Link-Layer Device Type Classification on Encrypted Wireless Traffic with COTS Radios

Conference Paper · August 2017

DOI: 10.1007/978-3-319-66399-9_14

4 authors, including:

Sandra Siby
École Polytechnique Fédérale de Lausanne
10 PUBLICATIONS **110** CITATIONS
SEE PROFILE

Ragav Sridharan
NIIT University
3 PUBLICATIONS **11** CITATIONS
SEE PROFILE

Nils Ole Tippenhauer
CISPA Helmholtz Center for Information Security
77 PUBLICATIONS **1,675** CITATIONS
SEE PROFILE

Link-Layer Device Type Classification on Encrypted Wireless Traffic with COTS Radios

Rajib Ranjan Maiti(✉), Sandra Siby, Ragav Sridharan, and Nils Ole Tippenhauer

Singapore University of Technology and Design (SUTD),
8 Somapah Road, 487372 Singapore, Singapore
{rajib_maiti,sandra_ds,ragav_sridharan,nils_tippenhauer}@sutd.edu.sg

Abstract. In this work, we design and implement a framework, PrEDeC, which enables an attacker to violate user privacy by using the encrypted link-layer radio traffic to detect device types in a targeted environment. We focus on 802.11 traffic using WPA2 as security protocol. Data is collected by passive eavesdropping using COTS radios. PrEDeC (a) extracts features using temporal properties, size of encrypted payload, type and direction of wireless traffic (b) filters features to improve overall performance (c) builds a classification model to detect different device types. While designing PrEDeC, we experimentally record the traffic of 22 IoT devices and manually classify that data into 10 classes to train three machine learning classifiers: Random Forest, Decision Tree and SVM. We analyze the performance of the classifiers on different block sizes (set of frames) and find that a block size of 30k frames with Random Forest classifier shows above 90% accuracy. Additionally, we observe that a reduced set of 49 features gives similar accuracy but better efficiency as compared to taking an entire set of extracted features. We investigate the significance of these features for classification. We further investigated the number of frames and the amount time required to eavesdrop them in different traffic scenarios.

Keywords: Encrypted network traffic · Classification · Machine learning

1 Introduction

According to [7], the number of IoT devices in 2016 has reached 6.4 billion and is expected to hit the 20 billion mark by 2020. This proliferation of IoT devices and wireless connectivity brings about questions of security and privacy. The large volume of data generated by IoT devices can allow those with access to the data to violate the privacy of the device owners.

Encryption techniques in protocols such as WPA2 are intended to preserve confidentiality of data transferred over wireless networks. However, traffic analysis can be done even on encrypted link-layer traffic, using header information

S.N. Foley et al. (Eds.): ESORICS 2017, Part II, LNCS 10493, pp. 247–264, 2017.
DOI: 10.1007/978-3-319-66399-9_14

and other parameters such as size and temporal properties. This was demonstrated in works such as [18,19], where features extracted from encrypted link-layer traffic were analyzed to determine a user's online browsing activity and smartphone app usage activity. In this paper, we investigate the possibility of analyzing eavesdropped WiFi link-layer traffic from devices to determine their type. If devices can be classified by type, an attacker can determine which types of devices are present in her surroundings, posing a privacy threat to the users of those devices. For example, an attacker's knowledge about device types on a company's premises would lead to information leakage about the presence and absence of people in particular locations (such as meeting rooms). The attacker might also obtain more details about the company's operations (for example, how many security cameras or laptops are present on the premises) or employees' lifestyles (how many employees are using fitness tracking devices). Furthermore, the attacker can use the knowledge of device types to exploit vulnerabilities associated with a specific type of devices. For example, the article in [13] reported more than 400,000 D-Link cameras, recorders and storage devices that were affected by a single vulnerability. An attacker might detect cameras in her environment and attempt camera-specific attacks on her surrounding devices. The attacker might also be correlate traffic to confidential activities such as the activation of an alarm system.

We propose the Privacy Exposing Device Classifier (PrEDeC), a framework that can be used to detect the presence of different types of devices in a wireless environment. PrEDeC takes eavesdropped encrypted link-layer traffic as input. PrEDeC does not require specialized equipment to obtain the traffic; it uses commercial-off-the-shelf (COTS) radios for eavesdropping. It extracts features from the traffic and uses trained classifiers on them to classify device types. In this paper, we focus on IoT devices such as IP cameras, Amazon smart speakers, smart printers and smartphones that communicate using the 802.11b standard.

We summarize our contributions as follows:

- We propose a framework, PrEDeC, that performs device type classification on eavesdropped encrypted link-layer wireless traffic using machine learning techniques.
- We present an implementation of the proposed framework and discuss which features of the traffic along with classification model are most relevant in device type classification.
- We investigate the accuracy of our implementation by performing experiments with a set of WiFi enabled devices in a controlled environment and compare the performance of three classifiers.

The paper is organized as follows. Section 2 briefly mentions the relevant background for our work. The attacker model and the classification framework is described in Sect. 3. Framework implementation is described in Sect. 4. Section 5 talks about the data sets used in our experiments, and the performance metrics for analysis. Related works are described in Sect. 7. We conclude the paper and discuss possible future work in Sect. 8.

2 Background

In this section, we briefly describe WiFi frame structure and the encryption technique used in WiFi.

2.1 WiFi Frame Structure

We consider wireless traffic that uses the 802.11 standard. A frame in 802.11 has the following fields as shown in Fig. 1:

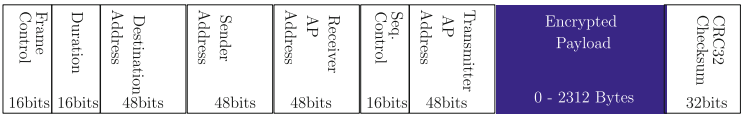

Fig. 1. IEEE 802.11 frame format.

1. Frame Control: This field consists of 11 sub-fields. It provides details about the frame's functions. It specifies the protocol version, type (management, control, data or reserved) and sub-type of the frame. It also has two sub-fields to indicate whether the frame is headed for a distribution system. Other sub-fields provide information such as the fragmentation, power management, order and re-transmission in the frame. Out of these fields, the type and sub-type play an important role in our analysis. Management frames facilitate communication among devices and have 12 sub-types. Control frames help in fair channel access among the contending devices and have 7 sub-types. Data frames carry payload in the frame body and have 9 sub-types.
2. Duration or ID: This field can take different values depending on the type and sender of the frame. It can be a station identifier, a duration or a fixed value. This field is not considered in our analysis.
3. Address: There are up to four address fields. These fields indicates the MAC addresses involved in sending and/or receiving the frame. These fields are used to correlate the device with its activity.
4. Sequence control: This field is used to indicate message order and help in identifying frame duplication. This field is not considered in our analysis.
5. Payload: This indicates the data content of the frame. Since the payload can be encrypted in case of WPA2, we do not use this field in our analysis. We only consider the size of the payload.
6. Frame Check Sequence: This field is used to check the integrity of the frame. This field is not considered in our analysis.

2.2 802.11 and WPA2

WiFi Protected Access 2 (WPA2) was developed to enhance the security mechanisms in WiFi networks. It introduced new protocols (4-way handshake, group key handshake) for key establishment and key change that would enable secure

communication among authenticated parties. It also uses a stronger encryption protocol, CCMP, to encrypt the data. A CCMP data unit consists of the frame header, a CCMP header, the data unit, a MIC (Message Integrity Code) and the FCS (Frame Check Sequence). Out of these fields, only the data unit and the MIC (which protects the integrity of the message) are encrypted. The frame header is not encrypted. In this work, we consider only the header information of the frame and the size of the payload for our classification problem.

3 Link-Layer Device Type Classification on Encrypted Wireless Traffic

In this section, we describe the system model, and the attacker model with possible exploits. We then present the Privacy Exposing Device Classifier (PrEDeC), a framework to train classifiers and apply them for device type prediction.

3.1 System and Attacker Model

We consider a scenario where there is a set of devices that can communicate wirelessly. The devices need not be active at all times. In addition, the number of devices in the scenario does not remain constant, i.e., devices can be introduced into or removed from the system.

The attacker model consists of an attacker who can passively eavesdrop the wireless traffic of these devices. The attacker does not have prior knowledge of the devices in the network. Furthermore, she does not perform active probing of the devices and does not attempt to decrypt the wireless traffic. The attacker uses commercial off-the-shelf radios (e.g. wireless adapters in a laptop or raspberry Pi) to perform traffic captures. She has a pre-trained classifier, and eavesdrops for a period of time which allows her to obtain sufficient traffic samples to perform device type classification. The attack scenario is visualized in Fig. 2.

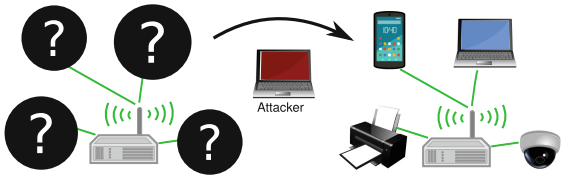

Fig. 2. Attack scenario. Encrypted traffic is observed and devices are classified.

3.2 PrEDeC

Privacy Exposing Device Classifier (PrEDeC) is a framework that enables an attacker to classify device types based on eavesdropped encrypted wireless traffic. An attack using PrEDeC consists of two phases:

Training Phase. This phase is run offline, before the actual attack. During the training phase, the attacker uses wireless traffic from a set of IoT devices to train a machine learning algorithm that can perform device type classification. The attacker performs three actions in this phase—data collection, feature extraction and selection, and model training in this phase. In the data collection step, she uses a COTS radio in monitor mode to passively eavesdrop wireless traffic. She has a number of IoT devices of different types, e.g., smartphone, camera, smart light, etc., from which she gathers traces. The eavesdropped traffic is stored in pcap files for analysis. The second step involves extracting features from the link-layer frames in the files. Features include the type and sub-type, size, direction, inter-arrival time and rate of frames. The feature extraction step produces signatures (a set of features) for every device (represented by its MAC address) present in the trace. The attacker uses certain statistical techniques to obtain a reduced number of features effective for efficient classification. The attacker does not consider MAC address and the manufacturer information that can be obtained from a MAC address as features believing that it may not have any correlation with the type of services provided by a corresponding device. The attacker appropriately labels (which indicates the device types) each of the MAC addresses present in the trace to properly correlate the high level activities performed by the corresponding devices. In the third step, the attacker provides the set of labels and signatures as input to train a model. The attacker can experiment with different models to obtain the best one for classification. The model training step also provides insight into which features were most helpful for classification. This, in turn, can be used as feedback to fine-tune the feature selection process. The attacker can repeat the three steps till she obtains a trained model with an optimized list of features for a large set of device types. She can then use this classification model to attack an unfamiliar environment.

Attack Phase. During the attack phase, the attacker eavesdrops wireless traffic from her target area. She performs feature extraction on the eavesdropped traffic and passes the list of signatures to her trained model. The model outputs the predicted device label for every MAC address present in the trace. The attacker can hence determine what type of devices are present in her environment.

4 Implementation of Framework

In this section, we describe the implementation details of the PrEDeC framework used to perform the attack described in Sect. 3. The framework is shown in Fig. 3. PrEDeC has six modules: data collector (optional), feature extractor, feature pruner, device annotator, model trainer and model tester. Each module is described in detail below.

4.1 Data Collector

The data collector performs passive eavesdropping of WiFi traffic and provides the sniffed WiFi frames as input to the rest of the framework. A number of

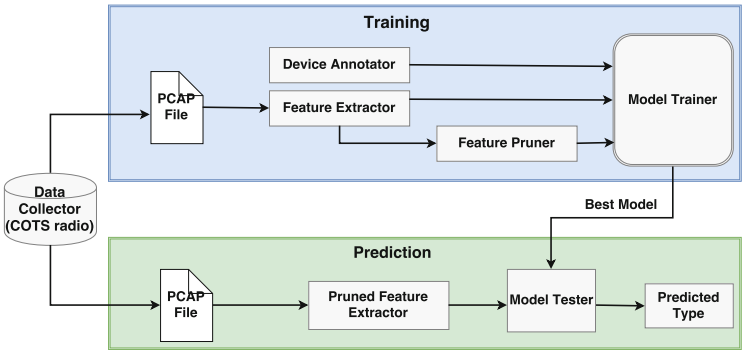

Fig. 3. PrEDec framework.

COTS tools can be used to capture the frames. In our case, we use a TP-Link TL-WN722N wireless adapter (in monitor mode) and Scapy (a python library for packet capture and analysis) to perform the traffic capture [15]. We set the adapter to perform channel hopping, so as to capture traffic on all fourteen WiFi channels. Since the adapter stays on a single channel at a time, frames on channels the adapter is not listening on will not be captured. Hence, the data collector captures a subset of the overall traffic. The sniffed frames are written to pcap files, which are then provided to rest of the framework as input.

4.2 Feature Extractor

The feature extractor extracts features from the frames collected by the data collector. We implement the feature extractor using Scapy. The feature extraction process consists of two steps. In the first step, certain fields are extracted from every frame in the PCAP file. These include the header information, the frame size and the timestamp of frame's capture. We refer to these fields as *basic* features. In the second step, the set of basic features are grouped based on the MAC addresses to compute, what we term as, *processed* features. However, since there may not be any correlation between the service(s) provided or activities performed and the MAC address used by a device, we do not use MAC address or manufacturer information (that can be obtained from a MAC address) as a feature in our work. This step takes one input parameter, called as *block size*. Block size indicates the minimum number of frames that is required to start computing the processed features. If an input pcap file contains more than the block size number of frames, we divide the frames into groups containing the block size number of frames and process each group separately. The processed features can be broadly grouped into five categories: rate, fraction, ratio, load and delta. Parameters used to compute the processed features are shown in Table 1.

Features in the *rate* category describe the rate at which frames are received or sent by a device. They are computed by calculating the number of frames of a particular type or sub-type to the observation window size. Features in

Table 1. Parameters used to calculate processed features.

Notation	Description
$T(f)$	Timestamp of a frame f
$First(mac, tp)$	$T(f)$ of first frame of type/sub-type tp containing MAC address mac
$Last(mac, tp)$	$T(f)$ of last frame of type/sub-type tp containing MAC address mac
$W(mac, tp)$	Observation window size = $Last(mac, tp) - First(mac, tp)$
$C(mac, tp)$	Number/size of frames with MAC address mac and type/sub-type tp
$S(mac, tp)$	Number/size of frames sent by a given mac of type/ sub-type tp
$R(mac, tp)$	Number/size of frames received by a given mac of type/ sub-type tp
$len(mac, tp)$	Size of frame of type/sub-type tp for mac
$gaps(mac, tp)$	Inter-arrival times of frame of type/sub-type tp for mac

the *fraction* category are of two types—aggregated and individual. Aggregated fraction features provide an indication of the contribution of each device to the total traffic. Individual fraction features provide an indication of the frame type composition for each device. The *ratio* category features look at the direction of the traffic and group it into sent and received traffic. Features in the *load* category calculate the mean and standard deviation of the sizes of different frames types. The *delta* category features consider the inter-arrival times of different frame types and calculate the mean and standard deviation of these times. The formulas to calculate the processed features are shown in Table 2. Features in the rate, fraction and ratio categories are computed in terms of both the number of frames and the sum of the sizes of the frames.

Table 2. Processed feature categories.

Feature	Notation	Calculation
Rate	$t(mac, tp)$	$\frac{C(mac,tp)}{W(mac,tp)}$
Fraction	$f_a(mac, tp)$	$\frac{C(mac,tp)}{C(all\_mac,tp)}$
	$f_i(mac, tp)$	$\frac{C(mac,tp)}{C(mac,all\_tp)}$
Ratio	$r_s(mac, tp)$	$\frac{S(mac,tp)}{C(mac,tp)}$
	$r_r(mac, tp)$	$\frac{R(mac,tp)}{C(mac,tp)}$
Load	$s_m(mac, tp)$	$mean(len(mac, tp))$
	$s_d(mac, tp)$	$std(len(mac, tp))$
Delta	$d_m(mac, tp)$	$mean(gaps(mac, tp))$
	$d_d(mac, tp)$	$std(gaps(mac, tp))$

4.3 Device Annotator

The device annotator assigns a label to each MAC address seen in an environment during the training phase. A label is determined by the functionality of

a device. If a device has multiple functionality, the label will be assigned based on the discretion of the user of the classifier. In this work, we perform manual labeling of the devices. Table 3 shows the set of devices in our experimental setup—their manufacturers, function(s) performed and the labels assigned by the device annotator, along with a numerical ID. In the rest of the paper, we shall use a device and a MAC address anonymously to indicate a same thing.

Table 3. Devices used in our experiments and their labels.

Label	Manufacturer	Function (s)	Device label
I	Belkin	WiFi access point	Access point
II	Withings/Netatmo	Surveillance camera	Camera
III	Intel	Laptop	Laptop
IV	Belkin	Access point using Ether MAC address	Other AP
V	Alfa	Personal desktop computer with WiFi adapter	PC
VI	HP	Printer, scanner, and fax machine	Printer
VII	Raspberry Pi	WiFi monitoring device	Rasp. Pi
VIII	Philips	Smart light controller	Smart light
IX	LG	Smartphone	Smartphone
X	Amazon Echo	Smart speaker	Speaker

4.4 Feature Pruner

Feature pruner removes those features that may be redundant for efficient classification. Features are extracted, by the extractor module, without looking into any feature-to-feature dependency or correlation. Feature pruner does some simple statistical analysis such as calculating standard deviation and variance inflation factor to identify a set of important features to be used for efficient classification. It follows the steps below to obtain a reduced set of features that can have higher impact on classification:

1. Removes a feature having a constant value (i.e., standard deviation $= 0$) across all the devices, assuming it may not have any impact on the classification. Sometimes, such features can be seen due to the fact that frames with certain sub-types are not seen in the network.
2. Removes a feature from a pair of features having high correlation coefficient. It finds all pairs of independent features with an absolute value of Pearson correlation coefficient greater than 0.5. For each of those pairs, it finds the VIF (Variance Inflation Factor) and discards the one having greater VIF. The procedure is repeated until no pairs have high correlation coefficient. We use 'usdm' package in R for this purpose.

4.5 Model Trainer

The model trainer takes two inputs—the set of signatures from the feature pruner and the set of MAC addresses to device types mapping from device annotator. Note that MAC address is used only for annotating a signature, not for classification. The trainer produces a trained model which will then be used in future predictions. We experiment with three supervised machine learning algorithms for the model trainer—CART (Classification And Regression Tree), RF (Random Forest) and SVM (Support Vector Machine). We implement the model trainer module using the following R packages: 'rpart', 'randomForest' and 'e1071' for CART, RF and SVM respectively. We perform 10-fold cross-validation using the 'caret' package when building the models. The model trainer also provides a ranking of features based on their importance in the classification task. This can help to fine-tune the set of features for future classification tasks.

4.6 Model Tester

The model tester takes a trained model and a set of signatures from the feature extractor and produces a predicted set of labels corresponding to each MAC address present in the signature set. It also provides feedback on the contribution of each feature towards the classification task.

5 Experimental Setup

In this section, we describe the experimental setup and the metrics we use to evaluate PrEDeC.

5.1 Data Collection

In our experimental setup, we use 22 devices which are grouped into eleven types based on functionality (Table 3). We collect three sets of WiFi traffic data under different scenarios and durations. The amount of activity of the devices in the experiments varies based on the scenario. An overview of our test sets is shown in Table 4. Set I is collected in an office environment on a working day, over a period of about 3 h. In this set, the devices have medium usage. Set II is collected over the weekend (36 h of data) when the devices are relatively inactive. Set III is collected under more controlled settings. The devices are placed inside a shielded room [14] and are subjected to heavy usage (high activity scenario) for about 9 h. Note that since Set I and Set II are not conducted inside the shielded room, they have a large number of unknown devices, which we exclude from our analysis. In addition to this, in Set II, there is a OnePlus smartphone which produces several probe requests with random MAC addresses. We do not consider these addresses in our evaluation and plan to explore the impact of MAC randomization on classification in future work. We divide the datasets into training and test sets for our evaluation.

Table 4. Datasets used for performance evaluation.

Set	Purpose	Period (in hours)	Total size (in MB)	#packets (×1000)	#MAC	#Types	#Unknown devices	#known devices
I	Train	2.2	109	450	244	10	224	20
I	Test	0.53	22	100	241	10	219	22
II	Train	28	157	700	252	10	240	22
II	Test	8	40	183	239	10	227	12
III	Train	6.00	891.3	2272	38	10	23	15
III	Test	2.82	372.7	972	37	10	22	15

5.2 Performance Metrics

We use the following metrics to evaluate the performance of PrEDeC:

1. Accuracy – The ratio of the number of correct predictions to the total number of predictions.
2. Precision – The ratio of the number of correct predictions to the total number of predictions for a particular type of device.
3. Recall – The ratio of the number of correct predictions to the number of devices present of a particular type.
4. F-score – The harmonic mean of precision and recall, i.e., f-score $= (2 * \text{precision} * \text{recall})/(\text{precision} + \text{recall})$

While the accuracy measures the overall correctness of the predictions, precision and recall provide an indication of performance on individual device types. F-score is the combined performance of precision and recall of a classification. These metrics can be derived from the confusion matrix.

6 Performance Evaluation

In this section, we evaluate the performance of our classifier, PrEDeC. The feature extractor of the classifier extracts 853 processed features for each distinct MAC address from the corresponding group of frames. Recall that feature extractor, at first, divides the frames in blocks, and then groups the frames in a block based on MAC address. Classification is performed in two phases: first, using all features (about 476 in number) having non-zero standard deviation, and then with features (about 49 in number) having low or no pairwise correlation.

We apply our classification on each data set separately and observe very different performance on every data set. For example, Set I and Set III produce relatively low accuracy (from 70–80% on an average for all the classifiers), whereas Set II produces very high accuracy (about 90% on average). Delving deeper, we observe that Set II has got the data set when the status of individual devices did not change much as it was collected during the weekend. Hence, signatures in both the training and testing data are very similar for every device.

However, Set I and Set III consist of traces where devices were actively used. Hence, the devices changed their statuses, which leads to relatively diverse signatures for them. We combine the data sets into one set to get a good mix of different training and testing signatures. The results reported in the rest of the paper are obtained from this combined set of data. Note that total number of signatures will vary depending on block size, which is an input parameter to our framework. To evaluate performance, we consider the same block size for training and testing in our experiments.

6.1 Classification Using All Features

We first investigate the performance of three classifiers—CART, RF and SVM for different block sizes. This set of experiments is performed using all features having non-zero standard deviation (476 features). Figure 4A shows the variation of overall accuracy with different block sizes (let us denote block size by Ω) for the three classifiers. Accuracy in the CART classifier varies from 45% to 60%, and it achieves maximum accuracy when $\Omega = 5$k, 10k, 20k or 30k. The SVM classifier shows a relatively higher variation in accuracy, between 35–90%, and it achieves more than 80% accuracy only when $\Omega = 30$k. Accuracy turns out to be more than 90% in the RF classifier irrespective of block size. Note that the block size and the number of signatures are inversely proportional. The RF classifier has high overall accuracy with Ω = 30k, 40k or 50k.

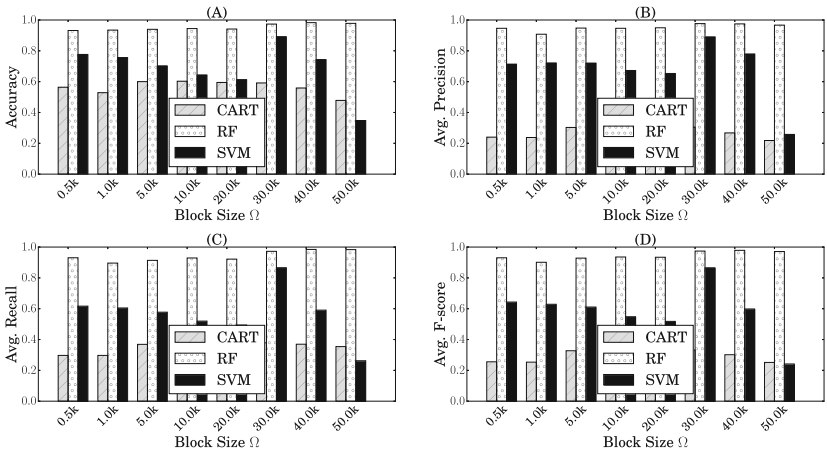

Fig. 4. (A) Overall accuracy (B) Average Precision (C) Average Recall (D) Average F-score for different block sizes in CART, RF, SVM.

Figure 4B shows that the precision (averaged over all device types) varies between 21–27% and 25–90% in CART and SVM respectively. In fact, we observe that the CART classifier fails to detect some types of devices such as the smart

speaker during our experiments (results not reported due to space constraints). It shows almost the same precision over $\Omega = 5$k to 30k. SVM classifier shows a similar trend as in the case of accuracy, with $\Omega = 30$k producing the highest precision. It is observed that SVM sometimes fails to classify a few types (for example, the Raspberry Pi) when Ω becomes more than 30k. The precision remains above 90% in RF classifier, and $\Omega = 30$k shows the highest precision though there is no large reduction in precision with higher Ω. This classifier detects all types with more or less same precision with any Ω.

Figure 4C shows that the recall (averaged over all device types) varies between 30–38% and 25–85% approximately in CART and SVM classifier respectively. SVM classifier finds all the devices only in case of $\Omega = 30$k. For other values of Ω, it fails to find 4–5 types of device on average. Again, it is interesting to observe that RF classifier provides consistently good (more than 90%) recall with any block size. Finally, F-score (averaged over all device types) shows that RF classifier outperforms the other two classifiers by a high margin in general, except for that in SVM with $\Omega = 30$k. This set of experiments clearly shows that the RF classifier performs much better than other models irrespective of block size. The block sizes of both 30k and 40k show better results in all three classifiers with any metric in general. However, we chose smaller block size (30k) as it can potentially improve run-time efficiency. Hence, the results in the rest of this paper are reported based on RF classifier with 30k block size.

6.2 Classification Using Optimized Set of Features

We analyze the classification performance with a reduced set 49 of features after VIF analysis (see Sect. 4.4). We report the results only for the RF classifier with 30k block size. Table 5 shows the change in the performance with the reduced feature set when classifying the same set of data used in the previous section. The confusion matrix is shown in Table 6. We notice that the accuracy, precision, recall and f-score reduce by approximately 2%. However, since the reduction in number of features is very high ($476 - 49 = 427$ features are removed after VIF), this analysis can be useful in improving the efficiency of the framework without a significant reduction in accuracy.

Table 5. Change in accuracy, precision, recall and f-score due to feature pruning.

	Accuracy	Precision	Recall	F-score
All features	0.98	0.98	0.97	0.97
% Change with reduced features	−1.64	−1.31	−3.17	−2.29

Furthermore, we look at the distribution of importance of the 49 features in RF classifier to find other avenues for feature reduction. The importance value of each of these features, grouped by category, can be seen in Table 7.

Table 6. Confusion matrix of RF classifier with 30k block size. Columns and rows indicate the number of the actual tags and the predicted tags respectively. Type IDs here are as shown in Table 3.

Type ID	I	II	III	IV	V	VI	VII	VIII	IX	X
I	36	0	0	0	0	0	0	0	0	0
II	0	125	0	0	1	0	0	0	2	1
III	0	0	36	0	0	0	0	0	1	3
IV	0	0	0	32	0	0	0	0	0	0
V	0	4	0	1	111	0	0	1	1	0
VI	0	0	0	0	0	36	0	0	0	0
VII	0	0	0	0	0	0	6	0	0	0
VIII	0	0	0	0	0	0	0	29	0	0
IX	0	2	2	0	0	0	0	0	29	0
X	0	1	0	0	0	0	0	0	0	32

Table 7. Mean and standard deviation of the importance values of the features in reduced set, for each category of features.

Category	Parameter	Mean % importance	Standard deviation
Rate	Frames count	1.21	1.32
	Frame size	0.76	0
Fraction	Frames count	0.02	0.02
	Frame size	2.73	0.90
Ratio	Frame count	0.03	0.02
	Frame size	4.76	4.35
Load	Mean	3.14	3.59
	Standard deviation	1.16	1.61
Delta	Mean	2.12	3.59
	Standard deviation	2.70	0
Number	Frame count	0.21	0.32
	Frame size	2.68	1.74

We observe that only 2 features have a percentage importance greater than 10%. These belong to *ratio* and *delta* category of features. The feature contributing least belongs to *delta* category. We report the distribution of values to the types of devices for two features, the most important and the least important ones (Fig. 5). The most important feature belongs to ratio category, and it is the ratio of the size of the data frames received by a MAC address to the size of all the data frames seen in a block. The least important feature belongs to delta category and it is the mean time gap of the management type frames with sub-type reserved for a MAC address. We see that the printer, camera, smart

light and Raspberry Pi show much lower range of values in the ratio feature. The delta feature shows lower spread in Raspberry Pi and smart speaker. We plan to explore the classification efficiency by selecting features based on their importance values in future work.

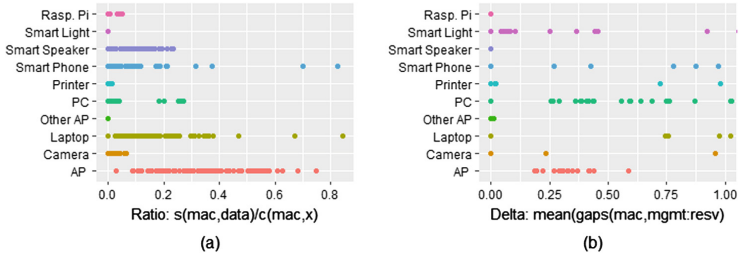

Fig. 5. Distribution of values of the most (A), and least (B) important feature in RF classifier belong to ratio and delta categories respectively (see Sect. 4.2).

6.3 Optimal Traffic Size

In this section, we investigate the number of frames needed to build a set of effective signatures and the time period required to eavesdrop them. We consider only the 30k block size in this analysis.

We see that it takes significantly different time durations on average to sniff 30k frames in the three data sets: about 550 s, 4200 s and 200 s for Set I, Set II and Set III respectively. In addition, standard deviation on time durations is very high in each of the data sets. We then look into average number of frames per group used to build signatures. Recall that frames in a block are grouped based on MAC address. It is observed that there is no direct correlation (Pearson correlation coefficient $= -0.06$) between the number of frames and the observation window size for any particular device in our (both individual and combined) datasets. However, we look for an average frame count and corresponding observation window size to get a rough estimation of the time required to sniff packets in our scenarios. We plan further investigation in this direction as future work.

Table 8 shows the average number of frames and the average observation window size used to build signatures for different types of devices in different data sets. For example, on an average, 1450 frames have been used to build signatures for the cameras and 771 s are needed to sniff those many frames in our combined data set. This analysis can be useful for an attacker to get an estimation of number of WiFi frames and time to eavesdrop them to achieve a certain accuracy.

6.4 Open Environment Experiment

We examine the performance of PrEDeC in an open, i.e., uncontrolled, environment using the RF classifier. We collect 1.1 million frames in 3.64 h in a lounge

Table 8. Number of frames (np) and corresponding window size (ws) per device types when 30k blocks are used.

	Set I		Set II		Set III		Combined set	
Device type	np	ws	np	ws	np	ws	np	ws
AP	5230	525	17088	4310	22245	290	19550	1182
Camera	257	359	2060	4229	1474	271	1450	771
Laptop	35	256	4683	4304	7024	288	5133	1041
Other AP	17	183	214	4126	21	213	67	1143
PC	114	446	1138	4198	142	279	331	1086
Printer	800	512	7271	4301	742	289	2161	1178
Raspberry Pi	16	251	1593	4154	-	-	1120	2983
Smart Light	483	258	150	4093	24	255	102	1181
Smart Phone	844	318	-	-	3094	257	2492	273
Smart Speaker	3543	427	382	4178	1121	269	974	1193
Average	1133.9	353.5	3842.1	4210.3	3987.4	267.8	3338	1203.1

area at our university. The RF classifier is trained using the data sets with 30k block size mentioned in the previous sections. We note that in this scenario we do not have the ground truth. However, by intuition, we would expect to see a larger number of PCs, laptops, and smartphones in a university. We would also expect to see a number of access points and IP cameras.

Table 9. Predicted device types in an open university environment.

Device type	AP	Camera	Laptop	Other AP	PC	Printer	Rasp. Pi	Smart Light	Smart phone	Smart speaker
Count	1	43	34	1	184	3	0	5	256	2

Table 9 shows the results of our prediction using PrEDeC classifier. We get a total of 3446 signatures with 529 unique MAC addresses in this dataset. We observe that PrEDeC detects larger proportion of smartphones and PCs as compared to the other devices. However, PrEDec detects only one access point despite the presence of multiple access points in the test environment. Additionally, the classifier detects 43 cameras, which is higher than the expected number. On closer inspection, we observed that several access points were classified as cameras. This is probably because we have only one access point in our training set. We hope to increase the accuracy of classifying access points by including more number of access points while training our classifier.

7 Related Work

Significant research was conducted in the area of device fingerprinting using link-layer information. Cache [4] used the duration field in 802.11 frames to identify various WiFi drivers. Franklin et al. [6] performed passive fingerprinting of 802.11 drivers by analyzing the inter-arrival times of probe request frames sent by different drivers. They employed a Bayesian classification method to classify 17 drivers. Their approach was fine-tuned in [5] to distinguish different operating systems using the same driver. Since different device types can have the same driver, our work has a more fine-grained classification in comparison to these works. Pang et al. [12] identified four metrics (network destinations, SSID probes, broadcast frame sizes, MAC protocol fields) that would help identify users from a network trace. Out of these, three metrics could be used even with link-layer encryption. In comparison to [12], our work focuses on device classification rather than user identification, and uses a larger set of features.

Hardware specific features such as clock skews [2,8] or radiometric signatures [3,17] were used to identify unauthorized access points and wireless cards. However, these features require precise measurement techniques or modification of the wireless monitor's driver. Hence, we do not consider these techniques in detail in relation to our work.

Another related area of research investigates classification of application behavior rather than device classification. Alshammari and Zincir-Heywood [1] evaluated the performance of five learning algorithms in identifying SSH and Skype traffic using flow based features. Korczyński and Duda [9] identified Skype service flows from TLS encrypted traffic. AppScanner [16], was a framework that used bursts and flows in the network traffic to fingerprint Android apps (using Random Forest). Our work is partly motivated by these works which suggests that encrypted link layer traffic can also become a useful tool to detect the presence of certain type of services provided by the WiFi enabled devices, and such detection would not require any active access to network infrastructure.

Zhang et al. [19] implemented a system to classify the type of online user activity based on link-layer traffic features such as the size distribution, direction, inter-arrival time and type of frames. They used SVM (Support Vector Machine) and RBFN (Radial Basis Function Network) algorithms to perform the classification. Wang et al. [18] employed similar techniques to detect apps used by a user on a smartphone by analyzing the link-layer traffic. They extracted features such as frame size, frame direction and frame inter-arrival time to train a Random Forest classifier and tested their implementation on 13 apps. [18,19] are the most comparable to our work since we utilize a similar, though larger, set of features in our classification framework. In addition, we focus on classifying device types rather than user activity.

Relatively little work has been done so far in IoT device type classification. Miettinen et al. [11] developed the IoT SENTINEL, a system that could identify the type of IP-enabled devices connected to an IoT network and restrict traffic from those identified as vulnerable. Meidan et al. [10] introduced ProfilIoT, a system that analyzed TCP sessions to distinguish IoT and non-IoT devices and

identify the device class. Both these works employed machine learning techniques to perform the classification. However, in contrast to our work, they look at wired network traffic and make use of higher layer traffic information for their classification. A comparison of the proposed framework, PrEDeC, and selected related work is shown in Table 10.

Table 10. PrEDeC features vs. related work.

Related work	Layer	Classified entity	Wired/ wireless	Active/ passive	COTS
Proposed work	Link	Device type	Wireless	Passive	Yes
Franklin et al. [6]	Link	Device driver	Wireless	Passive	Yes
Jana and Kasera [8]	Link	Device type	Wireless	Passive	No
Alshammari and Zincir-Heywood [1]	Transport	Service	Wireless	Passive	Yes
Zhang et al. [19]	Link	User	Wireless	Passive	Yes
Miettinen et al. [11]	Multiple*	Device Type	Wired	Passive	No

*uses link, network, transport and application layer features.

8 Conclusion

In this paper, we designed PrEDeC, a framework that allows an attacker to classify device types using encrypted link-layer WiFi traffic obtained by passive eavesdropping with COTS radios. We extracted 853 features from the WiFi frames based on properties such as size, timing and type. We optimized them to a set of 49 features and evaluated three well-known machine learning algorithms: Decision Tree, Random Forest and Support Vector Machine, to classify devices. We evaluated the performance of our classifier using a data set collected in our lab environment of three different traffic scenarios, totaling about 48 h and 5.2 million frames. Our results showed that an accuracy of about 95% can be achieved using the Random Forest classifier. Our investigation revealed that an average of 3300 packets and an observation window size of about 1200s to sniff them are needed to build effective signatures to achieve this accuracy in our scenarios. It is observed that this estimation can vary significantly depending on the status of the devices present in the target network, and further investigation in this direction is planned as a future work. Finally, we have tested our classifier in an open and uncontrolled university area, and we are successful in detecting devices like laptops, smartphones and desktop computers in high numbers. However, our analysis in this case shows that it requires a larger number of devices of certain types like smart lights and cameras to achieve more precise classification.

References

1. Alshammari, R., Zincir-Heywood, A.N.: Machine learning based encrypted traffic classification: identifying SSH and Skype. In: IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009, pp. 1–8. IEEE (2009)

2. Arackaparambil, C., Bratus, S., Shubina, A., Kotz, D.: On the reliability of wireless fingerprinting using clock skews. In: Proceedings of ACM Conference on Wireless Security (WiSeC), pp. 169–174. ACM (2010)
3. Brik, V., Banerjee, S., Gruteser, M., Oh, S.: Wireless device identification with radiometric signatures. In: Proceedings of the Conference on Mobile Computing and Networking (MobiCom), pp. 116–127. ACM (2008)
4. Cache, J.: Fingerprinting 802.11 implementations via statistical analysis of the duration field. In: Uninformed.org, vol. 5 (2006)
5. Desmond, L.C.C., Yuan, C.C., Pheng, T.C., Lee, R.S.: Identifying unique devices through wireless fingerprinting. In: Proceedings of ACM Conference on Wireless Security (WiSeC), pp. 46–55 (2008)
6. Franklin, J., McCoy, D., Tabriz, P., Neagoe, V., Van Randwyk, J., Sicker, D.: Passive data link layer 802.11 wireless device driver fingerprinting. In: Proceedings of the USENIX Security Symposium, Berkeley, CA, USA (2006)
7. Gartner (2015). http://www.gartner.com/newsroom/id/3165317
8. Jana, S., Kasera, S.K.: On fast and accurate detection of unauthorized wireless access points using clock skews. IEEE Trans. Mob. Comput. **9**(3), 449–462 (2010)
9. Korczyński, M., Duda, A.: Classifying service flows in the encrypted Skype traffic. In: 2012 IEEE International Conference on Communications (ICC), pp. 1064–1068. IEEE (2012)
10. Meidan, Y., Bohadana, M., Shabtai, A., Guarnizo, J.-D., Ochoa, M., Tippenhauer, N.O., Elovici, Y.: ProfilIoT: a machine learning approach for IoT device identification based on network traffic analysis (poster). In: Proceedings of the Security Track at ACM Symposium on Applied Computing (SAC), April 2017
11. Miettinen, M., Marchal, S., Hafeez, I., Asokan, N., Sadeghi, A.-R., Tarkoma, S.: IoT sentinel: automated device-type identification for security enforcement in IoT. arXiv preprint, December 2016. arXiv:1611.04880v2
12. Pang, J., Greenstein, B., Gummadi, R., Seshan, S., Wetherall, D.: 802.11 user fingerprinting. In: Proceedings of the Conference on Mobile Computing and Networking (MobiCom), pp. 99–110 (2007)
13. SENRIO (2016). http://blog.senr.io/blog/400000-publicly-available-iot-devices-vulnerable-to-single-flaw
14. Siboni, S., Shabtai, A., Elovici, Y., Tippenhauer, N.O., Lee, J.: Advanced security testbed framework for wearable IoT devices. ACM Trans. Internet Technol. (TOIT) **16**(4), 26 (2016)
15. Siby, S., Maiti, R.R., Tippenhauer, N.O.: IoTScanner: detecting privacy threats in IoT neighborhoods. In: Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security, pp. 23–30. ACM (2017)
16. Taylor, V.F., Spolaor, R., Conti, M., Martinovic, I.: Appscanner: automatic fingerprinting of smartphone apps from encrypted network traffic. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 439–454. IEEE (2016)
17. Ureten, O., Serinken, N.: Wireless security through RF fingerprinting. Can. J. Electr. Comput. Eng. **32**(1), 27–33 (2007)
18. Wang, Q., Yahyavi, A., Kemme, B., He, W.: I know what you did on your smartphone: inferring app usage over encrypted data traffic. In: 2015 IEEE Conference on Communications and Network Security (CNS), pp. 433–441. IEEE (2015)
19. Zhang, F., He, W., Liu, X., Bridges, P.G.: Inferring users' online activities through traffic analysis. In: Proceedings of ACM Conference on Wireless Security (WiSeC), pp. 59–70. ACM (2011)