Similarity-based clustering for IoT device classification

Guillaume Dupont[1], Cristoffer Leite[1], Daniel Ricardo dos Santos[2], Elisa Costante[2], Jerry den Hartog[1], Sandro Etalle[1]
[1]*Eindhoven University of Technology,* [2]*Forescout Technologies*

***Abstract*—Classifying devices connected to an enterprise network is a fundamental security control that is nevertheless challenging due to the limitations of fingerprint-based classification and black-box machine learning. In this paper, we address such limitations by proposing a similarity-based clustering method. We evaluate our solution and compare it to a state-of-the-art fingerprint-based classification engine using data from 20,000 devices. The results show that we can successfully classify around half of the unclassified devices with a high accuracy. We also validate our approach with domain experts to demonstrate its usability in producing new fingerprinting rules.**

***Index Terms*—Internet of Things; Classification; Clustering.**

I. Introduction

There is a growing number and variety of devices connecting to enterprise networks due to trends such as BYOD and the rapid deployment of IoT [1]. This situation makes it critical for organizations to have an accurate inventory of these devices, including information such as device type (e.g., computer, IP camera, server), vendor, and operating system. This inventory enables security controls with monitoring, vulnerability assessment, and network access control and allows organizations to detect attacks leveraging rogue devices, among others [2]–[4].

Creating and maintaining such an inventory requires the ability to classify devices from their traffic on the network. However, tools commonly used to identify connected devices are not suited to the current enterprise environment. While traditional device classification relies on manually-defined fingerprints, it has limitations such as scalability and low performance in terms of *coverage/accuracy* and *granularity*. The first is the ability to classify/correctly classify a device, and the second requires it to be specific enough to be useful.

Machine Learning (ML)-based alternatives have been proposed to overcome these issues. However, most of these approaches are "black-box", i.e., the features and classification algorithms used do not allow for straightforward interpretation and actionability of the results. These solutions also require a large amount of labelled data for training and use models that require retraining when adding new types of devices. Moreover, related works adopting these solutions suffer several limitations, such as the narrow focus on IoT devices, thus missing the enterprise-wide picture, and evaluating limited datasets, both in terms of the number of devices and in the variety of their types.

In this paper, we address the limitations of manual fingerprint-based and black-box machine learning-based device classification by proposing a similarity-based clustering method. Our solution uses unsupervised learning with a semantic-driven feature selection aiming to improve the coverage, accuracy, and granularity of an existing fingerprint-based classification engine. Unsupervised algorithms overcome the need for labelled data, which helps to keep up with the growth of the number and diversity of devices.

The underlying idea is that devices with similar attributes have a similar type, allowing us to cluster them together and give them the same labels. Our solution can be broken down into the following steps. First, we select semantically meaningful attributes of devices as features. Second, we define distance functions to evaluate the similarity between devices based on these features. Third, we cluster devices which are similar based on the notion of distance we defined. Finally, the clusters are analyzed, resulting in a number of classification suggestions that can be automatically applied or manually reviewed by an expert.

We evaluate our solution and compare it to a state of the art fingerprint-based classification engine using data from 20,000 devices. The results show that we can successfully classify around half of devices that were left unclassified by the other engines. We also validate our approach with domain experts to demonstrate its effectiveness to solve industry-wide problems.

The rest of this paper is organized as follows. Section II discusses the background and related work. Section III states the classification problem that we aim to solve, while Section IV describes our methodology to solve this problem. Section V shows the implementation and experimental results. Finally, Section VI concludes the paper and discusses possibilities of future work.

II. Background and Motivation

Device classification (also known as "device fingerprinting" [5]–[7]) works by applying a classification function on data about networked devices [8]. This data can be collected passively or actively. Passive techniques are non-intrusive, capturing packets as they move on a network. Active techniques interact with devices by sending them packets and observing responses or change of behaviour.

Most commonly used classification approaches (e.g., nmap[1], which is active, or p0f[2], which is passive) are rule-based, comparing data collected from a device against a set of

[1]https://nmap.org/
[2]http://lcamtuf.coredump.cx/p0f3/

fingerprints (or signatures) [9]–[11] that are manually designed. However, fingerprint-based classification is limited in terms of coverage, accuracy, and granularity [8].

Device classification is a well-researched problem, and recently the focus has been on classifying IoT devices [8], [12]–[25]. While many proposed approaches intend to address the same problem, they diverge in ways that we explore below.

First, the granularity of classification ranges from simply distinguishing IoT and non-IoT devices [26] to identifying the specific model and software version of a device [12].

Second, data collection ranges from passive techniques using network flows [14], [20], [24], packet headers [12], [16], [27], application layer data [15], [19], [21], or a combination of those methods [17], [22], [23], [25], to active techniques [1], [13]. Some research also investigates how to enrich network data with information found online [8].

A common limitation of these works is the scope of devices considered, which usually consists in smart home IoT devices and other consumer products. This setting does not reflect the broader range of devices found in enterprises, such as connected medical devices or operational technology equipment. In addition, the evaluation of classification methods demonstrated are often conducted in a lab setting with little or no actual user operation, which does not accurately represent "real-life" network deployments.

Recent solutions leverage machine learning to circumvent the expensive process of designing fingerprints manually. The majority of the work implements supervised learning algorithms [14], [17], [19]–[21], [23], [24], [27], including deep learning [16], [22]. These models not only require training with a sufficient amount of data for the devices to be classified, but they also require to be retrained each time a new device is introduced. This constraint limits such approaches to scale adequately as new devices are introduced. Additionally, these algorithms function as a black box, providing little insights into the rationale for classification. A network operator as an end-user would have limited information about why a certain device has been classified as such, and no means of troubleshooting if the classification is inaccurate.

Techniques like clustering have been applied in the past to classify network traffic [28], [29]. Such algorithms are ideal to find similarity between entities and are well-suited to device classification by considering the semantic of the features and defining a proper similarity measure [30]. From the discussion above and to the best of our knowledge, there no other work that focuses on applying clustering with semantic-driven feature selection to classify devices.

III. Terminology

A *device* is a networked-enabled system fulfilling certain functions and delivering value to an organization. Let us assume a set of devices D and a set of types are given. A *type* characterizes the main function of a device, such as printer, or IP-camera. Device types are organized in a *taxonomy*, which is a finite directed tree with nodes labelled with strings, as shown in Figure 2. The label of the root is the empty string and no two children of a node share the same label. We also interpret the taxonomy as a partial order on the nodes, with the root being the smallest element. The *level of a node* is its depth in the tree, i.e. the length of the path to the root. We denote a specific node by the sequence of labels from the root, e.g., "/IT/Networking/Router" denotes a router. Let us assume a taxonomy of devices types, which we call DT, is fixed.

Definition III.1. A *device classification function* $c : D \to DT$ assigns a device type in DT to each device in D, which we call the label of the device. □

Given a classification function, the *level of a device* is the level of its label. A device is called *unlabelled* if its level is 0 and *labelled*, otherwise. Depending on its level, a labelled device can be qualified as *weakly-labelled* (the level is 1) or *well-labelled* (the level is at least 2). In the taxonomy DT, a level 0 label gives no information about a device, and a level 1 label gives very limited information. Hence devices with a label of level 0 or 1 (i.e. unlabelled and weakly-labelled devices) are referred together to as *to-be-classified* devices.

The fitness of a device classification function is measured in terms of *coverage* (percentage of well-labelled devices), *granularity* (the levels of the devices) and *accuracy* (percentage of correctly labelled devices compared to some ground truth).

Definition III.2. We define the *device classification problem* as finding a device classification function that optimizes some fitness metric(s). □

IV. Methodology

We use similarity-based clustering to solve the device classification problem stated in Section III. Figure 1 shows an overview of the methodology.

Fig. 1. Overview of the methodology

First, we capture semantically meaningful features describing the devices we want to classify. Second, we compute the distance between devices using distance metrics tailored for each feature. Third, we use a clustering algorithm that groups similar devices based on their distances. Fourth, we analyze the *quality* of resulting clusters and assess their *actionability*. Finally, we present the results to an operator. This white-box approach allows using the suggestions in different ways to improve the existing device classification function. Below, we explain each step in detail.

A. Feature selection

A device d can be represented as a tuple of features $(f_1, \ldots, f_n)$, where each feature f_i in a domain V_i captures an aspect of d. The device classification results are expressed in term of features to user. They should have meaning and

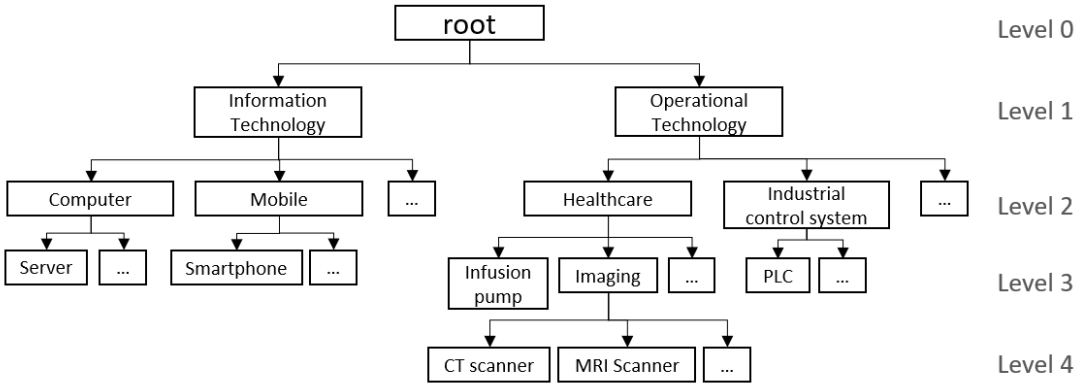

Fig. 2. Extract of a taxonomy of device types

be directly understandable by user. We call them semantically meaningful features.

B. Dissimilarity functions and distance

A dissimilarity function $dissim_i : V_i \times V_i \mapsto [0, 1]$ computes the dissimilarity between two devices d and d' with respect to a single feature f. A dissimilarity score of 0 represents the highest similarity between two feature values (i.e., they are identical) and 1 the lowest. For each feature, we define a dissimilarity function based on the feature's domain V_i.

We then calculate the distance between two devices as follows. For a given feature vector $(f_1, \ldots, f_n)$ along with their respective dissimilarity functions $(dissim_1, \ldots, dissim_n)$, we define the distance D between two devices $d = (f_1, \ldots, f_n)$ and $d' = (f'_1, \ldots, f'_n)$ as the average dissimilarity per feature:

$$D(d, d') = \sum_{i=1}^{n} (dissim_i(f_i, f'_i))/n$$

with $D \in [0, 1]$, where 0 means that two devices have exactly the same values for every feature.

C. Clustering

We then input the distance D for a multiset of devices into a clustering algorithm. Its goal is to aggregate similar objects together while separating dissimilar ones in different clusters [31]. We adopt density-based clustering [32]. Density-based clustering searches for areas with a high density of objects separated by regions of lower density [31]. Additionally, it can find clusters of different sizes and shapes while also handling noise (i.e., objects that cannot be clustered) [33].

In our device classification context, density-based clustering is well suited as it follows a non-parametric approach. It does not require the specification of total clusters in advance nor makes assumptions regarding their shape, unlike the parametric approaches. We leverage density-based clustering to find clusters of similar devices comprising both well-classified and to-be-classified, allowing us to classify the latter.

D. Cluster analysis

Using clustering algorithms requires a way to evaluate the quality of the results [34]. This evaluation, known as *cluster validation*, has been the topic of extensive research [34]–[38]. Clustering quality can be measured by a Clustering Validation Index (CVI) [38], such as the Silhouette Index or Davies-Bouldin Index [39]. Most CVIs estimate the cohesion (also known as "compactness" or "tightness") and separation of each cluster, then combine these values into a quality measure [37]. However, these CVIs consider clustering as an application-independent mathematical problem and provide little information about the *utility* of the clustering for a given use case [40].

For this reason, we also introduce a new cluster validation index designed to assist in our classification problem. The first component of this validation evaluates the quality of a cluster. The second component measures the actionability of a cluster from the point of view of a user.

1) Structural quality of a cluster: To evaluate the structural quality of a cluster, we look at the dissimilarity between its devices. We first look per feature how similar they are for that feature and consider them good if they are similar on at least some features. The feature dissimilarity AD_i of a Cluster C is computed as the average of the dissimilarity $dissim_i$ between all devices in the cluster (with $avg(\emptyset) = 1$):

$$AD_i(C) = avg(dissim_i(f_i, f'_i) | d, d' \, in \, C | f_i, f'_i \neq NaN)$$

We consider the feature similar if it does not exceed a threshold t_{diss}. We say a cluster is structurally *bad* if the count of similar features ($\#\{AD_i(C) < t_{diss} | i = 1 \ldots n\}$) does not reach a minimum threshold t_{good}.

2) Cluster actionability: To evaluate the actionability of a cluster we look at the specific use case; what can the user do with the cluster. For our use case of classification we consider the following *suggested actions* that may be assigned to a cluster:

- **Classify to-be-classified devices**: suggests a label for to-be-classified devices in the cluster.

- **Increase from level n to level $n+x$ label**: suggests an improved level $n+x$ label for level n ($\geqslant 2$) devices in the cluster.
- **Fix misclassification**: suggests the label of specific devices in the cluster may be incorrect.

Taking the suggested action from an actionable cluster improves the cluster, either with regards to coverage (first action), granularity (second action) or accuracy (third action, assuming the indicated devices were indeed classified incorrectly). While we consider action 'classify to-be-classified devices' can be taken automatically, the other two would be presented to an operator. Of course we still need a procedure that actually assigns these labels, and we discuss the one we use next.

If labelled devices in a cluster share the same parent at Level-2, we consider this a good indication that this Level-2 label would also apply for to-be-classified in the same cluster. Formally, we call a clusters *level-2-coherent* when (the cluster has well-classified devices and) the level of the greatest lower bound (GLB) of the labels of all well-classified devices in the cluster is at least 2:

$$level(GLB(\{label(d) | d \in C \wedge level(d) \geqslant 2\})) \geqslant 2$$

For instance, a cluster with only well-classified devices labelled as "/Information Technology/Mobile/Smartphone" or "/Information Technology/Mobile/Tablet" is level-2-coherent, since these devices share the same Level-2 label, "/Information Technology/Mobile". If a cluster is level-2-coherent we assign 'classify to-be-classified devices' as action and suggest the level-2 ancestor of the well-classified devices as the label for to-be-classified devices.

When a cluster is level-2-coherent, we look at the number of devices for each label in that cluster. We characterize as *main label* the highest label that has at least $t\%$ devices of the well-classified devices under it (with $t > 50$). If the main label's level is greater than 2 and if there are devices in the cluster with a lower level label, we assign 'increase from level n to level $n+x$ label' as action and suggest this label for the latter devices, thus increasing their granularity.

When a cluster is not level-2-coherent but does have a main label of at least Level-2, we suggest 'Fix misclassification' action. The devices whose labels are not under the main label are the ones suggested to be misclassified.

V. Experimentation

This Section details the implementation of the methodology outlined in Section IV, with explanations about the experimental setup and results obtained.

A. Implementation

We detail below the features and similarity functions chosen, as well as the clustering algorithm and cluster validation.

Features and similarity functions. The features we used in our implementation are presented in Table I, with references motivating their use in device classification. Based on the value types of the selected features (strings and sets of strings), we choose two well-known similarity functions, namely the Levenshtein Index [41] and the Jaccard Index [42].

TABLE I
Selected features, including their source, example after preprocessing, and similarity function

Source	Features	Example	Function
DHCP	Class	'Printers'	Levenshtein
	Options	(53,61)	Jaccard
	Param.Request List	(1,121,3)	Jaccard
	Vendor	'Cisco'	Levenshtein
	Operating System	Windows Vista/7	Levenshtein
HTTP	Banner	(Mozilla/4.0, Microsoft)	Jaccard
	Headers	(Server Microsoft-IIS)	Jaccard
MAC	Vendor (24 bits)	'Hewlett Packard'	Levenshtein
	OUI	'cc9891'	Levenshtein
Nmap	Banner	(23/tcp telnet, 80/tcp http)	Jaccard
	OS	(Windows 7, Windows)	Jaccard
	Function	'Windows'	Levenshtein
	Open Ports	(139/tcp, 445/tcp)	Jaccard
p0f	Fingerprint	(Windows NT kernel 5.x)	Jaccard
Other	Group	(Network, Non Corporate)	Jaccard
	Network Function	'Windows Machine'	Levenshtein

The Levenshtein Index, also called edit distance, computes the minimal number of insertions, deletions, and substitutions to make two strings the same [41]. It allows to compare two strings of arbitrary length, thus it is suited to measure the distance between feature values of string type.

The Jaccard Index measures the similarity between two sets A and B as the cardinality of the intersection divided by the cardinality of the union of the sets, i.e. $J(A,B) = \dfrac{|A \cap B|}{|A \cup B|}$. The Jaccard Index is well-suited for features whose values are sets of strings. We implement a similarity function for those features (see Table I) that first transforms the set of strings into two sets of tokens and then computes the Jaccard Index.

For each feature, its similarity function is used to compute a pair-wise similarity matrix as described in IV.

Clustering. We selected the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) algorithm [32], [43] for clustering. Like other density-based clustering algorithms, HDBSCAN keeps parameter tuning to a minimum, not requiring to specify a number of clusters nor assuming certain cluster shapes, and it handles noise (i.e,. outliers). HBSCAN stands out by its capacity of finding clusters of varying densities and by being more robust to parameter selection [44]. HDBSCAN is also suitable for this use case because it does not require the distance between each pair of points, only expecting a connected graph. It allows us to use NaN values for uncomputed distances of missing features as defined in the previous subsection.

We use the implementation of HDBSCAN available at [44], which supports pre-computed distance matrices. This allows us to feed the algorithm with our devices distances computed with our distance metric (IV). We only have one parameter to tune: the minimum number of devices that should be together to be considered as a cluster. We set it to a value of 3. In the context of device classification there is no requirement regarding the size of a cluster. Keeping this parameter low

increases the chance for devices to be clustered with similar ones, which could ultimately help with classification.

Cluster analysis and interpretation of results. We analyze the clusters by implementing the cluster analysis presented in SectionIV. The evaluation of the goodness of clusters is done as follow: With the 16 features we selected, we define a good cluster as one that has least 3 features ($t_{good} = 3$) with an average dissimilarity below 0.2 ($t_{diss} = 0.2$). These thresholds are chosen based on our empirical experimentation.

Regarding the heuristic-based analysis, we implement a function that identifies the type of clusters as described in Section IV. It analyzes the labels of the devices, and based on the type of the cluster, it results in one or more suggested actions. For example, let us consider a cluster containing 20 devices: 15 labelled as /Information Technology/Mobile/Smartphone, 3 as /Information Technology/Mobile/ and 2 unlabelled (unknown). The resulting suggested actions for that level-2-coherent cluster are 1) Classify the two unlabelled devices and 2) Change Level-2 to Level-3 label for the three Mobiles.

B. Experimental setup

Dataset. For our experiments, we used datasets containing data up to 20,000 devices from a healthcare enterprise network. The devices found on these networks comprise a wide variety of types, ranging from traditional computers and servers, to IoT devices such as IP cameras. On some networks, industry-specific devices can also be found, such as patient monitors and other connected medical devices in hospitals.

Within the network, we capture device data by passively monitoring their communications with network monitoring tools such as p0f and actively scan them with nmap and other tools. This data is fed to Forescout's existing classification engine, which assigns for each device a unique ID and applies a set of fingerprints against the data to assign a label to that device. The classification engine has the limitations of fingerprint-based systems discussed in Section II.

To create a dataset, we extract and store the devices and respective data as a collection of key-value pairs in a JSON file. Each device is represented as a tuple consisting of its unique ID, its label (if found by the classification engine, otherwise "Unknown") and its values for the 16 features selected (see Table I).

In the end the dataset comprises 15,760 well-classified and 4,240 to-be-classified devices.

Evaluation metrics. To evaluate the fitness of our approach, we measured the coverage, granularity and accuracy of the clustering results as follows:

- **Coverage**: Number of to-be-classified devices having a suggestion of classification.
- **Granularity**: Number of level-2 devices with a classification suggestion to level-3 or higher.
- **Accuracy**: Number of devices potentially misclassified.

In our evaluation, we consider the initial classification from the fingerprint-based classification engine described above as baseline for the coverage (i.e., how many to-be-classified devices are in the dataset). It is non-trivial to obtain a comparable baseline for the two other metrics.

C. Validation

Before running the main experiment, we performed a 20-fold cross-validation to assess the solution's effectiveness by checking if the automatically assigned labels are correct. For this test, we used a sample of 20,000 well-classified devices with an accurate primary classification (i.e., ground truth). Using a database of 4.4 million devices from more than 200 different companies operating in various industry sectors, we observed that both in the overall database and in these companies' specific networks, there are around 21.2% of to-be-classified devices. With that, we removed the label of 4,240 devices randomly selected from the sample to reproduce the same distribution of to-be-classified devices expected in a typical database. The tests resulted, on average, in 1387 clusters (1349 good clusters and 39 bad clusters) and 6,609 devices unclustered (i.e., noise points).

As shown in Figure 1, a Level-2 label in the taxonomy represents the minimum information about the type of the device (i.e., a computer, a mobile device, an ICS, etc) and gives a good amount of details about the device. Correctly suggesting a label up to at least Level-2 means that it achieved a consistent label, referred to in this cross-validation as a Consistent Level-2 Classification. In case that the suggestion is also correct to its full extent (i.e. the highest level of the primary label), it means that it achieved a consistent classification with a good granularity level; in this case, it will be called a High-Granularity Classification. For example, suggesting the label 'IT/Computer' to a device with the removed primary label 'IT/Computer/Workstation' means a Consistent Level-2 Classification, while suggesting 'IT/Computer/Workstation' means a High-Granularity Classification.

The 20-fold cross-validation resulted in an average of 2174 devices with a new suggested label out of the initial 4,240 to-be-classified. By comparing the label suggested in the experiments for the clustered devices with the primary classification removed from them, the methodology achieved, on average, 99.5% of Consistent Level-2 Classification and 95.6% of High-Granularity Classification during the tests. These results indicate that the methodology classifies devices with reliable labels and excellent precision, validating the following results.

D. Results

We execute our program with the dataset created and analysed the results from the main experiment described on Section V-B. Among the 20,000 devices, 12,595 devices are grouped into 1310 different clusters, while the remaining 7405 devices are considered as noise points. From the 1310 clusters, 1,285 are actionable and suggestions are provided which can help classifying a total of 1539 to-be-classified devices (out of the initial 4,240). The amount of 1539 to-be-classified devices candidate to classification can be broken down into 802 unlabelled devices (out of 2,234 unlabelled)

and 737 level 1 devices (out of 2006). Moreover the output of our experimentation suggests 36 level 2 devices (within 29 clusters) that can be improved with a level 3 classification, and 533 devices (from 120 clusters) that could be potentially misclassified.

The results can be used in two different use cases as mentioned in Section IV. In the case of automatic classification, 1539 to-be-classified devices (802 unlabelled and 737 level 1 devices) can be labelled automatically. In the case of manual classification the end user is presented with the 249 clusters and their respective classification suggestions mentioned above. Assuming all the suggestions being validated by the user, it results in a total of 2108 devices with an improved classification.

VI. Conclusion

This paper addressed the limitations of device classification based on traditional device fingerprinting and black-box machine learning by proposing a semantic similarity-based clustering method.

We evaluated our solution and compared it to the state-of-the-art fingerprint-based classification using data from 20,000 devices. The results show that we can successfully classify a good amount of the to-be-classified devices among these with 99.5% consistency and 95.6% high granularity. Also, the validation of our approach using real-world databases demonstrated its effectiveness to solve industry-wide problems.

The results demonstrated that the method meets the expectations. The proposed solution delivered consistent labels with high granularity to unclassified devices automatically while still allowing a specialist to perform manual verifications on these labels. Thus, the method can reduce a good portion of the work involved in manual and fingerprint-based classification.

On top of that, the method improves over traditional black-box classification mechanisms by showing classification suggestions to an operator. This white-box approach adds actionability and increases the results' reliability by combining automatic classification with manual verification.

Future work.

As future work, we plan to assess the possibility of improving the structural quality metrics to increase the number of devices receiving a label suggestion while maintaining the achieved consistency and granularity rates. We also intend to analyse the applicability of this method on distributed environments with privacy-sensitive information.

Acknowledgments

The authors received support to develop this research from the Project SeCoIIA (Secure Collaborative Intelligent Industrial Assets).

References

[1] P. Bajpai, A. K. Sood, and R. J. Enbody, "The art of mapping iot devices in networks," *Network Security*, vol. 2018, no. 4, pp. 8–15, 2018.

[2] A. Cui and S. J. Stolfo, "A quantitative analysis of the insecurity of embedded network devices: results of a wide-area scan," in *Proceedings of the 26th Annual Computer Security Applications Conference*, 2010, pp. 97–106.

[3] Office of Inspector General, "Cybersecurity management and oversight at the jet propulsion laboratory," https://oig.nasa.gov/docs/IG-19-022.pdf, 2019.

[4] Y. Mirsky, T. Mahler, I. Shelef, and Y. Elovici, "Ct-gan: Malicious tampering of 3d medical imagery using deep learning," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019.

[5] F. Veysset, O. Courtay, O. Heen, I. Team *et al.*, "New tool and technique for remote operating system fingerprinting," *Intranode Software Technologies*, vol. 4, 2002.

[6] M. Smart, G. R. Malan, and F. Jahanian, "Defeating tcp/ip stack fingerprinting." in *Usenix Security Symposium*, 2000.

[7] G. Taleck, "Ambiguity resolution via passive os fingerprinting," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2003, pp. 192–206.

[8] 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)*. USENIX Association, Aug. 2018, pp. 327–341.

[9] H. J. Abdelnur, R. State, and O. Festor, "Advanced network fingerprinting," in *Recent Advances in Intrusion Detection*, R. Lippmann, E. Kirda, and A. Trachtenberg, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 372–389.

[10] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 94–104, 2016.

[11] D. Herrmann, K.-P. Fuchs, and H. Federrath, "Fingerprinting techniques for target-oriented investigations in network forensics," *Sicherheit 2014–Sicherheit, Schutz und Zuverlässigkeit*, 2014.

[12] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. 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)*. IEEE, 2017, pp. 2177–2184.

[13] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, "Can we classify an iot device using tcp port scan?" in *2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*. IEEE, 2018, pp. 1–4.

[14] A. Hsu, J. Tront, D. Raymond, G. Wang, and A. Butt, "Automatic iot device classification using traffic behavioral characteristics," in *2019 SoutheastCon*. IEEE, 2019, pp. 1–7.

[15] P. R. Pêgo and L. Nunes, "Automatic discovery and classifications of iot devices," in *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, 2017, pp. 1–10.

[16] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Z. Yang, "Automatic device classification from network traffic streams of internet of things," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. IEEE, 2018, pp. 1–9.

[17] M. R. Santos, R. M. Andrade, D. G. Gomes, and A. C. Callado, "An efficient approach for device identification and traffic classification in iot ecosystems," in *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2018, pp. 00 304–00 309.

[18] J. N. Suárez and A. Salcedo, "Id3 and k-means based methodology for internet of things device classification," in *2017 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE)*. IEEE, 2017, pp. 129–133.

[19] 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," *arXiv preprint arXiv:1709.04647*, 2017.

[20] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Characterizing and classifying iot traffic in smart cities and campuses," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2017, pp. 559–564.

[21] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying iot devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, 2018.

[22] K. Yang, Q. Li, and L. Sun, "Towards automatic fingerprinting of iot devices in the cyberspace," *Computer Networks*, vol. 148, pp. 318–327, 2019.

[23] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, and M. Gurusamy, "Deft: A distributed iot fingerprinting technique," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 940–952, 2018.

[24] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "Audi: Toward autonomous iot device-type identification using periodic communication," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1402–1412, 2019.

[25] N. Ammar, L. Noirie, and S. Tixeuil, "Autonomous iot device identification prototype," in *2019 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 2019, pp. 195–196.

[26] J. Ortiz, C. Crawford, and F. Le, "Devicemien: network device behavior modeling for identifying unknown iot devices," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, 2019, pp. 106–117.

[27] 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*. ACM, 2017, pp. 506–509.

[28] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, ser. MineNet 06. New York, NY, USA: Association for Computing Machinery, 2006, pp. 281–286.

[29] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, "Detecting behavioral change of iot devices using clustering-based network traffic modeling," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7295–7309, Aug 2020.

[30] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Advances in neural information processing systems*, 2003, pp. 521–528.

[31] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011.

[32] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 2013, pp. 160–172.

[33] Y. Zhu, K. M. Ting, and M. J. Carman, "Density-ratio based clustering for discovering clusters with varying densities," *Pattern Recognition*, vol. 60, pp. 983–997, 2016.

[34] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of intelligent information systems*, vol. 17, no. 2-3, pp. 107–145, 2001.

[35] U. Maulik and S. Bandyopadhyay, "Performance evaluation of some clustering algorithms and validity indices," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 12, pp. 1650–1654, 2002.

[36] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 911–916.

[37] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. PéRez, and I. Perona, "An extensive comparative study of cluster validity indices," *Pattern Recognition*, vol. 46, no. 1, pp. 243–256, 2013.

[38] J. Hämäläinen, S. Jauhiainen, and T. Kärkkäinen, "Comparison of internal clustering validation indices for prototype-based clustering," *Algorithms*, vol. 10, no. 3, p. 105, 2017.

[39] S. Petrovic, "A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters," in *Proceedings of the 11th Nordic Workshop of Secure IT Systems*. Citeseer, 2006, pp. 53–64.

[40] U. Von Luxburg, R. C. Williamson, and I. Guyon, "Clustering: Science or art?" in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 65–79.

[41] G. Navarro, "A guided tour to approximate string matching," *ACM computing surveys (CSUR)*, vol. 33, no. 1, pp. 31–88, 2001.

[42] Y. Jiang, G. Li, J. Feng, and W.-S. Li, "String similarity joins: An experimental evaluation," *Proc. VLDB Endow.*, vol. 7, no. 8, p. 625636, Apr. 2014. [Online]. Available: https://doi.org/10.14778/2732296.2732299

[43] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 1, pp. 1–51, 2015.

[44] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *The Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017.