Managing IoT Cyber-Security Using Programmable Telemetry and Machine Learning

Arunan Sivanathan, Hassan Habibi Gharakheili, and Vijay Sivaraman

***Abstract*—Cyber-security risks for Internet of Things (IoT) devices sourced from a diversity of vendors and deployed in large numbers, are growing rapidly. Therefore, management of these devices is becoming increasingly important to network operators. Existing network monitoring technologies perform traffic analysis using specialized acceleration on network switches, or full inspection of packets in software, which can be complex, expensive, inflexible, and unscalable. In this paper, we use SDN paradigm combined with machine learning to leverage the benefits of programmable flow-based telemetry with flexible data-driven models to manage IoT devices based on their network activity. Our contributions are three-fold: (1) We analyze traffic traces of 17 real consumer IoT devices collected in our lab over a six-month period and identify a set of traffic flows (per-device) whose time-series attributes computed at multiple timescales (from a minute to an hour) characterize the network behavior of various IoT device types, and their operating states (*i.e.,* booting, actively interacted with user, or being idle); (2) We develop a multi-stage architecture of inference models that use flow-level attributes to automatically distinguish IoT devices from non-IoTs, classify individual types of IoT devices, and identify their states during normal operations. We train our models and validate their efficacy using real traffic traces; and (3) We quantify the trade-off between performance and cost of our solution, and demonstrate how our monitoring scheme can be used in operation for detecting behavioral changes (firmware upgrade or cyber attacks).**

***Index Terms*—IoT, device monitoring, flow characteristics, machine learning.**

I. Introduction

THE INTERNET-OF-THINGS (IoT) such as security cameras, smart-lights, smoke-alarms, and smart-bins, continues its reach to smart environments including homes, building, enterprise campuses and even cities [2]. With 10 billion IoT devices connected today, the installed base is expected to reach 22 billion in next five years [3]. However, research studies [4], [5] have shown that this revolutionary network technology comes with a glaring problem of cyber-security which cannot be overlooked. Malware and botnets, such as Mirai, Persirai, Reaper, and IoTroop [6]–[8], have been able to adversely affect these devices and their networks, enabling destructive cyber-campaigns.

The lack of effective security on IoTs [9]–[11] presents a number of challenges for network operators of large organizations who are looking to bring these devices online at scale. Even governments recently started to develop cyber-security guidelines and regulations [12]–[15] for manufacturers of IoT devices, preventing unauthorized access, modification or information disclosure. Implementing device-level security would definitely help protect against automated attacks [16], but its efficacy can vary across manufacturers and device types depending upon devices capabilities and their mode of operation [17]. In a parallel effort, IETF has approved an Internet standard called "Manufacturer Usage Description" (MUD) [18] to protect IoT devices. This framework allows manufacturers to formally specify the intended behavior of their devices that can be used to generate and enforce access control lists (ACLs) [19] for IoT devices, limiting their network behavior to only a tight set of services. Although MUD policies can reduce the surface of attacks on IoTs they are still insufficient, since ACL rules do not restrict temporal variation of traffic flows (*e.g.,* traffic with unwanted volume or pattern cannot be prevented if endpoints and protocols conform to MUD rules).

Therefore, it is crucial for organizations to maximize visibility into their IoT infrastructure [20], and thus better manage security risks of these vulnerable devices [6]. Network administrators need to know all connected devices and their expected operations on the network, and continuously monitor their activities ensuring IoTs behave "normally" [6]. Existing traffic monitoring solutions are either purely software-based (hence unscalable to high traffic rates), or customized hardware-based (hence inflexible and expensive) [21]. Network operators, today, widely use NetFlow [22] (an embedded switch instrumentation) to obtain aggregate measurement of traffic flows. However, it comes at cost of CPU resources on the switch [23] for generating, collating, and exporting flow records. To reduce this overhead, operators statistically mirror packet samples (*e.g.,* sFlow [24]) to a remote collector for extracting flow information that inevitably leads to reduced accuracy. On the other hand, special-purpose hardware appliances (*i.e.,* deep packet inspection engines) offer both accuracy and performance in traffic monitoring; but are prohibitively expensive for many network operators.

In this paper, we aim to monitor behavior of IoT devices on the network using a combination of Software Defined Networking (SDN) telemetry and machine learning methods.

Manuscript received June 14, 2019; revised October 26, 2019 and January 21, 2020; accepted January 23, 2020. Date of publication February 4, 2020; date of current version March 11, 2020. This submission is an extended and improved version of our paper presented at the INFOCOM 2017 SmartCity Workshop [1]. The associate editor coordinating the review of this article and approving it for publication was R. Badonnel. *(Corresponding author: Arunan Sivanathan.)*

The authors are with the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW 2052, Australia (e-mail: a.sivanathan@unsw.edu.au; h.habibi@unsw.edu.au; vijay@unsw.edu.au).

Digital Object Identifier 10.1109/TNSM.2020.2971213

We believe that the SDN paradigm by its nature provides flow-level isolation and visibility in a low-cost and scalable manner. For accurate detection of devices and tracking of their dynamic behaviors, we employ machine learning algorithms to learn key patterns of traffic flows. Our **first** contribution is to identify a set of TCP and UDP flows (for each IoT device) and highlight characteristics attributes, computed from time-series of flows at multiple time-scales, distinguishing various IoT device types and their operating states (booting, active, or idle) on the network. Our **second** contribution develops a multi-stage architecture consisting of a set of inferencing models that use flow-level attributes to automatically recognize traffic of IoT devices from non-IoTs, classify types of IoT devices, and identify operating states of each IoT during normal operation. We train our models and validate their performance to obtain high accuracy using real traffic traces. **Finally**, we demonstrate the efficacy of our scheme in detecting network behavioral changes due to firmware upgrade or cyber-attacks. Also, we quantify the trade-off between performance and cost of our monitoring solution for real-time deployment. Our solution builds upon our preliminary work [1] by identifying cost-effective attributes, and enhancing the architecture of inferencing that can detect changes in IoT devices. We believe that our real-time monitoring solution empowers network operators to better manage cyber-security risks of their IoT infrastructure.

The rest of this paper is organized as follows: Section II describes relevant prior work. In Section III we present our dataset and traffic flows, and characterize attributes of various IoT devices and their operating states. We propose the architecture of IoT traffic inference and evaluate its performance in Section IV, followed by a discussion on the operational trade-off and use of the proposed system in Section V. The paper is concluded in Section VI.

II. Related Work

Network Telemetry: Network traffic measurement has been a subject of interest to academia and industry. Many different methods have been proposed and practically used ranging from traditional port-based counting using SNMP [25] and packet sampling [24] to flow-based telemetry [21], [22] and WiFi packet sniffing [26], [27].

Modern telemetry methods can be categorized into (a) packet-based [24], [28], [29], and (b) flow-based [21], [22], [30]. sFlow [24] is one of commonly used methods that randomly samples (*i.e.,* one in N) packets from the network switches. Due to its random sampling, sFlow tends to collect packets from elephant flows (those that carry heavy traffic and are long in duration), and hence mice flows are likely to get missed which results inaccurate measurement. To address this issue, Everflow [28] proposes to collect specific packets (*e.g.,* TCP SYN, FIN, and RST) using the match and mirror functionality of data-center switches. Planck [29] estimates the throughput of flows at very tight time-scales by mirroring traffic of multiple ports to a monitoring port at which a collector performs high-rate sampling. Overall, packet-level telemetry can only provide partial visibility into network traffic flows.

Commercial switches equipped with NetFlow [22] engines export flow records (IPFIX). Netflow capable switches have the ability to export IPFIX records containing a rich set of information [20] (*e.g.,* port number, DNS, cipher suites) from the network traffic. However, they come with two major limitations: (a) they only export a flow record once it expires (not real-time), and (b) computational cost is high for updating and maintaining flow records inside the switch [31]. FlowRadar [21] overcomes the limitations of Netflow by incorporating an encoded hash table (data structure for flow counters) with low memory overheads and exporting flows periodically (*e.g.,* 10 ms). However, FlowRadar is still not supported by commercial switches available on the market. In this work, we use flow-level telemetry provided by SDN APIs [30] which enables us to measure traffic flows at low-cost with reasonable resolutions.

Traffic Classification: Traffic classification is widely used for various applications such as network management [32], QoS [33], and cyber-security [34]. Over the past few years, IoT traffic classification has attracted attention of researchers [35] to identify IoT devices, their states, and detect their abnormal behavior. Additionally, some works [36], [37] attempted to infer user activities from the network traffic of IoT devices.

Work in [38] attempts to correlate network activities of Nest Thermostat and Nest smoke-sensor to the user activity by looking at the distribution of payload size for various network flows. Similarly, authors of [39] show that the existence of certain IoT devices (behind NAT) can be identified by rate of traffic going to certain endpoints on the Internet. However, these works do not automatically detect or classify IoT devices.

Work in [40] 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 [41] 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. Although these classifiers show a good performance in the device classification, the cost of attribute extraction is high since they involves packet inspections. Lastly, work in [42] proposes a framework to group devices based on their semantic type (*e.g.,* camera, fitness/medical device, environmental sensor). Grouping devices potentially results a model with broad boundaries since often various devices of a given type (*e.g.,* cameras from different manufacturers) distinctly differ in their network behavior. Therefore, broad models would yield high rate of false classification during testing phase. In this paper, instead, we tighten our models, and hence increase the accuracy of classification and sensitivity of models to behavioral changes, by choosing each class mapped to one specific IoT device (*e.g.,* camera of a specific manufacturer).

In the context of cyber-security, work in [43] claims that machines can be trained to detect anomalies in IoT traffic

TABLE I
FLOW RULES SPECIFIC TO EACH DEVICE, PROACTIVELY INSERTED INTO SDN SWITCH FOR REAL-TIME TELEMETRY

Flow description	srcETH	dstETH	srcIP	dstIP	Protocol	srcPort	dstPort	Priority	Action
DNS query (DNS↑)	<devMAC>	*	*	*	17	*	53	100	forward
DNS response (DNS↓)	*	<devMAC>	*	*	17	53	*	100	forward
NTP query (NTP↑)	<devMAC>	*	*	*	17	*	123	100	forward
NTP response (NTP↓)	*	<devMAC>	*	*	17	123	*	100	forward
SSDP query (SSDP↑)	<devMAC>	*	*	*	17	*	1900	100	forward
outgoing remote (Rem.↑)	<devMAC>	<gwMAC>	*	*	*	*	*	10	forward
incoming remote (Rem.↓)	<gwMAC>	<devMAC>	*	*	*	*	*	10	forward
incoming local (Loc.↓)	<devMAC>	*	*	*	*	*	*	1	forward

generated by DDoS attacks using attributes such as packet size, inter-packet interval, average bandwidth and count of distinct IP addresses observed during a short duration (*i.e.,* 10-seconds). Work in [44] develops a machine to detect volumetric attacks by monitoring flow rules obtained from MUD profile of IoT devices. Anomaly (and attack) detection is beyond the scope of this paper, but we use attack traffic generated by authors of [44] to demonstrate how our traffic monitoring engines enable network operators for further investigation (manually or by other systems).

III. TRAFFIC FLOWS AND ATTRIBUTES

In this section, we begin by analyzing real traffic traces collected in our lab. We then identify traffic attributes to distinguish IoT devices from non-IoTs, classify individual IoTs, and determine their operating states.

A. Traffic Trace Dataset

We used two sets of full PCAP traffic traces collected from our testbed. The first dataset (*i.e.,* DATA1) was collected from a network consisting of more than thirty IoT and non-IoT devices for a duration of 6 months (*i.e.,* 01-Oct-2016 to 31-Mar-2017) [20]. We select 17 IoT devices, those whose trace was present for at least 60 days in packet traces. These devices include Amazon Echo, August doorbell, Awair air quality, Belkin motion sensor, Belkin switch, Dropcam, HP printer, LiFX bulb, NEST smoke sensor, Netatmo weather, Netatmo camera, Hue bulb, Samsung smart camera, Smart Things, Triby speaker, Withings sleep sensor, and Withings scale. Note that our dataset contains traffic traces of six non-IoT devices including Android phone, Android tablet, Windows laptop, MacBook, and two iPhones. Our dataset constitutes: (a) traffic generated by the devices autonomously (*e.g.,* periodic NTP), and also (b) traffic generated due to users interacting with the devices (*e.g.,* Belkin motion sensor responding to detection of movement, Amazon Echo responding to voice commands issued by a user, LiFX lightbulb changing color and intensity upon user request, and so on.

The second dataset (*i.e.,* DATA2) consists of traces with state annotation for selected IoT devices including Amazon Echo, Belkin switch, Dropcam, and LiFX bulb. We developed a software tool to automatically interact with these four devices over two days and annotate their traffic traces. Annotations indicate three operating states of IoT devices, namely "*boot*" (*i.e.,* getting connected to the network), "*active*" (*i.e.,* interacting with users), and "*idle*" (*i.e.,* not being booted or actively used). For the boot state, we used a TP-Link HS110 smart plug (whose traffic is not considered in our analysis) supplying power to these four devices. We wrote a script to automatically turn off/on this smart plug resulting a boot state for the subjected IoT device. For the active state, we used an app called "**RepetiTouch Pro**" and a text-to-speech engine called **espeak** [45]. The former records and replays interactions of a real user with three IoT devices including Belkin switch, Google Dropcam camera, and LiFX lightbulb via their manufacturer app – the user interactions (*i.e.,* turning on/off the switch, streaming video from the camera, and turning on/off the bulb) were recorded on an Android tablet which was connected to the local network of our testbed. The latter periodically asks scripted questions (*e.g.,* "How is the weather in Sydney Australia") from Amazon Echo. For the idle state, we used all traffic traces that were annotated as neither boot nor active, during data collection period.

B. Traffic Flows and Attributes

We showed in our prior work [1] that individual IoT devices exhibit identifiable patterns in their traffic flows such as DNS/NTP/SSDP signaling profiles, activity cycles, and volume patterns. Inspired by recent proposals [46], [47] on network telemetry using SDN, we consider a set of flow rules that collectively characterize traffic signature of IoT devices. For each device, these flow rules are pro-actively inserted into SDN-enabled switch(es) to which IoT devices are connected, as shown in Fig. 1. We use MAC address as the identifier of a device – one may use IP address (without NAT), physical port number, or VLAN for a one-to-one mapping of a physical device to its traffic trace. For real-time monitoring, in-built counters of these flow rules are periodically (*i.e.,* every minute) measured via the SDN controller that will form traffic attributes of each device. Note that flow counters provided by real SDN switches are highly accurate [47], though it may vary across different vendors [48] due to customized implementations.

Table I shows eight flow rules which we use 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" (*e.g.,* Internet) traffic outgoing from and incoming to the device that passes through the gateway, and **(8)** all "local" (*i.e.,* LAN) traffic incoming to the device. Note that

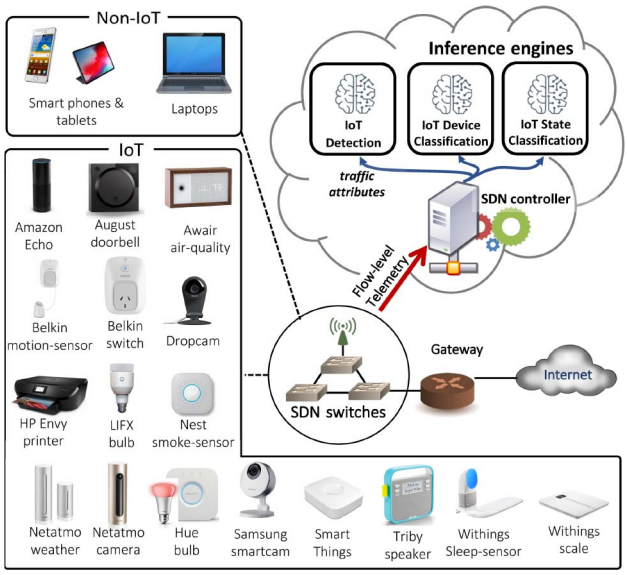


Fig. 1. System architecture of network telemetry and inference engines.

we do not measure incoming SSDP traffic to IoT devices in order to avoid capturing (and mixing with) discovery activities of other devices on the local network. Note that rules priority (the second last column in Table I) are used to split the traffic of each device into three levels: signaling packets (*i.e.,* priority 100), other remote packets (*i.e.,* priority 10), and local packets (*i.e.,* priority 1). We note that SDN-enabled switches that are currently available in the market typically support a large number of flow rules without experiencing performance degradation. For example, a NoviSwitch[1] provides a massive table with up to 1 million flow rules in TCAM for wildcard matches while offering up to 400 Gbps throughput. This means that with insertion of 8 OpenFlow rules, one switch can essentially manage monitoring of more than 100K IoT devices.

For each of eight flows (mentioned above), we use two key attributes [1] namely ***average packet size*** and ***average rate***. Also, note that traffic attributes can better characterize network behavior of individual devices if they are computed at multiple time-scales [49]. We, therefore, collect packet counts and byte counts per each flow every minute, and compute attributes at time-granularities of 1-, 2-, 4-, 8-, 16-, 32-, 64-minutes. This way, we generate fourteen attributes for each flow that means a total of 112 attributes per device.

In order to synthesize flow rules, we wrote a native SDN software switch emulator [50] that replays the network traffic from PCAP traces, and performs packet-by-packet service (matching packet headers against flow table entries, updating statistics, applying required actions) inside a software SDN switch – prototype implementation on a physical SDN switch is beyond the scope of this paper. The emulator records counters of flow bytes and packets periodically (*e.g.,* one minute). We use another script to generate instances of traffic attributes for each device every minute from the counter outputs. An instance is a vector of 112 attributes with a label (*e.g.,* Amazon Echo:boot).

[1]https://noviflow.com/noviswitch/

C. Traffic Characteristics of IoT Devices

We now highlight traffic characteristics of individual IoT devices that can be learned to distinguish them from non-IoTs, classify their device type, and identify their operating states.

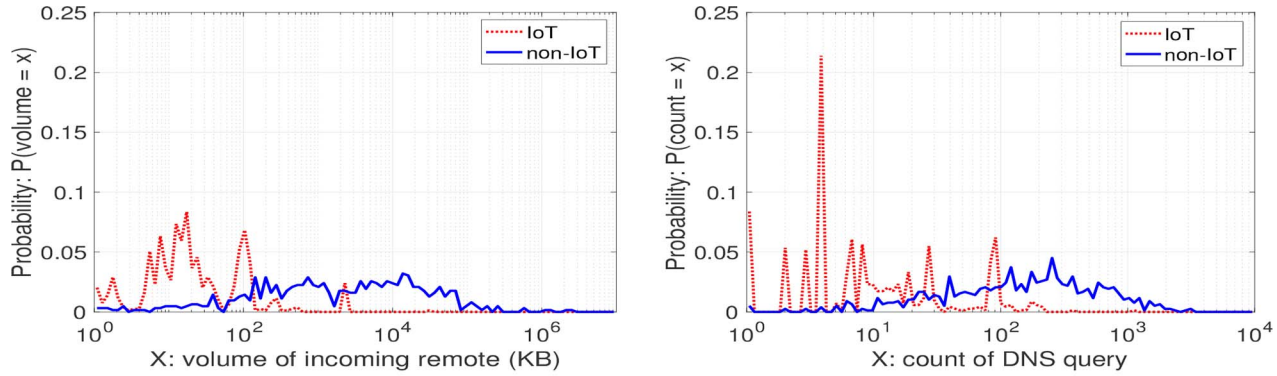
IoT versus non-IoT: We begin with traffic attributes that differentiate IoT devices from non-IoTs. Fig. 2 shows the probability density of two representative attributes, namely remote traffic volume at 32-minute resolution, and DNS query count at 64-minute resolution. We can see in Fig. 2(a) that IoT devices tend to transfer small volume of traffic from remote (*i.e.,* Internet) network and 90% of instances they download less than 500 KB every half-an-hour. However, for non-IoTs this value is widely spread between 10 KB to 100 MB and mostly they transfer more than 500 KB. In terms of DNS activity in Fig. 2(b), IoTs display identifiable patterns of query count mostly less than 100 (*e.g.,* 22% of instances with 4 queries per hour), while non-IoTs have a wider range of DNS query count (*i.e.,* 10 to 3000 DNS queries over an hour) with almost equal probabilities.

IoT device types: Focusing on IoT devices, we now consider three traffic attributes, namely NTP responses count at 16-min resolution, upload volume of remote traffic at 8-min resolution, and volume of SSDP responses at 8-min resolution, as shown in Fig. 3. We quantitatively compare traffic characteristics of four representative IoT devices from three different manufacturers (*i.e.,* Amazon, Belkin, and LiFX). It is observed from Fig. 3(a) that LiFX bulb (depicted by solid green lines) sends three NTP responses every 16-minute interval for more than 90% of instances. This measure varies between 7 to 42 responses for Amazon Eco (depicted by dashed red lines). For Belkin power switch and motion sensor (depicted by solid blue and dotted pink lines), we see two significant peaks at count of 1 and 2 NTP responses, each with a different probability – Belkin switch seems more active (compared to Belkin motion), with 70% probability of generating 2 NTP responses at 16-minute resolution.

For download volume of remote traffic attribute at 8-minute resolution, shown in Fig. 3(b), we see a relatively unique pattern in the probability density function for each of these four devices: for Belkin switch and Belkin motion it peaks at 573 bytes and 3KB, respectively, while LiFX bulb and Amazon Echo each exhibits a range of values, [0.5, 3] KB and [7, 33] KB, respectively.

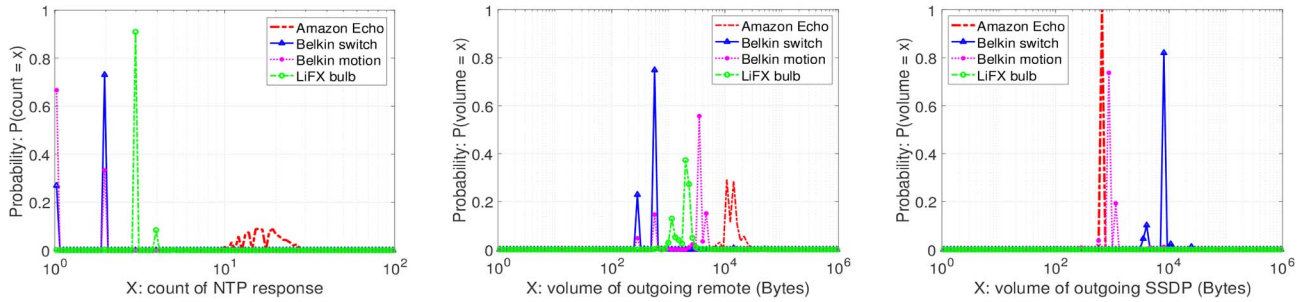
Considering upload volume of SSDP traffic in Fig. 3(c), Belkin switch seems distinctive from Belkin motion (*i.e.,* probability of 82% for 8 KB volume in Belkin switch compared to 73% chance for 800 bytes volume in Belkin motion). Amazon Echo displays a strong pattern with peak of 100% at volume of 650 bytes. Lastly, we observe that LiFX does not use SSDP protocol at all, and thus lacks this attribute in its traffic profile.

Operating states of IoT: We now look at selected traffic attributes of Amazon Echo, Belkin switch, and Dropcam at the three operating states, shown in Fig. 4. We focus on: download volume of remote traffic for Amazon Echo since



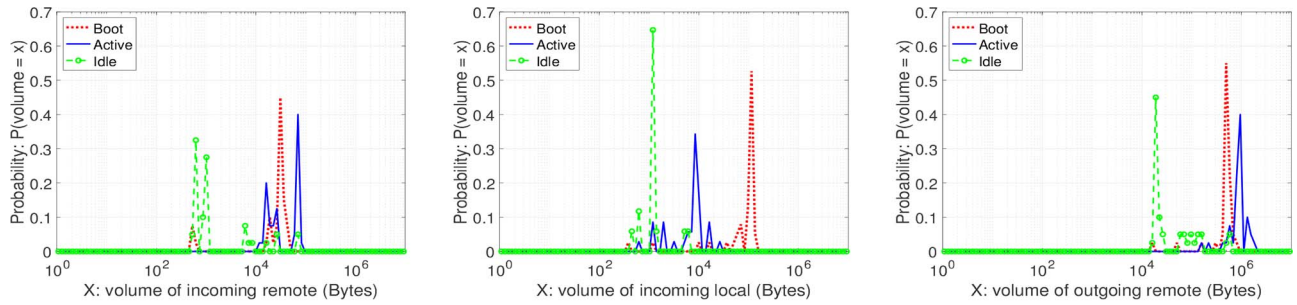
(a) Volume of remote traffic at 32-mins resolution. (b) Count of DNS query packets at 64-mins resolution.

Fig. 2. Histogram of traffic profile to compare IoT and non-IoT devices: (a) remote traffic volume over 32-minute, and (b) DNS query count over 64-minute.



(a) Count of NTP response packets (16-min). (b) Volume of remote traffic (8-min). (c) Volume of SSDP traffic (8-min).

Fig. 3. Histogram of traffic profile for representative IoT devices: (a) NTP response count, (b) download volume of remote traffic, and (c) upload volume of SSDP traffic.



(a) Amazon Echo: remote traffic (2-min). (b) Belkin switch: local traffic (1-min). (c) Dropcam: remote traffic (2-min)

Fig. 4. Histogram of traffic profile for IoT devices at three operating states: (a) Amazon Echo, (b) Belkin switch, and (c) Dropcam.

it frequently communicates with its cloud servers; download volume of local traffic for Belkin switch since it receives command from user mobile app connected to the local network; and upload volume of remote traffic for Dropcam since it tends to send videos to its cloud servers. We can see that the three operating states are fairly distinct in chosen attributes shown in Fig. 4. It is observed that all three devices exchange smaller volume of traffic during their idle state (shown by dashed green lines) compared to active and boot states. As an example,for Amazon Echo, shown in Fig. 4(a), 75% of idle instances receive between [0.5, 1] KB from remote servers at 2-minute resolution, while the probability density function for boot and active instances peaks at 30 KB and 70 KB, respectively. Additionally, we observe for Belkin switch that the volume of local traffic during boot state is larger than active state. This is because this device sends SSDP discovery when it boots up that results in arrival of responses from all SSDP-capable devices on the network. Therefore, a peak at 110 KB is seen for boot state (dotted red line) in Fig. 4(b).

IV. IoT Traffic Inference Engines

In this section, we develop a multi-stage architecture to automatically inferring IoT traffic, train a set of models, and evaluate their performance.

A. Inference Architecture

For a given device on the network, we have three objectives: (a) to determine if the device is IoT or non-IoT, and if it is detected as IoT: (b) to classify its device type (*e.g.*, Amazon Echo, Dropcam) and (c) to identify the operating

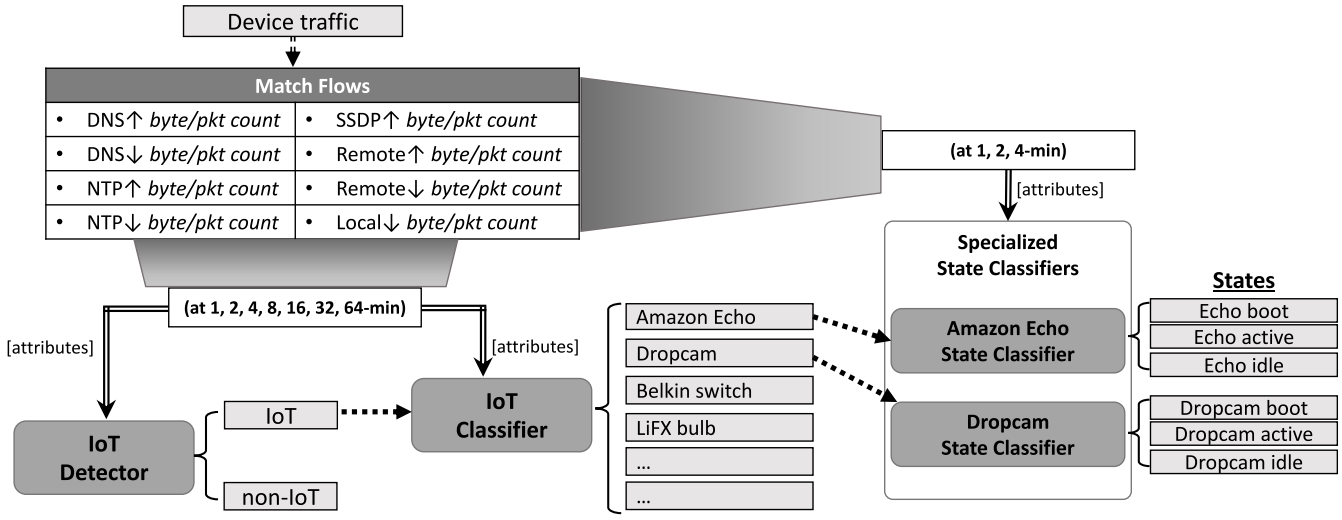


Fig. 5. Hierarchical architecture of IoT traffic inference engines.

state of IoT (*i.e.,* boot, active, idle). To meet these objectives we need a set of trained models: a bi-class classifier to distinguish IoT devices from non-IoTs (*i.e.,* IoT detector), a multi-class classifier to determine the type of a given IoT device (*i.e.,* IoT classifier), and a set of multi-class classifiers to identify IoT operating states (*i.e.,* a state classifier for each device type). Note that state classifiers are specialized models, each learns traffic patterns of one device in the three states of operation. State classifiers tend to have narrower views, and hence become more sensitive to change of behavior for their respective devices (compared to the device classifier with a broader view). It is important to note that minor traffic variations in different operating states can only be well learned with device-specialist models training a model of states classifier with instances from various devices will lead to an inaccurate prediction. Therefore, these specialized models are able to enhance the visibility of network operators into subtle changes [51] in their IoT infrastructure.

There exist a number of techniques [52] such as Neural Networks, Support Vector Machines (SVMs), and Decision Trees that can be used to train models to infer predefined classes. Neural networks have proven to be very effective in classifying input data with high dimensions, but they demand a large amount of training data. Also, neural networks are seen as black-box models since it becomes difficult to interpret their reasoning process. Performance of SVMs is very sensitive to selection of hyper-parameters, and hence it becomes difficult to train an accurate model. On the other hand, decision tree-based techniques are widely used since it is easier to generate (reasonably) accurate models with relatively small amount of data. Importantly they generate trees which can be readily interpreted. Note that decision tree algorithms are prone to over-fitting which can be avoided by use of ensemble decision trees. In this work, we employ Random Forest [53] which builds an ensemble of decision trees, each uses a random subset of attributes. It is best known for its performance in various classification tasks [20], [47], [54].

Fig. 5 illustrates our hierarchical architecture for IoT traffic inference. It consists of three layers of random-forest classifiers (*i.e.,* an IoT detector, an IoT classifier, and a set of IoT state classifiers). Once a new device connects to the network, the programmable switch is pushed by additional flow rules (Table I) pertinent to the device. We first feed the IoT detector model by full set of periodic flow-level attributes (*i.e.,* at time-scales of powers of 2 between 1-min to 64-min). It is important to note that our inference engine does not rely on a single output of the IoT detector for instances of a given device. Instead, it develops its trust to the detector (the model in the first layer) by receiving consistent outputs for a sequence of instances (at least 30 instances in Section IV-B). Upon confirmation of an IoT device with sufficiently enough monitoring time, the second model (*i.e.,* IoT classifier) is called by the full set of attributes – a device will not be checked by the second layer of inference, if it is confirmed as non-IoT at the first layer. The confirmation of the IoT classifier output triggers a pertinent state classifier at the third layer of our architecture. Our state classifier models consume a subset of attributes, only up to 4-minute resolution. This is because that change of states (*e.g.,* boot) results in short-term effect on device traffic pattern–considering long-term attributes may reduce the ability of the model to accurately detecting the operating state in real-time.

B. Models Training and Performance Evaluation

We now label instances to train our classifiers, and generate models needed for the three layers of the inference architecture, shown in Fig. 5. We next evaluate their performance using test instances. For both training and testing the traffic classifiers we use Weka [55] tool.

Instances: Recall from Section III-B that our instances are computed every minute. Since two of our models consume full-set attributes (*i.e.,* 1-min to 64-min) we downsample our instances by the factor of 15 to avoid over-fitting for the IoT detector and the IoT classifier – it is likely to have heavily-correlated instances generated within 15 minutes. Note that

TABLE II
PERFORMANCE METRICS OF THE IOT DETECTOR MODEL

IoT/non-IoT	TP	FN	FP	Precision	Recall	F_1	TP avg. confidence	FN avg. confidence
IoT	0.987	0.013	0.022	0.979	0.987	0.983	0.968	0.635
non-IoT	0.978	0.022	0.013	0.987	0.978	0.983	0.947	0.701

the risk of over-fitting is less for the state classifiers given that they only use short timescale attributes.

We have collected a total of 115,237 instances of IoT and non-IoT devices from our DATA1 (in Section III-A) and 10,423 instances of four IoT devices (*i.e.,* Amazon Echo, Belkin switch, Dropcam, and LiFX bulb) with state annotation from our DATA2 (in Section III-A). We have different number of instances across various devices in our dataset, depending upon their presence and activity on the testbed, and their interactions with the lab users. Among all devices, NEST smoke-sensor has the lowest number (*i.e.,* 865) of instances since it communicates once a day for a short period of time. The highest count of belongs to Dropccam with 11,873 instances as it was online more than 90% of days during the 6-month period of packet capture and it frequently communicates with its cloud-servers whenever it is on the network.

Metrics: Since classes are not evenly distributed in our datasets, we use three metrics including weighted "*precision*", "*recall*", and "F_1 *score*" along with confusion matrix to evaluate the performance of each model. These metrics are defined as follows:

$$precision = \frac{TP}{TP + FP} \tag{1}$$

$$recall = \frac{TP}{TP + FN} \tag{2}$$

$$F_1 = \frac{2 \times precision \times recall}{precision + recall} \tag{3}$$

where *TP* is the rate of true positive, *FP* is the rate of false positive, and *FN* is the rate of false negative. Note that F_1 conveys the balance between precision and recall values and is computed by the harmonic mean of these two values in Eq. (3). All metrics take a value between 0 and 1. For each class in our multi-class models, we obtain *FP* and *FN* by summation across all "incorrect" labels (*i.e.,* false alarms).

In addition to correctness of classification, we record confidence-level of our random-forest models for all instances, correctly classified and incorrectly classified ones. Ideally, we expect our models to display high confidence (*i.e.,* close to 1) when they predict a correct class for an input instance, and low confidence (*i.e.,* close to 0) when they predict an incorrect class. Lack of confidence indicates that the tested instance contains attributes different from those that were learned before (*i.e.,* new or unseen pattern). We next look at individual models at various layers of the inference architecture

IoT Detector: In our DATA1, there exist 1212 instances labeled as non-IoT and 114,025 instances labeled as 17 types of IoT. For training set, we randomly choose 800 instances (*i.e.,* 66%) from non-IoT and 50 instances from each class

Fig. 6. Confusion matrix of the IoT detector.

of IoT device (*i.e.,* a total of 850). Remaining instances in DATA1 are used to test this bi-class classifier.

Fig. 6 shows the confusion matrix of the IoT detector model. The rows show actual labels (*i.e.,* IoT or non-IoT), columns show predicted labels, and cell numbers are in percentage rounded to single decimal point. Table II shows all performance metrics of this model. It is seen that 98.7% of IoT test instances and 97.8% of non-IoT test instances are correctly classified, as shown by diagonal values of the confusion matrix in Fig. 6. Looking at the last two columns of Table II, the confidence-level of this model is fairly high on average (*i.e.,* 0.968 and 0.947) for correct classification and is relatively low on average (*i.e.,* 0.635 and 0.701) for incorrect classification. Also, the three performance metrics namely precision, recall, and F_1 all indicate reasonable high values on average as 0.983, 0.982, and 0.983 respectively. To check consistency of the IoT detector model (mentioned in Section IV-A), we plot in Fig. 9 the CCDF of duration of continuously mis-detecting IoTs from non-IoTs. It can be seen that continuous mis-detection for more than 15 minutes is extremely rare (with probability of 0.1%). Therefore, we only conclude from the IoT detector if it gives consistent output for at least 30 successive instances.

IoT Classifier: For this model, we split IoT instances of the DATA1 into chronological sets of training and testing, given sufficient number of instances available in our dataset over six-month period. This way the performance of the model is evaluated over time. We, therefore, use instances collected during the first 3 months (*i.e.,* 01-Oct-2016 to 31-Dec-2016) for training and the remaining instances (*i.e.,* collected between 01-Jan-2017 and 31-Mar-2017) for testing.

Fig. 7 depicts the confusion matrix of the IoT classifier. We observe that the model performs well in predicting most of classes. For example, the correct prediction rate for Amazon Echo, August doorbell, Belkin switch, or Dropcamp is more than 97%. However, the model performance does not seem acceptable for certain classes. For example, it is seen that 12.0% of NEST smoke-sensor instances are misclassified as Withings scale and 39.8% of HP printer instances are misclassified as Belkin switch. Additionally, for Awair air-quality sensor only 77.1% of test instances are correctly classified

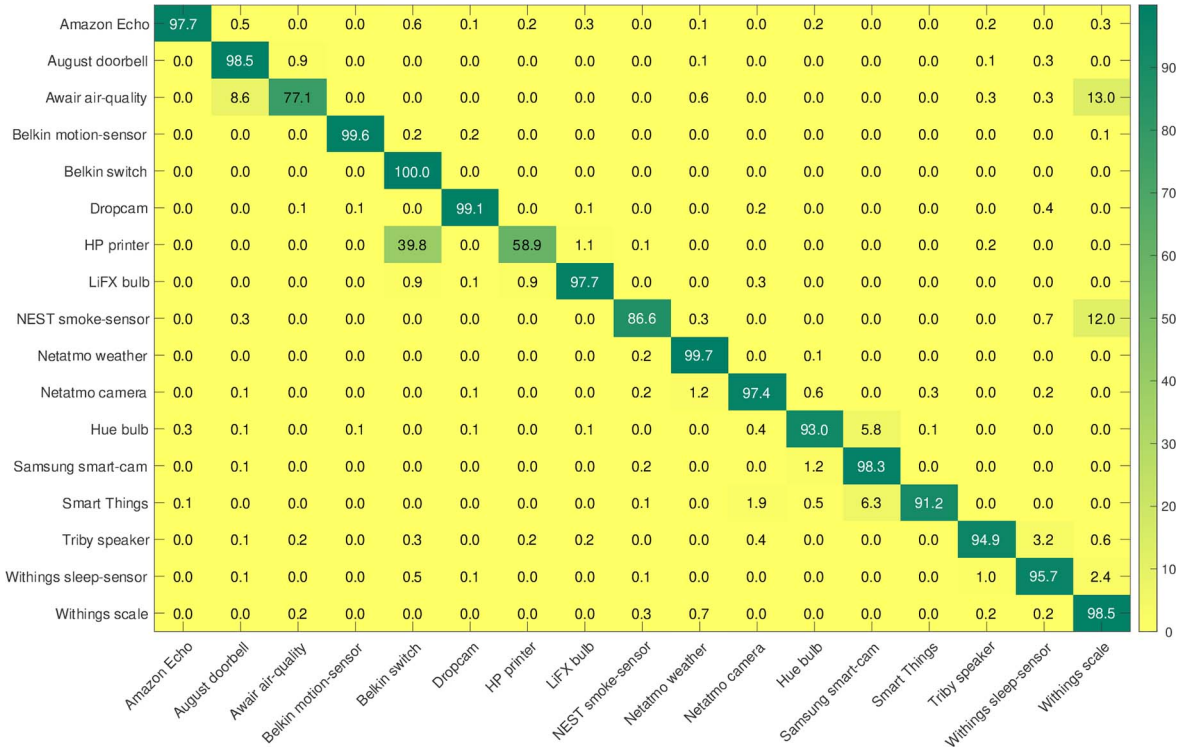
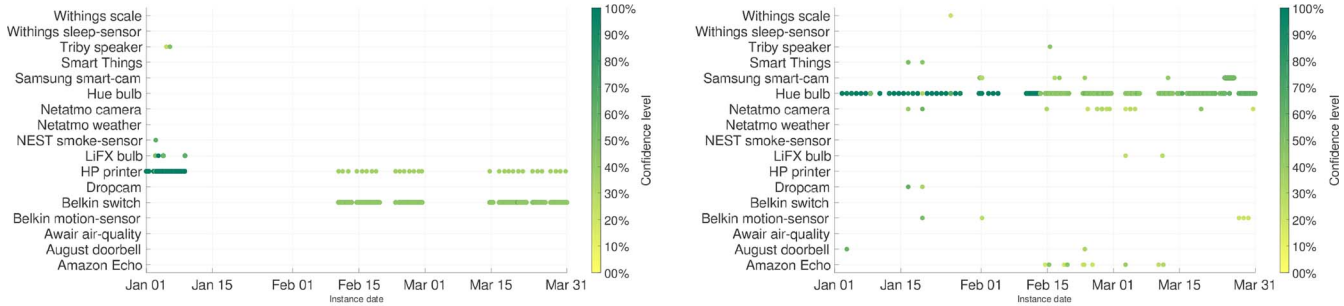


Fig. 7. Confusion matrix of IoT classifier trained by the first three months worth of data.



(a) HP printer.

(b) Hue light-bulb.

Fig. 8. Time trace of IoT classifier outputs with test traffic instances from: (a) HP printer, and (b) Hue light-bulb.

while 8.6% and 13.0% are misclassified as August doorbell and Withings scale, respectively. We note that the model displays a low confidence on average for incorrect prediction of these three classes, *i.e.,* 0.485 for Awair air-quality, 0.535 for HP printer, and 0.389 for NEST sensor – due to space constraints we omit detailed table of performance metrics per individual classes. Additionally, we find that even though 94.0% of Hue bulb instances are correctly classified but the average confidence of our model is 0.572 (*i.e.,* undesirably low).

To better analyze the poor performance of the model in certain classes, we plot in Fig. 8 the time trace of model outputs with test traffic instances from HP printer and Hue light-bulb. Each circle represents an instance and its color shows the model confidence. A color bar on the right side of plots shows the mapping of confidence values to colors – dark green indicates high confidence and yellow indicates low confidence. Starting from Fig. 8(a), we observe that instances of HP printer from first week of January are mostly classified correctly supported by high confidence levels (*i.e.,* dark green circles). The printer goes offline for about a month and comes back online on 11-Feb-2017, since then its traffic is mostly misclassified as Belkin switch with consistently low confidence levels from the model (*i.e.,* light green circles). This clearly shows that the behavior of HP printer changed when it restarted in mid-February – we manually inspected traffic traces and verified that it was due to a legitimate firmware upgrade (*i.e.,* benign changes). Moving to Fig. 8(b), we see classifier outputs for Hue bulb traffic instances during the whole testing period. Though instances are mostly predicted correctly, the confidence level starts falling, from an average of 0.93 to average 0.50, on 15-Feb-2017. Again this behavioral change led to a manual inspection by which we verified that it was legitimate.

Given these observations, we augment our training set with two weeks of data (*i.e.,* from 12-Feb-2017 to 25-Feb-2017) for duration over which new legitimate traffic patterns emerged.

TABLE III
PERFORMANCE METRICS OF THE IOT CLASSIER MODEL (AFTER RE-TRAINING)

IoT/Non-IoT	TP	FN	FP	Precision	Recall	F_1	TP avg. confidence	FN avg. confidence
Amazon Echo	0.977	0.023	0.000	1.000	0.977	0.989	0.994	0.430
August doorbell	0.989	0.011	0.001	0.999	0.989	0.994	0.974	0.509
Awair air-quality	0.936	0.064	0.002	0.998	0.936	0.966	0.850	0.616
Belkin motion-sensor	0.996	0.004	0.000	1.000	0.996	0.998	0.825	0.340
Belkin switch	0.999	0.001	0.001	0.999	0.999	0.999	0.990	0.604
Dropcam	0.990	0.010	0.001	0.999	0.990	0.994	0.987	0.462
HP printer	0.977	0.023	0.001	0.999	0.977	0.988	0.985	0.591
LiFX bulb	0.980	0.020	0.001	0.999	0.980	0.990	0.892	0.617
NEST smoke-sensor	0.976	0.024	0.001	0.999	0.976	0.987	0.818	0.472
Netatmo weather	0.997	0.003	0.002	0.998	0.997	0.997	0.935	0.824
Netatmo camera	0.973	0.027	0.001	0.999	0.973	0.986	0.980	0.371
Hue bulb	0.917	0.083	0.001	0.999	0.917	0.956	0.975	0.505
Samsung smart-cam	0.997	0.003	0.006	0.994	0.997	0.996	0.989	0.367
Smart Things	0.969	0.031	0.001	0.999	0.969	0.984	0.980	0.437
Triby speaker	0.941	0.059	0.001	0.999	0.941	0.969	0.785	0.452
Withings sleep-sensor	0.957	0.043	0.004	0.996	0.957	0.976	0.968	0.504
Withings scale	0.979	0.021	0.003	0.997	0.979	0.988	0.953	0.560

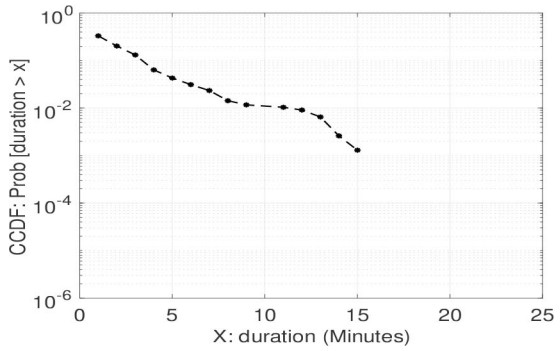


Fig. 9. CCDF of duration of continuously mis-detecting IoT from non-IoT.

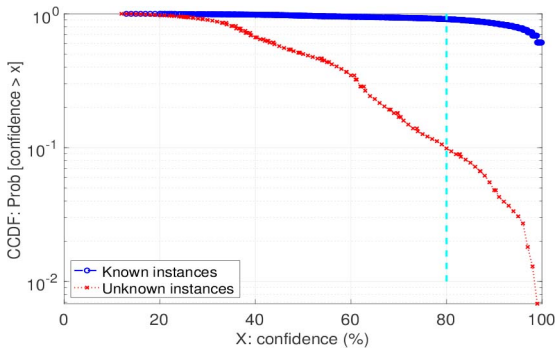


Fig. 10. CCDF of classifier's confidence-level for unknown and known device types.

Fig. 11 shows the performance of the IoT classifier after it is re-trained. It is seen that the confusion matrix is almost diagonal with the TP rate of more than 90% for all classes (*i.e.,* on average 97.4%). We list in Table III all performance metrics of the IoT classifier after re-training. We observe that the model average confidence is boosted across all classes – specifically it reaches to 0.975 for Hue bulb instances. Also, three performance metrics consistently display an acceptable performance of classification with 0.998, 0.973, and 0.986 for average precision, recall, and F1 score. It is important to note that there might be some similarities in traffic generated from different device types sourced from one manufacturer or different manufacturers [20] (*e.g.,* DNS queries or NTP frequency of Belkin motion sensors and power switch), but the collection of our attributes have strong predictive power in uniquely identifying devices based on their flow-level network activity.

Classifying Unknown IoT: Let us now consider a scenario where an unknown device type is presented to our IoT classifier. To evaluate the performance of classifier against unknown devices, we generate a set of device classification models (17 models) each is trained by 16 classes – we exclude one device type from training dataset of individual models. This enables us to test each model with corresponding known device types and an unknown device type (the one which is excluded in training). Unsurprisingly, our classifiers display low confidence-level when tested with unknown instances. We plot in Fig. 10 the CCDF of confidence-level of all classifiers for known and unknown device types separately. It is clearly seen that 90% of unknown instances give confidence less than 80%, while 92% of known instances yield a confidence-level greater than 80%. We will see in Section V-B how low-confidence classification would trigger an investigation by the network administrator.

IoT State Classifiers: From our DATA2, we generated different number of instances of four IoT devices with state labels including: Amazon Echo (boot: 208, active: 74, idle: 1795), Belkin switch (boot: 110, active: 84, idle: 2688), Dropcam (boot: 145, active: 98, idle: 2639) and LiFX bulb (boot: 160, active: 84, idle: 2338). To train individual device-specific models, we randomly choose 40 instances from each of their respective states – remaining instances are used to test the models.

Fig. 12 shows the confusion maps of the four IoT state classifiers. Our first observation is that all four models well predict the active state – 100.0% TP rate in three models (Amazon Echo, Dropcamp, and LiFX bulb), and 95.5% TP rate in Belkin switch. Next, we see that boot instances are prone to be misclassified as idle, and vice versa (*e.g.,* 8.6% and 7.5% of boot instances respectively in Belkin switch and LiFX

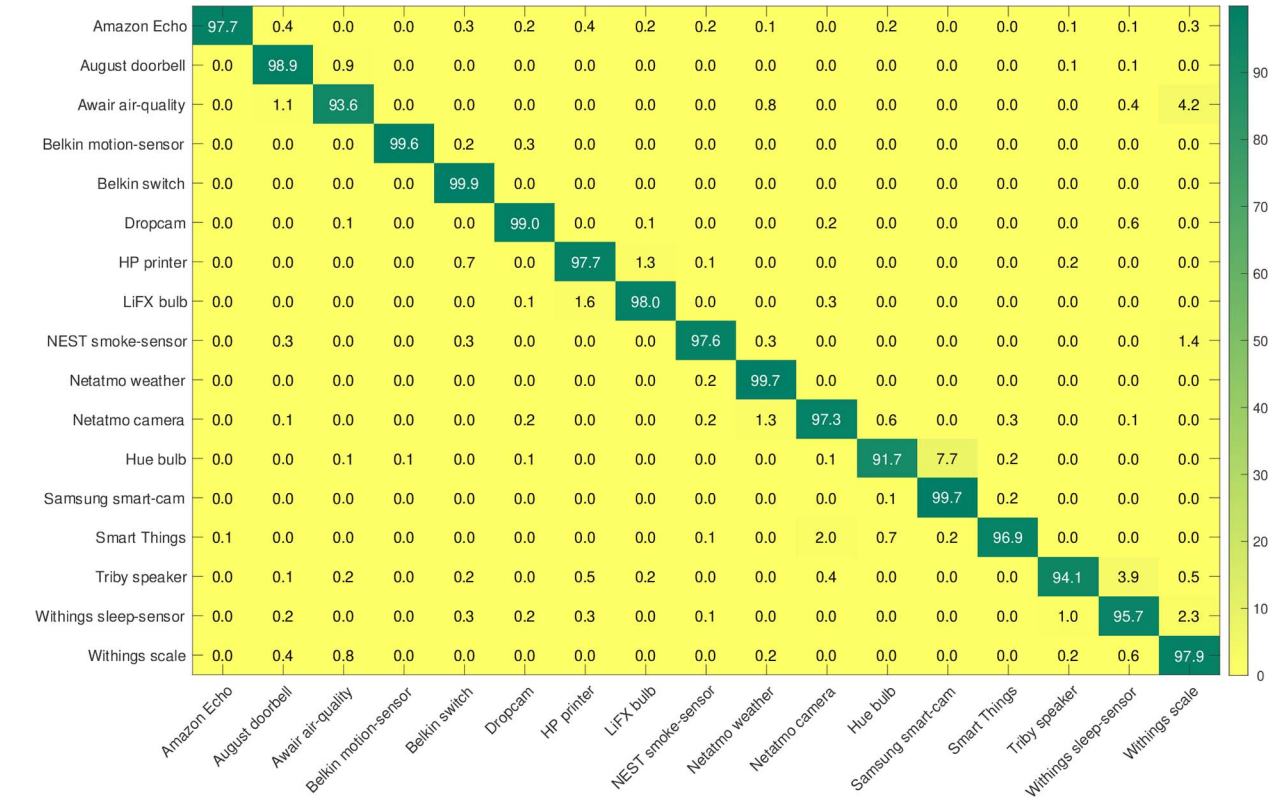
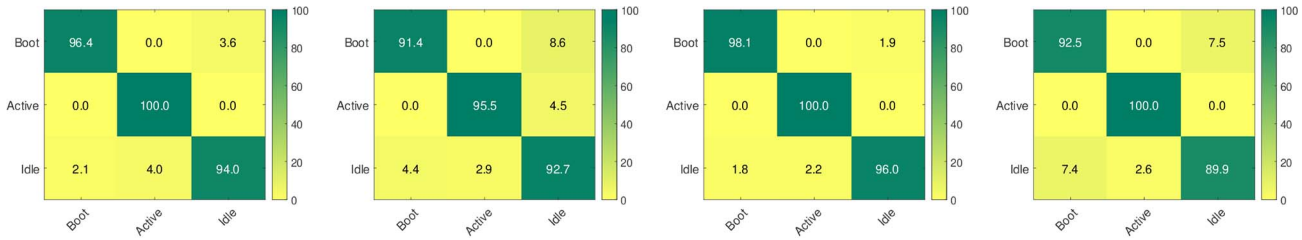


Fig. 11. Confusion matrix of IoT device classification re-trained by additional data.



(a) Amazon Echo. (b) Belkin switch. (c) Dropcam. (d) LiFX.

Fig. 12. Confusion matrix of IoT state classifiers: (a) Amazon Echo, (b) Belkin switch, (c) Dropcam, and (d) LiFX bulb.

bulb are misclassified as idle). This misclassification could be possibly because instances pertinent to state transitions (*e.g.,* boot to idle) are not precisely annotated.

V. Practical and Operational Considerations

In previous section, we evaluated the performance of our inference engines using all traffic attributes of devices during their normal operation. In this section, we first quantify the cost of our scheme in practice and show how we can optimize the trade-off between cost and performance. Next, we demonstrate how network operators can interpret the outputs of inference engines, managing their cyber-security risk.

A. Cost of Attributes

In order to quantify the cost of our scheme, we begin by examining the impact of individual attributes on the performance of traffic inference. Note that some of these attributes could be highly correlated, and hence become redundant. Also, some attributes may not be very relevant to class prediction, and hence can be removed.

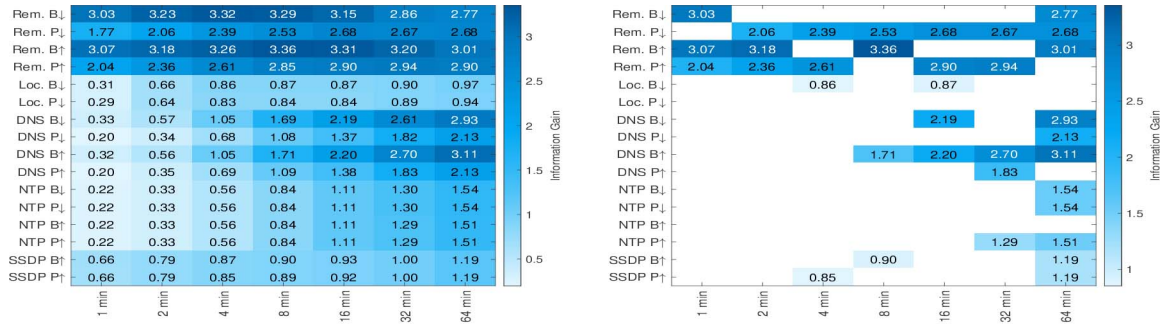
Redundant Attributes: We use a selection algorithm called Correlation-based Feature Subset (CFS) [56] with best-first searching method. CFS is a filter that uses a correlation-based heuristic to find a subset of attributes with the highest merit – *i.e.,* attributes highly-correlated with the class, yet uncorrelated with each other.

Importance of Attributes: In decision tree-based machine learning, Information Gain (IG) method is used to measure the wight of various attributes in accurate prediction. Important attributes carry more information (*i.e.,* large IG value) to distinguish classes, and unrelated attributes have no information. We now compute the IG value of attributes used for each of the three classifier types.

To better visualize the merit of various attributes, we illustrate in Fig. 13(a) the IG values computed for all 112 attributes used by the IoT classifier. Each cell represents an attribute

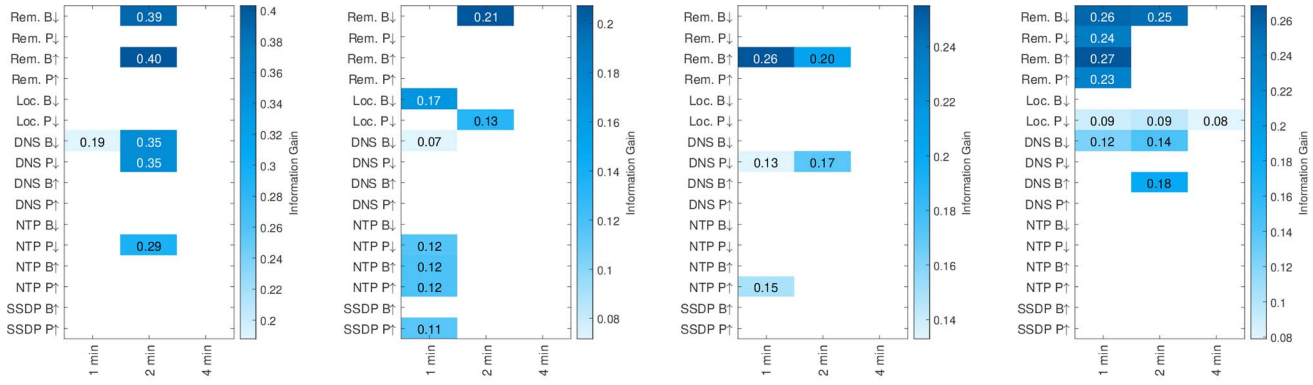
TABLE IV
FLOW ENTRIES (PER-DEVICE) NEEDED FOR NON-REDUNDANT ATTRIBUTES SET

Inference model	Rem.↑	Rem.↓	Loc.↓	DNS↑	DNS↓	NTP↑	NTP↓	SSDP↑	Num. of flow entries
IoT detector	✓	✓		✓	✓			✓	4
IoT classifier	✓	✓	✓	✓	✓	✓	✓	✓	8
State classifier - Amazon Echo	✓	✓			✓		✓		4
State classifier - Belkin Switch		✓	✓		✓	v	✓	✓	6
State classifier - Dropcam	✓				✓	✓			3
State classifier - LiFX	✓	✓	✓	✓	✓				5



(a) All attributes. (b) Subset of nonredundant attributes.

Fig. 13. Information gain value of: (a) all attributes, and (b) CFS-selected attributes, for the IoT classifier.



(a) Amazon Echo. (b) Belkin switch. (c) Dropcam. (d) LiFX bulb.

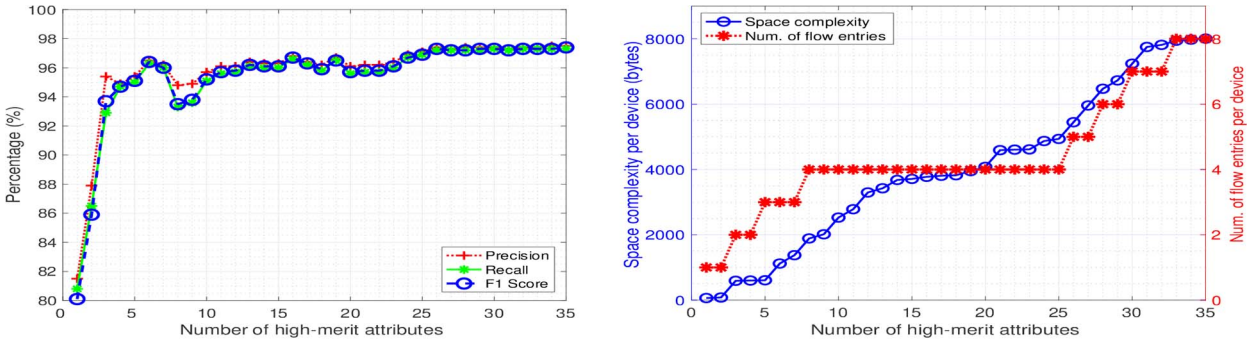
Fig. 14. Information gain of non-redundant attributes for state classier models: (a) Amazon Echo, (b) Belkin switch, (c) Dropcamp, and (d) LiFX bulb.

(*i.e.,* rows are flow counters and columns are various timescales), and is labeled (and color coded) by its IG value – the darker the cell, the higher the IG value. Fig. 13(b) shows a subset of 35 attributes selected by the CFS algorithm eliminating correlated (*i.e.,* redundant) attributes. Note that this subset still results the same performance of prediction as presented in the previous section.

We observe that the highest IG value 3.36 corresponds to "*outgoing remote byte-count over 8-minute*" followed by "*incoming remote byte-count over 4-minute*" with IG 3.32. Another observation is that byte-count of both incoming/outgoing remote over mid-term timescales (*i.e.,* 4-, 8-, 16-min) have higher information compared to other attributes, as shown by darker cells. Also, DNS counters over longer timescales (*i.e.,* 32- and 64-min) display a relatively high gain of information in predicting class of IoT devices.

Overall, attributes of two flow rules related to incoming local traffic and outgoing SSDP queries seem to have minimal impacts in IoT device classification. This is mainly because only a few of IoT devices in our lab (*e.g.,* Hue bulb, Bekin motion, Amazon Echo) communicate on the local network or send SSDP queries. Even though these flow rules (and associated attributes) may not seem important across all devices, they can precisely characterize and help identify devices which use them in their network traffic.

Moreover, we have analyzed the impact of attributes for other two types of classifiers. Fig. 14 shows the information gain value of non-redundant attributes for state classification models. We observe, for example, in Fig. 14(a) that attributes of only 4 flow rules (*i.e.,* incoming remote, outgoing remote, incoming DNS, incoming NTP) are needed for the state classification of Amazon Echo – there is no attribute selected for other 4 flows. Another observation is that attributes over very short timescales (*i.e.,* 1-min and 2-min) become important in classifying operating states of IoT devices. Also, we note that the variation of IG values for non-redundant attributes is less (*i.e.,* between 0.1 and 0.3) compared to the IoT classifier model (*i.e.,* between 0.86 and 3.36). For the IoT detector model,



(a) Performance metrics of classification.

(b) Cost of attributes and flow rules.

Fig. 15. Impact of attributes on: (a) performance, (b) cost, for the IoT classifier.

we found that attributes over longer timescale (*i.e.,* outgoing/incoming byte-count over 32-min and 64-min) have higher impact – we omit detailed results due to space constraints.

Cost Versus Performance: There exist two sources of cost in our inference scheme: (1) number of flow entries, and (2) space complexity of computing attributes. Given fixed size of TCAM on programmable (SDN) switches, efficient management of flow entries [57] becomes crucial to scale of scheme for deployment in a network with large number of IoT devices. Since our attributes are computed at multiple timescales up to 64-minute, we need to maintain time-series of flow counters accordingly (*i.e.,* 64 data-points each corresponds to a minute).

We, therefore, aim to reduce the cost by decreasing attributes, without significantly affecting performance. Table IV shows the number of flow entries needed by each inference model with non-redundant set of attributes – check-marked cells indicate the flows needed for attributes of models in each row. It clearly shows room for optimizing our approach by dynamic management of flow entries on the programmable switch. For example, the IoT detector model only needs 4 flow rules per device. Once a device is detected as IoT, additional 4 flows are needed by the IoT classifier (*i.e.,* a total of 8 flows). Once the IoT device is successfully classified, it may need less number of flows depending upon its specialized state classifier (some flows can be removed from the switch). The state classifier of Amazon Echo, Belkin switch, Dropcam, and LiFX respectively need 4, 6, 3, and 5 flow entries per each unit of device.

We further optimize by a careful trade-off between cost of performance. We: (a) first sort non-redundant attributes in descending order, (b) then accumulate attributes one-by-one from the sorted list, and (c) lastly, quantify the cost and performance at each step. Let us visualize this process for the IoT classifier in Fig. 15. We plot performance metrics and cost signals, each as a function of cumulative set of high-merit attributes. With 35 non-redundant attributes, it is seen: in Fig. 15(a) that average wighted precision, recall, F_1 score reach to 97.5%, 97.3%, 97.4% respectively, and in Fig. 15(b) that total cost per device would reach to 8KB of memory and 8 flow entries. We compute the space complexity by considering 8-byte values for a flow counter that can be stored as an unsigned long variable type. We note that with top 25 attributes we can achieve about 97% in all performance metrics which can save 4 flow entries (*i.e.,* 50% saving) and reduce the space complexity to 5KB (*i.e.,* 37% reduction). Obviously, operators of IoT networks may choose different strategies to balance their cost and performance depending upon their environment and resources.

B. Use of Inference Engines in Real-Time

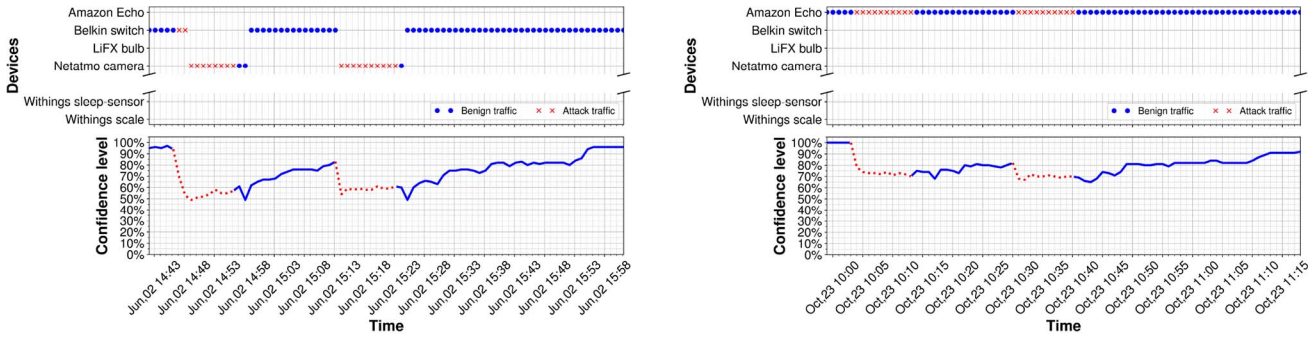
We now demonstrate the potential of our scheme that can help network operators detect behavioral changes due to malicious network activities or cyber-attacks.

Note that unlike general-purpose connected devices (*e.g.,* computers, phones), IoT devices display a limited number of identifiable states in their behavioral profile, and do not change their behavior (unless a firmware upgrade) during normal operation interacting with users and environment. This allows network operators to train data-driven inference engines that capture intended (normal) states of their IoT infrastructure, and hence better manage security risks by real-time monitoring the network behavior of devices. We discussed earlier in Section IV-B how legitimate firmware upgrade can be detected by our solution (*i.e.,* consistent misclassification and/or low confidence), as shown by Fig. 8. Note that sudden changes in outputs of inference models (if persist) for given device(s) can trigger an investigation by network administrators or inspection appliances.

We now test our classier models with attack traffic on IoT devices. We use a set of publicly available PCAP traces [44] that contain both benign and attack traffic (clearly annotated) corresponding to a few IoT devices we use in this work.

In Figures 16 and 17 we present results of three representative scenarios: (1) the output label of the IoT classifier changes persistently (*i.e.,* repeatedly misclassifying) accompanied by a sudden drop in confidence, (2) the output label of the IoT classifier does not change, but its confidence drops and persistently stays at low levels, and (3) the output of the IoT classifier remains normal (expected label with reasonable confidence), but the respective state classifier mis-behaves. In these plots, red crosses indicate time periods over which attack traffic is launched to the respective IoT device, and blue circles show purely benign traffic instances.

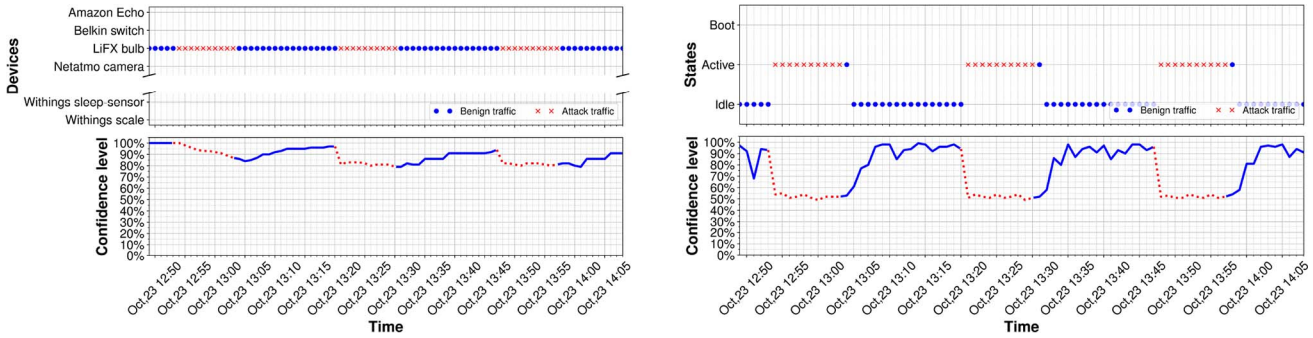
Fig. 16(a) illustrates the scenario 1 for Belkin switch. This time trace displays a situation where the IoT device



(a) IoT classifier with Belkin switch traffic.

(b) IoT classifier with Amazon Echo traffic.

Fig. 16. Time trace of device classifier outputs with benign and attack traffic of: (a) Belkin switch, and (b) Amazon Echo.



(a) IoT classifier with LiFX bulb traffic.

(b) LiFX state classifier with LiFX bulb traffic.

Fig. 17. Time trace of outputs for: (a) device classifier, and (b) state classifier, with benign and attack traffic of LiFx bulb.

experiences TCP SYN reflection attack twice, each for a duration of 10 minutes. It is seen that during attack periods (shown by red cross markers) the predicted label changes from Belkin switch to Netatmo camera with confidence less than 70%. Right after the attack, the output comes back to its original label and the confidence starts rising gradually.

Fig. 16(b) (representative of the scenario 2) displays a time trace of our monitoring scheme for Amazon Echo under UDP-based DoS attack over two periods of 10-minute each. We observe that these attacks do not change the predicted label of the IoT classifier, but cause the model confidence to decay rapidly (and remains below 80% persistently).

Lastly, Fig. 17(a) (representative of the scenario 3) illustrates a situation where attack traffic is not intense (*i.e.,* ping of death attack on LiFX bulb), and hence does not affect the broad view of the IoT classifier model. However, the specialized model of state classifier is significantly affected. During attacks period, LiFX bulb is persistently seen in active state which is not normal for a light-bulb since its activity (turning on/off or changing color) is expected to be relatively short. Additionally, the confidence of the state classifier quickly falls to a level of about 50% which is not normal again.

VI. Conclusion

Operators of smart environments face mounting pressure to enhance their visibility into their IoT infrastructure with many vulnerable devices. This paper developed a real-time monitoring solution for IoT devices using SDN-based flow-level telemetry combined with machine learning. We identified traffic flows that can collectively characterize the network behavior of IoT devices and their states such as booting, user interaction or idle. We then trained a set of classification models for a three-stage inference architecture using real traffic traces of 17 IoT devices collected over a period of 6 months. We validate their efficacy in detecting IoT devices from non-IoTs, classifying their type, and identifying their operating state. Lastly, we showed how we balance the trade-off between cost and performance of our scheme, and demonstrated how operators can use it to detect IoT behavioral changes (both legitimate and malicious).

References

[1] A. Sivanathan *et al.*, "Characterizing and classifying IoT traffic in smart cities and campuses," in *Proc. IEEE Infocom Workshop Smart Cities Urban Comput.*, Atlanta, GA, USA, May 2017, pp. 559–564.

[2] J. Gubbi, R. Buyya, S. Marusis, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] IoT Analytics. *State of the IoT 2018: Number of IoT devices*. Accessed: Dec. 21, 2019. [Online]. Available: https://bit.ly/2sR4eCI

[4] HP. (2014). *HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack.* [Online]. Available: http://bit.ly/38mNK4h

[5] F. Loi, A. Sivanathan, H. H. Gharakheili, A. Radford and V. Sivaraman, "Systematically evaluating security and privacy for consumer IoT devices," in *Proc. ACM CCS Workshop IoT Security Privacy (IoT S&P)*, Dallas, TX, USA, Nov 2017, pp. 1–6.

[6] "Cisco 2017 midyear cybersecurity report," Cisco, San Jose, CA, USA, Rep., 2017. [Online]. Available: https://www.cisco.com/c/dam/global/es_mx/solutions/security/pdf/cisco-2017-midyear-cybersecurity-report.pdf

[7] "Cisco 2018 annual cybersecurity report," Cisco, San Jose, CA, USA, Rep., 2018. [Online]. Available: https://www.cisco.com/c/dam/m/hu_hu/campaigns/security-hub/pdf/acr-2018.pdf

[8] M. J. Farooq and Q. Zhu, "Modeling, analysis, and mitigation of dynamic Botnet formation in wireless IoT networks," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 9, pp. 2412–2426, Sep. 2019.

[9] H. Suo, J. Wan, C. Zou, and J. Liu, "Security in the Internet of Things: A review," in *Proc. Int. Conf. Comput. Sci. Electron. Eng. (ICCSEE)*, vol. 3, 2012, pp. 648–651.

[10] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the Internet of Things: Perspectives and challenges," *Wireless Netw.*, vol. 20, no. 8, pp. 2481–2501, 2014.

[11] I. Andrea, C. Chrysostomou, and G. C. Hadjichristofi, "Internet of Things: Security vulnerabilities and challenges," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2015, pp. 180–187.

[12] Innovation-AUS. (Nov. 2018). *Call to Regulate the IoT Sector*. [Online]. Available: https://bit.ly/2qEMYMe

[13] IEEE-Spectrum. (Feb. 2019). *Japan to Probe IoT Devices and Then Prod Users to Smarten Up*. [Online]. Available: https://bit.ly/2I3RrBL

[14] InformationAge. (Apr. 2019). *U.S. Congress is Introducing a New Bill for IoT Security*. [Online]. Available: https://bit.ly/2CYE4zL

[15] Forbes. (May 2019). *U.K. to Introduce New Law for IoT Device Security*. [Online]. Available: https://bit.ly/2WpUI73

[16] I. Ali, S. Sabir, and Z. Ullah, "Internet of Things security, device authentication and access control: A review," 2019. [Online]. Available: http://arxiv.org/abs/1901.07309.

[17] V. Sivaraman, H. H. Gharakheili, A. Vishwanath, R. Boreli, and O. Mehani, "Network-level security and privacy control for smart-home IoT devices," in *Proc. IEEE 11th Int. Conf. Wireless Mobile Comput. Netw. Commun. (WiMob)*, Abu Dhabi, UAE, Oct. 2015, pp. 163–167.

[18] E. Lear, R. Droms, and D. Romascanu, "Manufacturer usage description specification," IETF, RFC 8520, Mar. 2019. [Online]. Available: https://www.rfc-editor.org/info/rfc8520

[19] A. Hamza, H. H. Gharakheili, and V. Sivaraman, "Combining MUD policies with SDN for IoT intrusion detection," in *Proc. ACM IoT Security Privacy (IoT S&P)*, Budapest, Hungary, Aug. 2018, pp. 1–7.

[20] A. Sivanathan *et al.*, "Classifying IoT devices in smart environments using network traffic characteristics," *IEEE Trans. Mobile Comput.*, vol. 18, no. 8, pp. 1745–1759, Aug. 2019.

[21] Y. Li, R. Miao, C. Kim, and M. Yu, "FlowRadar: A better NetFlow for data centers," in *Proc. USENIX NSDI*, Santa Clara, CA, USA, Mar. 2016, pp. 311–324.

[22] B. Claise, "Cisco systems NetFlow services export version 9," IETF, RFC 3954, Oct. 2004. [Online]. Available: https://www.rfc-editor.org/info/rfc3954

[23] SolarWinds. (Oct. 2010). *NetFlow Basics and Deployment Strategies*. [Online]. Available: https://bit.ly/2K4EZ6Q

[24] *sFlow*. Accessed: May 1, 2019. [Online]. Available: https://sflow.org/about/index.php

[25] D. Levi, P. Meyer, and B. Stewart, "Simple network management protocol (SNMP) applications," RFC, Rep. 3431, 2002.

[26] V. Srinivasan, J. Stankovic, and K. Whitehouse, "Protecting your daily in-home activity information from a wireless snooping attack," in *Proc. UbiComp*, 2008, p. 202–211.

[27] A. Acar *et al.*, "Peek-a-boo: I see your smart home activities, even encrypted!" Aug. 2018. [Online]. Available: https://arxiv.org/abs/1808.02741.

[28] Y. Zhu *et al.*, "Packet-level telemetry in large datacenter networks," in *Proc. Conf. Special Interest Group Data Commun. (SIGCOMM)*. New York, NY, USA, 2015, pp. 479–491.

[29] J. Rasley *et al.*, "Planck: Millisecond-scale monitoring and control for commodity networks," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 407–418, Aug. 2014.

[30] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[31] R. Hofstede, I. Drago, A. Sperotto, R. Sadre, and A. Pras, "Measurement artifacts in NetFlow data," in *Proc. Passive Active Meas. (PAM)*. Berlin, Heidelberg, 2013, pp. 1–10.

[32] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification," in *Proc. SIGCOMM Conf. Internet Meas.*, vol. 25, New York, NY, USA, 2004, pp. 135–148.

[33] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, and Y. Guan, "Network traffic classification using correlation information," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 104–117, Jan. 2013.

[34] M. Lyu, H. H. Gharakheili, C. Russell, and V. Sivaraman, "Mapping an enterprise network by analyzing DNS traffic," in *Passive and Active Measurement*. Puerto Varas, Chile, Mar. 2019, pp. 129–144.

[35] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for Internet of Things," *IEEE Access*, vol. 5, pp. 18042–18050, 2017.

[36] P.-M. Junges, J. François, and O. Festor, "Passive inference of user actions through IoT gateway encrypted traffic analysis," in *Proc. IFIP/IEEE Symp. Integr. Netw. Serv. Manag.*, Apr. 2019, pp. 7–12.

[37] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra, "Uncovering privacy leakage in BLE network traffic of wearable fitness trackers," in *Proc. Int. Workshop Mobile Comput. Syst. Appli. (HotMobile)*, St. Augustine, FL, USA, 2016, pp. 99–104.

[38] B. Copos, K. Levitt, M. Bishop, and J. Rowe, "Is anybody home? Inferring activity from smart home network traffic," in *Proc. SPW*, 2016, pp. 245–251.

[39] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted IoT traffic," in *Proc. Workshop Data Algorithmic Transparency*, May 2017, pp. 1–6.

[40] Y. Meidan *et al.*, "Detection of unauthorized IoT devices using machine learning techniques," 2017. [Online]. Available: http://arxiv.org/abs/1709.04647.

[41] 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 *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 2177–2184.

[42] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Z. Yang, "Automatic device classification from network traffic streams of Internet of Things," in *Proc. IEEE 43rd Conf. Local Comput. Netw. (LCN)*, Chicago, IL, USA, Oct. 2018, pp. 597–605.

[43] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer Internet of Things devices," in *Proc. IEEE Security Privacy Workshops (SPW)*, May 2018, pp. 29–35.

[44] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, "Detecting volumetric attacks on IoT devices via SDN-based monitoring of MUD activity," in *Proc. ACM Symp. SDN Res. (SOSR)*, San Jose, CA, USA, Apr. 2019, pp. 36–48.

[45] *eSpeak: Speech Synthesizer*. Accessed: Apr. 3, 2019. [Online]. Available: http://espeak.sourceforge.net/

[46] A. Sivanathan, D. Sherratt, H. H. Gharakheili, V. Sivaraman, and A. Vishwanath, "Low-cost flow-based security solutions for smart-home IoT devices," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, Bengaluru, India, Nov. 2016, pp. 1–6.

[47] H. H. Gharakheili, M. Lyu, Y. Wang, H. Kumar, and V. Sivaraman, "iTeleScope: Softwarized network middle-box for real-time video telemetry and classification," *IEEE Trans. Netw. Serv. Mang.*, vol. 16, no. 3, pp. 1071–1085, Sep. 2019.

[48] L. Hendriks, R. de O. Schmidt, R. Sadre, J. A. Bezerra, and A. Pras, "Assessing the quality of flow measurements from OpenFlow devices," in *Proc. IEEE Tandem Matching Anal. (TMA)*, Louvain La Neuve, Belgium, Apr. 2016, pp. 1–8.

[49] H. Jiang and C. Dovrolis, "Why is the Internet traffic bursty in short time scales?" *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 241–252, Jun. 2005.

[50] A. Sivanathan. (2018). *SDN Sim*. [Online]. Available: https://github.com/arunmir/sdn-sim

[51] EY. (Apr. 2017). *Cybersecurity Compromise Diagnostic: Hunting for Evidence of Cyber Attackers*. [Online]. Available: https://go.ey.com/2X7yTIS

[52] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Pittsburgh, PA, USA, Jun. 2006, pp. 161–168.

[53] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[54] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1257–1270, Aug. 2015.

[55] E. Frank, M. A. Hall, and I. H. Witten, *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*, 4th ed. Burlington, MA, USA: Morgan Kaufmann, 2016.

[56] M. A. Hall, "Correlation-based feature subset selection for machine learning," Ph.D. dissertation, Dept. Comput.Sci., Univ. Waikato, Hamilton, New Zealand, 1998.

[57] A. Vishnoi, R. Poddar, V. Mann, and S. Bhattacharya, "Effective switch memory management in OpenFlow networks," in *Proc. ACM Distrib. Event Based Syst. (DEBS)*, Mumbai, India, May 2014, pp. 177–188.

Arunan Sivanathan received the bachelor's degree from the University of Peradeniya, Sri Lanka, in 2012. He is currently pursuing the Ph.D. degree with the School of Electrical and Telecommunication Engineering, University of New South Wales, Sydney. He later joined the University of Jaffna, Sri Lanka, as a Lecturer from 2013 to 2016. His primary research interests include security of Internet of Things and data analytics on machine-to-machine communication.

Hassan Habibi Gharakheili received the B.Sc. and M.Sc. degrees in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2001 and 2004, respectively, and the Ph.D. degree in electrical engineering and telecommunications from the University of New South Wales, Sydney, Australia, in 2015, where he is currently a Lecturer. His current research interests include programmable networks, learning-based networked systems, and data analytics in computer systems.

Vijay Sivaraman received the B.Tech. degree from the Indian Institute of Technology Delhi, India, in 1994, the M.S. degree from North Carolina State University in 1996, and the Ph.D. degree from the University of California at Los Angeles in 2000. He has worked with Bell-Labs as a Student Fellow, in a silicon valley start-up manufacturing optical switch-routers, and as a Senior Research Engineer with the CSIRO, Australia. He is currently a Professor with the University of New South Wales, Sydney, Australia. His research interests include software defined networking, network architectures, and cyber-security particularly for IoT networks.