**Automated IoT Device Fingerprinting Through Encrypted Stream Classification**

Jianhua Sun1, Kun Sun2, and Chris Shenefiel3

1 College of William and Mary, Williamsburg, USA

[jianhua@cs.wm.edu](mailto:jianhua@cs.wm.edu)

2 George Mason University, Fairfax, USA

[ksun3@gmu.edu](mailto:ksun3@gmu.edu)

3 Cisco Systems, Inc., Raleigh, USA

[cshenefi@cisco.com](mailto:cshenefi@cisco.com)

**Abstract.** The explosive growth of the Internet of Things (IoT) has en- abled a wide range of new applications and services. Meanwhile, the mas- sive scale and enormous heterogeneity (e.g., in device vendors and types) of IoT raise challenges in efficient network/device management, applica- tion QoS-aware provisioning, and security and privacy. Automated and accurate IoT device fingerprinting is a prerequisite step for realizing se- cure, reliable, and high-quality IoT applications. In this paper, we pro- pose a novel data-driven approach for passive fingerprinting of IoT device types through automatic classification of encrypted IoT network flows. Based on an in-depth empirical study on the traffic of real-world IoT devices, we identify a variety of valuable data features for accurately characterizing IoT device communications. By leveraging these features, we develop a deep learning based classification model for IoT device fin- gerprinting. Experimental results using a real-world IoT dataset demon- strate that our method can achieve 99% accuracy in IoT device-type identification.

**Keywords:** IoT *·* Device fingerprinting *·* Encrypted traffic analysis.

# Introduction

Recent years have witnessed the explosive growth of the Internet of Things (IoT), where smart objects such as cameras, printers, and routers are interconnected to facilitate a wide range of new applications and services (e.g., smart home and smart city). According to a Gartner report [2], over 20 billion devices will be connected to Internet by 2020. Meanwhile, the enormous heterogeneity and vast scale of IoT raise challenges in efficient network/device management, applica- tion QoS provisioning, and security and privacy. The proliferation of security vulnerabilities has promoted the emergence of a new generation of malware [9, 20, 13] that target explicitly at IoT devices. For example, the Mirai [9] malware compromised over one million IoT devices and conducted DDoS attacks against several major server infrastructures.

Automated and accurate identification of IoT device types (i.e., device finger- printing) is a crucial prerequisite for realizing secure, reliable, and high-quality IoT applications. From the perspective of device management, each IoT device has to be individually configured depending on its type. Given the sheer number of heterogeneous devices deployed in a smart environment, manual configura- tion is onerous and error-prone. To achieve application QoS guarantee, different priorities may be assigned to devices of different types. In Internet of Vehicles (IoV), it is imperative to prioritize network flows conveying traffic emergency information over those from entertainment devices under high traffic load. To ensure network security, vulnerable devices should be provisioned with more restricted security rules.

In recent years a variety of techniques for IoT device fingerprinting have been proposed [9, 6, 14, 15, 24, 26, 27]. One major approach employs active banner

grabbing [9, 6, 14, 15] to collect device-specific information from the application- layer responses and identifies device types based on a database of matching rules. Unfortunately, this approach is inapplicable to fingerprinting IoT devices that adopt SSL/TLS to secure their communications.

Another avenue of research tackled device fingerprinting through passive IoT traffic classification [24, 26, 27]. However, none of these approaches is manifested to be effective in accurately classifying encrypted IoT traffic. Moreover, most of them merely leverage limited set of traffic features, thus suffering from classifi- cation performance degradation in case that sophisticated malware is capable of altering a device’s communication pattern.

In this paper, we propose a passive data-driven approach for automated IoT device fingerprinting through encrypted flow classification at the network edge. Motivated by the limitations of prior works, we identify two challenges associated with encrypted flow classification based IoT device fingerprinting. First, IoT de- vices increasingly use SSL/TLS to secure their communications. Therefore, it is infeasible to collect application-layer data about IoT devices through deep packet inspection (DPI), which necessitates accurate yet privacy-preserving method for classifying IoT traffic. Second, since the IoT traffic only accounts for a minor portion of the entire Internet traffic, identifying IoT traffic from the remain- ing network traffic is largely a needle-in-the-haystack problem. This situation is worsened by the heterogeneous deployment of traditional and IoT devices in real-world IoT environment.

To resolve these challenges, we first propose to collect substantial teleme- try information about IoT device communications at the network edge. Based on these information, we identify and extract a wide range of data features to characterize the network flows generated by both IoT and conventional non-IoT devices. Our feature set comprehensively integrates traditional flow features, device-specific behavioral features, and TLS-related features by analyzing the unencrypted TLS handshake data. The collection of these features does not re- quire packet payload analysis through DPI, making it applicable to the charac- terization of encrypted IoT traffic. To demonstrate the potential of the identified features for IoT device fingerprinting, we perform an in-depth empirical analy-

sis over real-world IoT traffic with a focus on characterizing the disparate TLS usage of IoT devices.

Second, leveraging these data features collectively, we use supervised machine learning (ML) classifiers to accurately differentiate between IoT traffic and non- IoT traffic relevant to typical client-server applications. For the purpose of IoT device fingerprinting, we develop a deep learning based multinomial classifier to classify encrypted IoT TLS flows. Our classification model can achieve 99% ac- curacy for IoT device fingerprinting based on the classification of single network flow, which is more responsive than prior multi-flow based classification methods. Moreover, using all available data features considerably enhances the robustness of the fingerprinting method. Compared to fingerprinting techniques that lever- age limited features (e.g., packet sizes and timing), our approach is more resilient against sophisticated attackers that can obfuscate IoT device traffic.

In summary, we make the following contributions:

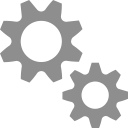
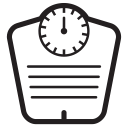
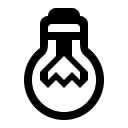
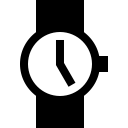
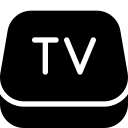
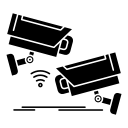
* We perform an in-depth empirical analysis of SSL/TLS encrypted IoT traffic and identify a variety of valuable data features for accurately characterizing IoT device communications.
* We propose a novel data-driven approach for passive fingerprinting of IoT device types based on automated classification of encrypted TLS flows of IoT devices.
* We develop a deep learning based classification model for IoT device finger- printing. Using a large dataset consisting of 21 real-world IoT devices, our model can achieve 99% accuracy in IoT device-type identification.

The remainder of the paper is organized as follows. Section 2 introduces the assumptions and the used dataset. Section 3 gives an overview of the proposed approach. Section 4 discusses the feature extraction process in detail. An empir- ical analysis of IoT TLS usage is presented in Section 5. Section 6 presents the evaluation results. Discussion and related work are presented in Section 7 and 8, respectively. Section 9 concludes the paper.

# Assumptions and Data Preliminaries

The IoT fingerprinting scenario we consider is a typical local area network (e.g., enterprise network or campus network), as shown in Figure 1, where a large range of IoT devices and non-IoT devices are deployed heterogeneously. The devices connect to the Internet via an access gateway. Common encryption techniques such as SSL/TLS can be employed by the devices to ensure data privacy and integrity.

The primary goal of IoT device fingerprinting is to identify the semantic type of new or unseen devices connected to the local network by solely relying on net- work traffic classification. For this purpose, our traffic analysis assumes a role of passive observer with the capability of monitoring the traffic traversing the gate- way router. The raw packets can be collected using common packet capturing



Internet

Classiﬁcation

Fingerprinting

Telemetry

Gateway

IoT Devices

Non-IoT Devices

**Fig. 1.** Edge-based IoT Device Fingerprinting

libraries, such as tshark [25] and NFQueue [33]. We do not manipulate the net- work traffic (e.g., deep packet inspection) or use the packet contents. Instead we extract useful telemetry information from the raw packets (e.g., protocol headers and traffic statistics) and leverage the information for IoT device fingerprinting. **Data preliminaries.** The dataset used in our analysis was originally col- lected by Sivanathan et al. [32] in a smart environment as depicted in Figure 1. It contains traffic from 21 IoT devices (e.g., smart cameras and switches) and 3 non-IoT devices over 3 weeks operation. We observe that 14 out of 21 IoT devices use TLS to secure their communications, which accounts for 55% of the

total captured IoT traffic.

# Approach Overview

In this section we define the problem of IoT device fingerprinting and present an overview of our fingerprinting approach.

## Problem Definition

We formulate IoT device fingerprinting as a classification problem. That is, we perform supervised multi-class classification, where a classifier is trained on la- beled instances and tested by assigning a label out of multiple predefined labels to each unlabeled instance. Given a set of IoT device types *D* = *{d*1*, . . . , dn}*, each instance is of the form (**fi**, *di*), where **fi** is the feature vector that represents a network flow (i.e., a bidirectional flow defined by the network five-tuple) gen- erated by a device *di* that belongs to *D*. This way, IoT device fingerprinting is stated as follows: assign a device type in *D* to each new network flow based on its features.

## Overview

The IoT device fingerprinting framework is composed of four major components: *packet capture/parsing*, *feature extraction*, *classifier training*, and *device type fingerprinting*.

*Packet capture/parsing.* Traffic from devices in the monitored network are collected with common packet capturing libraries such as tshark and NFQueue. The raw packets are parsed and reassembled into successive bidirectional flows based on the *flow key* represented by conventional network five-tuple: the source and destination addresses, the source and destination ports (for TCP and UDP traffic), and the protocol number.

*Feature extraction.* We identify and extract a series of network flow features from the reassembled flows, which uniquely characterize the IoT device traffic. These features not only allow to distinguish IoT traffic from conventional net- work traffic, but also enable accurate IoT device fingerprinting. We classify these features into two broad categories: aggregate and intraflow features. Aggregate features characterize flow statistics and device activities spanning multiple flows. Intraflow features consist of flow metadata (from packet header), time-series fea- tures (e.g., packet lengths and inter-arrival times), and TLS-related features (for SSL/TLS encrypted traffic). All the extracted *m* features are represented as a vector **f** *∈* R*m*. The feature extraction process is elaborated in Section 4.

*Classifier training.* This component is responsible for creating ML-based clas- sifiers through model training based on the extracted flow features. We could build binary classifier to differentiate IoT devices from conventional non-IoT de- vices. For the task of IoT device fingerprinting, a multiclass classifier is trained on the labeled instances (**fi**, *di*). The resulting model is stored for IoT device type fingerprinting.

*Device type fingerprinting.* For a new flow represented by the feature vector **f** , the trained classifier *C* is applied to identify the device type. Classifier *C* outputs a vector of posterior probabilities **p** = *{p*1*, . . . , pn}*, where *pi* is the probability of flow **f** originating from device *di*. The device type is identified as the *di* with maximal probability.

# Feature Extraction

Feature extraction is the process of transforming real world observation into a vector of values that describe the underlying process. With the purpose of IoT device fingerprinting, we identify a series of network flow based features that uniquely characterize the IoT traffic. These features not only allow us to dis- tinguish IoT traffic from conventional network traffic, but also enable accurate IoT device fingerprinting. We classify these features into two broad categories: aggregate (interflow) and intraflow features. Intraflow features consist of the normal flow features, “side-channel” features, and TLS-related features (for SS- L/TLS encrypted traffic). In contrast, aggregate features are interflow features that characterize flow statistics and device activities spanning multiple flows.

Here we elaborate on the feature extraction process and explain the rationale of leveraging these features for encrypted flow classification.

## Aggregate Features

IoT traffic typically constitutes (1) background traffic generated by the device autonomously (e.g., NTP queries for time synchronization) and (2) traffic gen- erated due to user interactions (e.g., the smart camera transmits image data to the cloud server when home invasion occurs). Aggregate features provide a high-level characterization of device activities. They are collected from multiple consecutive flows within specific duration. One such feature is the interflow time during which the IoT devices remain inactive, as it indicates the overall fre- quency of device activity. For this we compute the mean and standard deviation of the interflow times as features.

We also identify other device-specific aggregate features including the re- spective numbers of involved application-layer protocols, contacted destination servers, and unique DNS queries issued by the device within a fixed duration. We analyze the IoT dataset discussed in Section 2 and compute the daily average of the three aggregate features for each device. As indicated in Figure 2, each IoT device presents a distinct profile of feature combination. Compared to non-IoT devices (as listed in Table 1), IoT devices communicate with significantly fewer destination servers and generate fewer unique DNS queries accordingly. The ap- plication protocols involved in IoT communication are more restricted. This is intuitively reasonable since the majority of IoT devices are purpose-limited with restricted functionality (e.g., transmitting data to the cloud server for processing or receiving commands from the control server), while the real user/application behavior is enormously heterogeneous.

**Table 1.** Aggregate Features of Non-IoT Devices

|  |  |  |  |
| --- | --- | --- | --- |
| Non-IoT  Device | Destination  Servers | Application  Protocols | Unique  DNS Requests |
| MacBook Pro | 985 | 6 | 193 |
| Android Smartphone | 348 | 7 | 184 |
| Windows PC | 862 | 8 | 205 |

## Intraflow Features

*Flow Metadata.* Flow metadata includes useful information conveyed by the packet header and statistical features of the entire flow. The features we extract include the source and destination ports, the number of inbound/outbound pack- ets and bytes, the total duration of the flow in seconds, the mean packet size, and the peak and mean packet rate. Due to the dynamic allocation of IP addresses and the widespread use of CDNs, we exclude IP addresses from the feature set.

30

Destination Servers Application Protocols Unique DNS Queries

25

20

Daily Average

15

10

5

0

Amazon Echo

Belkin Motion Sensor Belkin Switch

Blipcare Blood Pressure Meter

Dropcam HP Printer

Insteon Camera LiFX

Smart Bulb

NEST Smoke Alarm Netatmo

Weather Station

Netatmo Welcome

Pixstar Photo Frame

Samsung Smartcam

SmartThings TPLink Camera

TPLink Smart Plug

Triby Speaker Withings Sleep Sensor

iHome Withings

Baby Monitor

Withings Smart Scale

**Fig. 2.** Aggregate Features of IoT Devices

Since IoT devices typically generate short bursts of traffic sporadically [32], these per-flow features serve as coarse-grained description of IoT device activity.

*Packet Sizes and Inter-arrival Times.* The series of packet sizes and inter- arrival times within a flow reveal the fine-grained pattern of IoT communication. From the application level, these features provide a descriptive profile of user behavior. For example, an IoT device such as a smart camera may periodically transmit packets of uniform size to its cloud server; while a PC would gener- ate flows with more diversity in the packet sizes/inter-arrival times given the complexity of user activities (e.g., web browsing and video streaming).

For each flow the packet sizes and inter-arrival times are extracted without considering the TCP retransmission packets or packets lacking payloads (e.g., pure TCP ACK). We used Markov chain to model the time series features. To model the series of packet sizes, we separate the IP MTU size into equally sized bins *B* = *b*1*, . . . , bn* and allocate each packet size value to the appropriate bin. A matrix *S* is constructed where each entry, *S*[*i, j*], keeps track of the number of transitions between the bin *bi* and bin *bj*. We then normalize each entry of *S* as the transition probability and flattened *S* into a feature vector. The series of packet inter-arrival times is modeled in the same way.

*{ }*

*Byte Distribution.* The full byte distribution provides information about the encoding of the data. This feature is modeled as a histogram giving the frequency of occurrence for each byte value in the application payload for a flow. Specif- ically, we used an array to record the count for each byte value of the packet payloads in a single flow. In our analysis an array length 256 is adopted. The probability of each byte value is computed by dividing the byte counts by the total number of byte counts recorded by the array.

100

80

Percentage of Flows

60

40

20

0

100

80 60

IoT

Non-IoT

Percentage of Flows

Percentage of Flows

60 40

40

20

20

0

0

Ciphersuite

TLS Extension

Client Key Length

* + 1. Advertised Ciphersuites b. Advertised Extensions c. Public Key Length

**Fig. 3.** TLS Client Features for IoT versus Non-IoT Devices.

*Unencrypted TLS Handshake Information.* Even for SSL/TLS encrypted traf- fic, we can still extract valuable information from the unencrypted TLS hand- shake process. Such information serves as characteristic feature of the com- munication endpoints. The ClientHello, ServerHello and Certificate mes- sages contain plain text data. In particular, the ClientHello message advertises lists of supported ciphersuites and TLS extensions ordered by the client pref- erence. The ServerHello message conveys the ciphersuite and TLS extensions selected by the server. The server certificate information can be derived from the Certificate message. The client public key length can be determined from the length of the ClientKeyExchange message.

Our classification model uses the list of offered ciphersuites, the list of ad- vertised extensions, and the public key length as features. In total 124 unique ciphersuites were observed in the dataset, and a binary vector of length 124 was created where a one is assigned to each ciphersuite in the list of offered ciphersuites. We also observed 14 unique TLS extensions, and a binary vector of length 14 was created where a one is assigned to each extension in the list of advertised extensions. The public key length was represented as a single integer value. Therefore, we extracted 139 TLS features for classification.

To demonstrate the discriminatory power of TLS features for IoT device fingerprinting, in Section 5 we provide a detailed characterization of TLS usage in IoT.

## Integration of Aggregate and Intraflow features

To integrate aggregate features with the intraflow features, we used a disjoint sliding window method to calculate the aggregate features within the window, and then augment each intraflow feature vector within a window with the cor- responding aggregate features.

# Characterizing TLS Usage in IoT

In this section, we compare the TLS usage of IoT devices with conventional non-IoT devices and show that they exhibit distinct characteristics. From the

dataset we extract 7820 unique TLS flows of 9 IoT devices and 12557 flows of 3 non-IoT devices. Each unique flow originates from a successful TLS handshake. Merely 12 or less unique flows are extracted from 5 IoT devices, which may be attributed to the extensive use of TLS session resumption to avoid expensive TLS renegotiation. In fact we observed that 25% of IoT clients advertised the 0x0023 (SessionTicket TLS) extension.

## TLS Client Features

The difference between IoT and non-IoT devices regarding the TLS client fea- tures is manifested in Figure 3. First, the top 15 most advertised ciphersuites in IoT are listed along with the corresponding percentage in non-IoT flows. We see less variation in the distribution of the offered ciphersuites in IoT as IoT devices consistently advertise limited and fixed number of ciphersuites. In contrast, due to the diversity in applications, larger variation in ciphersuite distribution is ob- served in non-IoT traffic. Nearly 100% of IoT devices offer weak ciphersuites such

as 0x000a(TLS RSA WITH 3DES EDE CBC SHA), 0x0016(TLS DHE RSA WITH 3DES EDE CBC SHA),

0xc003 (TLS ECDH ECDSA WITH 3DES EDE CBC SHA); while non-IoT devices ad- vertise them with significantly lower frequency. This implies that TLS imple- mentations in IoT may be outdated or lack thorough security considerations.

**Table 2.** TLS client features for each IoT device.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Device Category | Number of Unique  TLS Feature Vectors | Number of  Offered Ciphersuites | Number of  Offered Extensions | Most Frequently  Advertised Extensions | Client’s Public Key |
| Amazon Echo | 2 | 91 | 7 | heartbeet  supported groups ec point formats | 2048-bit (RSA)  512-bit (ECDHE RSA) |
| Dropcam | 1 | 3 | 4 | SessionTicket TLS  heartbeat supported groups ec point formats | 512-bit (ECDHE RSA) |
| HP Printer | 4 | 85 | 13 | signature algorithms | 768-bit (DHE RSA)  2048-bit(RSA) |
| Netatmo Welcome | 2 | 69 | 5 | server name  supported groups ec point formats | 512-bit (ECDHE RSA) |
| PIX-STAR Photo-frame | 1 | 79 | 6 | All 6 extensions | 1024-bit (DHE RSA) |
| Samsung Smartcam | 1 | 49 | 4 | All 4 extensions | 512-bit (ECDHE RSA) |
| SmartThings | 1 | 1 | 1 | signature algorithms | 1024-bit (DHE RSA) |
| Triby Speaker | 1 | 80 | 8 | All 8 extensions | 512-bit (ECDHE RSA) |
| iHome | 1 | 8 | 1 | signature algorithms | 2048-bit (RSA) |
| Android Smartphone | 14 | 78 | 14 | supported groups ec point formats signature algorithms | 512-bit (ECDHE RSA)  768-bit (DHE RSA)  256-bit (ECDHE RSA)  2048-bit (RSA)  1024-bit (DHE RSA) |
| MacBook Pro | 9 | 89 | 14 | supported groups ec point formats | 512-bit (ECDHE RSA)  768-bit (DHE RSA)  256-bit (ECDHE RSA)  2048-bit (RSA)  1024-bit(DHE RSA) |
| Windows PC | 7 | 96 | 14 | supported groups ec point formats | 512-bit (ECDHE RSA)  768-bit (DHE RSA)  256-bit (ECDHE RSA)  2048-bit (RSA)  1024-bit(DHE RSA) |

We also observed more diverse distribution in TLS extensions advertised by non-IoT devices. Up to 8 extensions are advertised in over 50% of non-IoT flows. In contrast, only 4 extensions are advertised by 50% of IoT flows. Three extensions were advertised in over 50% of non-IoT flows and rarely observed in IoT flows: 0x0017 (extended master secret), 0x0010 (ALPN), and 0xff01 (renegotiation info). This indicates more lightweight TLS implementation with limited functionality in IoT due to the computation and resource constraint. We also observed that the extension 0x000f (heartbeat) is advertised with a much higher frequency by IoT devices which indicates the behavior of IoT devices fre- quently contacting remote servers for liveliness check. Moreover, 40% of IoT clients advertised 0x0015 (padding) extension compared to 20% of the enter- prise clients, which indicates the existence of more aged IoT clients that have outdated TLS implementation.

The distribution of client public key length is also different. Most non-IoT devices used a 512-bit (ECDHE RSA) public key. 2048-bit (RSA) and 1024- bit (DHE RSA) were more widely used by IoT devices. One interesting obser- vation is 27% of non-IoT flows used 256-bit (ECDHE RSA) to generate keys for CHACHA20 POLY130 cipher, but no IoT clients used this cipher to encrypt their connection.

The discriminatory power of TLS client features for IoT device fingerprinting becomes more obvious when looking into the TLS usage of each device. Table 2 summarizes the TLS client features for each device. Four IoT devices offered the number of ciphersuites comparable to the non-IoT devices, while only one IoT device (i.e., HP Printer) offered the same number of extensions. The public keys used by the IoT devices are also quite limited with all devices using at most two public keys for their connections, which indicates the simplicity of IoT devices. In contrast, we observed more variation in the used public keys in non-IoT devices due to the diversity of user applications.

Table 2 also lists the number of unique TLS client feature vectors advertised by each device. Here a feature vector is considered unique if the concatenation of ciphersuite and extension features is different. We can see that each IoT device offered quite distinct ciphersuites and extensions, which allows accurate identi- fication of the specific device category. 6 out of 9 IoT devices even consistently offered a single fixed set of ciphersuites and extensions. 2 IoT devices offered 2 ciphersuite vectors and only the HP Printer offered 4. This also implies that IoT devices typically employ the same client to communicate with the remote servers.

## TLS Server Features

Figure 4 illustrates the difference regarding TLS server features. 60% of TLS sessions selected 0xc030 (TLS ECDHE RSA WITH AES 256 GCM SHA384) and 25% of TLS sessions selected 0x0035 (TLS RSA WITH AES 256 CBC SHA). In contrast, the distribution of ciphersuites selected by non-IoT device sessions is more heav- ily tailed. This corroborates the client-side analysis that most IoT devices used specialized clients to communicate with limited number of remote servers.

100

40 80

Percentage of Flows

Percentage of Flows

30 60

20 40

10 20

0 0

Ciphersuite

1. Selected Ciphersuite

Selected TLS Extension

1. Selected Extension

80

IoT

Non-IoT

Percentage of Flows

60

Percentage of Flows

60

40 40

20 20

0

Validity of Certificate (days)

0

2 53 1 5 71 13 10 9 6 30

Number of SAN Entries

1. Validity of Certificate (d) Number of SAN Entries

**Fig. 4.** TLS Server Features for IoT versus Non-IoT Devices.

Similar to the selected ciphersuites, the servers that IoT devices communi- cated with selected fewer TLS extensions. There is much greater diversity in the selected TLS extensions with 0xff01 (renegotiation info) and 0x000b (ec point formats) being the most frequent.

By analyzing the plain text server certificate data (e.g., certificate subject), we found that non-IoT devices mostly connected to common servers such as

\*.google.com, \*.icloud.com and \*.facebook.com. Moreover, the correspond- ing distribution of certificate subjects is very long tailed. We also analyzed two other features extracted from server certificates: the certificate validity and the number of Subject Alternative Name (SAN) entries. As depicted in Figure 4, the validity of the certificate is sharply divided for IoT devices and non-IoT devices. The distribution of the validity of non-IoT devices’ certificates is long tailed with greater diversity. Almost all server certificates associated with IoT devices have no more than 2 SAN entries. This is intuitively reasonable consid- ering that non-IoT devices typically connected with several hundreds of distinct servers in a single day, whereas IoT devices only consistently communicate with its manufacture server or third-party cloud server for data uploading and pro- cessing.

Table 3 summarizes the TLS server features for each device. A unique com- bination of ciphersuite and extension is selected in IoT TLS flows. Due to the abundance of client applications, TLS sessions of non-IoT devices selected more diverse ciphersuites and extensions. Especially when considering the respective

**Table 3.** TLS server features for each device.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Device Category | Most Frequently  Selected Ciphersuite | Most Frequently  Selected Extensions | Number of  Distinct CS | Number of  SAN entries | Number of  Distinct SNI |
| Amazon Echo | TLS ECDHE RSA WITH AES 128 GCM SHA256 | renegotiation info  server name ec point formats | 1 | 2 | 5 |
| Dropcam | TLS ECDHE RSA WITH AES 128 CBC SHA | renegotiation info | 1 | 2 | 0 |
| HP Printer | TLS DHE RSA WITH AES 128 CBC SHA | renegotiation info | 2 | 3 | 0 |
| Netatmo Welcome | TLS ECDHE RSA WITH AES 256 GCM SHA384 | renegotiation info  server name ec point formats | 1 | 2 | 1 |
| PIX-STAR Photo-frame | TLS DHE RSA WITH AES 256 CBC SHA | renegotiation info  SessionTicket TLS | 1 | 1 | 1 |
| Samsung SmartCam | TLS DHE RSA WITH AES 128 CBC SHA | renegotiation info ec point formats  SessionTicket TLS | 1 | 1 | 0 |
| SmartThings | TLS DHE RSA WITH AES 128 CBC SHA | signature algorithms | 2 | 0 | 0 |
| Triby Speaker | TLS ECDHE RSA WITH AES 256 GCM SHA384 | renegotiation info  heartbeat server name ec point formats | 1 | 0 | 1 |
| iHome | TLS RSA WITH AES 128 CBC SHA256 | None | 1 | 2 | 0 |
| Android Smartphone | TLS RSA WITH AES 128 CBC SHA | renegotiation info | 135 | 53 | 263 |
| MacBook Pro | TLS ECDHE RSA WITH AES 128 GCM SHA256 | renegotiation info | 148 | 53 | 316 |
| Windows PC | TLS RSA WITH AES 256 CBC SHA | renegotiation info | 262 | 53 | 554 |

numbers of certificate subjects, SAN entries and unique server name indication (SNI) fields, we can observe significant difference between both categories of devices.

## Discriminative Power of TLS Features

Figure 5 shows a similarity matrix for the 14 IoT devices with respect to their TLS clients. The client offered ciphersuites, the advertised extensions and the public key length were concatenated as feature vectors, and the similarity be- tween two feature vectors was computed using Euclidean distance (2-norm)

/J,

*d*(*fi, fj*) = *i,j* (*fi − fj*)2 as the distance metric, and *fi* being the mean of the feature vectors for the *i*’th device type. Since each IoT device should be perfectly self-similar, the diagonal of this matrix are all 1.0. The TLS client fea-

ture for each IoT device is unique except for the Belkin Motion Sensor and the TPLink Smart Plug, which have similar TLS client parameters. Device-specific TLS parameters can be a promisingly powerful feature for classifying IoT device type.

# Evaluation

In this section, we describe the details of our ML-based IoT fingerprinting methodology, and demonstrate the discriminative power of the TCP/IP flow features for accurately differentiating between IoT devices and non-IoT devices as well as identifying the specific IoT device category.

## Algorithms

We performed the encrypted flow classification using a deep learning based model: Multi-layer Perceptron (MLP), and compared the results with three com-

Amazon Echo

Belkin Motion Sensor

Dropcam

HP Printer

iHome Insteon Camera1

LiFX

Smart Bulb Netatmo Welcome

Pixstar

Photo Frame Samsung Smartcam

SmartThings TPLink Camera

TPLink

Smart Plug Triby Speaker

Amazon Echo Belkin

Motion Sensor

Dropcam HP Printer

iHome

Insteon Camera1

LiFX

Smart Bulb Netatmo Welcome

Pixstar Photo Frame

Samsung Smartcam

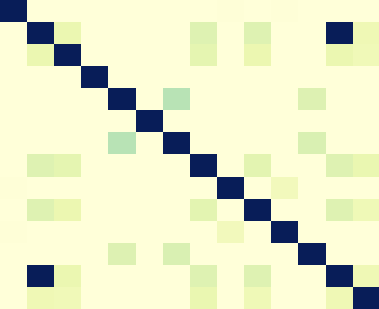
SmartThings TPLink Camera

TPLink Smart Plug

Triby Speaker

1*.*0

0*.*9



0*.*8

0*.*7

0*.*6

0*.*5

0*.*4

0*.*3

0*.*2

0*.*1

**Fig. 5.** Similarity matrix for different IoT devices with respect to the TLS client fea- tures

mon machine learning algorithms: Logistic Regression, Random Forest and Sup- port Vector Machine. To auto-tune the hyperparameters for the models, we used grid search and cross-validation over a set of standard values. We used Keras [5] with the TensorFlow [7] backend to implement the MLP; while the remaining ML algorithms are implemented using Scikit-learn [28]. An open source tool is used and extended for feature extraction [4].

**Multi-layer Perceptron (MLP).** MLPs are fully connected feed-forward neural networks with multiple hidden layers. Each layer consists of several neu- rons that compute a weighted sum of the inputs from the adjacent layer and produce the output through a non-linear activation function. Deep and dense MLPs can estimate highly nonlinear functions. However, due to the huge num- ber of parameters that need to be learned, it is complex process to train these models and there is the potential for overfitting.

We performed automatic search of the best hyperparameters for the MLP model using grid search and cross validation. Table 4 lists the values of the hyperparameters we tuned and the corresponding intervals within which the values are varied.

**Logistic Regression.** Logistic regression is one of the simplest classification algorithms. It generates an estimate of the probability that a feature vector belongs to a specific class. We used both l1-regularization and l2-regularization in the comparison. The model is more easily interpretable compared to MLP

**Table 4.** Tuned hyperparameters of the MLP

|  |  |  |
| --- | --- | --- |
| Hyperparamet | er Value | Tuning Space |
| optimizer | Adam | Adam, SGD,  RMSProp |
| learning rate | 0.001 | 0.001 .. 0.1 |
| momentum | 0.0 | 0.0 .. 0.9 |
| batch size | 32 | 8 .. 256 |
| training  epochs | 150 | 50 .. 200 |
| number of  hidden layers | 3 | 2 .. 5 |
| dropout | 0.1 | 0.0 .. 0.5 |
| activation | sigmoid | tanh,  sigmoid, relu |

as the model parameters quantify the significance of the corresponding data features.

**Random Forest.** A random forest combines decision tress with ensemble learning [10]. Each decision tree is built from a random sub-sample of the original data and produces a probability for each possible output class. The output of the ensemble is the average probability among all decision trees. Random forest is resistant to overfitting. Moreover, the variance is reduced without resulting in bias when the number of trees increases. For the Scikit-learn implementation, we used grid search and cross-validation to tune the depth of the trees and the number of features per split.

**Support Vector Machine (SVM).** An SVM projects original data to a high-dimensional feature space using a kernel function, where a hyperplane can be built to linearly separate different classes [12]. Our SVM implementation adopts the One-VS-Rest strategy and uses LinearSVC for classification.

## IoT vs Non-IoT

For the binary classification between IoT and Non-IoT device flows, we used a logistic regression classifier with l1-regularization [18]. As such, five machine learning classifiers were trained using different combinations of the collected data features, which incorporate the flow metadata (M), the time series features including the packet sizes (PS) and inter-arrival times (PT), the byte distribu- tion of the packet payload (BD), the TLS client parameters (i.e., the offered ciphersuites, advertised extensions, and the public key length) (TLS), and the aggregate features (A) spanning multiple flows.

To demonstrate the usefulness of the identified data features, we use 10-fold cross validation over 96680 IoT flows and 96680 Non-IoT flows, out of which 20000 flows are TLS flows. Table 5 shows the cross validation results. We can see that using all available data features achieves the best accuracy of 99*.*6%. Using the TLS features alone to classify TLS flows gives 90% accuracy, which demonstrates the usefulness of these features for characterizing IoT devices.

**Table 5.** Classifier accuracy for different subsets of features

|  |  |
| --- | --- |
| Data Features | Accuracy |
| M+PS+PT | 82.4% |
| M+PS+PT+BD | 93.5% |
| TLS | 90.1% |
| M+PS+PT+BD+TLS | 97.2% |
| M+PS+PT+BD+TLS+ | A 99.6% |

## IoT Device Fingerprinting

We also evaluate the effectiveness of the identified flow features for IoT device fingerprinting. For the multi-class classification, we use 541250 flows generated by all the 21 IoT device categories and data features described in Section 4. We randomly selected 75% flows to train the classifiers and used the remaining 25% flows for testing. The final classification results under different ML models are listed in Table 6. Using MLP and all flow features results in the best classification performance, with an overall accuracy of 99.8% for the 21-class classification based on a single network flow.

**Table 6.** Comparison of classifier performance

|  |  |
| --- | --- |
| Model | Accuracy |
| *l* 1-Logistic  Regression | 86.2% |
| *l* 2-Logistic  Regression | 81.5% |
| Random Forest | 72.7% |
| SVM | 58.4% |
| MLP | **99.8%** |

The confusion matrix of the best classification result is shown in Figure 6. Each matrix element *mij* represents the percentage of device *di*’s flows classified as that of device *dj*. The confusion matrix is perfectly diagonal, indicating the superior power of the identified network flow features for IoT fingerprinting. Notably, all samples of an Insteon Camera were identified as another device of the same category since it generated significantly less flows.

We also present the classification performance of MLP with different subsets of features in Table 7. With a subset of data features, our approach is able to differentiate the device types based on a single flow. Using a combination of the available data features significantly enhances the classification robustness.

## Algorithm Performance

We assess the performance of the used classification algorithms in terms of the training and testing time. The multi-class classification experiment discussed in Section 6.3 was repeated 10 times for each algorithm. Figure 7 and Figure 8 plot

Amazon Echo Belkin Motion Sensor

Blipcare Blood Pressure Meter

Dropcam

HP Printer iHome

Insteon Camera1 Insteon Camera2 LiFX Smart Bulb

True Device Category

NEST Smoke Alarm

Netatmo Weather Station Netatmo Welcome

Pixstar Photo Frame Samsung Smartcam

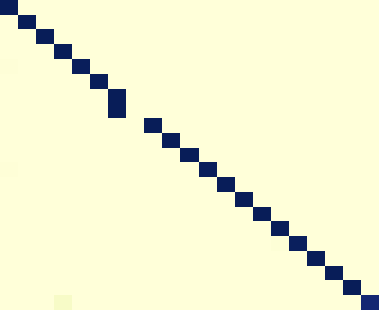
SmartThings TPLink Camera TPLink Smart Plug Triby Speaker

Withings Sleep Sensor

Withings Baby Monitor Withings Smart Scale

1*.*0

0*.*9



0*.*8

0*.*7

0*.*6

0*.*5

0*.*4

0*.*3

0*.*2

0*.*1

Predicted Device Category

Amazon Echo Belkin Motion Sensor

Blipcare Blood

Pressure Meter

Dropcam HP Printer

iHome Insteon Camera1 Insteon Camera2 LiFX Smart Bulb NEST Smoke Alarm

Netatmo

Weather Station

Netatmo Welcome Pixstar Photo Frame Samsung Smartcam

SmartThings TPLink Camera TPLink Smart Plug Triby Speaker Withings Sleep Sensor

Withings

Baby Monitor

Withings Smart Scale

**Fig. 6.** Confusion matrix for the 21-class IoT device type classification using MLP. The best overall accuracy is 99.8%.

**Table 7.** MLP classification accuracy for different subsets of the flow features

|  |  |  |  |
| --- | --- | --- | --- |
| Data Features | Accuracy | Precision | Recall |
| M+PS+PT | 0.81 | 0.84 | 0.82 |
| M+PS+PT+BD | 0.83 | 0.86 | 0.83 |
| M+PS+PT+TLS | 0.89 | 0.90 | 0.89 |
| M+PS+PT+BD+TLS | 0.92 | 0.93 | 0.93 |
| M+PS+PT+BD+TLS+ | A 0.99 | 0.99 | 0.99 |

the training and testing times in seconds with respect to the average classification accuracy of each algorithm. The random forest can be quickly trained and tested, while SVM is relatively slow to train and test. The MLP is cumbersome to train, but quick to classify new samples.

# Discussion and Future Work

**Robustness against adversarial attacks.** Attackers capable of controlling their devices’ network traffic may attempt to circumvent our fingerprinting tech- nique. For example, random or constant delays could be introduced into a flow. Similarly, packet sizes could be varied through data padding at different protocol levels. Since our method does not solely rely on a single feature (e.g., time series

35

L1-LogReg

L1-LogReg

Time(seconds)

Time(seconds)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6000 |  |  |  |  |
|  |  |  |  |
|
|
|  | L2-LogReg |  |  | L2-LogReg |
| 5000 | Random Forest | 30 |  | Random Forest |
|  | SVM |  |  | SVM |
| 4000  3000  2000  1000  0 | MLP | 25  20  15  10  5 |  | MLP |

0*.*4 0*.*5 0*.*6 0*.*7 0*.*8 0*.*9 1*.*0

Accuracy

0*.*4 0*.*5 0*.*6 0*.*7 0*.*8 0*.*9 1*.*0

Accuracy

**Fig. 7.** Model Training Times **Fig. 8.** Model Testing Times

feature), it is more resilient to such adversarial attacks compared to existing IoT device fingerprinting techniques. Modifying the TLS parameters supported by a device (i.e., advertised ciphersuites and extensions) may also degrade the normal functionality of the device itself. An attacker could also try to mimic the traffic pattern of a different device type. Considering that most IoT devices will communicate with a fixed set of servers, it would be extremely difficult for the attacker to generate similar requests to the servers, especially when the emulated device uses TLS for authentication and data encryption.

**Impact of TLS 1.3.** The analysis of TLS features in this paper is based on TLS 1.2. TLS 1.3 [3] eliminates a complete round trip time, making TLS more suitable for IoT devices. In TLS 1.3, the server selected extensions and certificate get encrypted. Since our approach only relies on TLS client features, the adoption of TLS 1.3 will not deteriorate the performance of IoT device fingerprinting.

**Scalability.** The proposed IoT device fingerprinting method can be deployed at the edge of a LAN. Our current implementation is applied to small scale IoT smart environment. A more scalable fingerprinting system could be realized using a distributed design. In our future work, we will explore using federated learning approach [21] for distributed optimization of our deep learning based fingerprinting model. Using federated learning, we could train a local model at the network edge and synchronize the model modifications through a centralized server. This solution is more scalable since only model parameters have to be synchronized, thus obviating the need to offload raw data to the centralized server.

**Applications of device fingerprinting.** In future work, we will consider the applications of automated IoT device fingerprinting from various perspec- tives. One potential application is to leverage device fingerprinting to instruct the QoS-aware provisioning of network resources. Another focus is on its secu- rity applications. For example, we will explore how to integrate our fingerprinting method into existing network access control (NAC) system. This could enable the NAC system to automatically detect unknown vulnerable devices and con- figure security rules accordingly.

# Related Work

**Operating System Fingerprinting.** In network security, fingerprinting has been used for information gathering and vulnerability assessment of the attack target. One prominent example is OS fingerprinting, which aims at determining the operating system of a remote host based on the implementation differences of a TCP/IP network stack. A variety of tools (Nmap [1] and P0f [35]) actively probe a target with purposely crafted packets and generate fingerprint from the response packets. The OS version is identified by matching the fingerprint with a database of existing OS categories. Hershel [30] and Faulds [31] focus on Internet-wide single-probe OS fingerprinting, which leverages the temporal features (i.e., TCP retransmission timeout) to classify the OS of the target. The objective of our work is to identify the IoT device type instead of the OS version, which precludes the usage of OS fingerprinting as different devices may run the same OS.

**IoT Device-Type Identification.** Various prior research have been pro- posed to utilize a multitude of data sources for passive IoT device type iden- tification. Early wireless communication fingerprinting techniques leverage the implementation characteristics of wireless network interface card (WNIC) hard- ware [11] or device-driver [16, 23]. However, the fingerprinting target is WNIC instead of physical device. These techniques also require the observer to be in the wireless range of the target device in order for the technique to work. Cor- respondingly, Kohno et al. introduced a passive physical device fingerprinting method that exploited the minute deviations in the clock skews to derive a clock cycle pattern as the device identity [19]. The analysis is not readily applicable to fingerprinting the semantic type of IoT devices.

State-of-the-art solutions for device-type identification in small scale net- works rely on supervised machine learning with various traffic features [29, 22, 24]. GTID [29] extracts device signatures from the packet inter-arrival times (IATs), which are used to train an Artificial Neural Network (ANN) to identify device type. The device identification process requires several hours of traffic for signature generation. Moreover, solely relying on packet IATs makes it sus- ceptible to spoofing attack where sophisticated attacker could emulate the IAT pattern of a different device. The authors of [22] design a device type detection technique that classifies encrypted link-layer traffic using random forest classifier. The technique only targets wireless devices, demanding passive eavesdropping using specialized radios. Similarly, Meidan et al. [24] performed IoT device type identification by applying random forest classifier to features extracted from TCP sessions.They only achieve acceptable accuracy through classification of 20 consecutive TCP sessions, while our method achieves high accuracy on single- flow classification.

Motivated by network security, IoT Sentinel [26] builds up a classifier on de- vice fingerprints extracted from the burst of network traffic generated during the initial setup phase of the device. After assessing the vulnerability of the iden- tified device type, security rules are enforced to constrain the communications of vulnerable devices. One limitation of IoT Sentinel is that it only operates

when a device is first introduced to a network. DIoT [27] uses unsupervised learning to classify devices into device types based on passive fingerprinting of periodic background traffic of IoT devices. For each device type, Gated recurrent unit (GRU) is used to establish the normal communication profile and subse- quently detect anomalies in communication patterns. The fingerprinting strategy of DIoT is subject to deterioration since it merely relies on the periodicity inher- ent in background traffic. Furthermore, only an abstract device type is identified instead of the semantic type.

Another avenue of research centers on Internet-wide detection of IoT devices. Shodan [6] is the first search engine for discovering Internet-connected devices. Internally it uses Nmap to identify the application and services of an online de- vice. Censys [14] employs banner-grabbing to gather device-specific information from the application-layer responses, and identifies devices based on a database of matching rules. Fingerprinting with Shodan and Censys demands cumbersome manual efforts for devising the matching rules. Recently, Feng et al. developed a search engine named ARE [15] for automatic device discovery and annotation in large scale. Device rules are built by correlating application-layer response data from IoT devices with the product descriptions (e.g., the device type and vendor) in relevant websites. The active probing performed by these techniques could be detected and defeated by sophisticated attackers through obfuscation. Moreover, the application layer data may not be readily available from banner grabbing. In contrast, our approach can automatically detect IoT device type using deep learning based classification.

**Network Traffic Analysis.** Network traffic analysis has been applied to various applications such as intrusion detection [36] and app identification [34]. These applications employ classical machine learning classifiers with different features extracted from the raw traffic (e.g., packet header, payload and statis- tics). For example, time-series features (packet lengths and inter-arrival times) are used for mobile application service usage classification [17]. Recent studies even achieve high malware detection accuracy on encrypted traffic using the plain text data contained in the TLS handshake [8]. These features are instructive for our passive traffic classification based IoT device fingerprinting. We perform fine- grained characterization of IoT devices and explore the power of a wide-range of features for IoT device identification.

# Conclusions

This paper proposed a novel data-driven approach for automated and accurate fingerprinting of IoT device types. Our fingerprinting method relies on passive classification of potentially encrypted IoT flows. We first performed an in-depth empirical study of real-world IoT traffic, based on which we identify a variety of useful data features that could accurately characterize IoT device commu- nications. Leveraging these features, we have developed a deep learning based classification model for IoT device fingerprinting. Using a real-world IoT dataset, our evaluation results demonstrate that the proposed method can achieve *∼* 99%

accuracy in IoT device-type identification based on single network flow classifi- cation.

# Acknowledgment

This work is partially supported by the U.S. ONR grants N00014-16-1-3214, N00014-16-1-3216, and N00014-18-2893 and U.S. ARO grant W911NF-17-1- 0447.

# References

1. Nmap, Network Security Scanner Tool (2012), {https://nmap.org/}
2. 20 billion IoT devices by 2020 (2017), [{https://www.gartner.com/newsroom/id/](http://www.gartner.com/newsroom/id/) 3598917}
3. The Transport Layer Security (TLS) Protocol Version 1.3 (2018), {https:// datatracker.ietf.org/doc/rfc8446/}
4. Joy (2019), {https://github.com/cisco/joy}
5. Keras: Deep Learning for humans (2019), {https://github.com/keras-team/ keras}
6. Shodan (2019), [{https://www.shodan.io/}](http://www.shodan.io/)
7. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghe- mawat, S., Irving, G., Isard, M., et al.: Tensorflow: a system for large-scale machine learning. In: OSDI. vol. 16, pp. 265–283 (2016)
8. Anderson, B., McGrew, D.: Machine learning for encrypted malware traffic clas- sification: accounting for noisy labels and non-stationarity. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1723–1732. ACM (2017)
9. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., et al.: Understanding the mirai botnet. In: 26th *{*USENIX*}* Security Symposium (*{*USENIX*}* Security 17). pp. 1093–1110 (2017)
10. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
11. Brik, V., Banerjee, S., Gruteser, M., Oh, S.: Wireless device identification with radiometric signatures. In: Proceedings of the 14th ACM international conference on Mobile computing and networking. pp. 116–127. ACM (2008)
12. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**(3), 273–297 (1995)
13. Costin, A., Zaddach, J.: Iot malware: Comprehensive survey, analysis framework and case studies. BlackHat USA (2018)
14. Durumeric, Z., Adrian, D., Mirian, A., Bailey, M., Halderman, J.A.: A search en- gine backed by internet-wide scanning. In: Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security. CCS ’15 (2015)
15. Feng, X., Li, Q., Wang, H., Sun, L.: Acquisitional rule-based engine for discovering internet-of-things devices. In: 27th *{*USENIX*}* Security Symposium (*{*USENIX*}* Security 18). pp. 327–341 (2018)
16. Franklin, J., McCoy, D., Tabriz, P., Neagoe, V., Randwyk, J.V., Sicker, D.: Passive data link layer 802.11 wireless device driver fingerprinting. In: USENIX Security Symposium. vol. 3, pp. 16–89 (2006)
17. Fu, Y., Xiong, H., Lu, X., Yang, J., Chen, C.: Service usage classification with encrypted internet traffic in mobile messaging apps. IEEE Transactions on Mobile Computing **15**(11), 2851–2864 (2016)
18. Koh, K., Kim, S.J., Boyd, S.: An interior-point method for large-scale l1-regularized logistic regression. Journal of Machine learning research **8**(Jul), 1519–1555 (2007)
19. Kohno, T., Broido, A., Claffy, K.C.: Remote physical device fingerprinting. IEEE Transactions on Dependable and Secure Computing **2**(2), 93–108 (2005)
20. Kolias, C., Kambourakis, G., Stavrou, A., Voas, J.: Ddos in the iot: Mirai and other botnets. Computer **50**(7), 80–84 (2017)
21. Koneˇcny`, J., McMahan, H.B., Yu, F.X., Richt´arik, P., Suresh, A.T., Bacon, D.: Fed- erated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)
22. Maiti, R.R., Siby, S., Sridharan, R., Tippenhauer, N.O.: Link-layer device type classification on encrypted wireless traffic with cots radios. In: European Sympo- sium on Research in Computer Security. pp. 247–264. Springer (2017)
23. Maurice, C., Onno, S., Neumann, C., Heen, O., Francillon, A.: Improving 802.11 fingerprinting of similar devices by cooperative fingerprinting. In: 2013 Interna- tional Conference on Security and Cryptography (SECRYPT). pp. 1–8. IEEE (2013)
24. Meidan, Y., Bohadana, M., Shabtai, A., Ochoa, M., Tippenhauer, N.O., Guarnizo, J.D., Elovici, Y.: Detection of unauthorized iot devices using machine learning techniques. arXiv preprint arXiv:1709.04647 (2017)
25. Merino, B.: Instant traffic analysis with Tshark how-to. Packt Publishing Ltd (2013)
26. Miettinen, M., Marchal, S., Hafeez, I., Asokan, N., Sadeghi, A.R., Tarkoma, S.: Iot sentinel: Automated device-type identification for security enforcement in iot. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). pp. 2177–2184. IEEE (2017)
27. Nguyen, T.D., Marchal, S., Miettinen, M., Dang, M.H., Asokan, N., Sadeghi, A.R.: Diot: A crowdsourced self-learning approach for detecting compromised iot devices. arXiv preprint arXiv:1804.07474 (2018)
28. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Ma- chine learning in python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)
29. Radhakrishnan, S.V., Uluagac, A.S., Beyah, R.: Gtid: A technique for physical deviceanddevice type fingerprinting. IEEE Transactions on Dependable and Secure Computing **12**(5), 519–532 (2015)
30. Shamsi, Z., Nandwani, A., Leonard, D., Loguinov, D.: Hershel: Single-packet os fingerprinting. IEEE/ACM Transactions on Networking **24**(4), 2196–2209 (2016)
31. Shamsi, Z., Cline, D.B., Loguinov, D.: Faulds: A non-parametric iterative classifier for internet-wide os fingerprinting. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. CCS ’17 (2017)
32. Sivanathan, A., Sherratt, D., Gharakheili, H.H., Radford, A., Wijenayake, C., Vish- wanath, A., Sivaraman, V.: Characterizing and classifying iot traffic in smart cities and campuses. In: Computer Communications Workshops (INFOCOM WKSHPS), 2017 IEEE Conference on
33. Sugiyama, Y., Goto, K.: Design and implementation of a network emulator using virtual network stack. In: 7th International Symposium on Operations Research and Its Applications (ISORA08). pp. 351–358 (2008)
34. Taylor, V.F., Spolaor, R., Conti, M., Martinovic, I.: Robust smartphone app iden- tification via encrypted network traffic analysis. IEEE Transactions on Information Forensics and Security **13**(1), 63–78 (2018)
35. Zalewski, M.: p0f v3 (2012), [{http://lcamtuf.coredump.cx/p0f3/}](http://lcamtuf.coredump.cx/p0f3/)
36. Zarpelao, B.B., Miani, R.S., Kawakani, C.T., de Alvarenga, S.C.: A survey of intru- sion detection in internet of things. Journal of Network and Computer Applications **84**, 25–37 (2017)