IoT Devices Recognition Through Network Traffic Analysis

Mustafizur R. Shahid, Gregory Blanc, Zonghua Zhang, Hervé Debar
CNRS SAMOVAR UMR 5157, Institut Mines-Télécom, France
{mustafizur.shahid, gregory.blanc, herve.debar}@telecom-sudparis.eu
zonghua.zhang@imt-lille-douai.fr

Abstract

The growing Internet of Things (IoT) market introduces new challenges for network activity monitoring. Legacy network monitoring is not tailored to cope with the huge diversity of smart devices. New network discovery techniques are necessary in order to find out what IoT devices are connected to the network. In this context, data analysis techniques can be leveraged to find out specific patterns that can help to recognize device types. Indeed, contrary to desktop computers, IoT devices perform very specific tasks making their networking behavior very predictable. In this paper, we present a machine learning based approach in order to recognize the type of IoT devices connected to the network by analyzing streams of packets sent and received. We built an experimental smart home network to generate network traffic data. From the generated data, we have designed a model to describe IoT device network behaviors. By leveraging the t-SNE technique to visualize our data, we are able to differentiate the network traffic generated by different IoT devices. The data describing the network behaviors are then used to train six different machine learning classifiers to predict the IoT device that generated the network traffic. The results are promising with an overall accuracy as high as 99.9% on our test set achieved by Random Forest classifier.

1. Introduction

The total number of IoT devices is expected to reach 75 billion by 2030 [9]. This growing IoT market introduces new challenges for network administrators [7]. With the huge diversity of IoT devices (thermostat, camera, smart bulb, etc), device recognition is of tremendous importance. Indeed, with the increase of IoT malware such as Mirai, securing IoT networks has become critical. At its peak, Mirai infected more than 600,000 devices around the world [2]. The infected devices were primarily used to perform DDoS attacks. In this context, knowing the type of device connected to the network will help to enforce security. For example, knowing that a device is a security camera from a specific manufacturer can help the network administrator to specify filtering rules that will not allow the camera to do anything else than what it is expected to do. Device type recognition can also be used to block the access to the network of devices considered to be vulnerable.

IoT device recognition can also be used by an attacker to discover vulnerable IoT devices by performing passive network traffic analysis. Indeed, in a wireless IoT network, it is easier for an attacker to capture network traffic to perform further analysis with the purpose of recognizing the type of connected devices even if the traffic is encrypted. Device fingerprinting through network traffic analysis can also help malware to identify vulnerable devices passively. Instead of actively looking for a device to infect, passive vulnerable device discovery will reduce the network signature of the malware making its detection even more difficult for intrusion detection system. Device type recognition also raises privacy concerns. Once a device has been identified, techniques presented in [1] can be used to further determine the current state of the device. In the case of a smart home, such information can help to infer what is happening inside the house leading to potential privacy breaches.

We define an IoT device as being a device that is intended to perform a specific task. Indeed, contrary to laptops, desktop computers or smartphones, IoT devices perform very specific tasks. For example, a smart plug or a smart bulb can be switched on or off. In the case of a smart bulb, one can also set its brightness. However, neither a smart plug nor a smart bulb is supposed to watch video on youtube or send emails. As a result, IoT devices network traffic follows a stable pattern. The generated network traffic being very predictable, it is well suited for machine learning techniques.

In this paper, we present a method to recognize IoT devices by analyzing the generated network traffic. To this purpose, the raw network traffic is preprocessed to extract bidirectional flows described by features such as the size of the first N packets sent and received, along with the inter-

arrival times. Because of the lack of publicly available data, we built an experimental smart home network composed of four devices to generate network traffic data. We then perform t-SNE on our dataset to visualize and have an insight of the data. Moreover, the visual clusters obtained point out the effectiveness of our selected set of features in differentiating the different IoT devices. Next, we train and test six different machine learning algorithms to perform network traffic classification. We are able to predict what device has generated the network traffic with an accuracy as high as 99.9% on our test set for the Random Forest classifier.

The rest of the paper is organized as follows: Section 2 reviews the related works and Section 3 describe the features used in our model. In Section 4, we present the experimental results including data visualization and classification. In Section 5, we discuss the possible improvement of the model. Finally Section 6 concludes and presents the possible future works.

2. Related Works

A few number of works focusing on IoT device fingerprinting have emerged due to the rapid development of IoT networks. Y. Meidan et al. propose a machine learning based network traffic analysis approach to identify IoT devices in order to create white lists of authorized devices [11][10]. They use features extracted from full TCP sessions (from SYN to FIN). T. D. Nguyen et al. present a system to detect compromised IoT devices [13]. They take advantage of the temporal periodicity of traffic generated by IoT devices. Features to identify devices include periodic flows characteristics, period accuracy, period duration and period stability. First, normal communication profiles are created for individual devices. A recurrent neural network is then used to detect any deviation from the expected behavior. M. Miettinen et al. present a method to identify the type of an IoT device being connected to the network in order to constrain the communications of vulnerable devices [12]. They use a wide variety of features extracted during the setup phase of the device. Contrary to the other mentioned approaches, they train one classifier per device type. B. Bezawada et al. describe a method to perform device behavioral fingerprinting [4]. They use a subset of the features from [12] along with payload based features to train classifiers. R. Doshi et al. perform network traffic classification to detect DDOS attack in IoT networks [6]. Our work on IoT device recognition through network traffic classification differs from existing ones in that we use a very different set of features that are easily extractable even from encrypted network traffic.

Other works examine privacy related issues like extracting sensitive information about the current state of the devices. Hence, A. Acar et al. point out how an adversary can determine current activities of the users in a smart home by profiling network traffic using machine learning algorithms [1]. N. Apthorpe et al. also show that simply analyzing network traffic transmission and reception rates can reveal sensitive user interactions with the device [3]. For example, network transmission and reception rates of the Nest indoor security camera can allow an adversary to determine if there is movement inside a smart home. Similarly, Copos et al. study two popular smart devices, the Nest Thermostat and the wired Nest Protect to show that an attacker can infer if a home is occupied or not, only by analyzing the network activity [5]. The primary goal of our work is not to explore privacy related issues but rather to provide a way of recognizing IoT devices connected to a network.

3. Features Description

The purpose of our work is to provide a mean to recognize devices based on their network behavior. Therefore, we need to define features that will appropriately describe the network activity. In past smart device identification works features are extracted from full TCP sessions [10][11]. The issue with this method is that we have to wait until the end of the session in order to be able to extract all the features. For some IoT devices such as the Nest security camera TCP sessions can last for days. Another approach is to focus only on the network activity during the setup phase to identify the device [12]. However, this method is of no help if we have missed the setup phase of the device. For example, if we want to sniff an existing network in which all the devices are already setup. Another constraint is that the features have to be extractable even if the network traffic is encrypted.

For the model to be easily implementable in real world networks, one needs to keep features as simple as possible. We propose to work with bidirectional flows identified by their source and destination IP addresses and ports. In the case of long TCP connections, we do not capture the whole session. A timeout is used to split long connections into several bidirectional flows. Each bidirectional flow is described by a feature vector composed of the following:

- The size of the first N packets sent
- The size of the first N packets received
- The N - 1 packet inter-arrival times between the first N packets sent
- The N - 1 packet inter-arrival times between the first N packets received

The size of the packets and the inter-arrival times have proven their effectiveness in works on application identification [14][8]. All feature vectors must have the same size.

Figure 1: Experimental smart home network

If a bidirectional flow contains less than N packets, the remaining fields of the vector are set to zero. The value of N is set empirically. The more devices we have to classify the greater N has to be so that the classifier will have enough information to accurately differentiate the bidirectional flows generated by the different devices. Meanwhile, N has to be as small as possible for performance related issues. In Section 4.3, we will examine the impact of the variable N on the overall accuracy of a classifier.

4. Experimental Results

In this Section, we describe the dataset generation process and then present the results of visualization and classification. For the rest of our work, the variable N, defined in Section 3, is equal to 10. That is, the total number of features is equal to 38. Other values of N will be tested in Section 4.3. A timeout of 600 seconds is used to split long connections into multiple bidirectional flows.

4.1 Smart Home Dataset

A small smart home network is built to generate network traffic. The experimental smart home consists of four IoT devices: a Nest security camera, a D-Link motion sensor, a TP-Link smart bulb and a TP-Link smart plug. The four devices connect to the Internet through a wireless access point. The network traffic is collected thanks to a Raspberry Pi placed between the wireless access point and the Internet as shown in Figure 1. The raw network traffic is then preprocessed the following way:

1. The MAC addresses of the devices are used to split the network traffic into different pcap files corresponding to different IoT devices. This will facilitate the labeling of the dataset.

Table 1: Total number of bidirectional flows per device

	train	test
D-Link Motion Sensor	867	207
Nest Security Camera	839	216
TP-Link Smart Bulb	821	219
TP-Link Smart Plug	695	163
Total	**3222**	**805**

2. The bidirectional flows along with their timestamp and protocol are extracted from the different pcap files. Only TCP flows are kept as all of the devices use HTTP or HTTPS for communications.

3. The bidirectional flows of the different IoT devices are merged to form a single dataset. The timestamp is used to reorder the flows.

Network traffic has been collected for a total of 7 days. The training set has been collected during the first five and a half days. The test set has been collected during the remaining days. The training set and the test set are composed of 3222 and 805 instances respectively. Table 1 shows the number of flows for each device in the training and test sets. For the rest of the work, data analysis is performed using the python libraries scikit-learn and tensorflow.

4.2. Data Visualization

In this section, we present the results of data visualization using t-Distributed Stochastic Neighbor Embedding (t-SNE) which is a non-linear dimensionality reduction technique. t-SNE has no knowledge of the label of the input data. It is an unsupervised learning algorithm. Data visualization is important to get an insight of our 38-dimensional data and to assess the discriminative power of our model. That is, we want to know if the features we have selected to describe the network behavior are discriminative enough to differentiate the network traffic generated by different devices.

t-SNE outperforms many other non-parametric data visualization and exploration techniques [15]. Another commonly used dimensionality reduction technique is Principal Component Analysis (PCA). The limitation of PCA is that like any other linear dimensionality reduction methods, it only focuses on placing dissimilar data points far apart in the lower dimension. It does not attempt to place similar data points close together. Differently, t-SNE attempts to represent similar data points close to each other while preserving the global structure of the dataset. Therefore t-SNE is a much better option for visualization.

The obtained visual representation of the data is shown in Figure 2. The data points form visual clusters corresponding to the network traffic generated by different IoT

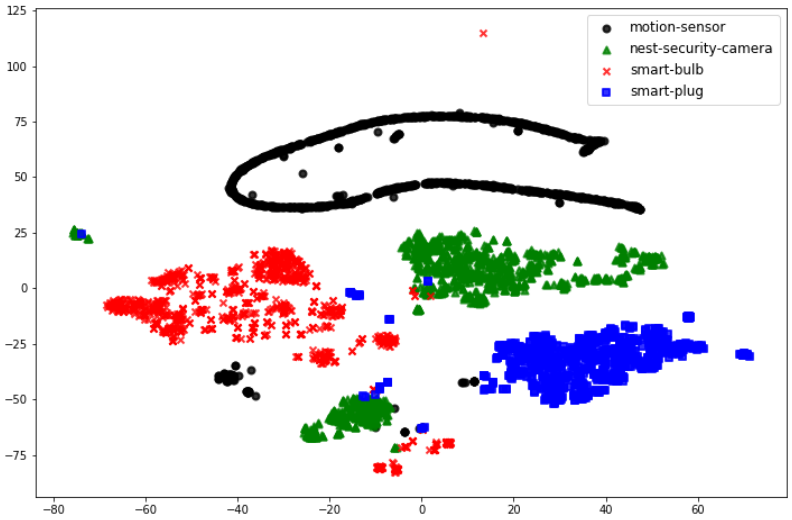

Figure 2: Dataset visualization using t-SNE

devices. Most of the data points corresponding to the same IoT device lie close to each other whereas data points from different IoT devices lie far apart. The obtained representation indicates that the features selected to describe the network traffic are discriminative enough to distinguish between the different IoT devices.

4.3. Classification

In this section we test different machine learning algorithms to classify the bidirectional flows according to the IoT device they belong to. Six different classification algorithms are tested: Random Forest, Decision Tree, SVM (with rbf kernel), k-Nearest Neighbors, Artificial Neural Network (ANN) and Gaussian Naïve Bayes. Most of these algorithms have regularization parameters also referred to as the hyperparameters that need to be tuned. During the training phase, a validation set consisting of 25% of the training set is used to fine tune the hyperparameters. Once the best parameters have been found, the classifier is retrained on the whole training set. The ANN is a fully connected feedforward neural network consisting of two hidden layers with 10 neurons each and using a 0.5 dropout rate.

The performance of the different algorithms are measured on the test set. The metrics used are accuracy, precision, recall and F1 score. The accuracy of the classifier is the proportion of flows that are correctly classified. To assess the overall precision, recall and F1 score, micro-averaging is used. Micro-averaging is preferred over macro-averaging because it takes class imbalance into account. Let us consider our 4-class classification problem. The four classes are $device_1$, $device_2$, $device_3$ and $device_4$. Let TP_i, TN_i, FP_i and FN_i be the number of true positive, true negative, false positive, and false negative respectively for $device_i$. The micro-average of precision, recall and F1 score are given by:

$$microAvPrecision = \frac{\sum TP_i}{\sum TP_i + \sum FP_i}$$

$$microAvRecall = \frac{\sum TP_i}{\sum TP_i + \sum FN_i}$$

$$microAvF1Score = 2\frac{microAvPrecision.microAvRecall}{microAvPrecision + microAvRecall}$$

The precision, recall and F1 score for each device are also calculated individually as shown in Tables 3, 4 and 5. This is done by simply considering the problem as if it was a binary classification problem, with the device we are calculating the performance for, corresponding to the positive class. In that case, precision, recall and F1 score is given by:

$$device_i precision = \frac{TP_i}{TP_i + FP_i}$$

$$device_i recall = \frac{TP_i}{TP_i + FN_i}$$

$$device_i F1score = 2\frac{device_i precision.device_i recall}{device_i precision + device_i recall}$$

Gaussian Naive Bayes is the algorithm that performs the worst with an overall accuracy of 91.9%. All other algorithms achieve a high performance with an overall accuracy on the test set ranging between 98.6% and 99.9%. The best performance is achieved by the Random Forest classifier with equally high overall accuracy, precision and recall. Despite the relative small size of our dataset, ANN achieves

Table 2: Overall performance on the test set of the different classifiers

	accuracy	micro-av. precision	micro-av. recall	micro-av. F1 score
RF	.999	.999	.999	.999
DT	.995	.995	.995	.995
SVM	.993	.993	.993	.993
KNN	.989	.989	.989	.989
ANN	.986	.986	.986	.986
GNB	.919	.919	.919	.919

Table 3: Precision on the test set of the different classifiers and for specific devices

	sensor	camera	bulb	plug
RF	1.	1.	.995	1.
DT	.986	1.	.995	1.
SVM	1.	.977	.995	1.
KNN	1.	.977	.986	.994
ANN	.986	.986	.978	1.
GNB	1.	.771	1.	.993

an overall accuracy of 98.9%. The performance of the ANN can be improved by collecting more network traffic data to increase the size of the dataset. These positive experimental results indicate that it is possible to recognize IoT devices with high accuracy by passively analyzing the network traffic.

Let N be the number of packets sent and received that are taken into consideration by a classifier as defined in Section 3. To analyze the impact of the variable N on the overall accuracy, we only consider Random Forest classifier as it is the classifier that achieved the best performance. Hence, we train multiple Random Forest classifiers with different values of N, ranging from 2 to 10, to find out the optimal value of N for our network. Figure 3 shows the overall accuracy achieved by training classifiers with different values of N. The overall accuracy increases as the value of N goes up. Surprisingly, when N is set to 2 the classifier still achieves a high accuracy of 98.9%. Such a high accuracy is reached even with a small value of N because our experimental network is very small and consists of only four different devices. Therefore, the size of the first two packets sent and received and the corresponding inter-arrival times, provide enough information to the classifier to accurately differentiate between the different IoT devices. Indeed, the greater the number of different devices connected to the network is, the more packets the classifier has to consider in order to accurately differentiate the flows corresponding to the different devices. The accuracy reaches the maximum value of 99.9% for N equal to 6 and higher for our experimental

Table 4: Recall on the test set of the different classifiers and for specific devices

	sensor	camera	bulb	plug
RF	1.	1.	1.	.994
DT	1.	.991	.995	.994
SVM	.995	1.	1.	.969
KNN	.990	1.	.986	.975
ANN	.995	.986	1.	.957
GNB	.971	1.	.785	.926

Table 5: F1 score on the test set of the different classifiers and for specific devices

	sensor	camera	bulb	plug
RF	1.	1.	.997	.997
DT	.993	.995	.995	.997
SVM	.997	.988	.997	.984
KNN	.995	.988	.986	.984
ANN	.990	.986	.989	.978
GNB	.985	.871	.880	.958

network. Hence, if we take resource efficiency into consideration in the case of our network, the optimal value of N for the Random Forest classifier is 6.

5. Discussion

One limitation of our study is the limited number of devices used. This is a common limitation of most IoT related works because of the lack of publicly available data [6][3][5]. The experimental smart home network is composed of only four devices. The model should be trained and tested on larger IoT networks containing a variety of devices. As the number of different devices connected to the network increases, the more resources will be required to train the model. With thousands of different types of device available in the market, it becomes difficult to deal with a single classifier for every device. If a new device has to be incorporated to the dataset or if a system update changes the network behavior of a single device, a new model should be retrained on the whole dataset. One solution is to train a single classifier for each device as in [12]. However, in that case all the different classifiers have to be run in parallel in order to recognize the device, increasing the resource used during the operational use of the model. An intermediate solution could be to train classifiers for groups of devices that share similar behaviors.

Another limitation is that our experimental network is composed exclusively of IoT devices which might not be the case in the real world. In a general-purpose network most activity will be generated by smartphones or laptops.

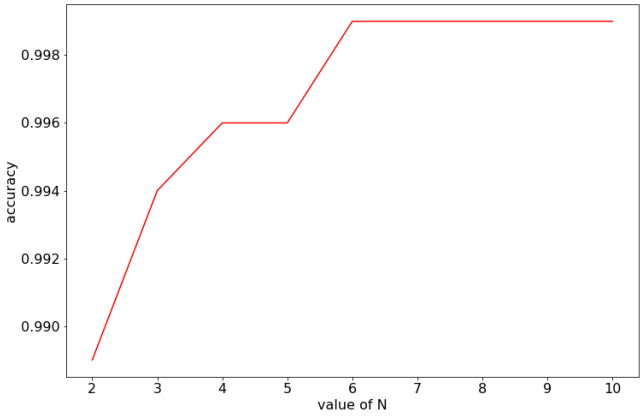

Figure 3: Overall accuracy achieved by Random Forest classifier for different values of N

Given the wide variety of tasks performed by a laptop, it is completely possible that some flows it generates share similar characteristics to the one generated by a security camera and hence end up being identified as being a security camera. However the proposed approach can be easily adapted to general-purpose networks in which smartphones or laptops are also connected. One can add an extra class to the classifier consisting of all the non-IoT flows. Then majority voting on a sequence of classified flows, similarly to what is proposed in [11], can be used to identify the device. For example, if the following sequence is obtained *(camera, non-IoT, camera, non-IoT, non-IoT)* then the device is considered as not being an IoT device.

6. Conclusion

The purpose of this work was to propose a method to recognize IoT devices by analyzing network traffic data. To describe the network behavior, we selected features such as the size of the first N packets sent and received, and their corresponding inter-arrival times. Visualization using t-SNE pointed out the effectiveness of our selected set of features in distinguishing the network traffic generated by different devices. Different machine learning algorithms were then used to classify the network traffic. An overall accuracy of 99.9% has been achieved by the Random Forest classifier. However, further studies need to be performed in order to assess our method on larger datasets. The method should be tested on networks composed of a greater number of different IoT devices.

References

[1] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and A. S. Uluagac. Peek-a-boo: I see your smart home activities, even encrypted! *CoRR*, 2018.

[2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou. Understanding the mirai botnet. In *Proceedings of the 26th USENIX Conference on Security Symposium*, SEC'17, 2017.

[3] N. Apthorpe, D. Reisman, and N. Feamster. A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic. *CoRR*, 2017.

[4] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray. Iotsense: Behavioral fingerprinting of iot devices. *CoRR*, 2018.

[5] B. Copos, K. Levitt, M. Bishop, and J. Rowe. Is anybody home? inferring activity from smart home network traffic. In *2016 IEEE Security and Privacy Workshops (SPW)*, 2016.

[6] R. Doshi, N. Apthorpe, and N. Feamster. Machine learning ddos detection for consumer internet of things devices. In *2018 IEEE Security and Privacy Workshops (SPW)*, 2018.

[7] M. A. Khan and K. Salah. Iot security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems*, 2018.

[8] W. Linlin, L. Peng, M. Su, B. Yang, and X. Zhou. On the impact of packet inter arrival time for early stage traffic identification. In *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2016.

[9] Louis Columbus. Roundup Of Internet Of Things Forecasts And Market Estimates, 2016. https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/.

[10] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici. Profiliot: A machine learning approach for iot device identification based on network traffic analysis. In *Proceedings of the Symposium on Applied Computing*, 2017.

[11] Y. Meidan, M. Bohadana, A. Shabtai, M. Ochoa, N. O. Tippenhauer, J. D. Guarnizo, and Y. Elovici. Detection of unauthorized iot devices using machine learning techniques. *CoRR*, 2017.

[12] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. Sadeghi, and S. Tarkoma. Iot sentinel: Automated device-type identification for security enforcement in iot. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017.

[13] T. D. Nguyen, S. Marchal, M. Miettinen, M. H. Dang, N. Asokan, and A.-R. Sadeghi. Dïot: A crowdsourced self-learning approach for detecting compromised iot devices. *CoRR*, 2018.

[14] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir. Application-awareness in sdn. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, 2013.

[15] L. van der Maaten and G. Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 2008.