Revisiting IoT Fingerprinting behind a NAT

Conference Paper · July 2021

DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00235

2 authors, including:

Elias Bou-Harb
University of Texas at San Antonio
120 PUBLICATIONS **1,862** CITATIONS

Revisiting IoT Fingerprinting behind a NAT

Christelle Nader and Elias Bou-Harb
The Cyber Center for Security and Analytics
University of Texas at San Antonio
San Antonio, USA
christelle.nader@my.utsa.edu, elias.bouharb@utsa.edu

***Abstract*—The growing usage of Network Address Translation (NAT) over the past couple of years has become a double-edged sword. On one hand, it provides an added measure of security for legitimate users. On the other hand, the anonymity provided by NAT could undoubtedly be leveraged by malicious actors. To this end, the objective of fingerprinting devices behind a NAT aims at properly comprehending the nature of such devices while aiding in proper network and security provisioning and characterization. While the problem scope is certainly not new, it has been evolving quite rapidly given the wide macroscopic and microscopic deployments of IoT devices, and have recently attracted significant attention from the research community. Throughout this paper, we revisit the task of classifying IoT devices deployed within a localized IoT realm. In contrast to the state-of-the-art, we explore the capabilities of unsupervised and semi-supervised shallow and deep learning methodologies in capturing the nature of such NATed devices. Initial results using empirical data indicate the failure of clustering methods in fingerprinting both IoT and non-IoT nodes behind a NAT. Further, and knowing that IoT devices typically possess relatively consistent traffic patterns, the results shed light on the unexpected capability of autoencoders in better capturing non-IoT devices (in contrast to IoT nodes) behind a NAT. In an effort to comprehend such results, we implement and evaluate an explainable mechanism that provides preliminary insights into this phenomena. Additionally, the developed semi-supervised Restricted Boltzmann Machine (RBM) approach generated comparative results to the state-of-the-art, without relying on the stringent and ad-hoc process of features' engineering, with relatively very good algorithm complexity and scalability. Results from this work put forward interesting future work in the area of network traffic analysis of NATed IoT devices, while highlighting the need for addressing the notions of explainability and concept drift.**

***Index Terms*—Network traffic analysis, Internet of Things, IoT fingerprinting, Network Address Translation, Explainable AI, Machine Learning**

I. Introduction

The usage of Network Address Translation (NAT) has grown exponentially over the last few decades as it allows several devices to share a limited number of public IP addresses in addition to providing Internet-wide services via port mapping. Consequently, the anonymity that NAT provides have induced the problem of identifying the nature of the NATed devices, which has always attracted the attention of the research community. For example, back in 2009, Rui et al. [1] proposed an algorithm that detects devices behind a NAT based on Support Vector Machine (SVM). Additionally, in 2014, Gokcen et al. [2] analyzed traffic flows in an attempt to identify and characterize NAT behavior.

Although the problem of device fingerprinting behind a NAT has been explored in the past in various contexts, the explosive growth of deployed IoT devices over the past few years has motivated researchers to bring new solutions to specifically address the IoT context. Indeed, NAT introduces numerous security issues and technical challenges in the IoT realm, including, but not limited to (1) under quantification or over-estimation of the number of vulnerable devices found behind a NAT [3], which hinders IoT-centric botnet characterization and attribution, (2) the issue of legitimate IoT device/type/vendor identification and characterization [4, 5], and (3) the sound and comprehensive analysis of IoT malware evolution residing on NATed IoT devices [6]. Broadly, fingerprinting IoT devices behind a NAT would aid in network and security provisioning, and cyber forensic triage.

As a result, Meidan et al. [7] were among the first to explore this problem by drawing upon IoT-relevant empirical data captured from a localized IoT environment. The authors proposed a state-of-the-art approach for inferring IoT devices connected behind a home NAT by leveraging several supervised learning algorithms along with engineering a number of features that they have employed.

Motivated by this research direction and the open problem of fingerprinting (IoT) devices behind a NAT, this paper explores shallow and deep learning methods in an attempt to fingerprint such nodes. The paper reports on the failure of some methods, the unexpected results of others (while attempting to explain the achieved results), while reporting on comparative state-of-the-art performance.

Specifically, this work makes the following three contributions:

- In contrast to [7], we propose a set of learning algorithms without relying on feature engineering processes. We leverage more than 1 million network flows obtained from [7] over a period of 37 days to perform the empirical experimentation. Our results show that one of the developed and implemented semi-supervised approach generates comparative results in contrast to the state-of-the-art supervised approach.
- We implement two unsupervised clustering algorithms, namely, Density Based Spatial Clustering of Applications with Noise (DBSCAN) and K-Means. We evaluate these two algorithms based on homogeneity, completeness, and silhouette scores. The results show that such unsupervised clustering algorithms fail to accurately fingerprint IoT

(and non-IoT) devices behind a NAT; as opposed to their known efficient use in properly identifying Internet-facing IoT devices in the wild [8–10].
- We devise and employ two semi-supervised machine learning algorithms, namely, Logistic Regression with autoencoders and Logistic Regression with a Restricted Boltzmann machine (RBM). The results indicate that the autoencoder approach is better capable of fingerprinting non-IoT devices behind a NAT; a somehow bewildering result given the heterogeneity of non-IoT traffic in contrast to the quite consistent IoT-generated traffic patterns. By designing and implementing an explainable mechanism, the results shed light on the impact of the NAT device which induces such outcome.

The remainder of this paper is organized as follows. In the next section, we detail the experimental setup and describe a number of data preprocessing steps. In Section III, we describe our experiments and thoroughly report and comment on their respective results. In Section IV, we discuss relevant literature while positioning the presented work. Finally, in Section V, we draw some conclusions and pinpoint a few endeavors which aim at paving the way for future work.

II. Experimental Setup

A. Data Collection

In this paper, we leverage empirical data collected by Meidan et al. [7]. To curate such data, the authors set up a network that represents a real-world scenario of various IoT

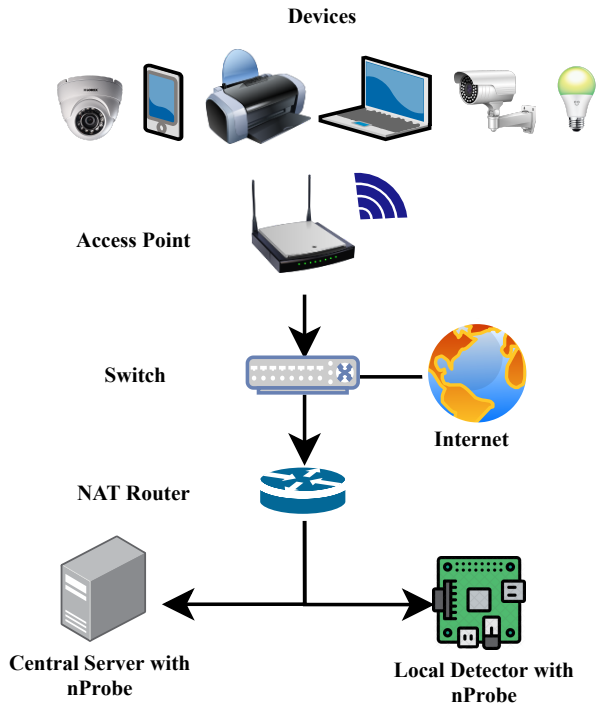


Fig. 1: The data curation setup [7]

and non-IoT devices connected behind a NAT, as illustrated in Figure 1. They first connected several commercial IoT devices as well as laptops and smartphones to a switch (Cisco Catalyst 2960-X, 1G ports) via a wireless access point. Second, they connected this switch to a NAT router (Cisco 3825) where NetFlow was installed. Subsequently, they installed nProbe on a server and a Raspberry Pi to collect NetFlow records from the router for further analysis. The IoT devices that they deployed represent seven manufacturers such as Amazon, Samsung, and WEMO, as well as three of the most popular types of home IoT devices, namely "Media/TV" (streamer, speaker), "Surveillance" (webcam), and "Home Automation" (light bulb, doorbell), as summarized in Table I.

Device	Manufacturer	Type
doorbell	Amazon	IoT
light_bulb	TP_Link	IoT
socket	Wemo	IoT
socket	Wemo	IoT
speaker	Sonos	IoT
streamer	Amazon	IoT
webcam	Amcrest	IoT
webcam	Samsung	IoT
access_point	TP_Link	Non-IoT
laptop	Dell	Non-IoT
laptop	Dell	Non-IoT
smartphone	Samsung	Non-IoT
smartphone	Samsung	Non-IoT

TABLE I: Devices used in our approach

In terms of the IoT devices, they distinguished between three operational phases:

- **Initial:** The device setup immediately after booting.
- **Active:** User interaction such as webcam viewing.
- **Idle:** Network connectivity without any user-driven traffic.

They chose to focus on the *idle* phase, where human interaction is lacking. Nevertheless, in this phase, the IoT devices can be very active, e.g., video streaming or sending measurements. The reason to focus on the idle stage is that IoT devices are in that phase most of the time, and thus it provided them with the highest coverage. In addition, during this phase, IoT devices have characteristic behaviors such as sending regular messages to their servers. Therefore, their traffic patterns are relatively consistent and stable; a fact that is desired when leveraging machine learning-based classification.

	# Instances	Percentage
IoT	1,397,704	89.29
Non-IoT	167,582	10.71

TABLE II: Dataset Overview

The data encompasses over 1 million network flows, collected throughout a period of 37 days. Indeed, this data is comprised of 1,397,704 IoT instances (89.29%) and 167,582 non-IoT instances (10.71%) as shown in Table II.

B. Data Preprocessing

For consistency, completeness and soundness purposes, we perform two data preprocessing steps on the raw captured Netflow records. Such records are summarized in Table III.

Netflow Feature	Description
IN_BYTES	Incoming counter for # bytes associated with an IP Flow
IN_PKTS	Incoming counter for # packets associated with an IP Flow
IPV4_DST_ADDR	IPv4 destination address
L4_DST_PORT	TCP/UDP destination port number
L4_SRC_PORT	TCP/UDP source port number
PROTOCOL	IP protocol byte (6: TCP, 17: UDP)
SRC_TOS	Type of Service byte setting when there is an incoming interface
TCP_FLAGS	Cumulative of all the TCP flags seen for this flow
DURATION	Time (seconds) between first/last packet switching
inter_arrival_time	Time (seconds) between successive flows of the same device

TABLE III: Summary of the Netflow Features

The steps taken include:

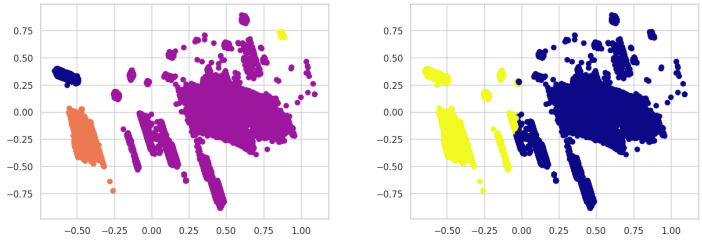
- Removing all of the zero variance Netflow features that have no contribution to classification (e.g. SRC_AS, DST_AS, INPUT_SNMP, OUTPUT_SNMP). Omitting the source IP addresses as well, given that they represent the IPs of the NAT device.
- Encoding the categorical feature "IPV4_DST_ADDR" and scaling all the numerical features such that all feature inputs would be in the range of [0,1].

III. Methodology and Empirical Results

This section details our proposed approach as well as its experimental results. In Section III-A, we implement two unsupervised clustering algorithms, namely, DBSCAN, and K-Means. In Sections III-B and III-C, we implement two semi-supervised machine learning approaches, namely, Logistic regression with autoencoders, and Logistic regression with RBM. At the end of each section, we assess the algorithms and draw some conclusions. We implement our algorithms using Python's Keras and Scikit-learn libraries. In addition, we evaluate our approach on a Ubuntu 18.04.5 LTS machine with a 62.5GB memory, an Intel Xeon(R) W-2145 CPU @3.70GHz x 16, and a hard disk of 251GB.

A. Clustering

In this section, we were interested in exploring the capabilities of unsupervised approaches in identifying IoT devices behind a NAT. To this end, we implement two unsupervised clustering algorithms. We chose two commonly used algorithms, namely K-Means and DBSCAN, due to their wide adoption in data clustering applications related to cyber forensics [11, 12].



(a) DBSCAN (b) K-Means

Fig. 2: DBSCAN and K-Means clustering results.

Since it is known that clustering algorithms operate more effectively with a lower dimensional space, we initially apply the Principal Component Analysis (PCA) technique on several random data samples starting with 10,000 flows and incrementing it each time by 20,000 flows. We notice that after 150,000 flows, the results are the same. Therefore, we find that applying the PCA technique on a random data sample of 150,000 flows gives us the best possible outcome.

Consequently, we show the clustering results in Figure 2. Figure 2a erroneously shows four clusters, given that we already know that we should perceive two (i.e., IoT and non-IoT). To grasp a better notion of the (lack of) soundness related to the observed clusters (including those of K-Means in Figure 2b), we further evaluate the deployed algorithms according to three metric scores:

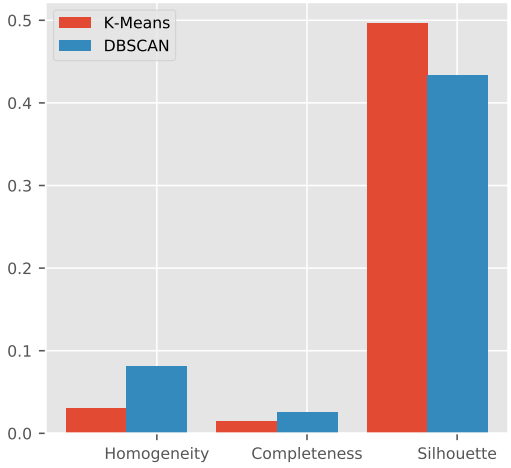


Fig. 3: Performance Evaluation of DBSCAN and K-Means

- **Silhouette Coefficient.** This score is calculated using the following equation on each sample: $(b-a)/max(a,b)$, where a is the mean intra-cluster distance, and b is the mean nearest-cluster distance between a sample and the nearest cluster that the sample is not part of.
- **Completeness Score.** This metric shows if all the data points that are a member of a given class are elements of the same cluster.
- **Homogeneity Score.** This metric shows if all of its clusters contain only data points which are members of

a single class.

Figure 3 provides comparative results of the three different metric scores for the two algorithms. Although DBSCAN's homogeneity and completeness scores are slightly higher than the scores of K-Means (i.e., 0.081 and 0.026 versus 0.031 and 0.015 respectively), these scores tend to 0, which demonstrates the ineffectiveness of such algorithms in properly clustering (IoT) devices behind a NAT. In addition, K-Means' silhouette coefficient (0.497) is somewhat higher than DBSCANs' (0.434). Nonetheless, values that are quite low (i.e., lower than 70-80%) indicate poor clustering outcomes. Thus, it is apparent that both algorithms fail to achieve any meaningful fingerprinting of devices behind a NAT.

B. Logistic Regression with Autoencoders

In this section, we devise and implement a semi-supervised machine learning algorithm that is based on Logistic Regression and autoencoders [13]. We note that semi-supervised learning is a combination of supervised (e.g. Logistic Regression) and unsupervised (e.g. Autoencoders) learning processes in which the unlabeled data is used for training a model. Indeed, in this approach, autoencoders (i.e., a special type of neural network architectures where the output is the same as the input) are used to learn the best possible representation of data which is then passed onto the Logistic model to learn the relationships in the representations in order to make the predictions.

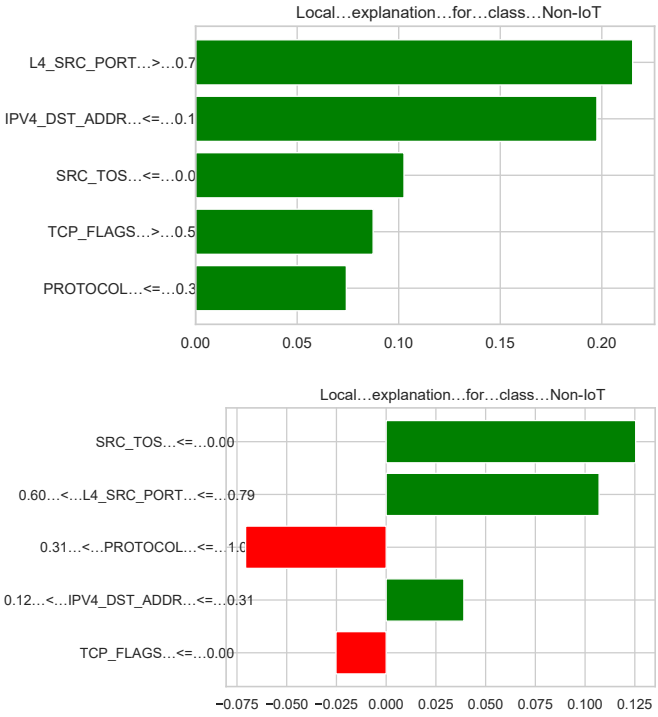
We first develop a network with one input layer and one output layer in which they have the same dimensions. We also employ an optimizer function ("Adam"[14]) because it is efficient, works well with training deep learning models, and can handle a large amount of data. Additionally, we use a loss function ("binary_crossentropy" [15]) because we are dealing with a binary problem, i.e., classifying IoT and non-IoT flows. Moreover, we choose a batch size of 256 and epochs of 30. Note that autoencoders do not need too many samples of data for learning good representations. Therefore, we experiment with several random data samples of various flows starting with 10,000 flows and incrementing it each time by 10,000 flows. We find that a random data sample of 40,000 flows gives us good representations. Further, as autoencoders endeavor to learn only one class in order to automatically distinguish the other class, we only feed it with IoT samples.

After the model is trained, we obtain the latent representation of the input (i.e., a simplified model of the input data) and train a simple linear classifier such as Logistic Regression with such obtained data.

	Precision	**Recall**	**F1-Score**	**Support**
IoT	0.67	0.75	0.71	558
Non-IoT	0.81	0.75	0.78	824
Accuracy			0.75	1,382
Macro Avg	0.74	0.75	0.75	1,382
Weighted Avg	0.76	0.75	0.75	1,382
Training Time	2.149 Seconds			

TABLE IV: Results of Logistic Regression with Autoencoders

The results of Table IV show that this approach achieved an overall accuracy of 75% and has a training time of 2 seconds. What is interesting in these results is that the algorithm was able to classify non-IoT devices behind a NAT with a higher precision and F1-Score than IoT devices, i.e., 81% and 78% versus 67% and 71% respectively. Indeed, we would have expected that IoT devices would yield a higher classification rate than non-IoT devices due to their more predictable and somehow limited/consistent traffic patterns.



(b) IoT Instance

Fig. 4: LIME's deep learning explainability of non-IoT and IoT instances

In an attempt to comprehend how we arrived at such results, we relied on an explainability mechanism. To this end, Ribeiro et al. [16] proposed a package called LIME that explains individual predictions for any black box classifier. In fact, LIME takes the raw data (and the classification model) as input and outputs a probability for each class. Thus, we implemented LIME to better understand the features involved in such classification results. We experimented with 10 random instances from our data and passed them as a parameter to the "explainer" provided by the LIME model.

We compare the results in Figure 4 for two instances which classify the flows as non-IoT and IoT with 76% and 86% accuracy respectively. The features involved in non-IoT classification are shown in green, whereas the features involved in IoT classification are in red. The explainability mechanism yielded two insightful observations. First, it seems that for proper IoT fingerprinting, the black-box model employed the

$PROTOCOL$ feature, which is the transport layer protocol, to achieve the fingerprinting. Thus, it is plausible to note that if the NATed device is using some UDP-based protocol, similar to the ones used exclusively in IoT contexts, such as MQTT and CoAP, then the classifier, despite the NAT device can easily fingerprint this behavior. Second, if the device "piggybacks" on traditional application layer protocols (i.e., HTTP) which employ TCP, it seems that it is not enough for the classifier to deem it as a non-IoT device. However, this, in combination with a NAT device-selected source port, would indeed be sufficient to deem the device as non-IoT. From these points, we can deduce that while IoT devices typically possess somehow uniform traffic patterns/flows in contrast to the heterogeneity of the traffic generated from non-IoT devices, it seems that the autoencoder is putting less emphasis on the actual traffic dynamics, while focusing more on the semantics of the actual protocol being operated on that device (which is reflected in the usages of the transport layer protocols, type of services, flags, and source ports). We believe that this output is quite interesting and undoubtedly begs for in-depth future exploration of network traffic analysis in conjunction with stronger explainability mechanisms, as applied to various security applications (including IoT fingerprinting).

C. Logistic Regression with RBM

In this section, we implement another semi-supervised machine learning algorithm that is based on Logistic Regression with Bernoulli Restricted Boltzmann Machine (RBM)[17]. Indeed, Bernoulli RBM estimates its parameters by using a Stochastic Maximum Likelihood (SML), also known as Persistent Contrastive Divergence (PCD). In addition, after experimenting with different solvers for the Logistic Regression model, we employ the "newton-cg" solver because it yielded the best results.

Compared to the previous approach in Section III-B, we utilize all the flows from [7] because we want to compare the results of this classifier to the state-of-the-art supervised approach [7]. In fact, the training set consists of the first 30 days (approximately 81% of each device, on average), and the testing set consists of the remaining 7 days (approximately 19%).

Parameters	Possible Values			Best Value
Logistic C	1	6,000	10,000	10,000
RBM learning_rate	0.06	0.01	0.001	0.01
RBM n_components	50	100	200	50

TABLE V: Possible Values of Parameters for the Logistic RBM approach

Initially, we instantiate an RBM features' classifier which consists of pipelining an RBM model with a Logistic Regression model. Table V presents the possible values of one Logistic Regression parameter (C), and two RBM parameters (learning rate and number of components). As the classifiers have a large number of possible hyperparameter values, we implement the GridSearchCV function which converges towards the best parameter values for these models. As a result, the best parameter values are obtained as: Logistic C (10,000), RBM Learning Rate (0.01), and RBM Number of Components (50).

We then apply these values to the created classifier and evaluate it.

	Precision	Recall	F1-Score	Support
IoT	0.98	1.00	0.99	267,757
Non-IoT	0.95	0.71	0.81	18,370
Accuracy			0.98	286,127
Macro Avg	0.96	0.85	0.90	286,127
Weighted Avg	0.98	0.98	0.98	286,127
Training Time	267.143 Seconds			

TABLE VI: Results of Logistic Regression using RBM features

The results from Table VI show that this approach has an accuracy of 98% and has a training time of approximately 4 minutes. Even though Meidan et al.'s [7] approach has an approximate learning time of 2 seconds, the developed approach herein obtained similar performance, i.e., a weighted average precision of 98% versus an average precision of 99%, without the need for feature engineering or prior domain knowledge.

Complexity. To further assess this approach, we study its complexity by using the Big-O notation. To this end, we look for the complexities of each separate algorithm used in this approach, namely, the BernoulliRBM and the Logistic Regression coupled with the Newton Conjugate Gradient (CG). Indeed, the BernoulliRBM algorithm has a time complexity of $O(d^2)$, where d is the number of features used [18]. In addition, both the Newton CG and the Logistic Regression algorithms have a time complexity of $O(nd)$ [19, 20], where d is the number of features used and n is the number of samples used. We also note that the space complexity of the Logistic Regression algorithm is $O(d)$ [20]. By pipelining the BernouilliRBM and the Logistic Regression with the Newton CG, the overall time complexity arrives to $O(d^2) + O(nd)$. Since the number of features is constant, the terms containing d cancel out. Therefore, the overall time complexity of this approach is $O(n)$ which is a fair result.

Scalability. To further validate our results, we perform a scalability test which measures the performance of this approach as the number of flows increases. Due to the lack of data, we initially duplicate all the flows from [7]. We then perform several tests on this method, in which we increase the original data by 300K random flows each time. We evaluate each experiment according to these performance metrics: training time, prediction time (flow classification time), accuracy, and weighted average precision.

	# Flows ($\times 10^5$)					
	1.5	**1.8**	**2.1**	**2.4**	**2.7**	**3.0**
Training Time (sec)	267.143	247.642	265.678	317.702	333.164	359.871
Prediction Time (sec)	0.269	0.328	0.344	0.402	0.425	0.456
Accuracy	0.98	0.93	0.94	0.94	0.94	0.94
Weighted Avg Precision	0.98	0.88	0.93	0.91	0.88	0.88

TABLE VII: Scalability Experimental Results

The results from Table VII show that as the number of flows increases, the accuracy slightly drops and stabilizes at 94%. In addition, the weighted average precision also slightly decreases and fluctuates between 88% and 93%. Although these results show lower values, they are still considered good results. Moreover, we plot the training time and prediction time as shown in Figure 5. Both the training time and prediction time increase with the number of flows.

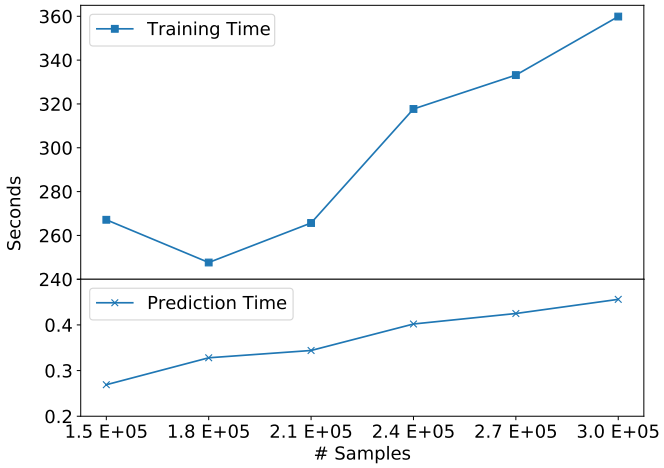


Fig. 5: Scalability Training vs Prediction Results

The training time augments from 267.143 sec ($\sim$ 4 min) to 359.871 sec ($\sim$ 6 min) whereas the flow classification time increases from 0.269 sec to 0.456 sec, which is very acceptable.

Summary. It is possible to identify IoT devices behind a NAT by using a semi-supervised machine learning algorithm that consists of a Logistic Regression model with RBM features. Such an approach can properly classify IoT and non-IoT devices behind a NAT with a 98% and a 95% precision respectively (using this specific dataset). In addition, the proposed methodology has comparative results with state-of-the-art [7] without resorting to features' engineering. We also find that the time complexity of this algorithm is $O(n)$ which is a fair result. Further, this approach scales well as it maintains a high accuracy (94%) and a weighted average precision (88%-93%).

IV. Related Work

In this section, we elaborate on three related topics that are relevant to our work herein. First, we discuss recent efforts pertaining to IoT device fingerprinting. Then, we focus on unsupervised IoT device classification. Finally, we enumerate the literature related to device identification behind a NAT.

IoT Fingerprinting. Pour el al. [8] leveraged macroscopic, passive empirical data provided by network telescopes to shed light on the IoT evolving threat landscape. The authors aimed at classifying compromised IoT devices from one way-network traffic by developing a multi-window convolutional neural network. By analyzing 3.6 TB of darknet traffic, their approach effectively uncovered 440,000 compromised IoT devices and 350 IoT botnets that were still active in the wild. Meidan et al. [21] developed ProfilIoT, a machine learning approach to accurately identify IoT devices connected to a network. The authors employed supervised learning in order to train a multi-stage classifier. The developed classifier distinguished between IoT and non-IoT devices by leveraging HTTP packet properties (user-agents). Differently, Perdisci et al. [22] developed IoTFinder, an independent system for large-scale IoT device identification. Indeed, their proposed approach leveraged distributed passive DNS data which is comprised of more than 40 million clients. This collected data was then passed on to a multi-label machine learning-based classifier that only uses DNS fingerprints to classify IoT devices. In contrast, Huang et al. [23] developed IoT Inspector, an open-source tool that allows users to observe traffic from smart home devices on their own home networks. In fact, by allowing users to download IoT Inspector, the authors were able to collect (and analyze) labeled network traffic from 54,094 smart home devices.

Unsupervised IoT classification. Sivanathan et al. [24] developed a modular device classification architecture that is based on an unsupervised one-class clustering method for each device in order to detect normal network behavior. Their approach can be used for automatic conflict resolution and noise filtering. After evaluating their scheme on 10 real IoT devices, they achieved an overall accuracy of more than 94%. In addition, Bhatia et al. [25] introduced a network-centric,

behavioral anomaly detection approach that can separate normal and anomalous traffic. They showed that their method can be incorporated in a larger system in order to identify compromised end-points despite IP spoofing. Moreover, Meidan et al. [26] proposed N-Balot, a network-based anomaly detection method that extracts behavior snapshots of the network and feeds them to autoencoders in order to detect anomalous traffic from compromised IoT devices. After empirically evaluating their approach with two of the most widely known IoT-based botnets, Mirai and BASHLITE, they found that N-Balot can accurately and instantly detect such attacks. Similarly, Nomm and Bahsi [27] presented an anomaly-based detection of IoT botnets by leveraging unsupervised learning models with reduced feature set size. They showed that their approach decreased the required computational resources and can detect IoT botnets with high accuracy.

Device Identification behind a NAT. Griffioen and Doerr [3] leveraged a weakness in the packet generation algorithm and random number generation of the Mirai IoT malware in order to detect NAT behavior via IP churn. In fact, they used a network telescope of 65K IP addresses for a period of 9 months to quantify and detect IP churn. By doing so, they found that NAT leads to an underestimation of the total Mirai spread when solely counting IP addresses, as multiple infections are located behind the same public IP address. Additionally, Khatouni et al. [28] proposed a passive supervised machine learning methodology to detect hosts behind NAT devices by using flow level statistics without any application layer information. Indeed, the authors captured a large dataset and performed an extensive evaluation with four existing approaches from the literature. Their results showed that their methodology can identify NAT behaviors and hosts with high accuracy. Differently, Yang et al. [29] proposed a methodology to identify NATs for online IoT devices based on Tri-Net; a semi-supervised deep neural network. Indeed, Tri-Net learned features on three layers, namely network, transport, and application layer in a small labeled data set. After evaluating this approach on a real-world dataset with more than 8 million online IoT devices, the authors were able to find 2,511,499 IoT devices connected to the Internet via NAT with a precision and recall of up to 92%.

This paper contributes to the same line of research, but focuses instead on the niche problem of IoT fingerprinting of NATed devices. Further, it explores non-supervised learning methods while reporting on some (negative) results, or unexpected outcomes (with an attempt for explainability) while also demonstrating comparative results to the state-of-the-art.

V. Concluding Remarks

This work complements current device classification methods behind a NAT by offering a new unsupervised and semi-supervised machine learning methodologies without feature engineering that detects IoT devices behind a home NAT. This work initially introduces two unsupervised clustering algorithms, namely K-Means and DBSCAN. As a result, we found that such algorithms are inefficient in properly detecting (IoT) devices behind a NAT. Subsequently, we devise two semi-supervised ML algorithms that encompass a Logistic Regression classifier with either Autoencoders or RBM. Consequently, the results show that the algorithm that features autoencoders has a higher classification rate for non-IoT devices rather than IoT devices. By relying on an explainability technique, we find that several features (e.g., $L4\_SRC\_PORT$ and $PROTOCOL$) affect these rates. Moreover, the evaluation of the Logistic Regression with RBM features' classifier provides sound fingerprinting of IoT devices behind a NAT with a state-of-the-art accuracy of 98%.

As for future work, we will be attempting to infer IoT devices behind a NAT on a much larger scope. To this end, we will leverage network telescopes as well as the data sets provided by Huang et al. [23] to identify IoT devices behind a NAT on an Internet-scale perspective. We will continue to explore the notions behind explainable AI as applied on network traffic analysis. Moreover, with the rise of the ZHtrap [30] botnet, which uses Tor for communications, it would be interesting to examine if a flavor of our developed methods could be applied to the Tor network in order to detect IoT devices behind Tor proxies. Finally, the notion of concept drift poses challenges to the deployed machine learning models [31]. Therefore, we plan on testing the robustness of our developed approach against concept drift on various security application, including the fingerprinting of NATed compromised IoT devices and exploring the problem of IoT device identification.

Acknowledgment

We thank the reviewers in advance for their constructive feedback. This work was supported by a grant from the National Science Foundation, Office of Advanced CyberInfrastructure #1907821.

References

[1] R. Li, H. Zhu, Y. Xin, Y. Yang, and C. Wang, "Remote nat detect algorithm based on support vector machine," in *2009 International Conference on Information Engineering and Computer Science*. IEEE, 2009, pp. 1–4.

[2] Y. Gokcen, V. A. Foroushani, and A. N. Z. Heywood, "Can we identify nat behavior by analyzing traffic flows?" in *2014 IEEE Security and Privacy Workshops*. IEEE, 2014, pp. 132–139.

[3] H. Griffioen and C. Doerr, "Quantifying autonomous system ip churn using attack traffic of botnets," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, 2020, pp. 1–10.

[4] X. Feng, Q. Li, H. Wang, and L. Sun, "Acquisitional rule-based engine for discovering internet-of-things devices," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 327–341.

[5] F. Shaikh, E. Bou-Harb, N. Neshenko, A. P. Wright, and N. Ghani, "Internet of malicious things: Correlating active and passive measurements for inferring and characterizing internet-scale unsolicited iot devices," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 170–177, 2018.

[6] E. Cozzi, P.-A. Vervier, M. Dell'Amico, Y. Shen, L. Bilge, and D. Balzarotti, "The tangled genealogy of iot malware," in *Annual Computer Security Applications Conference*, 2020, pp. 1–16.

[7] Y. Meidan, V. Sachidananda, H. Peng, R. Sagron, Y. Elovici, and A. Shabtai, "A novel approach for detecting vulnerable iot devices connected behind a home nat," *Computers & Security*, vol. 97, p. 101968, 2020.

[8] M. S. Pour, A. Mangino, K. Friday, M. Rathbun, E. Bou-Harb, F. Iqbal, K. Shaban, and A. Erradi, "Data-driven curation, learning and analysis for inferring evolving iot botnets in the wild," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, 2019, pp. 1–10.

[9] C. J. Dietrich, C. Rossow, and N. Pohlmann, "Cocospot: Clustering and recognizing botnet command and control channels using traffic analysis," *Computer Networks*, vol. 57, no. 2, pp. 475–486, 2013.

[10] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection," 2008.

[11] E. Bou-Harb, M. Debbabi, and C. Assi, "A systematic approach for detecting and clustering distributed cyber scanning," *Computer Networks*, vol. 57, no. 18, pp. 3826–3839, 2013.

[12] ——, "A statistical approach for fingerprinting probing activities," in *2013 International Conference on Availability, Reliability and Security*. IEEE, 2013, pp. 21–30.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[17] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[18] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.BernoulliRBM.html

[19] T. P. Minka, "A comparison of numerical optimizers for logistic regression," *Unpublished draft*, pp. 1–18, 2003.

[20] P. Kumar, "Time complexity of ml models," 2019. [Online]. Available: https://medium.com/analytics-vidhya/time-complexity-of-ml-models-4ec39fad2770

[21] 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, pp. 506–509.

[22] R. Perdisci, T. Papastergiou, O. Alrawi, and M. Antonakakis, "Iotfinder: Efficient large-scale identification of iot devices via passive dns traffic analysis," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2020, pp. 474–489.

[23] D. Y. Huang, N. Apthorpe, F. Li, G. Acar, and N. Feamster, "Iot inspector: Crowdsourcing labeled network traffic from smart home devices at scale," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 2, pp. 1–21, 2020.

[24] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, "Inferring iot device types from network behavior using unsupervised clustering," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*. IEEE, 2019, pp. 230–233.

[25] R. Bhatia, S. Benno, J. Esteban, T. Lakshman, and J. Grogan, "Unsupervised machine learning for network-centric anomaly detection in iot," in *Proceedings of the 3rd acm conext workshop on big data, machine learning and artificial intelligence for data communication networks*, 2019, pp. 42–48.

[26] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[27] S. Nõmm and H. Bahşi, "Unsupervised anomaly based botnet detection in iot networks," in *2018 17th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2018, pp. 1048–1053.

[28] A. S. Khatouni, L. Zhang, K. Aziz, I. Zincir, and N. Zincir-Heywood, "Exploring nat detection and host identification using machine learning," in *2019 15th International Conference on Network and Service Management (CNSM)*. IEEE, 2019, pp. 1–8.

[29] Z. Yan, N. Yu, H. Wen, Z. Li, H. Zhu, and L. Sun, "Detecting internet-scale nats for iot devices based on trinet," in *International Conference on Wireless Algorithms, Systems, and Applications*. Springer, 2020, pp. 602–614.

[30] R. Lakshmanan, "New mirai variant and zhtrap botnet malware emerge in the wild," Mar 2021. [Online]. Available: https://amp.thehackernews.com/thn/2021/03/new-mirai-variant-and-zhtrap-botnet.html

[31] "CADE: Detecting and explaining concept drift samples for security applications," in *30th USENIX Security Symposium (USENIX Security 21)*. Vancouver, B.C.: USENIX Association, Aug. 2021. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/yang