Inferring IoT Device Types from Network Behavior Using Unsupervised Clustering

Arunan Sivanathan, Hassan Habibi Gharakheili, and Vijay Sivaraman
School of Electrical Engineering and Telecommunications,
University of New South Wales, Sydney, Australia
Emails: {a.sivanathan@unsw.edu.au, h.habibi@unsw.edu.au, vijay@unsw.edu.au}

***Abstract*—The Internet-of-Things (IoT) is increasingly becoming a major challenge for network administrators to monitor and manage connected devices and sensors, ranging from smart-lights to smoke-alarms and security-cameras. In addition to new device offerings, manufacturers tend to automatically perform firmware upgrade from their cloud servers to change functionalities of existing devices that are operational in the field. This makes it difficult to re-train device classification models in order to capture legitimate changes dynamically. In this paper, we develop a modular device classification architecture that allows us to dynamically accommodate legitimate changes in network IoT assets, either addition of a new device type or upgrades of existing types, without replacing the entire set of models. Our contributions are twofold: (1) We identify key traffic attributes that can be obtained from flow-level network telemetry to characterize individual IoT devices. We develop an unsupervised one-class clustering method for each device to detect its normal network behavior. (2) We tune individual device-specific clustering models and use them to classify IoT devices in real-time. We enhance our classification by developing methods for automatic conflict resolution and noise filtering. We evaluate the efficacy of our scheme by applying it to traffic traces of ten real IoT devices, and demonstrate its ability to achieve overall accuracy of more than 94%.**

I. Introduction

The Internet-of-Things (IoT) continues to expand its reach into homes, offices, enterprise campuses, and even cities, as more devices are rapidly connected to networks for collecting and sharing data. Network operators today are unable to identify all IoT devices on their networks, lack real-time visibility into the network behavior of known IoT assets [1], and are unsure whether connected devices behave legitimately or not. As a result, unmonitored IoT devices have already caused data breaches or been hijacked to carry out large-scale attacks on the Internet [2].

IoT devices are typically purpose-built with limited functionalities – they communicate with a specific set of endpoints (*i.e.,* servers) using a small number of TCP/UDP flows. Therefore, a growing number of traffic classification proposals are emerging based on supervised machine-learning techniques (*e.g.,* multi-class decision-trees or neural-networks) that use packet-level [3], flow-level [4], or a combination of packet-level and flow-level [5] traffic attributes for monitoring IoTs behavioral patterns on the network. In our prior work [5] we showed that generating the model for multi-class classifiers becomes practically challenging when a new device type is added to the network or the behavior of existing device types legitimately changes (due to firmware upgrades by device manufacturers) – it is needed to re-generate the entire model of all classes. In order to avoid over-fitting the generated model to specific classes, we need to carefully balance (*i.e.,* representing classes equally) the training dataset comprising instances of all device types. However, certain devices need much more instances to capture their normal behavior.

In this paper, we employ a set of one-class clustering models (one per IoT device), each can be independently trained and updated. Our first contribution identifies IoT traffic attributes that can be computed from real-time flow-level telemetry. We show how clusters of attributes can characterize network behavior of each device. Our second contribution develops a classification scheme using a set of device-specific clustering models augmented by refinement and filtering methods. We apply our classification solution to real traffic of ten IoT devices and demonstrate its accuracy of more than 94%.

II. Related Work

Automatic detection of IoT devices from the network traffic has been the subject of research [3]–[8] over the past few years. Authors of [6] simply use the set of IP addresses (of servers) that each device communicates with to identify IoT devices on the network. This method can not be very reliable since typically an elastic IPv4 address allocation is employed for dynamic cloud computing (*e.g.,* Amazon AWS). Work in [7] shows that traffic pattern of encrypted network flows, measured outside NAT, can reveal the IoT devices used inside a home network. However, authors do not develop an automatic method for discovery of IoT devices behind the NAT. Work in [3] develops a supervised machine learning model using over 300 attributes (packet-level and flow-level) of IoT traffic. Authors highlighted the most important attributes as packets Time-To-Live (minimum, median, and average), ratio of transmitted-bytes to received-bytes, total number packets with reset flag, and the Alexa rank of servers which the device communicates with. Work in [8] employs 16 binary attributes (indicating the use of various protocols at application, transport, network and link layers) along with remote IP address/port numbers, and size and raw byte value of packets from IoT traffic to train a supervised multi-class classifier. Some researchers [4] argue that traffic attributes need to be automatically learned (from raw sequence of packet payloads in TCP flows) instead of being hand-crafted. We believe that extraction of packet payloads makes it difficult for this method to scale.

III. Clustering Flow-Level Attributes of IoT Traffic

In this section, we first outline our IoT dataset, network telemetry, and traffic attributes. We, next, show how clusters of attributes will characterize individual IoT devices.

A. Flow-Level Telemetry and Traffic Attributes

Dataset: We use a publicly available dataset [2] that contains more than 6 weeks worth of packet traces. It consists of active and idle periods of 10 real IoT devices namely Amazon Echo, TPlink switch, Belkin motion sensor, Belkin switch, LiFX Bulb, Netatmo camera, Hue bulbs, iHome switch, Samsung Smart camera, and Google Chromecast. Note that the original dataset also contains some attack traffic (clearly annotated) which we removed for this study.

Flow-Level Telemetry and Attributes: We showed in our prior work [5] that individual IoT devices exhibit identifiable patterns in their traffic flows such as activity cycles and volume patterns, and DNS/NTP/SSDP signaling profiles. Inspired by recent proposals [9], [10] on network telemetry using SDN, we identify a consistent set of flow rules for each device that are inserted into the SDN-enabled switch (to which IoT devices are connected) for real-time monitoring.

Counters of these flow rules are periodically (*i.e.,* every minute) measured by the SDN controller that will form traffic attributes of each device. We use eight flow rules to measure network traffic of each IoT device with the following order: **(1,2)** DNS outgoing queries and incoming responses on UDP 53, **(3,4)** NTP outgoing queries and incoming responses on UDP 123, **(5)** SSDP outgoing queries on UDP 1900, **(6,7)** other "remote" traffic (*e.g.,* Internet) outgoing from and incoming to the device that passes through the gateway, and **(8)** all "local" traffic (*i.e.,* LAN) incoming to the device. Note that we do not monitor SSDP traffic incoming to IoT devices to avoid capturing (and mixing) the discovery activities of other devices on the local network. We have used MAC address as the identifier of a device – one may use IP address (*i.e.,* without NAT), physical port number, or VLAN for a one-to-one mapping of a physical device to its traffic trace.

We use two key attributes [11] namely ***average packet size*** and ***average rate*** for each of eight flows (mentioned above). We also note that traffic attributes can better characterize individual devices if they are computed at multiple time-scales. We, therefore, collect per-flow packet and byte counts every minute, and compute attributes at time-granularities of 1-, 2-, 4-, and 8-minutes. This way we generate eight attributes for each flow that means a total of 64 attributes per device.

Extracting Attributes: In order to synthesize the flow entries and thereby extract attributes from our traffic traces, we use our native packet-level parsing tool [11]. It takes raw PCAP files as input, develops a table of flows (like in an SDN switch) and exports byte/packet counters of each flow at a configurable resolution (*e.g.,* 60 sec). Lastly, we generate a stream of instances (*i.e.,* a vector of attributes periodically generated every one minute) corresponding to each device separately. We split our instances into 4 weeks for training and 2 weeks for testing. The second column in Table I summarizes the number of training /testing instances per each device.

TABLE I: Device instances and clustering parameters.

	Instance count		**Unsupervised classifier parameters**	
Device	Training	Testing	Principal components	No. of clusters
Amazon Echo	27,102	11,677	18	256
TPlink switch	38,210	21,176	10	128
Belkin motion	38,228	21,375	10	128
Belkin switch	21,037	12,613	14	256
LiFX	25,903	10,348	12	256
Netatmo cam	13,528	10,563	14	256
Hue bulb	17,329	10,409	16	256
iHome	37,865	19,924	12	128
Samsung cam	38,226	20,904	14	256
Chromecast	17,395	8,462	15	256

B. Attributes Clustering

Our primary objective is to train a number of models (one per IoT device) where each model recognizes traffic patterns of a particular device (*i.e.,* class) and rejects data from all other classes – *i.e.,* one-class classifier generates "positive" outputs for a known/normal instances, and "negative" otherwise. This approach enables us to re-train each model independently. Also, device-specialized models can better distinguish anomalies (outliers) [2] – anomaly detection is beyond the scope of this paper. There are a number of algorithms for one-class classification. The most popular and efficient method is K-means which finds groups (*i.e.,* "clusters") of instances for a given class that are similar to one another. Each cluster is identified by its centroid, and an instance is associated with a cluster if it is closer to the centroid of that cluster than any other centroids.

To provide insights into traffic characteristics of IoT devices, we show in Fig. 1 clusters of instances for Amazon Echo, Belkin switch, and Chromecast in our dataset. Note that our instances are multi-dimensional (*i.e.,* 64 attributes), and thus can not be easily visualized. We, therefore, employ Principal Component Analysis (PCA) to project data instances to two-dimensional space just for illustration purpose – data instances are shown as dots and cluster centroids are shown as crosses. Only 10% of all instances are shown in each cluster for better visualization. For example, four dots in cluster A1 (for Amazon Echo) approximately represent 40 instances. Dashed circles depict the boundary of clusters. We draw these circles to include 99% of data points in each cluster (*i.e.,* closer to the centroid), and thus exclude 1% of farther instances to avoid impurities in our training dataset – actual clusters are not in the form of circles. These boundaries will be used to determine if a test instance belongs to clusters of a class or not.

It is seen that instances of Amazon Echo, Belkin switch, and Chromecast are respectively grouped into 16, 4, and 8 clusters. We observe that instance clusters of Amazon Echo are fairly spread across the 2D space. For Belkin switch, clusters are mainly spread across the principal-component-1 while their principal-component-2 is limited between -20 and 20. Lastly, Chromecast instances are densely concentrated within limited

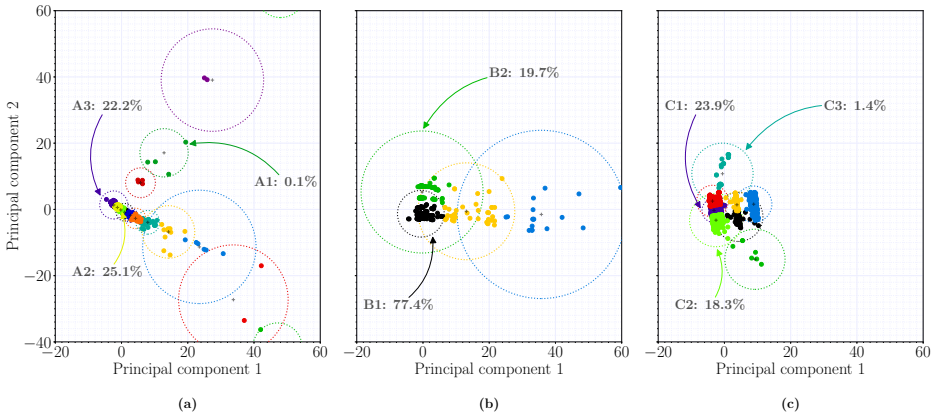

Fig. 1: Clusters of data instances in two-dimensional space for representative IoT devices: (a) Amazon Echo, (b) Belkin switch, and (c) Chromecast.

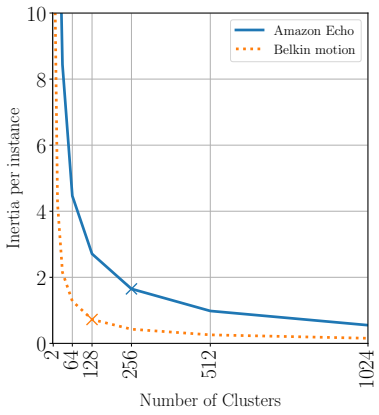

Fig. 2: Elbow method for selecting optimal number of clusters.

regions across both components. Note that each cluster (within a device class) has a probability of covering instances (of that device) depending upon device-specific traffic patterns seen in the dataset. For example, most probable clusters for Amazon Echo are A2 (25.1%), A3 (22.2%), for Belkin switch are B1 (77.4%), B2 (19.7%), and for Chromecast are C1 (23.9%), C2 (18.3%). These clusters indicate the dominant traffic characteristics of their respective devices.

IV. Unsupervised Classification of IoT Devices

In this section, we describe the architecture of our one-class classifier followed by mechanisms to resolve conflicts among multiple models and remove noises. We then evaluate the performance of our proposed scheme.

A. Clustering Models: Generation, Tuning, and Testing

Prior to generating clustering models we need to pre-process our raw dataset. First, since the scales of various attributes are widely different (*i.e.,* several orders of magnitude), we normalize each attribute independently to prevent the algorithm outweighing large-value attributes (*e.g.,* Mbps) over smaller attributes (*e.g.,* a few bps) [12]. We, therefore, scale individual attributes using Z-score method. Second, note that our data is 64-dimensional that can be computationally expensive for real-time prediction and also affect the performance of clustering. It is common to project data instances into a lower dimension space via PCA [13] which results linearly uncorrelated principal components. These orthogonal components enable K-means to detect clusters more clearly. We choose number of PCA components to retain optimum "cumulative variance".

Following dimension reduction, we apply K-means algorithm with varying K values (*i.e.,* 2^i where $i = 1, ..., 10$). We note that setting K to small values would not generate an accurate model of network behaviour for IoT devices, and large values increase the computational cost in both training and testing phases. Also, a very large K results in smaller-size clusters (*i.e.,* a rigid classifier) which cannot detect legitimate instances with small deviations from training data – *i.e.,* over-fitting. We find the optimal number of clusters using the elbow method [14]. Fig. 2 shows the average square distance of instances from the cluster centers (*i.e.,* Inertia per instance) versus clusters count, for two devices. We choose the optimal cluster number for each device when the derivative of Inertia per instance exceeds -0.01 (marked by '$\times$'). We observe that the model for Amazon Echo needs 256 clusters for optimal performance, and this measure is 128 clusters for Belkin motion. The last column of Table I shows the model parameters for individual devices obtained from the methods mentioned above.

Having clustering models generated, we can test an instance (of IoT device traffic attributes) against each model by finding the nearest cluster (in each model) to the test instance, as shown in Fig. 3. We now demonstrate a sample of test instance by considering the two-dimensional space of clusters in Fig. 1. Let's assume that the test instance has the principal component 1 and 2 respectively equals to 0 and 20. The nearest clusters to this test instance are A1 of Amazon Echo, B2 of Belkin switch, and C3 of Chromecast. Since the instance falls outside the A1 boundary, the Amazon Echo model results a negative output while the other two models each gives a positive output.

Conflict Resolution: Each model learns the behavior of one device and multiple devices may have similar traffic behavior (*e.g.,* DNS, NTP or SSDP) for a short period of time [4]. This leads to multiple positive outputs generated from device models for a test instance. We use the probability of clusters (within each model) to solve the conflict among multiple models which generate the positive output – higher probability indicates the "winner" model. For our test example, the test instance (0, 20) is classified as Belkin switch since the probability of B2 is 77.4% which is higher than the probability 1.4% of C3 from Chromecast.

Temporal Noise Filtering: We note that our real-time classification scheme may still mis-classify instances for one epoch time (*i.e.,* a minute). For example, a MAC address which is consistently classified as Amazon Echo for hours (or even days) may suddenly get classified as another device – this output should be treated as a noise unless it persists for successive epochs. We, therefore, filter these temporal noises by applying the majority vote selection over a moving window of last three decisions generated by our conflict resolver – one may want to increase this window size depending upon desired responsiveness to changes in traffic behavior.

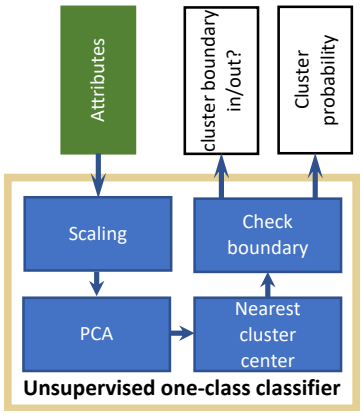

Fig. 3: Use of each clustering model for a test instance.

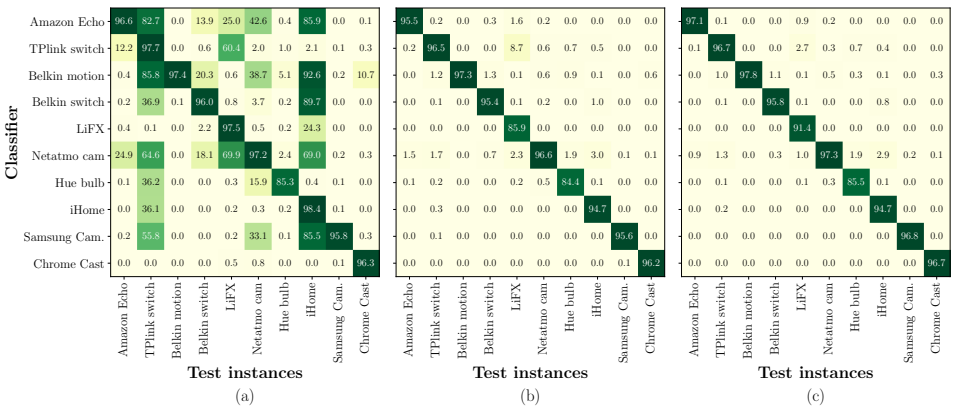

Fig. 4: Confusion matrix of device classification: (a) raw output of clustering models, (b) refined output after conflict resolution, (c) final output after temporal noise filtering.

B. Performance Evaluation

We now evaluate the performance of our device inference scheme for test instances of the dataset. Fig. 4 shows the confusion matrix of the classification for three stages. Each clustering model (listed by rows) is presented by test instances of all devices (listed by columns). Starting from raw outputs, we can see in Fig. 4(a) that 9 out of 10 models correctly detect 95% of instances from their own class, as shown by diagonal elements of the matrix. However, high rate of false positives are also observed in non-diagonal elements of Fig. 4(a). For example, more than 85% of iHome instances are incorrectly detected by the model of Amazon Echo. Considering the raw output of models, we observe that 43% of test instances are detected by more than one model (in addition to their respective device model), and 3% of test instances are completely missed by all models (*i.e.,* no positive output from any model).

Fig. 4(b) shows the confusion map after applying the conflict resolver to the output of the device-specific models. It clearly shows a significant enhancement in the performance of classification by selecting only one model with the highest cluster probability (for a given test instance). Note that the average false positive rate has reduced to less than 0.5%. Also, we observe that the conflict resolver slightly reduces true positives for almost all device models. The most impacted instances correspond to LiFX where true positives have dropped from 97.5% to 85.9% – the model of TPLink switch incorrectly detects 8.7% of LiFX instances.

Lastly, the best outcome is obtained in Fig. 4(c) when we apply the temporal noise filtering to the output of the conflict resolver. We observe that both true positives and false positives are improved. Specifically, LiFX instances now experience a palatable 91.4% true positive while false positive claims from the model of TPLink switch model is reduced to 2.7%. Overall, our device classification scheme gives an average true positive 94.5% and false positive 1%.

V. Conclusion

Real-time traffic monitoring is of paramount importance for network operators who manage a diverse set of IoT devices. In this paper, we have developed a modular classification to infer IoT devices from their network behavior using a set of clustering models. We have identified a set of network flows for IoT devices that result in attributes computed from real-time per-flow telemetry, and optimized and trained clustering models. Lastly, we augmented our machine learning-based scheme by conflict resolution and noise filtering methods, applied it to traffic traces of ten real IoT devices, and achieved more than 94% accuracy.

References

[1] Cisco, "Cisco 2017 Midyear Cybersecurity Report," Tech. Rep., 2017.

[2] A. Hamza *et al.*, "Detecting Volumetric Attacks on IoT Devices via SDN-Based Monitoring of MUD Activity," in *Proc. ACM SOSR*, San Jose, California, USA, Apr 2019.

[3] Y. Meidan *et al.*, "Detection of Unauthorized IoT Devices Using Machine Learning Techniques," *arXiv*, 2017. [Online]. Available: http://arxiv.org/abs/1709.04647

[4] J. Ortiz *et al.*, "DeviceMien: Network Device Behavior Modeling for Identifying Unknown IoT Devices," in *Proc. ACM IoTDI*, Montreal, Quebec, Canada, 2019.

[5] A. Sivanathan *et al.*, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," *IEEE Transactions on Mobile Computing*, 2018.

[6] H. Guo and J. Heidemann, "IP-Based IoT Device Detection," in *Proc. ACM IoT S$P*, Budapest, Hungary, August 2018.

[7] N. Apthorpe *et al.*, "A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic," in *Proc. DAT*, New York,USA, Nov 2017.

[8] M. Miettinen *et al.*, "IoT Sentinel Demo: Automated Device-Type Identification for Security Enforcement in IoT," Tech. Rep., 2017.

[9] A. Sivanathan *et al.*, "Low-Cost Flow-Based Security Solutions for Smart-Home IoT Devices," in *Proc. IEEE ANTS*, Banglore, India, Nov 2016.

[10] H. Habibi Gharakheili *et al.*, "iTeleScope: Intelligent Video Telemetry and Classification in Real-Time using Software Defined Networking," *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1804.09914

[11] A. Sivanathan *et al.*, "Characterizing and Classifying IoT Traffic in Smart Cities and Campuses," in *Proc. IEEE INFOCOM workshop on smart cities and urban computing*, Atlanta, USA, May 2017.

[12] I. Bin Mohamad and D. Usman, "Standardization and Its Effects on K-Means Clustering Algorithm," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 6, no. 17, pp. 3299–3303, Sep 2010.

[13] C. Ding *et al.*, "K-means Clustering via Principal Component Analysis," in *Proc. Int. Conf. Machine Learning*, Banff, Alberta, Canada, July 2004.

[14] D. Ketchen *et al.*, "The Application of Cluster Analysis in Strategic Management Research: An Analysis and Critique," *Strategic Management Journal*, vol. 17, no. 6, pp. 441–458, 1996.