WiFi-based IoT Devices Profiling Attack based on Eavesdropping of Encrypted WiFi Traffic

Mnassar Alyami,[1] Ibrahim Alharbi,[1] Cliff Zou,[1] Yan Solihin,[1] and Karl Ackerman[2]

[1]*Dept. of Computer Science*
University of Central Florida
{mnassar.alyami@knights. | ia@knights. | changchun.zou@ | yan.solihin@}ucf.edu

[2] *Sophos*
karl.ackerman@sophos.com

2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC) | 978-1-6654-3161-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/CCNC49033.2022.9700674

***Abstract*—Recent research has shown that in-network observers of WiFi communication (i.e., observers who have joined the WiFi network) can obtain much information regarding the types, user identities, and activities of Internet-of-Things (IoT) devices in the network. What has not been explored is the question of how much information can be inferred by an out-of-network observer who does not have access to the WiFi network. This attack scenario is more realistic and much harder to defend against, thus imposes a real threat to user privacy. In this paper, we investigate privacy leakage derived from an out-of-network traffic eavesdropper on the encrypted WiFi traffic of popular IoT devices. We instrumented a testbed of 12 popular IoT devices and evaluated multiple machine learning methods for fingerprinting and inferring what IoT devices exist in a WiFi network. By only exploiting the WiFi frame header information, we have achieved 95% accuracy in identifying the devices and often their working status. This study demonstrates that information leakage and privacy attack is a real threat for WiFi networks and IoT applications.**

***Index Terms*—Internet of Things, Traffic Analysis, Privacy Attack, Device Fingerprinting; Eavesdropping**

I. Introduction

IEEE 802.11 Wireless network (WiFi) is increasingly connected to a wide variety of Internet-of-Things (IoT) devices, including smart locks, baby monitors, blood pressure monitors, voice assistants, etc. Hence, the security and privacy of the WiFi network are increasingly targeted by attackers. Examples of security attacks on WiFi-connected IoT devices and applications include the Mirai malware that caused distributed denial-of-service (DDoS) [1] and worms in smart bulbs that allowed attackers to control all nearby compatible IoT lights [1]. Privacy attacks are just as concerning. WiFi traffic analysis allows an observer to "fingerprint" devices to infer private user activities, for example by monitoring a camera's bitrate, the adversary can infer object movements inside a building [2]. The privacy and security concerns are sometimes related. For example, the adversary's ability to identify the type of IoT devices in the network allows him/her to infer what vulnerabilities are present to exploit.

In this paper, we focus on privacy concern that arises from the ability of the attacker to fingerprinting IoT devices in the WiFi network. In particular, while there is abundant work in this area [3]–[7], they all rely on an assumption that the attacker is a *network insider*, i.e. the attacker has to either join the WiFi network prior to fingerprinting, or be able to wiretap the Internet-side network link of the WiFi network. The assumption, if true, provides a false sense of security, in that one may conclude that as long as the WiFi network uses good password scheme, device fingerprinting is avoided.

In this paper, we explore a hypothesis whether an *out-of-network attacker* can effectively fingerprint IoT devices *without* joining a WiFi network. In contrast to in-network attack, there are many challenges for an out-of-network attacker: the attacker cannot see any plaintext of packet payload due to WiFi data-link layer encryption, the captured traffic is noisy (if mixed with neighboring networks), and very limited information (e.g., no IP address and port information), hence it is not immediately clear if the hypothesis is valid. However, if the hypothesis is true, the implications are serious. First, the attack is applicable to all WiFi networks that the attacker has close proximity to, instead of only networks with breakable password. Second, the attack does not require any special steps to be performed beforehand; the attacker can just walk or drive to close proximity and start analyzing traffic. Third, the attack is not traceable as it does not leave any footprint detectable by users or forensic examiners. Other implications are discussed in Section VII.

We show in this paper that fingerprinting IoT devices from outside the WiFi network is not only possible, but that it is easy to perform and can yield surprisingly detailed information about devices in the network. Overall, the paper makes the following contributions:

- We demonstrate the feasibility of fingerprinting attack via out-of-network WiFi eavesdropping. We show that it can be performed by practically anybody, without relying on any special equipment; consumer-grade laptops suffice.
- We explore two types of data (times series and summary) and train three machine learning algorithms with them to profile 12 real-world IoT devices working in different states, and report their prediction accuracy results and insights. We show that our best technique can fingerprint device types and device working modes (idle vs. busy) with 95% accuracy on average.
- We discuss the implications of our work and possible defense approaches.

The remainder of the paper is organized as follows. We review related work in Section II. In Section III, we present

the adversary model. Section IV demonstrates our attack system and the data collection of encrypted traffic. Data pre-processing approaches used for our device profiling are introduced in Section V. Then, we evaluate our attack on an experimental testbed in Section VI, including an analysis of the impact of monitoring time on the classification accuracy. In Section VII, we discuss the results, other implications, and potential defenses. Finally, we draw our conclusion and discuss future works in Section VIII.

II. Related Work

Device identification is one of many types of information that can be performed using network traffic classification, and has been a topic of interest from the early stage of the Internet. Studies looking at both WiFi and ethernet traffic showed that traffic classification could accurately identify various information, including IoT device types [4]–[7] and mobile phone app activities [8].

These prior works assumed traffic as seen by an in-network observer that collects the TCP/IP level packets. Such exposure reveals useful and distinguishable network characteristics of the device. For example, the authors in [4] note a single attribute signature for different IoT devices using the destination port number. Additionally, flow volume features such as packet interarrival time are shown to be adequate to conduct device fingerprinting using deep learning algorithms [7]. These methodologies are not applicable by an out-of-network adversary as the IP traffic in this context is encapsulated to the upper layer, encrypting all the influential network characteristics such as port number, protocol, cipher suites, etc. Different from existing works, we utilize a different set of features that are easily extractable even for an observer that is not part of the WiFi network.

The closest related work to our proposed WiFi profiling attack is the work conducted by Acar et al. [3], which used traffic analysis (WiFi, Zigbee, and Bluetooth) to identify devices, their states, and user activities, and presented traffic spoofing as a defense method. However, they assumed a rogue access point with *tcpdump* to collect WiFi traces. Thus, these studies [3]–[7] all assume an *in-network* observer, requiring physical access or extensive knowledge to break in and join the victim's encrypted network. In contrast, our assumption is an *out-of-network* adversary, who simply eavesdrops on WiFi traffic without performing elaborate steps to join or break into the network.

Another related research domain investigates hardware fingerprinting. For example, [9] extracts clock skew measurements to perform hardware fingerprinting (i.e., distinguishing a device among the same class of devices). However, the focus in this area is on the hardware basis rather than the device-specific features, which is unsuitable for our device type classification scenario.

Many studies have conducted WiFi traffic analytics with a variety of goals [9], [10], where the traffic can be collected outside the network using off-the-shelf WiFi monitoring devices. However, none of these studies take into consideration the missing rate reported with off-the-shelf network sniffers [11], or the missing frames resulted from the channel hopping eavesdropping, in which adapters will constantly miss transmitted frames on a specific channel while listening to all fourteen WiFi channels, each one at a time. In comparison, we run experiments to validate our sniffer's performance (See Section IV-A) and allow single-channel monitoring on the targeted network.

Many studies have been proposed to defend against traffic analysis attacks. These countermeasures largely focus on website fingerprinting [12]–[14] or achieving location anonymity [15] and hence are ineffective against device fingerprinting. Traffic reshaping techniques such as traffic morphing and padding [16] reduce classifier accuracy. Still, they will fail with time-based classification as both methods are limited to obfuscate traffic patterns based on packet size. Signal Jamming approaches use antennas to disrupt traffic flow at potential adversary locations by increasing the noise ratio. However, it causes interference and degrades nearby networks' performance and is illegal by law [17]. Thus, our attack remains effective against all the defenses proposed in the studies above.

In brief, different from related works, we explore and demonstrate the more realistic privacy attack to WiFi-based devices relying on out-of-network WiFi traffic monitoring. We implement a practical and accurate proof-of-concept attack assuming a realistic threat model. We report a fast detection of 30 seconds and discuss the implications. We also discuss a potential defense to mitigate the attack (See Section VII).

III. Threat Model and Assumptions

We assume an attacker that passively observes out-of-network WiFi traffic of the victim's WiFi router or access point (AP). To achieve that, the attacker must be physically located within the signal range of the AP. The attacker may be *wardriving* (or warcycling, warwalking, etc.), i.e., searching for WiFi networks from a moving vehicle, while using a listen-only sniffing tool that eavesdrops and collects raw traffic from nearby WiFi networks. The attacker does not have an ability to break into the network or join it. Most WiFi-based IoT devices operate at 2.4 GHz frequency and our study focuses on 2.4GHz, but our findings apply to 5GHz as well.

The goal of the attacker is to infer information of devices in a particular WiFi network, including how many unique devices, which type the devices are (e.g., light bulb, smart TV, laptop, etc.), and to some extent their operating status (idle or busy). The purpose of the attack is to gather important data that reveals potentially sensitive information. For example, the number of devices may reveal the family size, number of employees or customers in a business, etc. The number and types of devices may reveal socioeconomic status. The type of devices may reveal potential hardware/software vulnerabilities of some IoT devices that could be exploited by the attacker later.

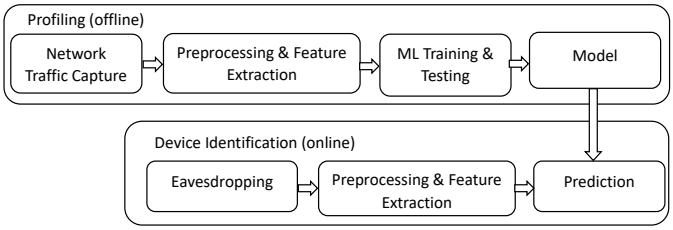
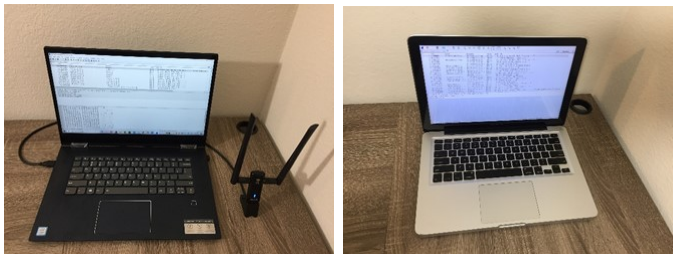


Fig. 1. IoT device profiling attack system.



(a) (b)

Fig. 2. Two ways of setting up out-of-network capturing. (a) Sniffing using Alfa WiFi interface and either Airodump-ng or Kismet software.(b) Sniffing using MacBook built-in WiFi interface and Airtool software.

IV. Capturing of Out-of-Network Encrypted WiFi Traffic

Figure 1 shows the system architecture that we assume the attacker uses. It consists of two stages: the training model for profiling (offline phase) and device identification (online phase). In the offline phase, the attacker sets up many IoT devices connected to a WiFi gateway simultaneously; network traffic is collected using sniffing tools and data is labeled for the device name using the MAC address. Then, we pre-process the data, remove noise (e.g., traffic belonging to other WiFi networks, beacon frames and broadcast frames), and extract useful features into a csv file for supervised learning (Section V). Afterward, we train these features through different machine learning (ML) algorithms and retain the highest accuracy model for online inference (i.e., device identification).

In the online phase, the attacker sniffs network traffic of victims' AP for a short time (such as 30 seconds) and stores the trace for pre-processing, like in the offline phase. As we shall explain in Section IV-B, our pre-processing and data cleaning by no means consider prior knowledge of the devices' information. Our methodology applies statistical and standard filtering to clean noise frames that do not represent data patterns. Then, we extract the features out of the pre-processed file using a python script. Finally, we utilize the stored model to predict each device's type and its working activity.

We will now discuss each step in the attack.

A. Out-of-Network WiFi Traffic Capturing

To capture frames, we first tested out several popular sniffing tools: Airodump-ng[1] and Kismet[2]. Airodump-ng and Kismet sniff raw 802.11 frames. Both are capable single-channel or multi-channel monitoring via frequency hopping between all channels. The main difference between the two is that Airodump-ng dumps the capture into a capture file format (such as pcap[3]), whereas Kismet stores the trace as an SQLite3 database. Airodump-ng's pcap output is in a compatible format to perform packet inspection using a network analyzer like Wireshark[4]. For the hardware, we used an external wireless adapter (Alfa AWUS036ACM) in the monitor mode, as shown in Figure 2.a, because most computer built-in WiFi cards are programmed to receive only packets addressed to the machine's interface card or a broadcast.

After analyzing the captured traffic, we observed a significant proportion of captured packets with small sizes, which contradicts the elephant-mouse internet traffic phenomenon [18]. The elephant flows of 1,500 bytes were never seen for all devices, including video packets of smart TV and cameras. Our investigation shows that both tools are limited to catch a limited range of packet sizes up to 472 bytes, which is sufficient for specific signal intelligence applications. For example, authors in [10] suggested rogue AP detection using Kismet by utilizing a few captured packets of any size to identify rogue AP based on the received signal strength because it differs from a legitimate AP.

Due to the limitations of Airodump-ng and Kismet, we evaluate a third tool called Airtool[5]. Airtool is a free WiFi traffic sniffer using Mac's built-in network interface card (NIC) (See Figure 2.b). It can be used to passively sniff WiFi traffic and store the traces in a pcap format for further analysis using Wireshark. To verify the completeness of traffic captured by Airtool running on an out-of-network MacBook, we simultaneously run another in-network laptop's Wireshark to record its own incoming/outgoing network traffic to the AP. Then we compared the two traces focusing on the traffic between the second in-network laptop and the AP.

As illustrated in Table I, we found that Airtool collects more

[1]http://aircrack-ng.org/doku.php?id=airodump-ng
[2]https://www.kismetwireless.net
[3]https://en.wikipedia.org/wiki/Pcap
[4]https://www.wireshark.org/
[5]https://www.intuitibits.com/products/airtool

TABLE I
Airtool testing: comparison of passive out-of-network Airtool capture with in-network Wireshark capture.

Packets/Frames Size Range	Wireshark	Airtool	
	#Packets	#Frames	#Data frames
0-19	0	2428	0
20-39	0	5593	199
40-79	2441	0	0
80-159	260	2890	2883
160-319	108	239	239
320-639	173	194	190
640-1279	241	255	255
1280-2559	13574	13846	13846
Total	16797	25445	17612

frames than packets captured by Wireshark because it captures additional control and management frames at the data-link layer (most frames have sizes between 0 to 39), which do not appear in Wireshark's in-network traffic capturing. Airtool interprets each captured frame as a WiFi data-link layer frame, while in Wireshark, each WiFi frame is interpreted as an Ethernet II frame. Thus, for the same WiFi packet, the Wireshark capture has a smaller size than the data-link layer frame interpretation captured by Airtool. This is the reason why 2441 packets in the size range of 40-79 in Wireshark capture all appear in the upper packet size range (80-159) in the Airtool capture. Furthermore, after filtering out all control and management frames in the Airtool capture (shown in the last column), the comparison still holds, and therefore there are no noticeable missing packets by Airtool. Thus, we use Airtool for our evaluation testbed.

B. Pre-Processing of Captured WiFi Traffic

After the Airtool software captures the encrypted WiFi traffic, the resulting traces in pcap file format are analyzed using Wireshark. More specifically, the raw trace data is pre-processed using the following steps:

1) We extract the bidirectional flows associated with the MAC address of the WiFi router under investigation. This is needed since Airtool could capture WiFi traffic from multiple APs in the neighborhood. Only data frame types are kept because all other control and management MAC-layer frames do not represent the profiling data pattern. To filter out non-data frames, we use the following display filter in Wireshark:[6]
   `< (wlan.sa = = "Router's MAC" || wlan.da = = "Router's MAC") && wlan.fc.type = = 2>`
2) We export the pcap files as csv for steps 3 and 4.
3) We remove noise frames that were generated by some MACs. These noise frames are easy to filter out since they mostly appeared with a single frame. They are filtered out by keeping only traffic frames that have bi-direction communication traffic (i.e., having both send and receive frames).
4) We replace the MAC addresses with the corresponding device names and their working status to facilitate dataset labeling. This step is only performed in the offline training phase. We skip this step in the online attack phase. The label is used for classification training data and testing verification purposes.
5) We use a Python script to extract and calculate statistical features we will discuss in Section V-E and finally obtain the needed dataset for both offline training and performance testing.

[6](wlan.sa) and (wlan.da) keywords filter by the source and destination MAC address respectively, and (wlan.fc.type = = 2) filters to keep all data type frames by removing WiFi control and management frames.

V. Data Processing and Profiling Based on Machine Learning

In this section, we first discuss what data fields we can observe and utilize in out-of-network monitoring. Next, we develop and demonstrate two data processing approaches to generate representative data suitable to feed into machine learning (ML) classification: Time-series and Summary data.

A. Observable Data Fields in Out-of-Network Monitoring

For out-of-network monitoring on a secured WiFi network, everything at and above the data-link layer is encrypted by a WiFi protocol (e.g., WPA-PSK). The only observable information is the MAC-layer frame header, plus the frame observation timestamp and signal strength. The MAC-layer frame header has the following useful attributes: source MAC address, destination MAC address, frame type, and frame size.

Initially, we thought that the signal strength might be a good attribute to utilize, which could reflect device hardware property and its distance to the AP (e.g., mobile or stationary). However, from our experiments, we found that the signal strength is affected by many unpredictable factors such as neighboring WiFi networks, and by the reflections, absorption, and deflection of the surrounding objects and rooms, etc. For this reason, we do not consider signal strength in our device profiling.

B. Preliminary Data Analysis

To provide insights into the IoT traffic, we analyzed a captured traffic seen over 5 minutes for three IoT devices chosen for illustrative purposes: A smart light bulb, a smart plug, and a WiFi-based printer. We changed their working mode in the third minute to monitor their behavioral change. In other words, in the middle of our observation, we turned the light bulb and the plug off, and stopped printing, so the printer would be idle.

The header-based features are either a (a) flow-related or (b) volume-related feature that can be observed within some time window. Flow-based features refer to the frequency and duration of transmission, whereas volume-based features refer to the traffic size in bytes. Although it is limited information, it is sufficient to yield signatures as we can observe unique statistical differences among various types of devices. For example, we can easily notice in Figure 3 a clear distinction for the printer in terms of the number of received packets, unlike the bulb and the plug. From a different angle, both the bulb and plug exhibit distinct behavior in terms of packets sizes. Specifically, both devices seem to send most packets at a unique size, 82 bytes for the light bulb and 92 bytes for the plug. Figure 4 shows the word cloud of top packet sizes initiated from the two devices. Hence, we believe the adversary can build signatures with a set of efficient statistics.

The adversary can also infer the event of working status change. We captured a little spike of flow in the data as seen in Figure 3 with all devices because the AP initiated a communication to send an off signal to the bulb and plug and to stop the printer. Such behavior is a transient network

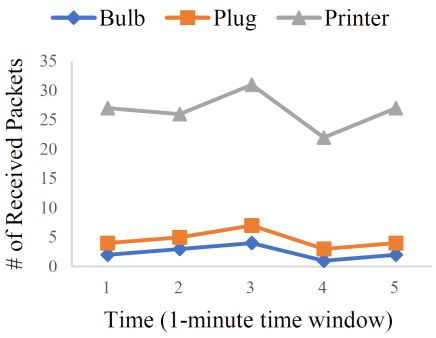


Fig. 3. Traffic flow of three devices.

behavior that can only be observed momentarily from the traffic flow (in part of seconds), not from statistical features based on some time window (+20 seconds). Therefore, it is difficult to infer the operational mode of the devices that are exhibiting a minimal traffic change. On the other hand, the working status of the other devices with higher network capability and memory storage such as Alexa, smart TV, and WiFi cameras, are remarkably observable due to the significant drop in the flow when transitioning to the idle state. For instance, Alexa will receive very few packets when it is idle as it is not receiving voice commands from the user to search the Internet (e.g., checking the weather or deliveries).

C. Machine Learning Algorithms

We choose several ML models, including Random Forest (RF), Support Vector Machine (SVM), and Naïve Bayes (NB) for learning and inference. RF was reported in [19] to provide superior performance on a network traffic classification problem among 11 ML models. Additionally, we chose SVM and NB due to the two sequence sizes in our time-series dataset. SVM has a good performance on multidimensional datasets as appeared with IoT devices that have intensive traffic. On the other hand, NB excels in relatively small data, suitable for devices that generate an infrequent and small amount of network traffic [20].

In our IoT device classification, we always consider one additional class called 'unknown', which contains all devices that the classifier cannot determine their classes with a predefined confidence.

D. Device Profiling based on Time-series Data

Time-series data processing is straightforward. We transform the monitored data-link layer trace into a series of three-feature entries. Each monitored data frame is transformed into three numeric values, including the inter-arrival time T (i.e., the time difference between two consecutive packets), direction D (i.e., 0 represents sent and 1 for received packets by an IoT device), and the packet size S. Assume we create a sequence from N frames. Then we can obtain the series $\{T_0, D_0, S_0\}, \{T_1, D_1, S_1\}, \ldots, \{T_N, D_N, S_N\}$.

The main drawback of this approach is that it requires sufficient amounts of packets to create each data point for machine learning training or classification. Since we deal with a heterogeneous system monitored in a fixed time window, some devices initiate a considerable amount of data communication (such as security cameras), and others generate very few packets (such as smart plugs). For example, if we consider collecting a 100-packet series from the plug and camera, we need over 30 minutes of capturing for the plug while a single second of monitored data is enough for the camera. To balance between the two groups, we adopt a two-level classification strategy starting with a traffic intensity threshold. The first level splits devices into two groups based on traffic density, whether high or low. Then, we accordingly utilize an appropriate sequence size that aligns with the device traffic intensity. The second level calculates the prediction probability using the ML algorithms. If the probability is above a specific threshold, it gives the prediction; otherwise, the instance will be classified as 'unknown' device.

E. Device Profiling based on Summary Data

In this approach, we profile IoT devices with various traffic features observed over a specific time window. As shown in Figure 5, we divide the trace of n seconds into a fixed window size of W seconds, such that we start with a time window $[t_{start} \ldots t_{end}]$, and recursively increment both ends by W as long as $t_{end} + W \leq n$. Hence, for n seconds of monitored data, we can obtain $\frac{n}{W}$ data points.

To efficiently generate more sample data points, we use a sliding window $s = \frac{W}{2}$ in which we increment the window $[t_{start} \ldots t_{end}]$ by s seconds instead of W seconds, which yields $(\frac{n}{W} \times 2) - 1$ data points. Each device is then identified by its MAC address for each time window W for feature extraction and labeling.

After creating our dataset, we build three classifiers, using the three mentioned ML algorithms (SVM, NB, and RF). In

(a) Smart light bulb. (b) Smart plug.

Fig. 4. Word cloud of sent packet sizes from two devices.

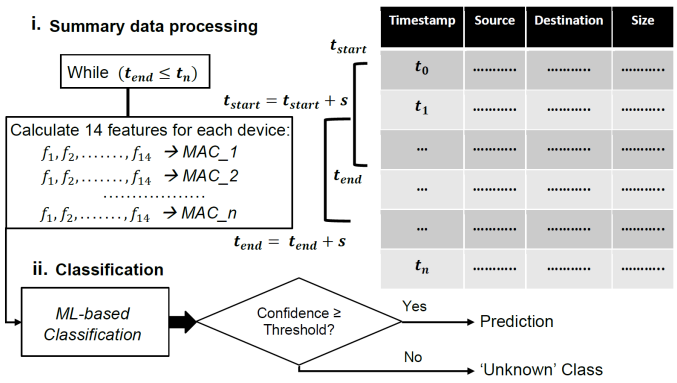


Fig. 5. Summary data processing and classification.

the following, we list all 14 extracted features from monitored packets within each time window:

1) The number of packets sent from the device to AP.
2) The number of packets received by the device from AP.
3) The variance of inter-arrival time.
4) The average number of consecutively sent packets before seeing a received packet.
5) The average number of consecutively received packets before seeing a sent packet.
6) Total number of bytes in sent packets.
7) Total number of bytes in received packets.
8) Number of different sizes in sent packets.
9) Number of different sizes in received packets.
10) Maximum packet size.
11) Mode of sent packet lengths (i.e., the packet size that appeared most in the monitoring window).
12) Mode of received packet lengths.
13) The variance of sent packet size distribution.
14) The variance of received packet size distribution.

VI. Evaluation

In this section, we first discuss our testbed to collect the dataset and our evaluation metrics. Then, we present our evaluation results on the testbed trace.

A. Testbed Setup and Evaluation Metrics

We set up a testbed with 10 different IoT devices and 2 non-IoT (a smartphone and laptop) devices, and allow them to connect to the Internet via a WiFi router. To collect ground truth data, we capture the trace for all devices for a sufficient monitoring time (1 hour). We list in Table II our devices along with their operational modes for the capture setup of our dataset. These working scenarios are by no means exhaustive of devices that have unlimited functionalities, but they encompass various common usages. Using the captured raw packets, we construct two datasets (time-series and summary data formats as explained in the previous section) and randomly split each dataset's instances into two groups, approximately 75% of the instances for training and 25% for testing.

TABLE II
IoT dataset capture setup.

Device	15 min	15 min	30 min
Laptop	Browsing	Streaming	Idle
iPhone	Social Media		
TV	Internet Television App		
TV fire stick	Streaming		
Amazon Echo	Receive a query/control command regularly	Play media (e.g., Music)	
Google Home			
Printer	Occasionally Printing		
Bulb	ON		OFF
Plug			
Baby Monitor			
Doorbell			
Camera			

We evaluate our classification models using the following aspects: Accuracy, Precision, Recall and F1 Score. Let us denote true prediction as T, broken further into true positives TP and true negatives TN. Likewise, false prediction is denoted as F, broken into false positives FP and false negatives FN. Accuracy is measured as $\frac{T}{T+N}$, Precision is measured as $\frac{TP}{TP+FP}$, Recall is measured as $\frac{TP}{TP+FN}$, and F1 is measured as $2 \times \frac{Precision \times Recall}{Precision+Recall}$.

B. Model Accuracy Results

Table III shows the accuracies of various ML models with the time-series data vs. summary data. In this testing, we set time window size of 30 seconds. The non-IoT devices include a laptop and iPhone in our testing. For all cases, the results show that using the summary data achieves much higher accuracies than using the time-series data. In addition, RF model achieves the best prediction accuracies than the other two ML models.

We think that time-series patterns can be better learned from long-term observed trace more than short-term, which requires significantly longer time observation (e.g., +30 minutes) to perform the attack. However, such a long time of eavesdropping is an unrealistic attack scenario.

As shown in Table III, the summary data profiling yields more accurate results than time-series data, across all models and all types of devices. We think that the summary data approach yields better results because its features are more useful for profiling heterogeneous devices. For instance, a TV sends a regular number of packets per time window, in contrast to Google Home traffic that varies over time. This is not easily captured in the time-series data but is distinguishable from the packet size perspective in the summary data.

Furthermore, the table shows that RF outperforms the other two algorithms in all cases with summary data. We further plot all evaluation metrics on Figure 6. The figure confirms RF's superiority: it outperforms SVM and NB in all metrics, and achieves at least 95% in all metrics.

C. Working Status Detection and Detection Speed

We further investigate if our RF model can detect the status of various devices that have two working modes (busy vs. idle). The accuracy results are presented in Figure 8. The figure shows that the classification accuracies are above 90% with a few exceptions: iPhone (idle state) and Amazon Echo (Active and Idle states). The worst accuracy occurs for Amazon Echo, where 28.1% of Amazon Echo in the busy state is incorrectly classified as an Amazon Echo in idle state.

TABLE III
Comparing the accuracies of ML models using time-series vs. summary data.

		SVM	NB	RF
Time Series	Non-IoT	0.34	0.25	**0.41**
	IoT	0.57	**0.74**	0.68
Summary Data	Non-IoT	0.51	0.41	**0.94**
	IoT	0.65	0.77	**0.96**

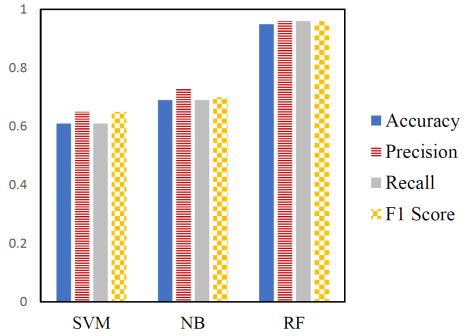


Fig. 6. Overall performance of summary data-based profiling.

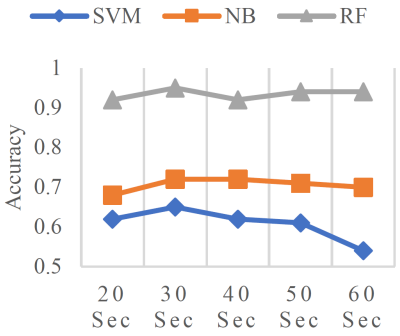


Fig. 7. Impact of time window size on accuracy.

The reason behind this misclassification is that it takes a few seconds to execute some commands (e.g., turning the light on), hence the generated traffic is very small, making the traffic pattern appears similar to the idle state. As explained in Section V-B, we found that some devices like the printer, plug, doorbell, smart bulb, and baby monitor do not have distinct traffic statistics when changing their working states, thus in our experiments, each of these IoT devices has only one model without distinguishing its working state.

We also analyzed the impact of the time window size on the accuracy (See Figure 7), by varying it from 20 seconds to 60 seconds. We observed an increase in accuracy by 3% when moving to 30 seconds with RF, but the accuracy is flat from 30 to 60 seconds. Therefore, we adopted the 30-second time window as the default.

VII. Discussion

Our results have validated the hypothesis that an out-of-network attacker can effectively fingerprint IoT devices without joining a WiFi network. Features such as number of packets, inter-arrival time, packet sizes, and their distributions, are sufficient in fingerprinting the types of devices and in most cases, their working modes. The attack is easy to carry out (no need to join or break into a network, and no special equipment is required), does not take a long time to perform (30 seconds provides 95%+ accuracy), and does not leave any detectable footprints.

Implications: The attack raises substantial privacy concerns. First, it enables covert business surveillance. An adversary can drive near the business area to infer the level of economic activity, clients' socioeconomic groups, estimated revenues, revenue trends, or even discover potential vulnerable devices to target in further attacks. In more complex settings, it can provide environmental awareness that can be used to track mobile devices (e.g., cars, drones, phones, etc.). For instance, a swarm of drones can be deployed over a large area to classify and identify signal-emitting devices in the covered region and track their movement and interactions. This may reveal how devices interact with each other, and reveal the geographic movement of each device.

Does out-of-network profiling attack scale? Our proof-of-concept attack results are accurate on our experimental testbed. However, a key question we have not investigated is, can our attack results be extended to the actual open world? We plan further investigation in this direction as future work.

Potential Defenses: A robust defense against device fingerprinting at the *data-link* layer protection is expensive to implement. For example, encrypting MAC header information may conceal device identities hence reduce the attack success probability [21]. However, it does not fully eliminate the attack and the resulting key management and encryption overheads may make it impractical. Other defenses at the *network* and *application* layers are ineffective at the data-link layer [12]–[14].

Another possible defense is obfuscation via virtual identifiers, e.g., virtual wireless clients (VWC) [22], [23] that generate multiple virtual NICs for each physical NIC. With VWC, a WiFi device's traffic will be dispersed among multiple virtual clients either randomly or using some rules. A large enough number of virtual clients may provide sufficient entropy to prevent an attacker to map different MACs to a single WiFi device for profiling.

VIII. Conclusion And Future Work

This paper presents a new privacy attack that an out-of-network eavesdropper can use to identify various IoT devices and, in most cases, infer their working modes, within a 30 second time window. We demonstrated this attack's feasibility on a testbed of 10 IoT and 2 non-IoT devices with a high accuracy of 95+%. For future work, we plan to investigate if additional modes (e.g., streaming, chatting, printing) and fingerprinting of apps can be achieved by an out-of-network observer.

Acknowledgment

This work is supported by the grant from Sophos Inc. and National Science Foundation under grant DGE-1915780.

References

[1] Yinxin Wan, Kuai Xu, Guoliang Xue, and Feng Wang. Iotargos: A multi-layer security monitoring system for internet-of-things in smart homes. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 874–883. IEEE, 2020.

[2] Jinyang Li, Zhenyu Li, Gareth Tyson, and Gaogang Xie. Your privilege gives your privacy away: An analysis of a home security camera service. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 387–396. IEEE, 2020.

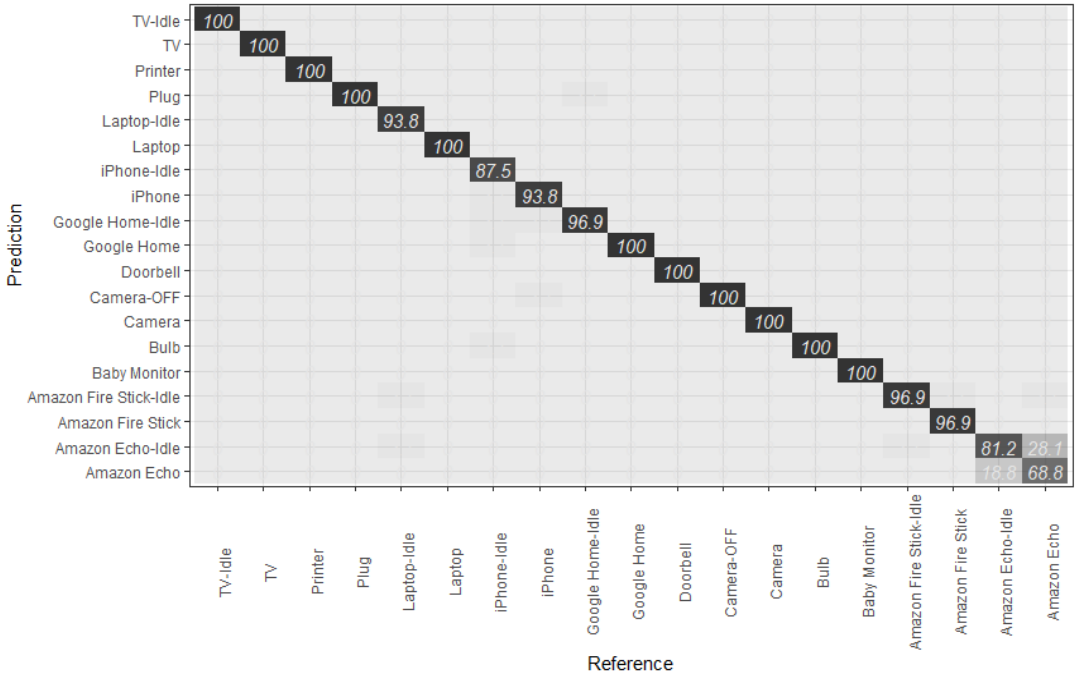


Fig. 8. Confusion matrix of our IoT devices classification based on summary dataset using RF machine learning algorithm.

[3] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and Selcuk Uluagac. Peek-a-boo: I see your smart home activities, even encrypted! In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 207–218, 2020.

[4] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying iot devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, 2018.

[5] Deepak Kumar, Kelly Shen, Benton Case, Deepali Garg, Galina Alperovich, Dmitry Kuznetsov, Rajarshi Gupta, and Zakir Durumeric. All things considered: an analysis of iot devices on home networks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 1169–1185, 2019.

[6] Mustafizur R Shahid, Gregory Blanc, Zonghua Zhang, and Hervé Debar. Iot devices recognition through network traffic analysis. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 5187–5192. IEEE, 2018.

[7] Sandhya Aneja, Nagender Aneja, and Md Shohidul Islam. Iot device fingerprint using deep learning. In *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, pages 174–179. IEEE, 2018.

[8] John S Atkinson. *Your WiFi is leaking: inferring private user information despite encryption*. PhD thesis, UCL (University College London), 2015.

[9] Tadayoshi Kohno, Andre Broido, and Kimberly C Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, 2005.

[10] BR Chandavarkar et al. Detecting rogue access points using kismet. In *2015 International Conference on Communications and Signal Processing (ICCSP)*, pages 0172–0175. IEEE, 2015.

[11] Pablo Serrano, Michael Zink, and Jim Kurose. Assessing the fidelity of cots 802.11 sniffers. In *IEEE INFOCOM 2009*, pages 1089–1097. IEEE, 2009.

[12] Ahmed Abusnaina, Rhongho Jang, Aminollah Khormali, DaeHun Nyang, and David Mohaisen. Dfd: Adversarial learning-based approach to defend against website fingerprinting. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 2459–2468. IEEE, 2020.

[13] Saman Feghhi and Douglas J Leith. An efficient web traffic defence against timing-analysis attacks. *IEEE Transactions on Information Forensics and Security*, 14(2):525–540, 2018.

[14] Wladimir De la Cadena, Asya Mitseva, Jens Hiller, Jan Pennekamp, Sebastian Reuter, Julian Filter, Thomas Engel, Klaus Wehrle, and Andriy Panchenko. Trafficsliver: Fighting website fingerprinting attacks with traffic splitting. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1971–1985, 2020.

[15] Jianqing Liu, Chi Zhang, and Yuguang Fang. Epic: A differential privacy framework to defend smart homes against internet traffic analysis. *IEEE Internet of Things Journal*, 5(2):1206–1217, 2018.

[16] Fan Zhang, Wenbo He, and Xue Liu. Defending against traffic analysis in wireless networks through traffic reshaping. In *2011 31st International Conference on Distributed Computing Systems*, pages 593–602. IEEE, 2011.

[17] Jammer enforcement. https://www.fcc.gov/general/jammer-enforcement/. Accessed: 2020-11-24.

[18] Kun-Chan Lan and John Heidemann. On the correlation of internet flow characteristics. Technical report, Citeseer, 2003.

[19] Sina Fathi-Kazerooni, Yagiz Kaymak, and Roberto Rojas-Cessa. Identification of user application by an external eavesdropper using machine learning analysis on network traffic. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2019.

[20] FY Osisanwo, JET Akinsola, O Awodele, JO Hinmikaiye, O Olakanmi, and J Akinjobi. Supervised machine learning algorithms: classification and comparison. *International Journal of Computer Trends and Technology (IJCTT)*, 48(3):128–138, 2017.

[21] Lior Khermosh, Zachy Haramaty, and Jeff Mandin. Implementing ieee 802.1 ae and 802.1 af security in epon (1gepon and 10gepon) networks, March 12 2013. US Patent 8,397,064.

[22] Omar Nakhila and Cliff Zou. Parallel active dictionary attack on ieee 802.11 enterprise networks. In *MILCOM 2016-2016 IEEE Military Communications Conference*, pages 265–270. IEEE, 2016.

[23] Omar Nakhila and Cliff Zou. Circumvent traffic shaping using virtual wireless clients in ieee 802.11 wireless local area network. In *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*, pages 52–56. IEEE, 2017.