ProfilIoT: A Machine Learning Approach for IoT Device Identification Based on Network Traffic Analysis

Yair Meidan[1], Michael Bohadana[1], Asaf Shabtai[1],
Juan David Guarnizo[2], Martín Ochoa[2], Nils Ole Tippenhauer[2], and Yuval Elovici[1,2]

[1] Department of Software and Information Systems Engineering, Ben-Gurion University, Beer-Sheva, Israel
[2] Singapore University of Technology and Design, Singapore

ABSTRACT

In this work we apply machine learning algorithms on network traffic data for accurate identification of IoT devices connected to a network. To train and evaluate the classifier, we collected and labeled network traffic data from nine distinct IoT devices, and PCs and smartphones. Using supervised learning, we trained a multi-stage meta classifier; in the first stage, the classifier can distinguish between traffic generated by IoT and non-IoT devices. In the second stage, each IoT device is associated a specific IoT device class. The overall IoT classification accuracy of our model is 99.281%.

CCS Concepts

•Security and privacy → Mobile and wireless security; •Computing methodologies → Machine learning;

Keywords

Internet of Things (IoT); Cyber Security; Machine Learning; Device Identification; Network Traffic Analysis.

1. INTRODUCTION

The term "Internet of Things" (IoT), is used as an umbrella keyword covering various aspects related to the extension of the Internet and Web into the physical realm, by means of the widespread deployment of spatially distributed devices with embedded identification, sensing and/or actuation capabilities [9]. Among the challenges the IoT poses to organizations are security and governance issues stemming from the proliferation of such devices and the ever increasing number of IoT-enabled organizational assets. In the future, organizations might not know exactly what IoT devices are connected to their network, a situation which threatens the security and integrity of the network and the devices.

In this work we address the challenge of IoT device identification within a network by analyzing and classifying network traffic data. Even if the prefixes of MAC addresses can be used to identify the manufacturer of a particular device, there is no standard to identify brands or types of devices. However, high-level network statistics, such as the the "ratio between incoming and outgoing bytes" and the "average time to live" have been used to identify malicious traffic in network communications [3]. We propose to use such network features to identify IoT devices. We believe our approach is generic and flexible enough to be applied in the rapid evolving IoT landscape and at the same time targeted enough to cater to efficient device identification.

Research questions. This research proposes a novel method to classify devices connected to an organization's network based solely on network traffic analysis. More specifically, we focus on the following questions: **a)** Can we accurately distinguish between IoT devices and non-IoT devices? **b)** For a specific IoT device (e.g., smart TV, IP camera), can we accurately model its network behavior and detect the device's presence within the network traffic?

Summary of contributions. a) To the best of our knowledge, we are the first to apply machine learning techniques to network traffic for IoT device classification and identification. **b)** We demonstrate that we can accurately distinguish between IoT and non-IoT devices using traffic analysis, machine learning, and HTTP packet property (user agent). **c)** We show that our approach can accurately detect the presence of a specific IoT device within the network traffic.

2. DATA ACQUISITION

To induce our model and evaluate it we collected traffic data from local IoT devices, PCs, and smartphones (see Table 1). As commonly practiced, these devices were connected to a Wi-Fi access point, and their network traffic data were recorded as *.pcap files [6] for further analysis.

Feature Extraction. The TCP packets were first converted by the feature extractor tool [4] to sessions (unique 4-tuples consisting of source and destination IP addresses

SAC'17, April 3-7, 2017, Marrakech, Morocco

ACM 978-1-4503-4486-9/17/04…$15.00
http://dx.doi.org/10.1145/3019612.3019878

Table 1: Devices included in the dataset

Specific Device Type	Device Type	Make and Model	Number of TCP Sessions
Baby Monitor	IoT	Beseye Baby Monitor Pro	2,072
Motion Sensor	IoT	Wemo F7C028uk	254
Printer	IoT	HP Officejet Pro 6830	70
Refrigerator	IoT	Samsung RF30HSMRTSL	7,008
Security Camera	IoT	Withings WBP02/WT9510	980
Socket	IoT	Efergy Ego	342
Thermostat	IoT	Nest Learning Thermostat 3	6,353
TV	IoT	Samsung UA55J5500AKXXS	4,854
Smartwatch	IoT	LG Urban	687
PC	Non-IoT	Dell Optiplex 9020	3,138
Laptop	Non-IoT	Lenovo X260	4,907
Smartphone	Non-IoT	LG G2	2,178
Smartphone	Non-IoT	Galaxy S4	643

and port numbers, from SYN to FIN). Then, each session was represented by a vector of features from the network, transport, and application layers, and enriched with publicly available data such as Alexa Rank [1] and GeoIP [2].

Data Partitioning. After constructing the labeled dataset, we chronologically divided it into three mutually exclusive sets. The first set, denoted DS_s, is used for inducing a set of single-session classifiers. The second set, denoted DS_m, is used for optimizing the parameters of a multi-session classifier. As practiced in machine learning research, the third set was used as a test set (denoted DS_{test}) for evaluating our proposed method and deriving performance measures.

3. PROPOSED METHOD FOR IOT DEVICE IDENTIFICATION

We propose a multi-stage process in which a set of machine learning based classifiers are applied to a stream of sessions that originate from a specific device (i.e., a specific IP address). The goal is to determine whether the traffic belongs to a PC, a smartphone or a specific (known) IoT device.

3.1 Notation

The notation we use to describe our method and the means of evaluating it are summarized below.

D: Set $\{d_1, \ldots, d_n\}$ of known IoT devices.
DS_s: Dataset for inducing single-session classifiers.
C_i: Single-session classifier for d_i, induced from DS_s.
tr_i^*: Optimal classification threshold for C_i.
DS_m: Dataset, sorted in chronological order, for inducing multi-session based classifiers.
DS_m^i: Subset of sessions in DS_m, origin device d_i.
$DS_m^i[a]$: The a^{th} session, originating from d_i in DS_m^i.
$|DS_m^i|$: The number of sessions in DS_m^i.
p_i^s: Posterior probability of a session s to originate from d_i; derived by applying C_i to session s.
s_i^*: The optimal (minimal) size of a sequence of sessions for which C_i classifies correctly most sessions in *any* sequence of sessions of size s_i^* in DS_m.
S^d: Sequence of sessions originating from device d.
C: Set $\{(C_1, tr_1^*, s_1^*), \ldots, (C_n, tr_n^*, s_n^*)\}$ of single-session classifiers for devices in D with optimal thresholds tr_i^* and sequence sizes s_i^*.
DS_{test}: Dataset used for evaluating the proposed method (sorted in chronological order).
DS_{test}^i: Subset of DS_{test}, originating from device d_i.
$DS_{test}^i[a]$: The a^{th} session (originating from d_i) in DS_{test}^i.

3.2 Model Training

Let D be the set of known devices (i.e., devices that we want to be able to identify based on their traffic). Deriving the device identification model consists of the following steps.

Induce single-session binary classifier. For each $d_i \in D$ we induce a single-session binary classifier, denoted by C_i, that given a feature vector of a single session (denoted by s), outputs a posterior probability p_i^s that the session was generated by device d_i. The single-session classifiers C are obtained using the DS_s dataset. For training C_i for device d_i we derive binary labels, such that all feature vectors of sessions that belong to d_i are labeled as d_i, and feature vectors of sessions that do not belong to d_i are labeled as 'other.' Thus, given an unlabeled feature vector extracted from a session s, we can apply all single-session classifiers C to obtain a vector of posterior probabilities $(p_1^s, \ldots, p_n^s)$.

Determine optimal thresholds for single-session classifiers. For each classifier C_i we determined the optimal classification threshold (cutoff value), denoted by tr_i^*, for labeling a given session s with probability p_i^s as d_i or "other." We used the DS_m dataset to evaluate the performance of C, the set of single-session classifiers, and for setting the threshold values of the classifiers. Each optimal threshold tr_i^* was selected such that it maximizes the accuracy of classifier C_i.

Determine optimal session sequence size s_i^* for each classifier. Here we derive the optimal session sequence size s_i^* for each classifier C_i that is used for defining the multi-session classifier. First, for each IP (device) in DS_m we apply the set C of single-session classifiers to all session feature vectors for obtaining classifications. Then we utilize tr_i^* and DS_m to analyze the classification results of each optimized classifier. Afterwards we look for the minimal number of consecutive session classifications, based on which a majority vote will provide zero false positives and zero false negatives on the entire DS_m. We denote this number by s_i^* and refer to it as the optimal size of the moving window. The lower s_i^* is for a given d_i, the smaller number of consecutive sessions we need to accurately determine whether the sessions that emanated from an IP were generated by d_i or not. Algorithm 1 describes how s_i^* is calculated.

To conclude, for every device d_i we have a classifier C_i with threshold tr_i^*, and upon a majority voting on its s_i^* consecutive classifications we can determine whether sessions that emanated from a given IP were generated by d_i with 100% accuracy. Note that it was easy to differentiate between IoT devices and PCs and smartphones based on a single session (see Section 4), so the rest of the discussion in this section will focus on identifying the specific IoT device. Table 2 presents the performance of the single-session classifiers after being optimized with tr_i^* and their optimal s_i^*.

3.3 Application for Device Identification

Algorithm 2, our final classification algorithm, is based on C: the trained classifiers and their corresponding parameters $(C_1, tr_1^*, s_1^*), \ldots, (C_n, tr_n^*, s_n^*)$. The classification algorithm receives a stream of session vectors that emanated from an IP and were generated by an unknown device. It checks if

Algorithm 1: Calculating s_i^*

```
1: procedure FINDSISTAR(D, DS_m, C_i)
2:     s_i^* ← 1
3:     for d_j in D do
4:         DS_m^j ← subset of DS_m with origin d_j
5:         a ← 1
6:         s ← 1
7:         while a + s − 1 <= |DS_m^j| do
8:             n ← 0
9:             for sess in {DS_m^j[a], ..., DS_m^j[a + s − 1]} do
10:                p_i^s ← CLASSIFY(C_i, sess)
11:                if p_i^s > tr_i^* then
12:                    n ← n + 1
13:            if i = j and n > s/2 then
14:                a ← a + 1
15:            else
16:                a ← 1
17:                s ← s + 2
18:        if s_i^* < s then
19:            s_i^* ← s
20:    return s_i^*
```

Table 2: Single-session based classifier performance

IoT Device	tr^*	Method	FNR	FPR	s^*
Printer	0.35	GBM	0.3	0	11
Sec. Camera	0.5	Random Forest	0	0	1
Refrigerator	0.2	XGBoost	0.001	0.001	3
Motion Sensor	0.2	XGBoost	0.012	0	3
Baby Monitor	0.3	XGBoost	0.006	0	9
Thermostat	0.2	Random Forest	0.011	0.004	45
TV	0.1	GBM	0.026	0.001	23
Smartwatch	0.8	XGBoost	0.184	0	77
Socket	0.25	Random Forest	0	0	1

the stream of sessions was generated by device d_i by classifying using C_i with s_i^* consecutive sessions, and checking whether most of the s_i^* sessions were classified as d_i. In order to optimize the search for the device, the device inspection order is determined by s_i^*, so the algorithm starts to inspect devices with the lowest s_i^*, and continues with ascending order of s_i^*. A possible modification is to take into account also the prior probability of a device being observed.

4. EVALUATION

We evaluate our method using the third dataset, DS_{test}. The results indicate that by analyzing network traffic we can distinguish between IPs that belong to IoT devices and IPs that belong to PCs and smartphones. Smartphones were classified by analyzing the "user agent" HTTP property, and thus the classification accuracy for smartphones was 100%. The classification of PCs was performed by classifying a session by a single-session classifier. The performance for PCs was almost perfect (a very low false positive rate of 0.003 and a very low false negative rate of 0.003).

Having accurately classified smartphones and PCs, we applied Algorithm 2 (IoT device classification) on DS_{test} for

Algorithm 2: IoT device classification

```
1: procedure CLASSIFYDEVICE(C, S^d)
2:     Sort C by ascending s_i^*
3:     for (C_i, tr_i^*, s_i^*) in C do
4:         a ← 1
5:         n ← 0
6:         while a + s_i^* − 1 <= |S^d| do
7:             for sess in {S^d[a], ..., S^d[a + s_i^* − 1]} do
8:                 p_i^s ← CLASSIFY(C_i, sess)
9:                 if p_i^s ≥ tr_i^* then
10:                    n ← n + 1
11:            if n > s_i^*/2 then
12:                return d_i
13:            else
14:                a ← a + 1
15:    return 'unknown'
```

evaluating the performance for IoT device identification. We note that Algorithm 2 is optimized to derive the type of an IoT device by analyzing a minimal number of consecutive sessions. In a worst case scenario it needs to analyze $\max(s_i^*)$ consecutive sessions. To properly evaluate the performance of our method we reran Algorithm 2 multiple times, and each time we omitted the first session of the sequence from the previous run. This was done to compensate for a possible bias that may occur when the sequence begins with different sessions. Given dataset DS_{test} sorted in a chronological order, let DS_{test}^i be a subset of sessions in DS_{test} originating from d_i, and let $DS_{test}^i[a]$ be the a_{th} session originating from d_i in DS_{test}^i. We used DS_{test} for evaluating the proposed method, and for each device $d_i \in D$ we repeated the evaluation by applying Algorithm 2 (i.e., the trained model) on all of the subsequences of the sessions in DS_{test}^i, starting from session $a \in \{1, \ldots, |(DS_{test}^i)| - s_i^* + 1\}$ and ending at $a + s_i^* - 1$ (with maximal value $a + s_i^* - 1 = |(DS_{test}^i)|$). so, for each device $d_i \in D$ we repeated the evaluation as follows:

```
1: for a in {1, ..., (|(DS_test^i)| − s_i^* + 1)} do
2:     s^d ← {DS_test^i[a], ..., DS_test^i[a + s_i^* − 1]}
3:     CLASSIFYDEVICE (C, s^d)
```

As seen in Table 3, the classification accuracy on DS_{test} was high. Out of 7,376 test cases (each defined by the first session in the sequence) 19 cases were misclassified and 34 were unclassified, thus the total accuracy was 99.281%. Note that the classification accuracy on DS_{test} was not 100%, so we executed Algorithm 1 (previously run on DS_m) once again, this time on DS_{test}. We then compared s_i^*'s obtained from DS_m to s_i^*'s obtained from DS_{test}. Classification accuracy measures on DS_{test}, plus the recalculated s_i^*'s, are presented in Table 4. We note that the required s_i^* for perfect results for all devices in DS_{test} should be higher. For perfect results on DS_{test} we recommend using an s_i^* which is 4.333 times higher than the ones computed by Algorithm 1 on DS_m .

5. RELATED WORK

Table 3: Accuracy (Algorithm 2) on DS_{test}

Tested Device	Number of sessions classified		
	Correctly	Incorrectly	'Unknown'
Printer	14	0	0
Security camera	325	0	1
Refrigerator	2334	0	0
Motion Sensor	83	0	0
Baby Monitor	663	5	15
Thermostat	2074	0	0
TV	1566	12	18
Smartwatch	151	2	0
Socket	113	0	0

Table 4: Accuracy and recalculation of s_i^* on DS_{test}

Device	tr^*	s^*	Method	FNR	FPR	Acc.	s^* on DS_{test}
Printer	0.35	11	GBM	0	0	1	5
Security Camera	0.5	1	Random Forest	0.004	0	0.999	3
Refrigerator	0.2	3	XGBoost	0	0.001	0.999	5
Motion Sensor	0.2	3	XGBoost	0	0	1	1
Baby Monitor	0.3	9	XGBoost	0.03	0	0.999	39
Thermostat	0.2	45	Random Forest	0	0	1	39
TV	0.1	23	GBM	0.014	0	0.997	45
Smartwatch	0.8	77	XGBoost	0	0	1	43
Socket	0.25	1	Random Forest	0	0	1	1

Device identification based on characteristics of communication properties such as signals and emissions was discussed in [12], which reports respective efforts since the 1960s. More recently, the authors of [5] leverage passive radio frequency analysis to identify network interface cards (NIC) that transmitted IEEE 802.11 frames. Our work, however, is more closely related to efforts on identifying devices based on logical characteristics of their network traffic. The idea of using network characteristics to identify different kinds of nodes in a network has been applied in several contexts. For instance, [10] describes a technique to identify rogue wireless access points that try to mislead victims to connect to them.

A large area of research is the identification of clients in a network that have been infected with malware (e.g., in the context of botnets) and the identification of the associated command and control servers. In [7, 11], techniques to cluster network traffic patterns associated with botnets are presented. The characteristics observed include the flow patterns between hosts, such as the number of connections and amount of data exchanged. Similarly, using machine learning and network traffic features, [4] presents an approach to detect malware related traffic. Their work is close to ours, as we use similar network traffic features. However, our work differs from this and the works on botnet detection cited above, because we build a classifier for IoT devices based on the network traffic communication data gathered from a heterogeneous set of devices.

To the best of our knowledge, there is no other work in the literature that applies machine learning on network traffic features to identify IoT devices. In [8], a logical framework to classify IoT devices is proposed, however they classify IoT devices into just two categories (i.e., high vs. low energy consumption) and provide a very preliminary proof of concept of their algorithm based on simulations.

6. CONCLUSION

In this paper, we demonstrated that an IoT device can be accurately identified based on characteristics of the network traffic it generates. Our method can classify an IoT device, including by brand and model, with 99.281% accuracy which makes our method feasible for real organizations. We believe that our novel approach can be used to automatically and accurately recognize (possibly unauthorized) connections of IoT devices to an enterprise's computer network, and thus mitigate violations of operational policies. In future research, we plan to explore applications and adapt our technology to additional scenarios, possibly including different network protocols and various data capturing points, to better understand how our approach scales and generalizes.

7. REFERENCES

[1] Alexa top sites. Retrieved April 14, 2016 from http://www.alexa.com/topsites.

[2] Geoip lookup service. Retrieved April 14, 2016 from http://geoip.com/.

[3] D. Bekerman. Network features. Retrieved April 14, 2016 from http://www.ise.bgu.ac.il/dima/network_traffic_features_set.pdf.

[4] D. Bekerman, B. Shapira, L. Rokach, and A. Bar. Unknown malware detection using network traffic classification. In *Proc. of IEEE Conference on Communications and Network Security (CNS)*, 2015.

[5] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless device identification with radiometric signatures. In *Proc. of ACM conference on Mobile computing and networking*, 2008.

[6] G. Combs et al. Wireshark-network protocol analyzer. *Version 0.99*, 5, 2008.

[7] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *Proc. of USENIX Security Symposium*, 2008.

[8] P. N. Mahalle, N. R. Prasad, and R. Prasad. Object classification based context management for identity management in internet of things. *International Journal of Computer Applications*, 63(12), 2013.

[9] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. Internet of Things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.

[10] I. H. Saruhan. Detecting and preventing rogue devices on the network. SANS Institute InfoSec Reading Room, sans. org, 2007.

[11] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas. Botnet detection based on network behavior. In *Botnet Detection: Countering the Largest Security Threat*, pages 1–24. Springer, 2008.

[12] K. I. Talbot, P. R. Duley, and M. H. Hyatt. Specific emitter identification and verification. *Technology Review*, page 113, 2003.