to the device-type, such as communication patterns.

3 RELATED WORK

Device fingerprinting has received considerable attention from the research community. General device fingerprinting has been described in [13, 17, 18], which explore several techniques ranging from packet header features to physical features such as clock-skews. Wireless device fingerprinting techniques have been discussed in [1, 5, 7, 15, 21]. These works explore the device-type identification by exploiting the implementation differences of a common protocol such as SIP, across similar devices. Physical layer based device fingerprinting has received considerable attention [3, 4, 11, 23, 27] where the focus is on analyzing the physical aspects of devices to fingerprint them. IoT device-type fingerprinting research is in early stages due to the evolving nature of the IoT industry. All these works focus on general wireless devices and their applicability to IoT devices is an open question. One of the major challenges in this domain is that IoT devices use numerous protocols and it would be nearly impossible to attempt such analysis on a per protocol and per device basis.

Vladimir *et al.* [3] developed a radiometric approach based on imperfections in analog components for fingerprinting network interface cards (NICs). Such variations result in imperfect emissions when compared with the theoretical emissions and manifest in the modulation of the transmitted signals of the device. They used machine learning approaches to perform the fingerprinting. However, this work relies on the availability of the frames, physical layer transmission, from the given device. This may not be feasible for an IoT network as the devices are spread over an area and might be interconnected via different switches and middle-boxes.

François *et al.* [5] describe approaches to fingerprint devices based on the usage of a common protocol. Broadly, this work focuses on distinguishing various implementations of the same protocol. This work describes interesting techniques to parse protocols and provides an approach for representing and analyzing the behavior of a given protocol. However, IoT devices speak a variety of protocols, which makes it difficult to apply these techniques.

François *et al.* [6] describe a protocol grammar based approach for fingerprinting. They characterize a device based on the set of messages transmitted. A message is represented using the protocol grammar syntax. To classify a given device, the messages emitted by the device are compared with syntactic trees of the stored fingerprints and depending on a similarity metric, the device label is assigned. However, this approach is again specific to protocols that are well known and whose grammar rules are available.

Gao *et al.* [9] develop a wavelet analysis technique to fingerprint wireless access points based on frame inter-arrival time deltas. This technique can be seen as a black-box approach. However, this approach relies on the fingerprinter being in or near the range of the access point to gather sensitive time information and needs the access point to route data to the fingerprinter. The approach does not apply to IoT devices as these devices are usually end-points and do not forward data to other devices.

Radhakrishnan *et al.* [23] described GTID for device-type identification on general purpose devices like smartphones, laptops and tablet PCs. Their work relies on the inter-arrival times of different packets to extract the relevant features specific to a particular application like Skype. However, most IoT devices are usually very conservative in terms of traffic generation and do not generate much traffic. Applying these techniques to IoT networks will require non-trivial modifications to the original set of algorithms. In contrast, our work extracts the behavior of an IoT device on the available traffic.

Franklin *et al.* [7] describe a passive fingerprinting method for identifying the different types of 802.11 wireless device driver implementations on clients. The authors explore the statistical relationship of the active channel scanning strategy in a particular device driver implementation. The lack of a standard for the scanning strategy results in observable differences. This technique is useful for identifying the type of device driver implementation but not of the type of device. This is because a manufacturer might reuse the same device driver implementation across several device-types.

In the IoT fingerprinting problem space, IoTSentinel by Miettinen *et al.* [20] and IoTScanner by Siby *et al.* [25] are the currently known solution frameworks. Miettinen *et al.* in [20] describes IoTSentinel, a framework for device fingerprinting and securing IoT networks. It focuses on device-type identification at the time of device registration into a network. This approach uses packet header based features to identify a particular device-type and applies machine learning models to perform the fingerprinting. One shortcoming of this work is that it is susceptible to the two threats outlined in our threat model. This is because packet headers can be easily spoofed. Nonetheless, this work is a useful reference framework. Our approach provides better accuracy and stronger security. IoTSentinel reports a mean identification rate of 50-100%, whereas our approach reports a mean identification rate of 93-99%. Our approach complements IoTSentinel, as our approach can periodically cross-verify the device fingerprints established at registration time.

Siby *et al.* describe IoTScanner in [25], an architecture that passively observes network traffic at the link layer, and analyzes this traffic using frame header information during specific observation time windows. This work is more concerned with discerning the distinct devices and their presence based on the traffic patterns observed during the traffic capture time window. A shortcoming of this approach is that two identical device-types could be classified as two different device-types due to the variations in traffic generated during traffic capture time window. This approach is useful for network mapping at a high level, but performing this analysis periodically can be cumbersome. In contrast, our approach can re-verify a fingerprint of a device with a short signature of only 5 packets.

4 OUR FINGERPRINTING APPROACH

In this section, we describe the building blocks of our behavioral model of an IoT device. We start by briefly describing our proposed approach and discuss the technical challenges we face and how we address them. This helps set the stage for describing next the *static* behavioral model of an IoT device in terms of the protocols used by the given device, followed thereafter for describing the *dynamic* behavioral model of IoT device in terms of the session interactions of the device.

4.1 Proposed Approach, Technical Challenges and Solutions

Our fingerprinting approach generates a behavioral profile that quantifies the behavior of a device-type. Behavioral fingerprinting is quite valuable since it allows us to monitor the device behavior throughout its life time. If there are deviations from the device's initial behavior, due to some malicious activity, we can detect such activity by periodically observing and validating against a behavioral profile. To generate such a behavioral profile, our approach is to model the behavior of the device *approximately* as a collection of protocols used, and the set of observed command and response sequences. We collect the network traffic that is flowing into and out of the device and extract features of interest as indicators of a device behavior. Finally, we aggregate the features using a statistical model and use it as a reference for identifying the device. To identify a target device, we observe a few packets from the device and compare it against the previously recorded behavioral profiles.

The first technical challenge in our proposed approach is to be able to observe all possible protocol interactions and command-responses of the device. Since it may not be possible to observe all possible interactions, our approach can only *approximately* model the device behavior. To solve this challenge, in the laboratory setting, we used the controlling smart-phone app to interact with the device to capture the command-response sequences. Also, we observed the device when the user was not interacting with the device to capture other non-interactive behavior of the device. Our approach is essentially simulating the passive observation of network traffic where the observer could be observing the traffic flows of a new device to build the behavioral profile.

The second challenge is that the types of interesting behavioral features are non-trivial to determine. Therefore, we use available features like packet header feature and payload based features for this purpose. For packet header based features, we extract the device specific features such as the protocols used and communication patterns of the device during the observational period. Our choice of payload features, coupled with some preliminary results, shows that our approach can work on encrypted traffic as well.

The third challenge in our proposed approach is that statistical models can be very difficult to generate on multi-variate data. Towards this, we apply general purpose machine learning tools as machine learning classifiers are very good at learning local features of interest in a collection of data. Since, we are modeling the behavior of a device as a collection of individual fingerprints, the machine learning classifiers are most suitable for our approach. Typically, one distinguishing feature is sufficient to classify a given device-type against several other device-types and machine learning algorithms have shown robustness in such classification.

The final and most important challenge is: *How much of the target device data needs to be observed before the fingerprint can be matched against a stored behavioral profile?* Ideally, a small number of packets allows a fingerprinter to be able to keep track of the devices periodically and observe any deviations from its behavior. Our approach therefore attempts to create a short fingerprint with a few packets being sufficient to match the fingerprint. Therefore, the problem is to determine, on an average, the number of device packets that encapsulate one or more behavioral features of a target device. To solve this problem, we determined the average number of packets based on the assumption that an individual protocol interaction session of a device encapsulates one or more behavioral features of the device. This is consistent with our model for the behavioral profile of a device, which maps the device behavior as collection of protocol interaction sessions. Using experimental analysis, we establish the approximate number of packets required for fingerprinting a device.

4.2 Static Behavioral Model

IoT devices use several protocols, depending on the network layer in context, at various stages of their operation such as ARP, EAPOL, HTTP, MDNS, DNS and so on. Therefore, the list of protocols used by an IoT device is a good indicator of the device behavior. In [20], the authors used this notion to capture the device behavior at registration time. However, many IoT devices use common network protocols and the set of protocols is not necessarily specific to a given device-type. As such, the list of protocols used by IoT device provides only a partial and "*static*" view of the device's operations. As we demonstrate later, the results of Figure 8 strongly emphasize this point. Further modeling is required to completely understand the dynamic nature of the device's behavior, which we describe next.

4.3 Dynamic Behavioral Model

Our modeling of the dynamics of an IoT device is based on the notion that an IoT device has several distinct command-response sequences. We call each of these command response sequences as a *session*. A session can also be viewed as a sequence of packets that have same source/destination IP addresses and source/destination port numbers, in both directions of the communication flow. For instance, consider that a device responds to (or sends) the following types of control messages: $C_1, C_2, \ldots, C_n$ and that the responses for each of these messages are (not necessarily in that order): $R_1, R_2, \ldots, R_m$. A typical protocol interaction can be as follows: $C_1 \rightarrow R_1 \rightarrow C_2 \rightarrow R_3 \rightarrow C_1 \rightarrow R_1$. Therefore, the device's behavior can be viewed as a collection of these sequences.

Any of these sequences, say S_i, corresponds to a valid fingerprint, F_i of the device. But, for fingerprinting, the algorithm can only examine a fixed sized sequence of packets as storing the session lengths of each device-type is impractical. Therefore, the challenge is in arriving at an estimate on the average (preferably, least) number of packets that need to be examined for fingerprinting the device.

One way to estimate this average is by considering the limited scope of IoT devices, which usually have short sessions consisting of 2 to 10 packets. Given this intuition, the average number of packets per session, across five such devices, is given by: $\frac{(2+4+6+8+10)}{5} = 6$ packets. To check this rough theoretical estimate, we used the data from our experiments (Please see Section 6 for experimental setup, devices and data collection details) to count the number of sessions and the packets per session across five sample devices. For this sample set of data, the average number of packets per session is: $\frac{(3.89+6.18+4.41+5.13+9.48)}{5} = 5.81$, which is very close to our theoretical estimate. The summary of this result is that, to fingerprint a given device we need to capture 5 ± 1 packets for any given device. We considered the average of the two bounds of this range, *i.e.*, $\frac{4+6}{2} = 5$, as the *minimum* number of device packets to be examined by the fingerprinting algorithm. Based on the models described in this section, we describe the feature selection for the machine learning models that will be used to create the behavioral profiles of IoT devices.

Table 1: Average Number of Packets Per Session

Device	Total Sessions' Packets	Sessions	Packets/Session
AWOX Speaker	12755	3274	3.89
D-Link Camera	8600	1390	6.18
MUSAIC Speaker	1346	305	4.41
OMNA Camera	8253	1608	5.13
TP Link Light	1660	175	9.48

5 MACHINE LEARNING FEATURES FOR BEHAVIORAL PROFILING

We use two available types of features from the network packets: packet header features and payload based features. Broadly speaking, the packet header features are useful in quantifying the static behavioral model of the device, and the payload based features are useful in quantifying the dynamic behavioral model of the device.

5.1 Packet Header Features

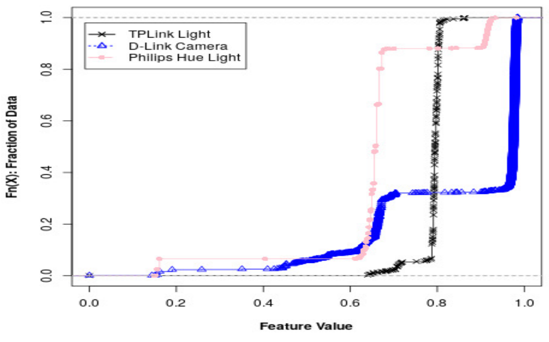
For the static behavioral model, we use a subset of the features, shown in Table 2, from those outlined by Miettinen *et al.* in [20]. Essentially, these features are extracted from the packet headers of the traffic data from the device. These features are binary, *i.e.* they have values of 0 or 1 for the absence or presence of a feature, respectively. Note that, unlike the work in [20], we do not consider network specific features like IP addresses, source or destination ports *etc.*, as these features are not necessarily dependent on the device behavior.

Table 2: Packet Header Features

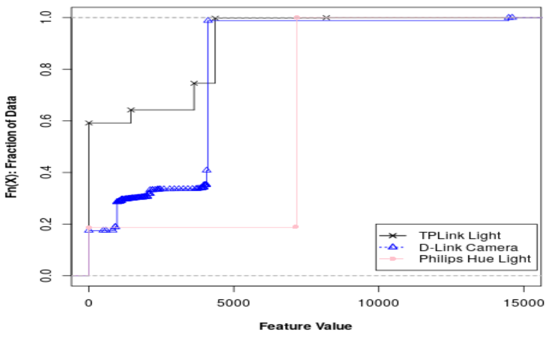
Protocol Layer/Type	Features
Link Layer	ARP
Network	IP/ICMP/ICMPv6/EAPoL
Transport	TCP/UDP
Application	HTTP/HTTPS/DHCP/BOOTP/SSDP/DNS/MDNS/NTP
IP Options	*Padding/Router Alert*

5.2 Payload Based Features

Primarily, we consider the use of three important features: entropy of payload, TCP payload length and TCP window size. To validate

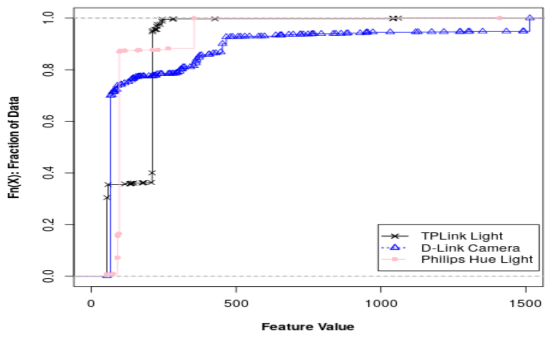


(a) Entropy

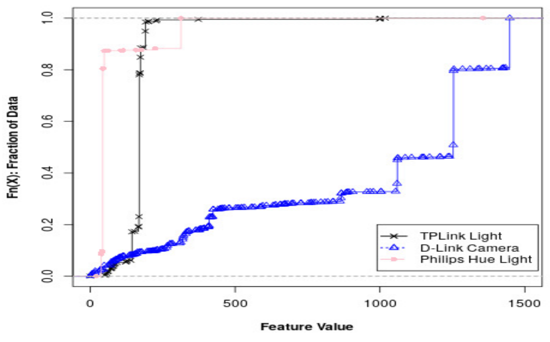


(b) TCP Window Size

Figure 2: ECDF of Payload Based Features



(a) Packet Length



(b) Payload Length

Figure 3: ECDF of Packet Length vs Payload Length

the intuition behind each feature, we tested the empirical cumulative distribution function (ECDF) of the feature for four different types of devices. The ECDF of a real-valued random variable X, or just distribution function of X, evaluated at p, is the probability that X will take a value less than or equal to p. For x-axis distribution, we used the feature values in the dataset, and for y-axis, we used the probability that feature value will take values less than or equal to p and the notation $Fn(X)$ denotes the fraction of data with probability p.

Entropy. The entropy of the payload is indicative of the information content inside a packet, which correlates to message types and sizes. From the analysis described by Khakpour *et al.* in [12], if a packet is carrying plain-text then the entropy of the payload is lowest and, if the packet is carrying audio/XML/JSON encoded or compressed or encrypted data, then the entropy will increase proportionately in that order. To calculate Shannon entropy of a sequence of m bytes with a symbol length of 8-bits or 1 byte, the following formula is used:

$$h_m = -\sum_{i=1}^{256} p_i \log_2 p_i$$

where p_i is the probability of the occurrence of byte value i in the m bytes, *i.e.*, $p_i = \frac{count(i)}{m}$. By using entropy as a feature, we are only focusing on the nature of the data and not on the data itself. We performed a statistical analysis of this feature across a few devices and show the result in Figure 2(a).

TCP Window size. This feature has been suggested by Alvin *et al.* in [18] as method to fingerprint general purpose devices. The intuition behind this feature is that the TCP window size depends on the memory of the IoT device and the speed of its processing. Small constrained devices, such as light bulbs, typically tend to have small window sizes and more powerful devices, such as video cameras, have variable and larger window sizes. Figure 2(b) shows the variation of the TCP window sizes across different categories of devices and show the variability of this feature among these devices. Such variability is the key factor for effective machine learning based classification. In the case that a device uses only UDP, this factor in itself is sufficient to distinguish against devices using TCP.

Payload Length. This is the length of the payload carried inside a TCP/UDP message, in other words, this is indicative of the length of the messages sent by a given device. This is a very device specific feature and shows significant variation from device to device as shown in Figure 3(b). This parameter is different from packet length that has earlier been used [20] and shows a higher variation across devices. We compared this feature with packet length across three sample devices and the results are shown in Figure 3. For instance, in Figure 3(a), 40% of the feature values have same values for all three devices, which does not aid the machine learning classifier for distinguishing among these devices. In comparison, Figure 3(b) shows that the payload lengths exhibit remarkable variations that might be very useful for a machine learning classifier to distinguish among these devices. Now, even if the messages are encrypted, for instance, the video camera feeds, the underlying block-cipher and padding result in deterministic patterns in the command and response message payload lengths. Typically, we observed that most control messages, exchanged by the device from the smart phone or over the local area network, are in plain-text. In various distinct sessions used by a device, the properties of payload are likely to remain uniform for sessions with same functionality, and therefore, are good indicators of the device behavior.

5.3 Behavioral Profile and Fingerprint

Based on above discussion, we now define the structure of a device-type fingerprint. From Section 1.1, the behavioral profile of a device is defined as a collection of various *fingerprints*. Based on the analysis of Section 4.3, the number of messages in a session contribute to the fingerprint of that session, which is 5 ± 1. We choose five packets as the number of session packets whose features correspond to a fingerprint of the device. This implies that any given set of five session packets will represent a fingerprint of the device and should be sufficient to identify the device. For each of the five packets we extract 20 features, *i.e.*, the 17 packet header features and the 3 payload based features, and group them together, to give us a feature vector of 100 features. We consider consecutive packets, *i.e.*, $p_i \rightarrow p_{i+1} \rightarrow p_{i+2} \rightarrow p_{i+3} \rightarrow p_{i+4}$, to generate a single feature vector, as the sequence of the packets is important to capture the session semantics. Specifically, in this work, we only consider the response packets of the device for the fingerprint and not the packets sent to the device[1]. This feature vector represents the fingerprint of the device with respect to the five chosen packets. Now, to create a behavioral profile from the network traffic captured from a device, we group the packets into groups of five and generate the feature vectors. The set of all such feature vectors corresponds to the observed behavioral profile of the device. These feature vectors can then be used to train a machine learning classifier that will be able to predict the device-type when presented with a target feature vector of the same device-type.

6 EXPERIMENTAL EVALUATION

In this section, we describe our experimental setup and the various devices on which the fingerprinting tests were carried out. We report several interesting results with different variations of features.

6.1 Experimental Setup and Data Sets

We tested our approach on the latest home IoT devices, listed in Table 3, available in the market. The *device label* corresponds to the unique identifier given to this device-type. The *category* corresponds to the general category under which one or more device-types are grouped, *e.g.*, AWOX light and iView light are grouped together. The *connectivity* refers to the physical layer connectivity supported by these devices. To enable data capture from these devices, we constructed a software bridge setup using a general purpose laptop running Kali Linux on an Intel® processor with 8 GB RAM. The laptop acted as a WiFi access point and removed the WPA/WPA2 encryption, giving us access to the actual packets sent by the device. This setup allowed us to capture all traffic, from both the wireless and wired devices as well as traffic passing through the network switch from the mobile phone control apps.

To collect the necessary data sets for training the machine learning classifiers, we emulated the normal usage of a device, *i.e.*, the device is in control of a smart phone app and/or the device performs some action without control messages. Our method for data set collection is as follows. First, the device is booted up and allowed to perform any initial configuration or firmware upgrade. Second, when the device is in steady state, we contacted the device through its smart app and started interacting with the device. Finally, we allowed periods of idle time for the device to perform some communication without user intervention. Depending on the device activity, we captured 1000 to 10000 packets of network traffic from each device. The various device operations are described in Table 3. In a real-world environment, our approach passively observes all traffic and generates the corresponding behavioral profile of the device. To generate a single data instance, based on the discussion in Section 4.3 and Section 5.3, we aggregated five consecutive packets into one feature vector. The resulting data instances for each device are shown in Table 3.

Note on Device Diversity and Scale. Although we have experimented on a small set (14) of device-types, we have ensured that these device-types represent a significant spectrum of home IoT device categories. In fact, we demonstrate that it is possible to identify a device-type into its particular category. Also, to ensure that our device-type fingerprinting approach emulates a real-world scenario, *i.e.*, to be able to identify a device-type when the traffic is a vast mix of different device-types, we collected sufficient data samples from each device-type and performed fingerprinting of each device-type using this large data set. Previous approaches like IoTSentinel [20] used a similar technique to ensure sufficient diversity of data.

6.2 Machine Learning Methodology

We used several classifiers available in Scikit-learn tool [22] such as k-nearest-neighbors, Decision trees, Gradient boosting and Majority voting. We describe Gradient boosting here, as this classifier gave consistently good results across all the experiments. Gradient boosting [8, 19] is a gradient descent based learning approach that produces a prediction model as an ensemble of weak prediction models. The learning starts with a "weak" model, typically Gradient Boost Regression Tree (GBRT) that tries to learn the data space and is iteratively improved by the next model that reduces the error of the previous model. The goal of gradient boosting is to combine weak learning models into a single strong model as shown: $F(x) = \sum_{m=1}^{M} \gamma_m h_m(x)$. Typically, h_m is GBRT of fixed depth, which is iteratively improved over M trials and γ_m is the regression parameter for that particular iteration. At each step, the

[1]However, we detected similar results even when the bi-directional traffic was considered for fingerprinting.

Table 3: Device Descriptions, Operations and Data Instances

Device Label: Device	Model	Category	Connectivity	Mode of Operations	Data Instances
1: TCP Light	GL30002-TP	Light	Wi-Fi	Connects through a Hub	1151
2: AWOX Light	SLCW13-14:D4:41	Hue light	Wi-Fi	Connects with a mobile app	2000
3: MUSAIC Music Speaker	MP10	Music Player	Wi-Fi, Ethernet	Connects with a mobile app	1003
4: D-Link Camera	DCS-932L	Camera	Wi-Fi, Ethernet	Connects through laptop	1991
5: iDevice Socket	IDEV0002	Socket	Wi-Fi, Bluetooth	Connects with a mobile app	415
6: iView Light	R60	Hue light	Wi-Fi	Connects with a mobile app	571
7: Lutron Hub	L-BDG2	Hub	Wi-Fi	Connects with a mobile app	108
8: Netatmo Climate	Home Coach	Climate Control	Wi-Fi	Connects with a mobile app	70
9: Omna Camera	DSH-C310	Camera	Wi-Fi	Connects with a mobile app	1072
10: Philips Hue Light	Hue 2.1	Light	Wi-Fi	Connects through hub	986
11: TPLink Light	Lb100	Hue Light	Wi-Fi	Connects with a mobile app	519
12: WEMO Outlet	Insight	Outlet	Wi-Fi	Connects with a mobile app	592
13: Wink Hub	2	Light	Wi-Fi	Connects with a mobile app	286
14: SmartThings Hub	-no model-	Hub	Wi-Fi	Connects with a mobile app	103

model is improved as follows:

$$F_{m+1}(x) = F_m(x) + \gamma_{m+1} h_{m+1}(x)$$

The h_{m+1} is chosen to minimize the loss function L in the current model's fitting of a data point x_i: $F_m(x_i)$ as shown:

$$F_{m+1}(x) = F_m(x) + \underset{h}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, F_m(x_i) + h(x))$$

For implementation we used Scikit-learn library [22] and we set the tool-kit specific parameters as follows: $n\_estimators = 100$, which denotes the number of weak learners, and the maximum depth of each tree is controlled by max_depth parameter. We set the $learning\_rate = 1.0$ and $max\_depth = 1$.

Evaluation Metrics. For testing, we label the device being tested as "1" and the remaining data as "−1". During classification, the "1" data instances correctly classified are denoted by, true positive (TP) and incorrectly classified are denoted by, false negatives (FN), and total "−1" correctly classified are denoted by, true negatives (TN) and incorrectly classified are denoted by, false positive (FP). We report the standard classification metrics such as, positive predictive value, $PPV = \frac{TP}{TP+FP}$, which indicates the ability of the classifier to truly identify the positive instances in a given data set; true positive rate, $TPR = \frac{TP}{TP+FN}$, which indicates the ability of the classifier to correctly identify the device when presented with all positive instances; and, accuracy, $ACC = \frac{TP+FN}{TP+FP+FN+TN}$, which shows the overall performance of the classifier against tested data instances.

We performed all experiments under two variations: (a) In the first variation, we included all the features described in Section 5. (b) In the second variation, to demonstrate the applicability of our approach for cases where data might be encrypted, we performed the experiments without the *"payload entropy"* feature.

6.3 Device-type Fingerprinting

In this experiment, we evaluated the accuracy of the classifier for device-type fingerprinting. For testing against a given device-type, we treated the class label of the device-type to be 1 and the rest of the 13 devices data as −1. Therefore, our learning model created an imbalance in the data wherein the positive labels were less than 10% of the total data set that consisted of 10887 data instances. We used five-fold cross-validation to avoid issues of over-fitting and to test the robustness of the classifier learning on unknown data instance classification.

For this experiment, although we collected data from 14 distinct devices, we could test the approach only on 10 devices due the following reasons. First, for the `Netatmo Climate` device, there were very less data instances as this device generated very less data during experimentation and we excluded fingerprinting this device. But, we still included it for testing against other device-types. Second, the hubs like `Lutron Hub, SmartThings Hub` and `Wink Hub` exhibit a different behavior compared to other devices. These devices act as conduits to other devices, by relaying commands and response, and do not have any specific behavior of their own. However, based on their observed behavior like advertising their presence and connecting to devices, we were able to perform fingerprinting of these devices to a limited extent and show these results as well.

On the remaining 10 device-types, as shown in Figure 4, this experiment achieved an high true positive rate of $99-100\%$ and when *"payload entropy"* was not included we achieved nearly identical results with a variation of ±2% on an average. In Figure 5, we show that our approach achieves an average PPV of over 99%, which shows that the learning model has very good ability to distinguish the tested device-type from among a host of other device-types. This result is very significant given the skewed nature of our data set, most of the times the classifier was correctly able to recognize that a particular data instance did not correspond to the target class label. In real-world networks, this is the most likely situation for fingerprinting and it is essential that the classifier does not generate too many false positives. Finally, in Figure 6, we show that the average accuracy across all the tested devices is consistently above 99%, which shows the quality of the machine learning classifier.

Experiments on Hubs. In Figures 7(a)-7(b), we show the results of fingerprinting the three hubs in our data set. For hubs with some activity, our approach worked quite well, $97-98\%$ TPR, and for other hubs it performed moderately well with a TPR of 86% .

Feature Robustness. Finally, we demonstrate the robustness and importance of the payload features we have used, *i.e.*, *payload size, entropy* and *TCP window size*, in the classification as compared to only packet header features used in [20]. As shown in Figure 8, the results confirm that the fingerprinting accuracy is high for the three features, we have used, showing their robustness and importance when excluding the packet header features.

6.4 Device category Fingerprinting

In this experiment, we explored the capability of our model in classifying devices into device-categories. For this, we generated a data set from the original data set by grouping devices into device-categories, *e.g.*, light bulbs. The data set is shown in Table 4, which

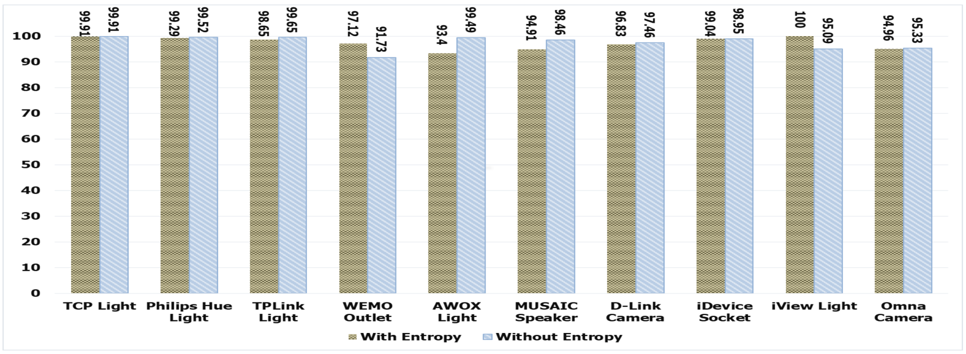


Figure 4: True Positive Rate

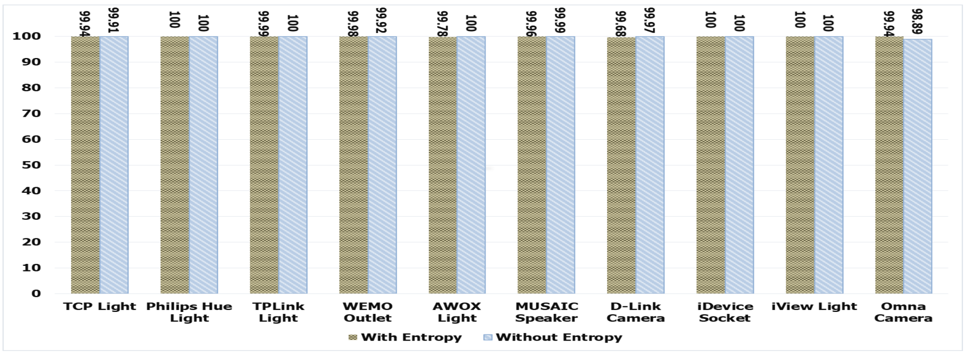


Figure 5: Positive Predictive Value

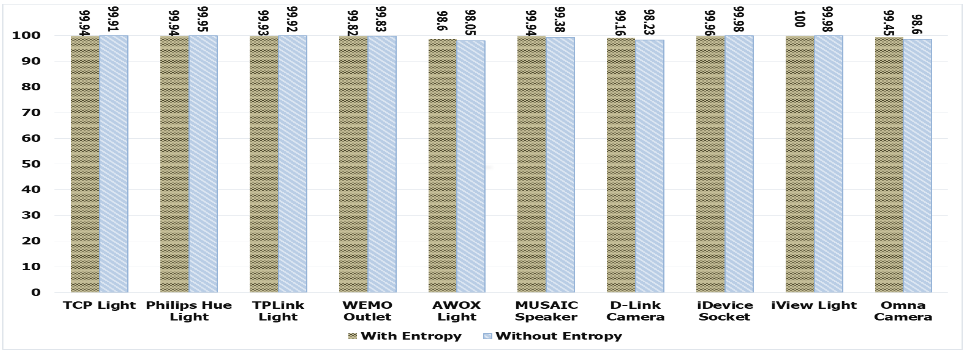
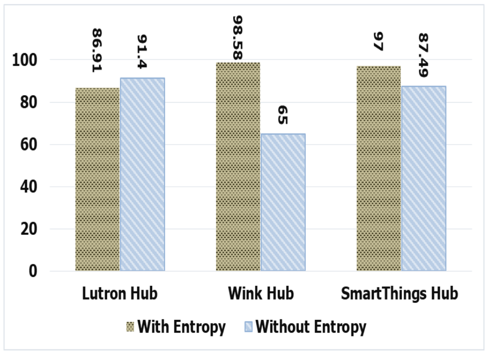
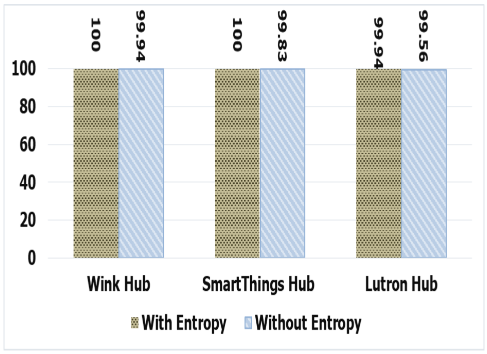


Figure 6: Detection Accuracy

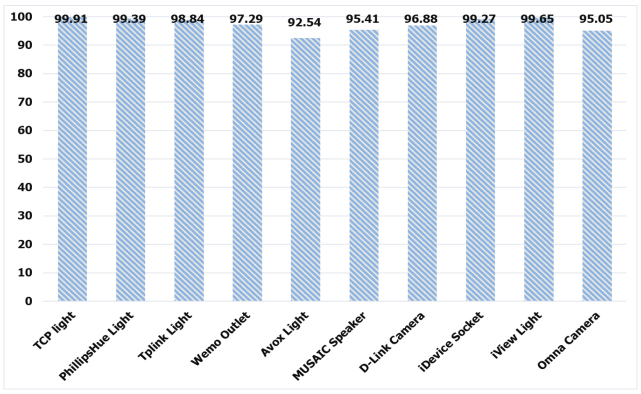


(a)

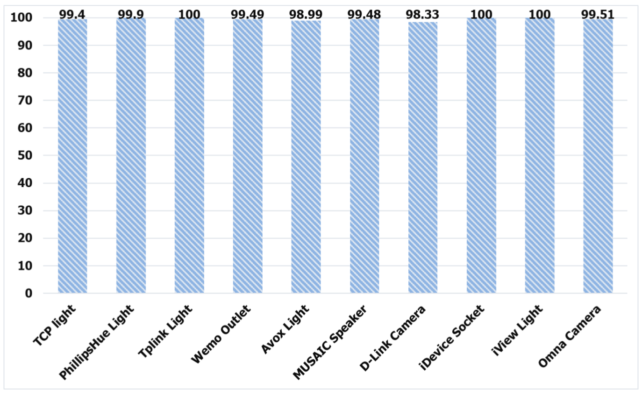


(b)

Figure 7: Fingerprinting Hubs: (a) TPR (b) PPV



(a)



(b)

Figure 8: Feature robustness: (a) TPR (b) PPV

Table 4: Device category Data

Device category	Data Instances
1: Light	2422
2: Music Player	1003
3: Camera	3063
4: Socket	415
5: Hub	210
6: Outlet	592
7: Colored Light	3090

Table 5: Device-instance Data

Device Label: Device	Data Instances
5 :iDevice Socket	415
6: iView Light	571
12: Wemo Outlet	592

consists of categories described in Table 3 (Note: "w/" means all features including *"payload entropy"* and "w/o" is excluding this feature). As shown in Table 6, the device category identification rate (TPR) ranged from 91-99% across the different device-categories with an average accuracy of 97-99%, which shows the robustness of the classifier even when trained against data instances from different device types in same category. This result is very significant as it demonstrates the feasibility of categorizing devices into common device-types. Our work is the first to report this kind of result.

6.5 Cross-instance Recognition

In this experiment, we collected data from different device instances of Wemo Outlet, iDevice Socket and iView light, shown in Table 5. The goal was to check how well the learning model could recognize different device instances. The training set used for this experiment was from Table 3, as would be done in a real-world scenario. The experiments reported a high recognition rate of 99.7-100%, indicating that the classifier was very successful in matching instances against previously stored profiles of the device-type.

6.6 Performance Across Multiple Classifiers

All the above experiments were repeated across multiple machine learning classifiers like: k-nearest-neighbors (kNN), Decision tree and Majority voting. We show the results for the device-type classification on two selected devices: TCP Light and D-Link camera, for all 20 features. Decision trees performed almost as good as Gradient boosting, mainly because Decision trees [16] are good at handling imbalanced data sets while the other classifiers reported lower accuracy. The classifiers reported a TPR ranging from 88-99%, shown in Figure 9, and an average accuracy ranging from 95-99%.

7 CONCLUSION

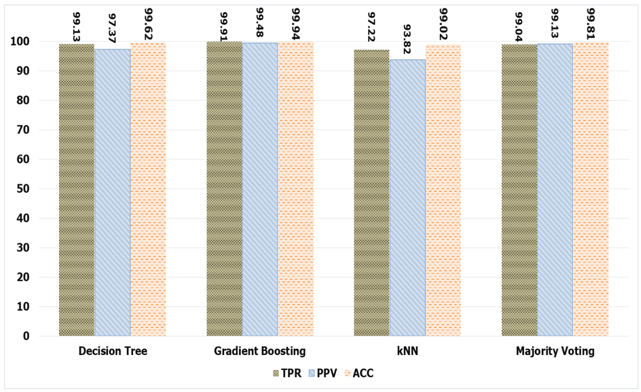
In conclusion, we re-affirm that the problem of IoT device fingerprinting is very important in the context of security. The identification of IoT device-types is a strong step towards identifying IoT device instances, which will be useful in establishing strong authentication of a device. The existing IoT devices have too much variation in protocols/functionality and it is difficult to come out with one general approach for fingerprinting. However, as our methodology showed, it is possible to fingerprint device-types with high accuracy. The high accuracy reported by our experiments show that it is possible to reduce false positives during device fingerprinting even in the presence of several other devices. Fingerprinting categories of devices is an entirely different challenge and we demonstrated some promising results in this direction. Our work is the first to report such cross-category identification of devices. There are many open questions remaining in fingerprinting and this will continue to be an interesting research area for the IoT domain for quite some time. One question is how to leverage device fingerprinting towards the development of strong authentication schemes for IoT devices that are difficult to clone. Such schemes should not require manufacturers to make changes to their device architectures or enforce unreasonable computational overhead on the devices themselves.

ACKNOWLEDGEMENT

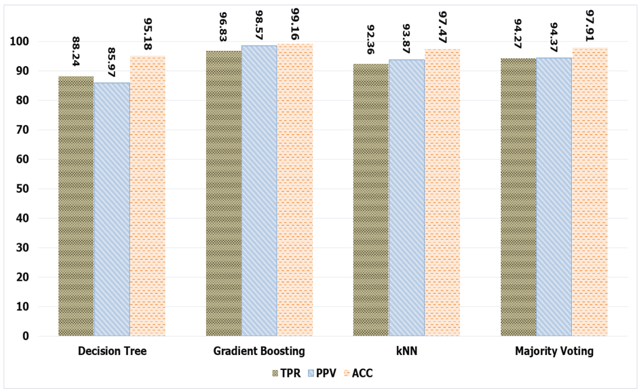
This work is supported in part by funds from NSF under Award No. CNS 1650573, CableLabs, AFRL, Furuno Electric Company, and SecureNok. A gift from CableLabs supported the equipment to carry out this research. In addition, this work is partially supported by the U.S. Department of Energy under award number DE-NE0008571 subcontracted through the Ohio State University. This material is also based upon work performed by Indrajit Ray while serving at the National Science Foundation. Research findings presented here

Table 6: Device Category Classification Metrics

Dataset	TPR		PPV		ACC	
	w/ Entropy	w/o Entropy	w/ Entropy	w/o Entropy	w/ Entropy	w/o Entropy
1: Light	99.30	99.26	99.51	99.26	99.73	99.67
2: Music Player	95.31	94.11	99.57	98.76	99.53	99.34
3: Camera	98.01	97.52	98.79	98.45	99.09	98.86
4: Socket	99.02	99.27	100	100	99.96	99.97
5: Hub	91.36	92.38	98.02	98.12	99.80	99.81
6: Outlet	97.12	96.29	99.66	99.65	99.82	99.78
7: Colored Light	95.50	94.63	98.86	97.16	98.40	97.67



(a)



(b)

Figure 9: Device-type Classification Metrics : (a) TCP Light (b) D-Link Camera

and opinions expressed are solely those of the authors and in no way reflect the opinions the DOE, the NSF or any other federal agencies.

REFERENCES

[1] Chrisil Arackaparambil, Sergey Bratus, Anna Shubina, and David Kotz. 2010. On the Reliability of Wireless Fingerprinting Using Clock Skews. In *Proc. of the Third ACM WiSec.* ACM, New York, NY, USA, 169–174.
[2] Sergey Bratus, Cory Cornelius, David Kotz, and Daniel Peebles. 2008. Active Behavioral Fingerprinting of Wireless Devices. In *Proc. of 1st ACM WiSec (WiSec '08).* ACM, New York, NY, USA, 56–61.
[3] Vladimir Brik, Suman Banerjee, Marco Gruteser, and Sangho Oh. 2008. Wireless device identification with radiometric signatures. In *Proc. of the 14th ACM MOBICOM.* ACM, 116–127.
[4] David Formby, Preethi Srinivasan, Andrew Leonard, Jonathan Rogers, and Raheem A. Beyah. 2016. Who's in Control of Your Control System? Device Fingerprinting for Cyber-Physical Systems. In *23rd Annual ISOC NDSS.*
[5] Jérôme François, Humberto J. Abdelnur, Radu State, and Olivier Festor. 2009. Automated Behavioral Fingerprinting. In *Proc. of the 12th RAID Symposium.* 182–201.
[6] Jérôme François, Humberto J. Abdelnur, Radu State, and Olivier Festor. 2010. Machine Learning Techniques for Passive Network Inventory. *IEEE Trans. Network and Service Management* 7, 4 (2010), 244–257.
[7] Jason Franklin and Damon McCoy. 2006. Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting. In *Proc. of the 15th USENIX Security Symposium, Vancouver, BC, Canada, July 31 - August 4.*
[8] Jerome H Friedman. 2002. Stochastic Gradient Boosting. *Computational Statistics & Data Analysis* 38, 4 (2002), 367–378.
[9] Ke Gao, Cherita Corbett, and Raheem Beyah. 2010. A passive approach to wireless device fingerprinting. In *Proc. of IEEE/IFIP DSN.* IEEE, 383–392.
[10] John Greenough. 2016. How the ?Internet of Things? will impact consumers, businesses, and governments in 2016 and beyond. http://www.businessinsider.com/how-the-internet-of-things-market-will-grow-2014-10?r=DE&IR=T. Last accessed: March 7th, 2018.
[11] Suman Jana and Sneha K Kasera. 2010. On fast and accurate detection of unauthorized wireless access points using clock skews. *IEEE Trans. on Mobile Computing* 9, 3 (2010), 449–462.
[12] Amir R Khakpour and Alex X Liu. 2013. An information-theoretical approach to high-speed flow nature identification. *IEEE/ACM Trans. on Networking* 21, 4 (2013), 1076–1089.
[13] Tadayoshi Kohno, Andre Broido, and Kimberly C. Claffy. 2005. Remote Physical Device Fingerprintin. *IEEE Trans. Dependable and Secure Computing* 2, 2 (2005), 93–108.
[14] Brian Krebs. 2017. Mirai IoT Botnet Co-Authors Plead Guilty ? Krebs on Security. https://krebsonsecurity.com/tag/mirai-botnet/
[15] Andreas Kurtz, Hugo Gascon, Tobias Becker, Konrad Rieck, and Felix Freiling. 2016. Fingerprinting mobile devices using personalized configurations. *Proc. on Privacy Enhancing Technologies* 1 (2016), 4–19.
[16] Rokach Lior. 2014. *Data Mining with Decision Trees: Theory and Applications.* Vol. 81. World Scientific.
[17] Richard Lippmann, David Fried, Keith Piwowarski, and William Streilein. 2003. Passive operating system identification from TCP/IP packet headers. In *Workshop on Data Mining for Computer Security.* 40.
[18] Alvin Martin, George Doddington, Terri Kamm, Mark Ordowski, and Mark Przybocki. 1997. *The DET curve in assessment of detection task performance.* Technical Report. National Inst of Standards and Technology Gaithersburg MD.
[19] Llew Mason, Jonathan Baxter, Peter L Bartlett, and Marcus R Frean. 2000. Boosting Algorithms as Gradient Descent. In *In Proc. of NIPS.* 512–518.
[20] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. 2017. IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. In *Proc. of 37th IEEE ICDCS.* 2177–2184.
[21] Jeffrey Pang, Ben Greenstein, Ramakrishna Gummadi, Srinivasan Seshan, and David Wetherall. 2007. 802.11 user fingerprinting. In *In Proc. of the 13th ACM MOBICOM.* ACM, 99–110.
[22] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12, Oct (2011), 2825–2830.
[23] Sakthi Vignesh Radhakrishnan, A. Selcuk Uluagac, and Raheem A. Beyah. 2015. GTID: A Technique for Physical Device and Device Type Fingerprinting. *IEEE Trans. Dependable and Secure Computing* 12, 5 (2015), 519–532.
[24] Senrio. 2016. 400,000 publicly available IoT devices vul- nerable to single flaw. http://blog.senr.io/blog/400000-publicly-available-iot-devices-vulnerable-to-single-flaw. Last accessed: 7th March 2018.
[25] Sandra Siby, Rajib Ranjan Maiti, and Nils Tippenhauer. 2017. IoTScanner: Detecting and Classifying Privacy Threats in IoT Neighborhoods. *arXiv preprint arXiv:1701.05007* (2017).
[26] A. S. Uluagac, S. V. Radhakrishnan, C. Corbett, A. Baca, and R. Beyah. 2013. A passive technique for fingerprinting wireless devices with Wired-side Observations. In *Proc. of IEEE CNS.* 305–313. https://doi.org/10.1109/CNS.2013.6682720
[27] Tom Van Goethem, Wout Scheepers, Davy Preuveneers, and Wouter Joosen. 2016. Accelerometer-based device fingerprinting for multi-factor mobile authentication. In *Int. Symp. on Engineering Secure Software and Systems.* Springer, 106–121.