2021 Fourteenth International Workshop on Selected Topics in Mobile and Wireless Computing

Clustering Unknown IoT Devices in a 5G Mobile Network Security Context via Machine Learning

Tony Hämmäinen
Dept. of Computer Science
Aalto University
Espoo, Finland
tony.hammainen@kolumbus.fi

Julen Kahles
R&D
Ericsson
Jorvas, Finland
julen.kahles@ericsson.com

***Abstract*—We propose a novel machine-learning pipeline for clustering unknown IoT devices in an industrial 5G mobile-network setting. Organizing IoT devices as few homogeneous device groups improves the applicability of network-intrusion detection systems. More specifically, we develop feature engineering methods that transform IP-flows into device-level data points, define distance metrics between the data points, and apply the DBSCAN algorithm on them. Our experiments on a simulated IoT device network with varying levels of noise show that our proposed methodology outperforms alternative methods and is the only one producing a robust grouping of the IoT devices with noise present in the traffic data.**

***Index Terms*—clustering, IoT, 5G, machine learning, network security, IP flows**

I. INTRODUCTION

The industrial market for the Internet-of-Things (IoT) devices is growing rapidly. Ericsson expects the installed base of cellular IoT devices to increase from one to four billion between 2018 and 2024 [1]. Traditionally, security solutions in this domain have relied on manual configuration, monitoring, and response processes; however, the expanding device attack surface calls for automated and scalable methods.

We are looking into a scenario where a mobile network operator is providing security services to its industrial clients with large quantities of IoT devices. In this context, the security provider does not directly operate the IoT devices, thus, the intended communication patterns of the devices are unknown to the security provider. Not knowing intended communication patterns makes intrusion detection a challenging task, as traditional methods rely on setting static rules for whitelisted traffic. Modern machine-learning-/anomaly-detection-based intrusion detection approaches bypass this problem by assuming that common traffic is legitimate while rare traffic is malicious. However, with a heterogeneous IoT-device base, this can lead to systematically misclassifying uncommon device types as malicious since their intended traffic is not common relative to other devices' traffic. For instance, if a factory has thousands of temperature sensors but only a handful of humidity sensors, the humidity sensors can get systematically flagged as malicious as a result of the assumptions behind anomaly detection.

Given this scenario, we make the following contributions in this paper[1]:

1) We propose a novel approach of clustering unknown IoT devices in a 5G mobile network into groups of homogeneous devices with the motivation that after clustering, any anomaly-based intrusion detection method can be applied without a high risk of misclassifying uncommon devices as malicious, as their legitimate traffic will now be common within their respective cluster.
2) In order to cluster the devices with high precision, we introduce a novel feature-engineering method, where device-level data points are formed as distributions of relative frequencies of triplets containing the network IP address, network port, and IP protocol, which we propose to call *the network socket identifier*.
3) We show through experiments that our approach of computing cosine distances between network-socket identifiers outperforms typical IP-flow feature-engineering approaches when measuring DBSCAN clustering accuracy by a large margin. The results indicate that using clustering as a preprocessing step for intrusion detection is viable only through using the feature-engineering method we introduce.

II. RELATED WORK

The need for security providers to know the intended communication patterns of their networks' devices has been recognized in the industry, as evidenced by Cisco having published the Manufacturer Usage Description (MUD) as an IETF RFC [2]. Its purpose is to let manufacturers define the intended communication patterns for their devices so that the trusted patterns can be used to enforce security. Having access to existing MUDs would render our work redundant, however, the current status is that the majority of existing devices do not have MUDs defined for them. Furthermore, in our setting, it is unfeasible for the security provider to map MUDs to devices as they can not access device specifications. Thus, we aim to close this gap by grouping the devices with the same intended communication patterns through clustering. To

This work has been funded by Ericsson Finland.

---

[1] The results this research presents are obtained within the Master's thesis work of the first author under the advisorship of the second author.

our best understanding, no prior work has attempted to use clustering for this purpose.

Cisco states in their MUD documentation that "numerous fingerprinting methods are being explored in the market to profile IoT devices to address this vulnerability", though dismissing them as being subject to spoofing [3]. For instance, one such fingerprinting tool is MUDgee, published by Hamza, Ayyoob, et al. [4]. A challenge with these fingerprinting tools is that they trust all captured traffic to be intended and legitimate: this assumes that no device has been compromised before the fingerprinting analysis. Hence, they are not suitable for networks with existing device bases that have possibly been subject to intrusions. We aim to solve this challenge through leveraging the power of data. We assume that it is unlikely for all devices to be compromised in the exact same way, thus, leveraging similarities between device communication patterns to identify legitimate traffic.

Several authors have studied deep-packet inspection as a means of classifying devices types [5]–[7]. Furthermore, several authors have studied packet interarrival times that identify timing-dependent characteristics (such as clock frequencies and packet-generation processes) [8]–[10]. Nonetheless, these approaches are not available to us due to our starting point of having access to IP-flow-level data only, which conceals packet-level details.

Many IoT-related intrusion detection approaches opt to go toward a supervised learning route which eliminates the need for the assumptions of anomaly detection that cause systematic bias [11]–[18]. However, a supervised approach requires having labeled ground-truth data samples available for all traffic types of all device types, which is impractical in our setting. Even though the ultimate purpose is different, the multiple-connection-derived (MCD) features from supervised learning can directly be applied in the feature engineering for clustering as well. Davis et al. have published a review of data preprocessing methods for anomaly-based intrustion detection [19]. The majority of approaches use simple descriptive statistics, such as averages, counts, and sums of each IP header field. We assume these techniques as benchmarks in our experiments.

Previous research on performing feature engineering on IP flows typically ignores or aims to aggregate high-dimensional categorical features, such as IP addresses, port numbers, and protocol numbers to simple statistics. Unlike previous research, our approach extracts the vast predictive power of these high-dimensional categorical features. This is possible because we compute pairwise distances between devices, which allows us to utilize the full dimensionality of a pair of data points without having to form a matrix containing all possible categorical values of all data points, which would be unfeasible resource-wise.

III. BACKGROUND

Our aim is to recognize homogeneous IoT device groups from their 5G network traffic data, more specifically, their time series of IP flows. Deriving homogeneous device groups from IP flows involves two phases. Firstly, the raw data must be processed into device-level data points, and then, a clustering algorithm is applied on the data points. In this section we first discuss the available data, and then explain what this means for the choice of a clustering algorithm.

A. Data characteristics

The following characteristics describe the context of the IoT devices in the industrial mobile-network setting:

1) We expect IoT devices to exist in massive numbers, which requires scalability from the system, both in terms of data-point counts and data-feature dimensionality (vast number of unique target addresses, ports, and protocols).
2) The amount of data generation per device varies greatly: some devices send large data streams continuously, while others send single packets infrequently.
3) True underlying device groups vary greatly in size. We expect cluster sizes to range from a few devices up to tens of thousands of devices in live networks.

The raw data captured from the 5G network consists of header information of IP packets, aggregated to the IP-flow level, listed in Table I. Looking at the data from a data-science perspective, we notice three characteristics that make the data challenging to deal with. These characteristics are (1) mixed data types, i.e., the data contains numerical, categorical, and boolean data types; (2) variable-length data, i.e., the amount of flows per device is dependent on the device's behavior; and (3) high dimensionality of categorical features, i.e., certain fields such as IP addresses, ports, and protocols contain a large amount of possible values, i.e. dimensions.

B. Data simulation

In order to obtain quantified performance measures for the clustering system, we need labeled data. As manually labeling the live network data is unfeasible, we resort to simulating the IoT network traffic using Ixia Breaking Point [20], an advanced tool widely used in the industry for testing and simulation purposes. We consider all the IoT devices to exist within a single radio access network from which they connect to servers that reside in the Internet. The data capture happens at the network probe, which is located within the Packet Core.

We assume the taxonomy introduced in IETF's RFC 7228 that categorizes IoT devices based on how constrained they are into classes 0, 1, and 2 [21]. Furthermore, we add class 3, which we define as smart/unconstrained. We model all device behavior based on the OMA Lightweight M2M (LwM2M) IoT device lifecycle management specification [22]. It includes operations between interfaces regarding device onboarding, boot-ups, updating, configuring, etc. Each process has its own traffic configuration including protocols, network addresses, ports, and data volumes.

We simulate a total of 4700 devices across twelve groups, distributed into four organizations. We define a device group as a set of devices that share all of their intended communication patterns. Device groups that belong to the same organization share some operations. The shared operation across device types could be, e.g., a login routine. Each device group has an assigned class that determines its application protocol and related operations.

Even though IoT devices mainly adhere to deterministic communication patterns, the networks experience external interference. This traffic, which we call noise, can consist of both legitimate and malicious traffic. For example, network administrators might establish one-off connections to devices for administrative purposes, or hackers might be scanning through the network looking for devices with vulnerabilities. In the simulation, we model noise as random connections, using random protocols, and random network IP addresses, while targeting well-known ports of the devices [23].

C. Clustering algorithm

The characteristics of the domain and its data pose constraints to the choice of the clustering algorithm. We have selected to use the density-based spatial clustering of applications with noise algorithm (DBSCAN) in our approach. DBSCAN fulfills three main features that make it a suitable choice in this context; others that do are mainly variants of DBSCAN. Firstly (1), it is able to derive the number of clusters from the data: while many clustering algorithms require a user-inputted count of clusters k, DBSCAN infers it from the data instead, based on hyperparameters epsilon (ε), indicating the minimum distance between points, and *minPts*, indicating the minimum number of nearby points. Secondly (2), DBSCAN is reasonably computationally efficient, being able to reach $\mathcal{O}(n \log n)$, whereas most alternative clustering algorithms capable of inferring cluster counts from the data are magnitudes slower, a crucial factor considering large quantities of IoT devices. Lastly (3), DBSCAN has the capability to deem data points as outliers, i.e., points that do not belong to any cluster. The networks of IoT devices are likely to include outliers, such as administrative personal computing devices, which are not homogeneous with other devices. From a security perspective, it is better to leave uncertain cases to be handled manually than make a false positive clustering.

IV. METHODS

DBSCAN requires querying pairwise distances between device-level data points. However, it does neither specify of what form the data points should be, nor which distance metric should be used to compute pairwise distances with. Thus, we need to aggregate our raw data into device-level data points and define a metric to compute distances between them. These steps have a radical effect on the predictive power of the clustering algorithm, yet there is room to customize these. We introduce our novel feature engineering solution and compare it to typical IP-flow aggregation and distance computation approaches.

A. Forming a device-level data point

Computing a distance between data points requires the data points to be in a standardized format. Since our raw data is of variable length, an aggregation is required. Overall, transforming raw IP-flow data into device-level data points consists of three steps: (1) feature selection, (2) applying aggregation functions, and (3) normalizing the features.

1) Feature selection: The list of available features is presented in Table I. Some of the features have been disregarded due to being used as the identifier, being redundant to other features, or being randomly assigned, thus bearing no predictive power.

Typical approaches to feature selection are dependent on the aggregation functions and the distance metrics used in succession, as these steps limit which data types can be handled. Typical solutions to the challenge of varying data types include (1) only considering continuous features, (2) only considering categorical features, (3) using specialized distance metrics that can deal with mixed data types, or (4) re-encoding the categorical features as numerical features (e.g., one-hot encoding).

Feature	**Type**	**Count of unique**	**Ref 1**	**Ref 2**	**Ref 3**	**Our mtd**
TEID	Categorical	$\gg$ 10 000				
IMSI	Categorical	$\gg$ 10 000	(X)	(X)	(X)	(X)
IMEI	Categorical	$\gg$ 10 000				
Ue IP	Categorical	$\gg$ 10 000				
Ue port	Categorical	$\sim$ 10 000				
Flow start	Continuous	-	X		X	
Flow end	Continuous	-				
Duration	Continuous	-	X		X	
Vol. rec	Continuous	-	X		X	
Vol. snt	Continuous	-	X		X	
IP prtcl no.	Categorical	$\sim$ 10	X	X	X	X
Ntwrk IP address	Categorical	$\gg$ 10 000	X	X	X	X
Ntwrk port	Categorical	$\sim$ 100	X	X	X	X
TCP flags	Categorical	$\sim$ 10	X	X	X	

TABLE I
SELECTED FEATURES FOR THE REFERENCE METHODS AND OUR METHOD

The chosen reference methods for feature engineering correspond to solutions (1), (2), and (3). In reference method (1), we consider both continuous features and categorical features with such aggregations that have a numerical output. Solution (4) is not feasible due to the high dimensionality of the categorical features in our data.

We hypothesize that *IP protocol number*, *network IP address*, and *network port* contain the majority of the predictive power; hence, our approach considers only them. The basis for the hypothesis emerges from the intended communication patterns being dependent on the applications running on the device, and applications being mappable to the combination of these three features.

2) Aggregation from flow level to device level: The typical approaches of aggregating IP flows utilize descriptive statistics as aggregation functions for each input feature [19]. Aggregation functions for the reference methods are listed in Table II.

In our method, we only consider three categorical input features: *IP protocol number*, *network IP address* and *net-*

Aggregation function	**Data type of input feature**	**Data type of aggregation output**	**Ref mtd 1**	**Ref mtd 2**	**Ref mtd 3**
Count	-	Numerical	X		X
AvgTimeBetween	Datetime	Numerical	X		X
Mean	Numerical	Numerical	X		X
Min	Numerical	Numerical	X		X
Max	Numerical	Numerical	X		X
Std	Numerical	Numerical	X		X
Median	Numerical	Numerical	X		X
Skew	Numerical	Numerical	X		X
Trend	Numerical	Numerical	X		X
Entropy	Categorical	Numerical	X		X
NumUnique	Categorical	Numerical	X		X
Mode	Categorical	Categorical		X	X

TABLE II
AGGREGATION FUNCTIONS USED FOR REFERENCE METHODS

work port. We hypothesize them to contain the majority of the predictive power given they define the socket where a server application is listening in client-server communication. Furthermore, instead of using them as separate features, we combine them to form a new categorical feature. We propose to call this novel concept *a network socket identifier*.

To aggregate the network socket identifier, we compute the frequencies of each distinct value observed over a sliding window of time. This produces a variable-length set of network socket identifier frequencies for each device.

3) Data point normalization: The fields of the IP flow headers are not directly comparable with each other. As is standard practice in data science, they thus need to be normalized with a suitable normalization method per data type.

For the reference methods, we follow industry standards. Reference method 1 contains numerical values, for which Z-score normalization is a match. Reference method 2 contains only categorical values, for which normalization is not applicable. The normalization for reference method 3 has been chosen based on Gower's distance, which necessitates MinMax-scaling numerical features between 0.0 and 1.0.

Our method differs from the reference methods in that only a single flow-level feature is used: the network socket identifier. The aggregation of calculating network socket identifier frequencies produces a set of value counts for each device. Due to the frequency values being calculated based on a single flow-level feature, it would not be necessary to perform normalization. However, our choice for the distance function, cosine distance, requires scaling each device-level data point to a unit-length vector w.r.t. the Euclidean norm. Intuitively, the output can be viewed as proportions of flows with respect to each server application, identified by their network socket.

B. Distance metric

For the reference methods, we follow data-type-dependent industry standards. For reference method 1, the Euclidean distance is chosen. Jaccard index is one of the most common metrics to compare finite sets. Thus, for reference method 2, we select the Jaccard distance, which can be trivially derived from the index. Combining categorical and continuous features in a distance metric requires making assumptions. We resort to assigning a fixed dissimilarity value-range for each feature (0.0 - 1.0). This is the basis for the Gower's distance, defined as the average dissimilarity of each feature, which we choose for reference method 3.

We select the cosine distance for our method due to three properties. Firstly, the cosine distance has an understandable meaning in a high-dimensional space: the angle between two feature vectors. Secondly, the distance range is bound between zero and one, which makes setting hyperparameter values for DBSCAN intuitive. Thirdly, since the cosine distance is calculated with a dot product, it is computationally efficient with sparse matrices - only features for which both feature vectors have a non-zero value need to be considered.

V. RESULTS

In this section, we first define and describe the evaluation metrics. Secondly, we show the results from our experiments and interpret their performance and applicability.

A. Evaluation metrics

For the evaluation and interpretation of the performance of the clustering system, we rely on completeness and homogeneity, which together form the V-measure, as defined by Rosenberg et al. [24]. Together, they provide an understandable description of what types of clustering errors are occurring, depicted in Figure 1. The computation of homogeneity and completeness is based on conditional entropy analysis.

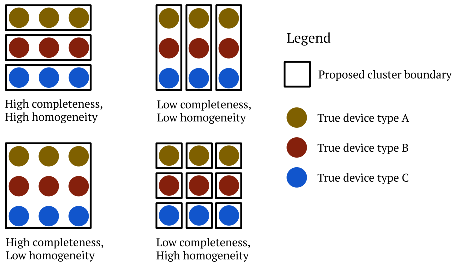

Fig. 1. Examples of proposed cluster boundaries that result in high and low completeness and homogeneity values

Furthermore, as homogeneity and completeness metrics assume that the clustering algorithm assigns all data points into clusters, we need to complement them with our own metric, unlabeledness, which quantifies the proportion outlier assignments by the clustering algorithm, a feature specific to DBSCAN. We define it as the count of outlier assignments divided by the count of all points. We prefer the clustering system to declare points as outliers rather than miscluster them together with data points they should not be clustered with. However, the simulated data set does not contain any true outliers, therefore, a non-zero value of unlabeledness means the clustering system is overly conservative with labeling.

B. Experimental results

To evaluate the methods, we have simulated 10 data sets with increasing degrees of noise, starting from 0% up to 90%.

As the proportion of noise in the data grows, the clustering task becomes increasingly difficult. In the following figures, we plot the evaluation metrics with respect to various values of the hyperparameter epsilon of DBSCAN, ranging from zero to one. This graph is plotted separately for different values of noise. Epsilon effectively dictates how close adjacent points have to be in order to be considered neighbors. For example, with a bound distance range of [0,1], an epsilon value of zero means all points are proposed to have their own individual cluster, while a value of one would have all data points assigned to a single cluster. The robustness of the system is directly proportional to the width of the range of epsilon values for which the clustering system produces good results: the wider the range, the more robust the system is.

In Figure 2, we show how reference method 1 performs. With low values of epsilon, the clustering system is incorrectly declaring the majority of data points as outliers. With epsilon values ranging from 0.2 to 0.5, the clustering performs decently: few points are left unlabeled. The homogeneity and completeness values are quite high, suggesting that the clustering is recognizing some true patterns from the data. However, neither homogeneity nor completeness reach a value of one, meaning that the system keeps making errors in both directions: it clusters together devices that should not be in the same cluster, and assigns devices that should be part of the same cluster into separate clusters. Epsilon values larger than 0.5 tend to result in low homogeneity, suggesting that the system is erroneously assigning most points into few large clusters, regardless of their types.

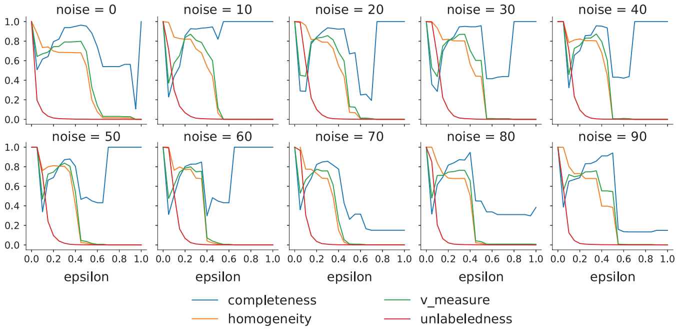

Fig. 2. Results for reference method 1: Numerical aggregations / Euclidean distance

In Figure 3, we show how reference method 2 performs. The clustering never assigns any points as outliers. This makes sense, as there are some categorical fields with very few distinct values, resulting in overlap. With low epsilon values, the clustering tends to propose decent results: with the exception of the noisiest data sets, homogeneity achieves a perfect score, while completeness stays above 0.8. This means that the clustering does not cluster together devices that should not be in the same cluster, but tends to split clusters into too many small clusters, resulting in a true device type being spread across clusters. With higher values of epsilon, the overall performance of the clustering drops considerably.

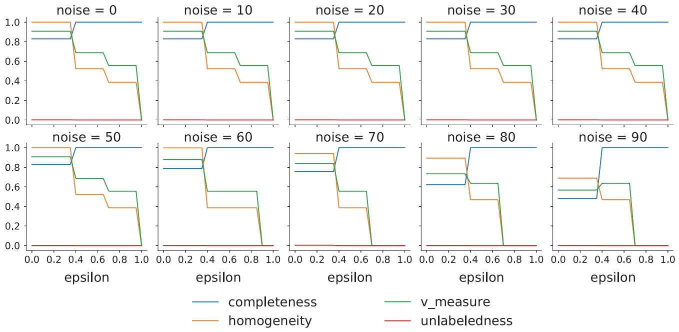

Fig. 3. Results for reference method 2: Categorical aggregations / Jaccard distance

Reference method 3 combines the aggregations used in methods 1 and 2. However, as shown in Figure 4, instead of providing better clustering results with the aid of additional available information, the difficulties of comparing data types meaningfully weigh the end-results down. The clustering system produces decent results with a small range of very low epsilon values, but quickly deteriorates to assigning all data points into a single large cluster.

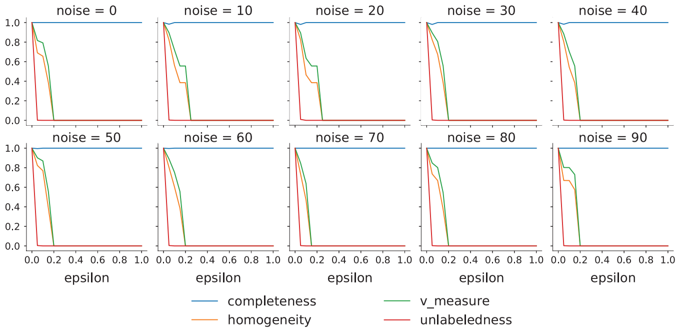

Fig. 4. Results for reference method 3: All aggregations / Gower distance

The results for our method are presented in Figure 5. Unlike the reference methods for any data sets, our method provides a range of epsilon values with a flawless clustering outcome for the data sets with degrees of noise ranging from 0% to 50%. With higher degrees of noise, the clustering produces a reduced completeness score with low epsilon values, and a reduced homogeneity score with higher epsilon values, the optimal epsilon range being around 0.3-0.4. Generally, we can deduce that with low degrees of noise, the system is very robust: a wide range of epsilon values will produce flawless outcomes. With higher degrees of noise, the system will perform decently, and with the choice of the epsilon value, it is possible to optimize either homogeneity or completeness.

VI. CONCLUSION AND FUTURE WORK

The main contribution of this work is presenting the novel concept of clustering IoT devices as a preceding step to intrusion detection to enhance its applicability: we show that IoT devices can be accurately and robustly clustered based solely on observed network traffic. Furthermore, we present

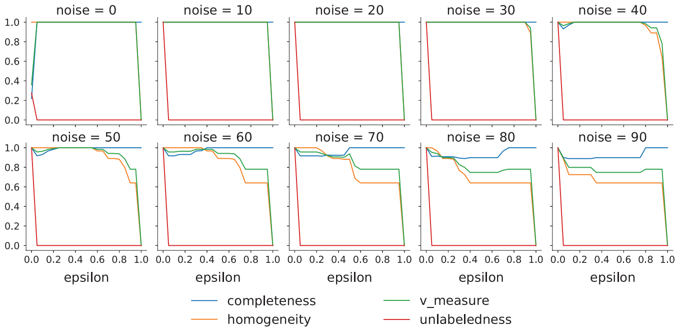

Fig. 5. Results for our method: Relative frequencies of network socket identifiers / Cosine distance

a novel feature engineering approach that outperforms alternative methods in our experiments. Our approach consists of first transforming connection-level IP-flow data into device-level points as relative frequencies of observed *network socket identifier* values (triplets of IP addresses, protocols, and ports), further computing pairwise distances between such data points using the cosine distance, and finally applying the DBSCAN clustering algorithm to them.

As future research work, we propose to (1) formally assess the performance of intrusion detection after having successfully clustered the IoT devices with respect to applying intrusion detection to the raw data, and (2) to investigate ways of optimizing the run-time complexity of the clustering system through better suited data partitioning methods.

With regard to to the former, it is not trivial to formulate a test setting to assess this. Traditionally, intrusion detection algorithms are measured on publicly available and well-known data sets; nevertheless, these data sets do neither aim to simulate IoT devices, nor specify device types. Due to this, our clustering approach can not be assessed using them. We can envision testing happening on live IoT networks, however, the challenge is to obtain reliable labels for intrusions.

With respect to the latter, KD-trees and ball trees are often integrated into implementations of DBSCAN to avoid unnecessary computations, as for example in the implementation by scikit-learn [25]. However, these methods do not perform well in high-dimensional settings. Thus, we suggest studying the applicability of partitioning methods that have taken recent advances in high-dimensional settings, such as proximity graphs, random projections, and locality sensitive hashing.

REFERENCES

[1] F. Jejdling, “Ericsson Mobility Report June 2019,” Ericsson, Stockholm, Sweden, Ericsson Mobility Report, June 2019.

[2] R. D. E. Lear and D. Romascanu, “Manufacturer Usage Description Specification RFC 8520. Retrieved March 2019,” 2019.

[3] Cisco.com, “Manufacturer Usage Description,” 2020. [Online]. Available: https://developer.cisco.com/docs/mud/

[4] A. Hamza, D. Ranathunga, H. H. Gharakheili, M. Roughan, and V. Sivaraman, “Clear as mud: generating, validating and applying iot behavioral profiles,” in *Proceedings of the 2018 Workshop on IoT Security and Privacy*, 2018, pp. 8–14.

[5] B. F. L. M. Sousa, Z. Abdelouahab, D. C. P. Lopes, N. C. Soeiro, and W. F. Ribeiro, “An intrusion detection system for denial of service attack detection in internet of things,” in *Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing*, 2017, pp. 1–8.

[6] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, “Iotsense: Behavioral fingerprinting of iot devices,” *arXiv preprint arXiv:1804.03852*, 2018.

[7] 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.

[8] T. Kohno, A. Broido, and K. C. Claffy, “Remote physical device fingerprinting,” *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.

[9] S. V. Radhakrishnan, A. S. Uluagac, and R. Beyah, “Gtid: A technique for physical deviceanddevice type fingerprinting,” *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp. 519–532, 2014.

[10] K. Gao, C. Corbett, and R. Beyah, “A passive approach to wireless device fingerprinting,” in *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*. IEEE, 2010, pp. 383–392.

[11] M. T. Khan, D. Serpanos, and H. Shrobe, “A rigorous and efficient run-time security monitor for real-time critical embedded system applications,” in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016, pp. 100–105.

[12] E. Anthi, L. Williams, and P. Burnap, “Pulse: an adaptive intrusion detection for the internet of things,” in *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, 2018.

[13] L. D’hooge, T. Wauters, B. Volckaert, and F. De Turck, “In-depth comparative evaluation of supervised machine learning approaches for detection of cybersecurity threats,” in *Proceedings of the 4th International Conference on Internet of Things, Big Data and Security*, 2019.

[14] C. Nixon, M. Sedky, and M. Hassan, “Practical application of machine learning based online intrusion detection to internet of things networks,” in *2019 IEEE Global Conference on Internet of Things (GCIoT)*. IEEE, 2019, pp. 1–5.

[15] E. D. Alalade, “Intrusion detection system in smart home network using artificial immune system and extreme learning machine hybrid approach,” in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*. IEEE, 2020, pp. 1–2.

[16] M. Belouch and S. E. hadaj, “Comparison of ensemble learning methods applied to network intrusion detection,” in *Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing*, 2017, pp. 1–4.

[17] A. Saeed, A. Ahmadinia, A. Javed, and H. Larijani, “Intelligent intrusion detection in low-power iots,” *ACM Transactions on Internet Technology (TOIT)*, vol. 16, no. 4, pp. 1–25, 2016.

[18] 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*, vol. 18, no. 8, pp. 1745–1759, 2018.

[19] J. J. Davis and A. J. Clark, “Data preprocessing for anomaly based network intrusion detection: A review,” *computers & security*, vol. 30, no. 6-7, pp. 353–375, 2011.

[20] Ixiacom.com, “Ixia Breaking Point,” 2020. [Online]. Available: https://www.ixiacom.com/products/network-security-testing-breakingpoint

[21] C. Bormann, M. Ersue, and A. Keranen, “Terminology for constrained-node networks rfc 7228. retrieved august 12, 2016,” 2014.

[22] O. M. Alliance, “Lightweight machine to machine technical specification,” Alliance, Open Mobile, Tech. Rep. 1, 2017.

[23] Nmap.org, “Nmap Network Scanning - Well Known Port List: nmap-services,” 2020. [Online]. Available: https://nmap.org/book/nmap-services.html

[24] A. Rosenberg and J. Hirschberg, “V-measure: A conditional entropy-based external cluster evaluation measure,” in *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007, pp. 410–420.

[25] scikit learn.org, “sklearn.cluster.DBSCAN,” 2021. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN