Inferring Connected IoT Devices from IPFIX Records in Residential ISP Networks

Arman Pashamokhtari*, Norihiro Okui†, Yutaka Miyake†, Masataka Nakahara†, Hassan Habibi Gharakheili*
*School of Electrical Engineering and Telecommunications, UNSW, Sydney, Australia
†KDDI Research, Inc., Saitama, Japan
Emails: {armanpasha, h.habibi}@unsw.edu.au, {no-okui, miyake, ms-nakahara}@kddi-research.jp

***Abstract*—Residential ISPs today have limited device-level visibility into subscriber houses, primarily due to network address translation (NAT) technology. The continuous growth of "unmanaged" consumer IoT devices combined with the rise of work-from-home makes home networks attractive targets for cyber attacks. Volumetric attacks sourced from a distributed set of vulnerable IoT devices can impact ISPs by deteriorating the performance of their network, or even making them liable for being a carrier of malicious traffic. This paper explains how ISPs can employ IPFIX (IP Flow Information eXport), a flow-level telemetry protocol available on their network, to infer connected IoT devices and ensure their cyber health without making changes to home networks. Our contributions are threefold: (1) We analyze near three million IPFIX records of 26 IoT devices collected from a residential testbed over three months and identify 28 features, pertinent to their network activity and services, that characterize the network behavior of IoT devices – we release our IPFIX records as open data to the public; (2) We develop a multi-class classifier to infer the presence of certain IoT device types in a home network from NATed IPFIX records. We also develop a *Trust* metric to track network activity of detected devices over time; and, (3) We evaluate the efficacy of our inferencing method by applying the trained classifier to IPFIX traces which yields an average accuracy of 96% in detecting device types. By computing a temporal measure of trust per each device, we highlight (on our testbed) a permanent behavioral change in third of devices as well as some intermittent behavioral changes in others.**

***Index Terms*—IoT, traffic inferencing, residential networks, IPFIX, machine learning**

I. Introduction

Home networks are becoming increasingly complex, yet neither ISPs nor subscribers have much visibility into connected devices and their network-level behavior. Technologies like network address translation (NAT) present only an opaque view of the home network to the global Internet [1], [2]. This makes it surprisingly challenging for ISPs to even detect and discover connected devices, as the traffic transmitted by every active device in a home would have the same IP and MAC address of the home gateway.

Consumer IoT devices have become popular by offering convenient functionalities to smart homes. It is anticipated that the number of smart homes will increase to about half a billion by 2025, which is more than two times larger than the same in 2020 [3]. Consumer IoT devices come in different categories, including but not limited to smart cameras, speakers, voice assistants, health devices, electrical plugs, and air-quality sensors.

IoT devices are believed to be more vulnerable [4] to cyber attacks than general-purpose IT devices. This is mainly due to a lack of sufficient security measures embedded into resource-constrained IoT devices. Moreover, typical home users often do not have adequate skills to protect their network and devices, hence creating risks for the entire Internet ecosystem. Recent reports [4], [5] highlight how a significant portion of deployed IoT devices, exposed with default passwords, are low-hanging fruits for attackers.

Traditional static tools, used for identifying IT assets (personal computers and smartphones), fall short when employed for agentless IoT devices [4] – not fully capture the heterogeneous behavior of IoT assets. However, obtaining continuous visibility into connected networks is an essential first step for securing these vulnerable devices [6]. Visibility means inferring device types and profiling their expected behavior by analyzing their network activity. Once device identities are determined, their dynamic behavior can be tracked, allowing for verification of their health or flagging any significant deviation from the norm.

ISPs enable IoT devices in homes to connect with their intended servers on the Internet. Stable and fast Internet connections can indulge botnets and malware in launching coordinated volumetric attacks (DDoS) using millions of infected IoTs [7] – this can affect the performance of ISPs core network and potentially make them accountable for carrying traffic of illegal campaigns [8]. On the other side, residential subscribers gradually express willingness in paying to ISPs [9] for improved security of their Internet-connected products. Therefore, ISPs seem to be relatively incentivized to play a role in providing network monitoring and security services to households.

Machine learning techniques are widely used by both industry and academia for enhancing network visibility in a variety of domains and use-cases [10]. Given their specialized functionalities, IoT devices (as opposed to IT devices) are proven [6] to display a finite set of identifiable patterns on the network, enabling operators to tightly model their expected (normal) behavior. Therefore, inferring types of IoT devices [11] and/or detecting their anomalous behaviors [12] can be achieved far more effectively than traditional signature-based techniques. Recently, several attempts have been made by researchers [2], [13]–[17] to help ISPs achieve this task. Existing works come with their own limitations like: (a)

requiring changes (hardware and/or software) to individual home gateways (difficult to scale) [2], [13], [15], [16]; (b) heavily dependent on the private identity (MAC/IP address) of devices (not applicable for post-NAT scenarios) [13]–[16]; and, (c) relying on signatures like domain name and/or IP block of manufacturers' server (not necessarily capturing the behavior of individual devices and hence yielding less reliable inferencing) [13], [17].

In this paper, we employ IPFIX data (a legacy flow-based telemetry) to detect consumer IoT devices in households. We first analyze near three million IPFIX records of 26 IoT devices in our testbed and extract 28 flow-level features which represent the network behavior of these devices – we release our IPFIX records as open data [18] to the public. We then train and tune a multi-class classifier model to infer the type of IoT devices from the features of their IPFIX records. We also deduce thresholds specific to each device class, and develop a trust metric per class, reducing the rate of misclassification as well as enabling us to monitor the behavioral health of detected devices. Lastly, we evaluate the performance of our classifier across device types and network services, and draw insights into how behavioral changes can be detected by our inferencing method.

II. Related Work

Gaining visibility into residential networks can enable value-add service offerings like quota management, parental control, and cyber security monitoring [1], [19], benefiting both ISPs and subscribers.

Inferencing from Residential IoT Traffic: Consumer IoTs are purpose-built devices to perform a finite (and often distinguished) set of functions on the network, and therefore create an opportunity for ISPs to automatically detect them and profile their network behavior. Prior relevant works [2], [14], [15] employed new or instrumented home gateways, making it far from trivial for deployment at scale. DEFT [15] developed a hierarchically distributed method for classifying IoT devices in home networks. The authors proposed to use SDN-based home gateways in each home that are coordinated by a central controller in the ISP cloud. Individual home gateways extract a mix of packet and flow features, and apply a trained classier (running as virtual network function on the gateway) to the traffic features – when unsure (or a new device discovered), locally computed features are sent to the central controller. Work in [2] requires NetFlow-enabled home gateway augmented by a "local detector" hardware for identifying vulnerable IoT devices in home networks using public databases like Common Vulnerability Exposures (CVE) and National Vulnerability Database (NVD). IoT-Sentinel [14] classifies IoT devices by collecting packet-based features using SDN-enabled gateways.

A recent work [17] developed a more scalable deterministic method for detecting IoT devices in home networks by collecting flow data from the core of ISP networks. The authors used IP addresses, port numbers, and domain names of the contacted cloud servers to determine the type of devices connected to each home network. Though the proposed method gives scalability, the resolution (by manufacturer and platform) and accuracy (70-80%) of their inferencing could be improved.

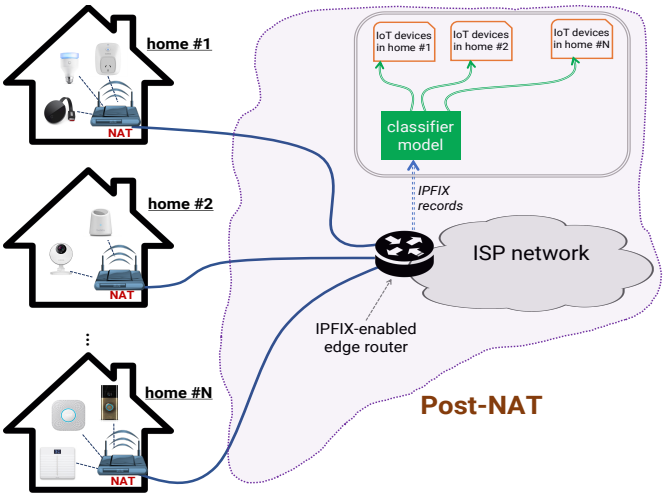

Fig. 1. System architecture of IPFIX classifier used by ISP.

Inferencing from IPFIX Telemetry: IPFIX has been used for different purposes like flow-based anomaly detection [20] and application classification [21]. To the best of our knowledge, this is the first work to use activity features of IPFIX records (collected from the edge of ISP networks) to infer IoT device types in home networks. This approach provides several advantages: (a) it requires no change to existing home networks; (b) statistical features of IPFIX records enable acceptable prediction even when inferencing is done post-NAT; (c) classifying individual flow records makes it stateless (no need to compute and/or maintain features over a window of time), and hence the complexity of inferencing is reduced; and, (d) IPFIX records contain no payload information, and thereby there is no privacy concern.

III. Inferencing Connected IoT Devices from IPFIX Records

Identifying the composition of IoT devices in households can be done either by collecting network telemetry data (packet-based and/or flow-based) from inside (pre-NAT) or outside home networks (post-NAT). The pre-NAT telemetry, indeed, reveals more information about the connected devices, particularly their unique identifiers (*e.g.,* MAC and/or IP addresses), enabling network operators to combine various telemetry records and map them to their unique device identifier. However, collecting pre-NAT telemetry is not a trivial exercise without making changes to legacy home gateways – prohibitively expensive for ISPs to deploy at scale for hundreds of thousands of households. On the other hand, the post-NAT approach provides a limited amount of data, primarily because NAT hides the entire internal network and identity of active devices – this makes it infeasible to directly associate flow records with their respective end-devices. That said, post-NAT inferencing and monitoring would be practically more attractive for ISPs, particularly for the ease of deployment at scale.

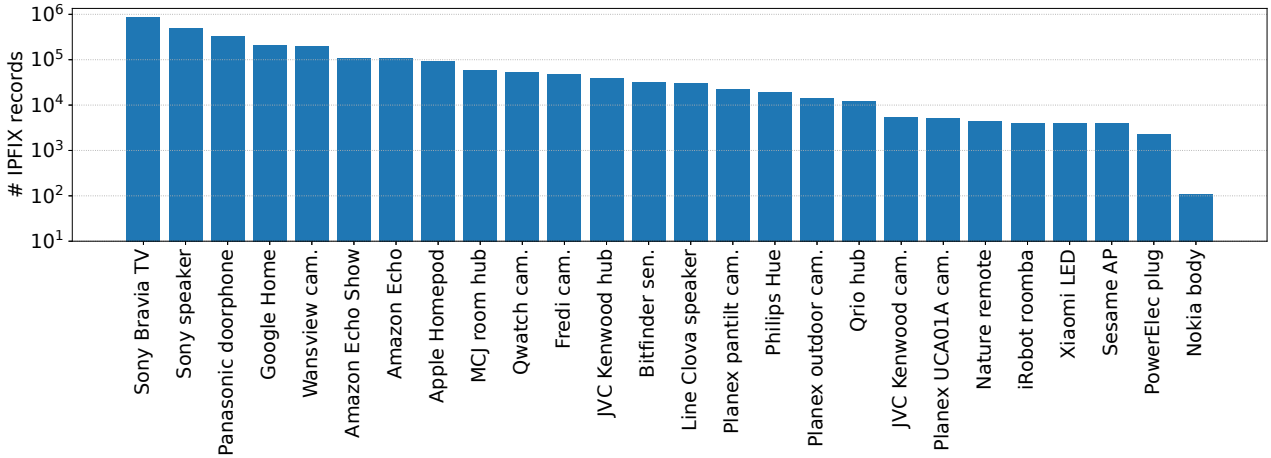

Fig. 2. Our dataset: number of IPFIX records across 26 IoT devices.

IPFIX [22] is a standard protocol for transmitting IP flow data from network devices, like switches and routers for network monitoring and analysis. IPFIX uses a five-tuple flow key (including source and destination IP address, source and destination transport-layer port number, and transport-layer protocol) to uniquely identify and aggregate packets belonging to the same flow. For connection-oriented flows, IPFIX uses session termination signals such as `TCP FIN` to detect the end of flows. For connection-less flows (*e.g.,* `UDP`), on the other hand, IPFIX uses two parameters, namely *idle-timeout* (by default five minutes) and *active-timeout* (by default 30 minutes). If no packet is exchanged over a flow for the idle-timeout or if a flow is active for more than the active-timeout, the flow is terminated and a corresponding IPFIX record is exported.

Fig. 1 illustrates the architecture of our inference system. Traffic from home networks is sent to the ISP network, where IPFIX-enabled edge routers are configured to export the respective IPFIX records. When a flow is terminated, its IPFIX record is presented to a classification model for determining its corresponding device type. Each IPFIX flow record will lead to the detection of a single IoT device (inside the home) that exchanged data with a server. The ISP will progressively append a list of IoT devices for each home, upon discovery of a new type by predicting the device label of IPFIX flows. It can be seen in Fig. 1 that with post-NAT inferencing (the scope of this paper) the ISP can only capture remote flows via which an IoT device inside a home network communicates with a remote server on the Internet; in contrast, the pre-NAT approach (practically challenging for ISPs, and beyond the scope of this paper) may provide richer visibility into every traffic including local and remote flows.

A. IPFIX Dataset and Features

Our residential testbed comprises 29 consumer IoT devices ranging from smart-camera, speaker, door phone, and TV to lightbulb, sensor, and vacuum cleaner. We collected PCAP traces from our testbed (pre-NAT) during January, March, and April 2020 – due to some technical issues in our lab, we were not able to collect the network traffic during February 2020. It is important to note that we collected our data in a pre-NAT setup to obtain ground-truth label (device MAC/IP address) of traffic data; however, we do not take any advantage of device identity (their MAC and/or IP address) in our inferencing process. We exclude the local flows from our raw data collected pre-NAT, and only keep remote flows for developing our inferencing models in this paper. In other words, our processed data truly represent a post-NAT telemetry scenario.

Our IoT network traces constitute: (i) traffic generated due to human users interacting with the devices at least twice a week in months January and March – for example, asking questions from smart speakers, checking air quality from smart sensors, switching the color of smart lightbulbs, watching a live stream from cameras; as well as (ii) traffic generated by the devices autonomously – for example, `DNS/NTP` activities that are unaffected by human interaction. During April, due to the COVID outbreak, devices were operated completely autonomously. For April 12-13 and 21-22, we have no traffic trace of the devices due to a power shutdown and the server outage, respectively. Note that PCAP files were configured to record up to a size limit of 9.6 MB (meaning time slots with a duration of approximately 20-30 minutes). Hence extracting IPFIX records from individual PCAPs may lead to a flow that crosses the boundary of subsequent slots and breaks into shorter flows. We, therefore, stitch PCAPs on a monthly basis, and then extract IPFIX records from the monthly PCAPs. We use the YAF tool [23] to generate IPFIX binary files from the PCAP files, followed by applying the Super Mediator tool [24] to generate IPFIX JSON files from the binaries. YAF comes with an option called `flow-stats` that allows exporting a richer set of flow-level information. In this paper, we use this option to embed all the possible features into individual IPFIX records.

Our IPFIX dataset [18] contains more than nine million IPFIX records in which about 70% of them are local flows that do not leave the NAT gateway (cannot be captured by the ISP edge routers). We use the IPv4 address of IPFIX records to filter local flows, including: (i) unicast flows with both source and destination from private address space (*i.e.,* `10/8`,

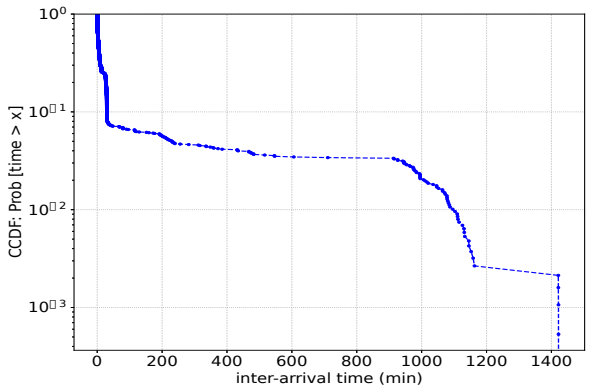

Fig. 3. CCDF of (daily) average inter-arrival time of IPFIX records per IoT device in our dataset.

`172.16/12`, `192.168/16`) reserved by the Internet Assigned Numbers Authority (IANA) [25], (ii) multicast flows (*i.e.,* with address in the range of `224/4`), (iii) link-local flows (*i.e.,* with address in the range of `169.254/16`), and (iv) broadcast flows. Three devices, namely Sony camera, Planex one-shot camera, and LinkJapan eSensor are found with only local flows in the dataset, hence excluded from our study in this paper. After excluding the local flows, there are about three million remote IPFIX records – for the rest of this paper, we will focus on remote flows only. Fig. 2 illustrates the number of IPFIX records we analyze per device during January, March, and April 2020. Highly active devices like Sony Bravia TV, Sony speaker, Panasonic doorphone, Google Home, and Wansview camera generated over 100K flows. Devices like MCJ room hub, Fredi camera, JVC Kenwood hub, Philips Hue, and Qrio hub are moderately active with 10K-100K flows. Lastly, we observe relatively less-active devices like Nokia body that generated only 107 flows during these three months.

Considering the frequency of network activities per device type, we plot in Fig. 3 the CCDF of average flow inter-arrival time, on a daily basis per device during the three months. The average inter-arrival time across all device types is about 50 minutes, while 15 device types generate a flow every 10 minutes on average. We observe that the longest daily average of flow inter-arrival time belongs to the Nokia body with more than 10 hours because this device only becomes active whenever the user interacts with it.

Moving to data fields of each IPFIX record, Table I summarizes a list of all potential elements (indicative of network activity of the flow) that will be used as part of features in §V. It is important to note that each IPFIX record represents a bi-directional flow [26], hence there are two sets of these features – each specific to a direction (outbound and inbound). For example, `smallPacketCount` (third row in Table I) shows the number of small packets that an IoT device (the initiator of flow) transmitted to an Internet-based server (remote endpoint), and accordingly, `reverseSmallPacketCount` indicates the number of small packets that the remote endpoint sent to the IoT device on the same flow.

In addition to the activity features listed in Table I, we capture the identity of each flow record by a set of binary features (Table II), highlighting their network service. We use

TABLE I
"UNIDIRECTIONAL" ACTIVITY FEATURES EXTRACTED FROM IPFIX RECORDS.

Feature	Description
`packetTotalCount`	# packets
`octetTotalCount`	# bytes
`smallPacketCount`	# packets containing less than 60 bytes payload
`largePacketCount`	# packets containing at least 220 bytes payload
`nonEmptyPacketCount`	# packets containing non-empty payload
`dataByteCount`	total size of payload
`averageInterarrivalTime`	average time (ms) between packets
`firstNonEmptyPacketSize`	payload size of the first non-empty packet
`maxPacketSize`	the largest payload size
`standardDeviationPayloadLength`	standard-deviation of payload size for up to the first 10 non-empty packets
`standardDeviationInterarrivalTime`	standard-deviation of inter-arrival time for up to the first 10 packets

four popular services including: `HTTP` (contributing to 6% of total flows in our dataset), `HTTPS` (35% of flows), `DNS` (20% of flows), and `NTP` (2% of flows) along with an aggregate of any other network services over `TCP` (3% of flows) or `UDP` (30% of flows) protocol. Note that 4% of the flows (in our dataset) use network control protocols like `ICMP` and `IGMP` that are not associated with either of the dominant transport-layer protocols namely `TCP` and `UDP`.

We found some device types that display distinct behaviors by certain features in their IPFIX records. For example, IPFIX records with `reverseSmallPacketCount` larger than 500 would belong to Sony speaker with a probability of 95%; also, 97% of the IPFIX records which their `reverseDataByteCount` falls between 300 KB and 400 KB, belong to Sony Bravia TV. That said, we observe some overlap of features across device types. This is mainly because each IPFIX record contains an "aggregate" set of information on the activity and identity of a single network traffic flow. For example, we found 16 devices share the `averageInterarrivalTime` feature, having a value less than 10 seconds on average; five devices have this feature valued between 10 and 20 seconds; and, five other devices have a value greater than 20 seconds in their IPFIX records.

In another example, we have three categories of devices in our dataset by only considering the `reversePacketTotalCount` feature: (a) Planex UCA01A camera with a high count (more than 200) of packets, (b) Xiaomi LED and iRobot Roomba with medium packet counts (between 100 and 200), and (c) other 23 device types with low packet counts (less than 70). Also, in terms of network services, devices like Fredi camera, JVC Kenwood camera, Sesame access point, Xiaomi LED, and all three Planex cameras never use `HTTPS`. Instead, all IPFIX records of a device like Qrio hub are `HTTPS`.

TABLE II
"BINARY" IDENTITY FEATURES OF POPULAR SERVICES EXTRACTED FROM IPFIX RECORDS.

Service	Protocol	Source/destination port numbers
HTTP	TCP	80, 8080, 8008, or 8888
HTTPS	TCP	443, 1443, 8443, or 55443
DNS	UDP	53, or 5353
NTP	UDP	123
others	TCP	*any*
others	UDP	*any*

These examples highlight that determining the class type of IPFIX records would certainly require a collection of features. Also, mapping these identified features to their correct class cannot be achieved by a set of deterministic rules. We, therefore, employ machine-learning techniques that have been widely adopted by researchers for network traffic inferencing [27] to learn the statistical attributes of IPFIX records for inferring their device class. We choose Random Forest, a powerful machine learning-based algorithm that builds a decision tree ensemble – allowing for the automatic creation of a sequential combination of feature rules (similar to what we did in the examples above) to predict the target class. In order to maximize the prediction performance, we extract every possible feature from the IPFIX records and let the Random Forest algorithm construct the optimal decision trees.

IV. CLASSIFICATION AND BEHAVIORAL TRACKING

As mentioned earlier, analyzing flows in a post-NAT scenario will reveal partial information about actual IoT devices connected to the home network; therefore, a machine learning-based model may misclassify some IPFIX records. Note that the primary objective of our inferencing is to determine whether an IoT device is present and active in a home network, or not. Therefore, the quality of each prediction is of top priority. This objective necessitates a method for discarding inconfident predictions, made primarily due to a lack of distinct features in the subject IPFIX flow. Note that one may attempt to aggregate individual IPFIX records over a time window in order to increase the amount of data passed into the inferencing model. However, this is not applicable in post-NAT deployment as there is no association between individual IPFIX records and devices.

Machine learning models often produce a measure of confidence for their prediction. The confidence-level is a number between 0 and 1. Higher confidence values indicate more reliable predictions. We, therefore, use the confidence-level of our Random Forest model to decide whether a prediction is "accepted" or "discarded" – certain thresholds are determined and applied to the confidence-level of the model's prediction. We note that thresholds can be applied globally [6] across all classes, or specifically per individual classes. Given the diversity in network behaviors as well as richness of training data across various IoT types, we choose to apply class-specific thresholds that are obtained from the training phase.

The model's confidence for correctly predicting instances of class L is denoted by C_L, and the corresponding threshold is denoted by λ which is calculated for each class separately. To set the thresholds, we use the distribution (average: μ_{C_L} and standard-deviation: σ_{C_L}) of confidence per each class during the training phase as baseline. In this paper, we consider two ways of thresholding per each class: (a) greater than the average confidence: $\lambda_1 = [\mu_{C_L}, 1]$, and (b) within one-sigma from the average confidence: $\lambda_2 = [\mu_{C_L} - \sigma_{C_L}, \mu_{C_L} + \sigma_{C_L}]$.

In §V, we will show how each of these thresholds would improve the accuracy of the model by discarding majority of mispredictions.

A. Tracking Behavior of Connected IoTs Post-NAT

Our primary objective is to identify IoT devices connected to home networks by classifying IPFIX flow records, collected post-NAT. For ISPs to obtain additional insights into the behavior of classified devices in each home, they may want to track devices' network activity, ensuring their cyber health and/or detecting possible behavioral changes. We note that tracking the behavior of IoT devices post-NAT can be quite challenging as our unit of data (IPFIX flow record) does not carry the identity (*e.g.,* MAC or IP address) of their respective end-device. In the absence of a solid meta-data for associating the IPFIX records with devices, we again resort to the output of our inferencing model.

For each device type discovered per household, we quantify and track a metric, called "*Trust*". The trust metric is computed based on the number of predictions for that type during a unit of time (a configurable parameter, say, one hour) with respect to a class-specific baseline obtained from the training phase. For each class L in our model, we record the expected (average) number of IPFIX records (denoted by $N_{e,L}$) and the expected rate of discarded IPFIX records (denoted by $D_{e,L}$) during training. A raw measure of trust is computed by:

$$T_L = \frac{N_{o,L}}{N_{e,L} \times D_{e,L}} \tag{1}$$

where, $N_{o,L}$ is the number of "observed" flows predicted as class L – obviously, those predictions receive an accepted level of confidence from the model. The intuition behind this measure (a raw trust) is to check the number of expected flows from a device with the number of flows that are indeed classified as that device type during the monitoring period. Trust values closer to one imply that corresponding devices behave normally. However, raw trust can become greater than one when a device is over-active (*e.g.,* under attack), or its flows are mispredicted as another device class. Either of these cases (*i.e.,* having the raw trust value greater than one) need to be interpreted as a misbehavior. Therefore we need to normalize the raw trust in such a way that it decays in case of deviation from normal behavior. We define normalized trust (will be simply referred to as trust in the rest of this paper) by:

$$T_{norm,L} = \exp\left(-\frac{(T_L - 1)^2}{2\sigma^2_{T_L}}\right) \tag{2}$$

where σ_{T_L} is the standard-deviation of T_L, computed during the training phase. Use of σ_{T_L} allows for customizing the

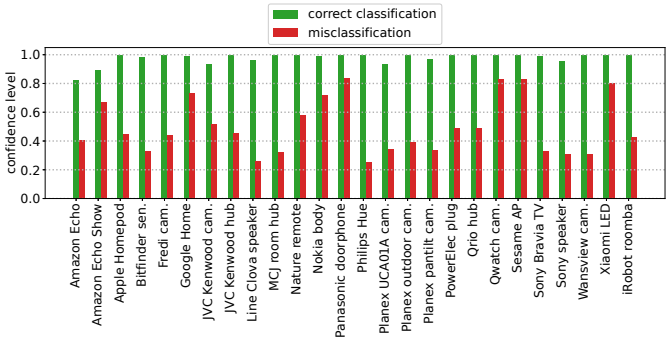

Fig. 4. Average confidence-level of correctly and incorrectly classified instances per device type for the training dataset.

change rate of trust across various device types – for a complex device like smart TV that displays a wide range of network activities the trust metric changes slower, while for a simple device like smart sensor it is will change relatively faster.

V. Evaluation Results

For evaluation, we split our data into two groups: training (1,450,215 IPFIX flows recorded in January and first half of March), and testing (1,274,786 IPFIX recorded in the second half of March and full April). We use Scikit-Learn Python library to train our Random Forest classifier.

A. Model Training and Tuning

We tune the parameters of our Random Forest model to maximize its performance for the chosen features (§III-A). We tune two important parameters, namely the number of decision trees, and the maximum number of attributes to consider at each branch split. For each combination of parameters, we quantify the model accuracy by 10-fold cross-validation, whereby the training dataset is randomly split into training (90% of total instances) and validation (the remaining 10% of total instances) sets, and the accuracy is averaged over 10 runs to produce a single performance metric. We compute the model's accuracy by averaging the rate of correct predictions (true-positives) across individual classes of our model. In other words, the average of all values across the main diagonal of the model's confusion matrix (will be discussed in Fig. 5). We found the best tuning parameters which yield the highest prediction accuracy (90%) to be 100 decision trees and a maximum of five features for finding the best split.

We next quantify the confidence of the model (on the training dataset) for correct classifications as well as incorrect classifications across various classes of device type, as shown in Fig. 4. Green bars indicate the average confidence for correctly predicted flows and red bars indicate the average confidence for mispredicted flows. A clear takeaway from this analysis is that the model is more confident when a flow is correctly classified and is relatively less confident when a flow is misclassified.

The second observation is that the two threshold methods (λ_1 and λ_2 we defined in §IV) affect our inferencing differently, as shown in Table III: λ_1 is found to be more conservative by discarding 18% of predictions, compared to

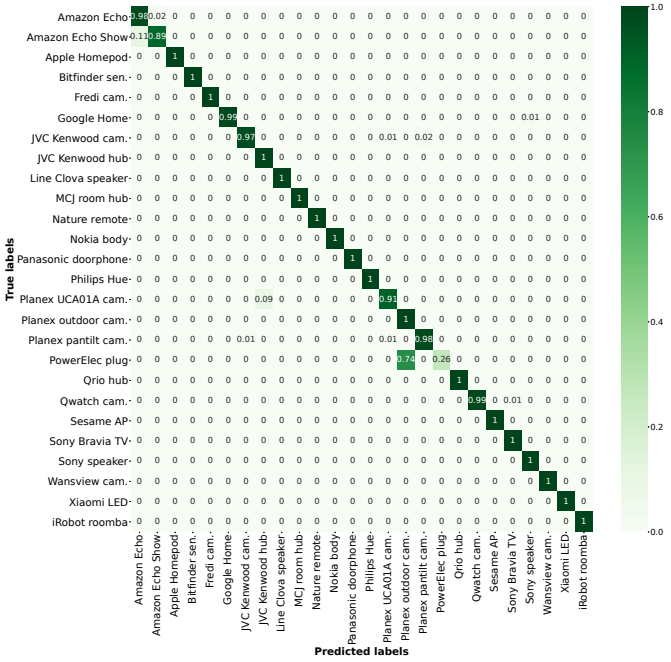

Fig. 5. Confusion matrix of accepted predictions – Random Forest model followed by applying class-specific confidence thresholds (λ_1).

TABLE III
Impact of λ_1 and λ_2 thresholds on training instances.

	Accepted		Discarded	
	Correct	**Incorrect**	**Correct**	**Incorrect**
λ_1	1,188,487	419	200,088	61,221
λ_2	1,323,119	9,451	65,456	52,189

8% discard rate resulted from applying λ_2; λ_1 is proven to discard more of incorrect predictions compared to λ_2 (99% versus 84%); considering the accepted predictions, λ_1 gives a slightly better accuracy than λ_2 (99.96% compared to 99.29%). Given the primary objective being the quality (purity) of predictions, we use the first method of thresholding (λ_1) for the testing phase of our evaluation.

B. Model Testing

Having tuned the parameters and obtained the class-specific confidence thresholds, we now evaluate the efficacy of the model by applying it to our testing dataset. Fig. 5 shows the confusion matrix of our inferencing (the model combined with λ_1 thresholds). The average accuracy across all classes is 96%, with 20 classes receiving more than 99% accuracy – the rate of correct predictions across accepted predictions. We found that applying the thresholds significantly improves the quality of model prediction. Note that the average accuracy of the model alone (without λ_1 thresholding) is 82%. Also, only 1% of the accepted instances are incorrectly predicted. Note that this quality of prediction is achieved at the cost of discarding a part of the correctly predicted flows (28% of correct predictions when the model is less confident than expected thresholds). Discard rate, on average, is about 16% per class.

Though our model gives an overall acceptable performance,

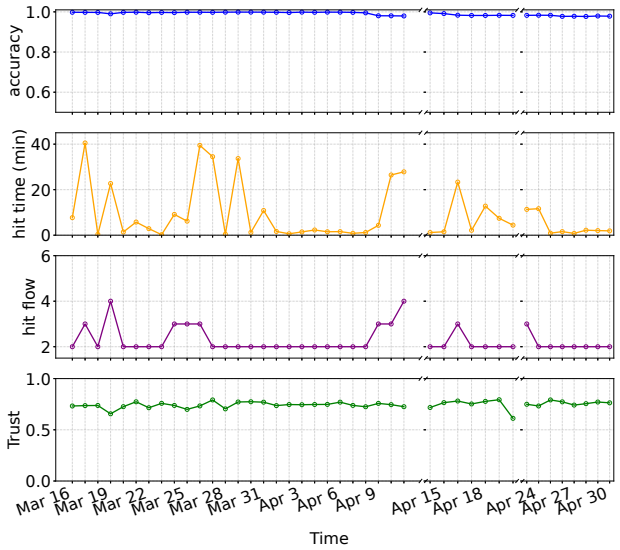

Fig. 6. Time-trace of average accuracy, trust, hit-time, and hit-flow of our inferencing across all classes on the testing dataset.

Fig. 7. Daily measure of Trust for four representative IoT devices on the testing dataset.

we find Power Elec plug performing not so well. We found that the model mispredicts most of Power Elec plug instances (relatively confidently) as Planex outdoor camera. Further investigations revealed that almost all of those misclassified flows correspond to `UDP/10001` service which is the second top service in IPFIX records of Planex outdoor camera (the mispredicted class). Analyzing the features of `UDP/10001` records from these two classes showed that the inter-arrival time and the number of outgoing packets in the flows of Power Elec plug significantly changed from the training to the testing phase, becoming similar to those of Planex outdoor camera flows, and hence resulted in the misprediction. Note that the ISP may choose re-train the model [11] after verifying certain legitimate changes, but re-training is beyond the scope of this paper.

Another takeaway from our evaluation is that the performance of our inferencing scheme varies across different protocols. The scheme yields a better accuracy (correctly predicted fraction of the accepted records) in `TCP` flows compared to that of `UDP` flows. Focusing on `TCP`, we see the highest true-positive rate in both `HTTPS` and "other" `TCP` flows with 99%, followed by `HTTP` flows with 94%. For `UDP` flows, `DNS` and "other" `UDP` services have 99%, followed by `NTP` with 75% true-positives. It is important to note that `NTP` flows receive the highest discard rate (99% of the `NTP` flows are predicted with fairly low confidence from the model). This is mainly because the vast majority of the `NTP` flows have the exact size of 76 bytes (including payload, `UDP` and `IP` headers) across all IoT classes in our dataset. We note that most of these IoT devices use the minimum packet structure defined by the `NTP` standard without any extension field, resulting in identical byte count feature for this specific type of flows.

C. Operational Insights

We track the temporal efficacy of our inferencing model in operation on a daily basis, as shown in Fig. 6. In addition to overall accuracy (top plot) which remains fairly high during our testing period, we capture two metrics: (a) *hit-time*, the time taken for our model to detect a device since its first-seen on the network, and *hit-flow* which indicates the number of flows generated by a device before the model correctly detects it. It can be seen from the second top plot in Fig. 6 that the average hit-time across all classes varies between 5 seconds and 40 minutes, highlighting the responsiveness of our inferencing scheme. Also, we observe that the average hit-flow mostly equals 2 (with a maximum of 4) – this suggests that discarding about a third of correctly classified flows does not incur an excessive delay in our inferencing.

Moving to the trust metric shown in the bottom plot of Fig. 6, we see consistently high values during the testing period on average (highlighting the overall health of devices). Only on 20th April, we observe a slight drop in the trust due to the early shutdown of our capturing server in preparation for a planned power outage on 21st and 22nd April – this resulted in just 3 hours worth of data captured on 20th April.

Dynamics of Trust Metric: To better understand the temporal dynamics of the Trust metric, we plot in Fig. 7 the daily Trust for a few representative classes which display noticeable changes in their behavior.

We categorize the behavioral changes by trust metric in two groups: (i) permanent declines which are probably caused by a legitimate firmware upgrade, leading to change in traffic features and/or the number of generated flows, and (ii) temporary declines that can be caused by an intermittent network condition or device usage/configuration for a short period of time, leading to generation of significantly different number of flows without necessarily affecting the traffic features. Fig. 7 illustrates the daily trust of four representative devices from our testbed: Amazon Echo Show (change in traffic features), and JVC Kenwood hub (change in the number of flows) represent a firmware upgrade scenario, while iRobot roomba and Sony Bravia TV represent temporary behavioral/usage changes affecting their flow count.

Amazon Echo Show has an average trust of 0.97 before April 9 on which it starts to decay and remains around 0.3 until the end of April. By further analysis, we found that the daily flow count was almost expected, but some significant changes were observed in certain traffic features (`reverseOctetTotalCount` and `reverseDataByteCount` dropped by about 30-40%). Another noteworthy change was in the `reverseAverageInterarrivalTime` feature that raised from 11 to 22 seconds after April 9. These changes caused the number of accepted flows to drop by 40%, compared to the expected value from the training phase.

JVC Kenwood hub starts with a very low trust value on 16^{th} and 17^{th} March, followed by a fluctuating trust measure averaged at 0.66 from 18^{th} March to 13^{th} April. Next, we observe a permanent drop, starting from 14^{th} April. JVC Kenwood hub generated 993 and 1019 flows during the first two days respectively (16^{th} and 17^{th} March) – in each day, about 900 flows received accepted prediction. Moving to the second half of April, the hub generated on average 220 flows per day, and around 170 of flows (daily) are predicted with an acceptable confidence. These values are far from the expected behavior of JVC Kenwood hub based on the training dataset where the expected number of flows is about 470 per day.

The iRobot Roomba displays a consistently high trust value over the testing period (about 0.94 on average) except for 20^{th} April. This device has an average of 50 flows per day over the testing period, except on 20^{th} April, it only had 6 flows. The shutdown of our capture server in preparation for a planned power outage on the following days caused this drop. As the expected flow count for iRobot Roomba is about 45 per day, the significant deviation on 20^{th} April resulted in a significantly low trust value for this device.

Sony Bravia TV shows a persistent trust measure throughout the testing phase except for April 14, when it generated a very high number of flows. On average, it generated $5,648$ flows per day during those days with a healthy trust metric. However, the flow count rose to $98,019$ on April 14 – of those flows, $\approx 44,000$ (about 3.5 times the expected baseline) received an accepted prediction.

VI. Conclusion

Residential networks continue to become richer and more vulnerable with the widespread adoption of consumer IoT devices. ISPs, therefore, recognize the need to deal with IoT security by obtaining visibility into these connected devices and their behavior. This paper discussed how an ISP can leverage IPFIX telemetry to achieve this task post-NAT, at scale, without making changes to home networks. We analyzed about three million IPFIX records (to be released as open data) from about thirty IoT devices to train a multi-class classifier model that infers connected IoTs in homes. We then improved the quality of our prediction by applying class-specific thresholds as well as developing a trust metric to track the health of the identified devices. Lastly, we showed that our method can classify the IPFIX flows of IoT devices with about 96% accuracy, and detect their behavioral changes.

References

[1] S. Grover *et al.*, "Peeking behind the NAT: An Empirical Study of Home Networks," in *Proc. ACM IMC*, Barcelona, Spain, Oct 2013.

[2] Y. Meidan, V. Sachidananda, H. Peng, R. Sagron, Y. Elovici, and A. Shabtai, "A Novel Approach For Detecting Vulnerable IoT Devices Connected Behind a Home NAT," *Computers & Security*, vol. 97, p. 101968, Oct 2020.

[3] Statista, "Number of Smart Homes forecast worldwide from 2017 to 2025," 2020. [Online]. Available: https://bit.ly/3ulFflm

[4] Palo Alto Networks, "Unit 42 IoT Threat Report," 2020. [Online]. Available: https://start.paloaltonetworks.com/unit-42-iot-threat-report

[5] D. Kumar *et al.*, "All Things Considered: An Analysis of IoT Devices on Home Networks," in *Proc. USENIX Security*, SC, USA, Aug 2019.

[6] A. Sivanathan *et al.*, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, Aug 2019.

[7] T. Micro, "Mirai Botnet Exploit Weaponized to Attack IoT Devices via CVE-2020-5902," 2020. [Online]. Available: https://bit.ly/2RRLy1X

[8] H. Kumar *et al.*, "Enhancing Security Management at Software-Defined Exchange Points," *IEEE TNSM*, vol. 16, no. 4, pp. 1479–1492, 2019.

[9] J. Blythe *et al.*, "What is security worth to consumers? Investigating willingness to pay for secure Internet of Things devices," *Crime Sci*, vol. 9, no. 1, pp. 1–9, Jan 2020.

[10] Cyber Edge, "Cyberthreat Defense Report," 2020. [Online]. Available: https://bit.ly/3xM1Idi

[11] A. Sivanathan, H. Habibi Gharakheili, and V. Sivaraman, "Detecting Behavioral Change of IoT Devices Using Clustering-Based Network Traffic Modeling," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7295–7309, Aug 2020.

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

[13] H. Guo and J. Heidemann, "IoTSTEED: Bot-side Defense to IoT-based DDoS Attacks (Extended)," USC/Information Sciences Institute, Tech. Rep. ISI-TR-738, Jun. 2020. [Online]. Available: https://bit.ly/3ec9eGS

[14] M. Miettinen *et al.*, "IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT," in *Proc. IEEE ICDCS*, Atlanta, USA, Jun 2017.

[15] V. Thangavelu *et al.*, "DEFT: A Distributed IoT Fingerprinting Technique," *IEEE IoT Journal*, vol. 6, no. 1, pp. 940–952, Feb 2019.

[16] S. Marchal *et al.*, "AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication," *IEEE JSAC*, vol. 37, no. 6, pp. 1402–1412, Jun 2019.

[17] S. J. Saidi *et al.*, "A Haystack Full of Needles: Scalable Detection of IoT Devices in the Wild," in *Proc. ACM IMC*, NY, USA, Oct 2020.

[18] A. Pashamokhtari, N. Okui, Y. Miyake, N. Masataka, and H. Habibi Gharakheili, "IoT IPFIX Records," 2021. [Online]. Available: https://iotanalytics.unsw.edu.au/iotipfixrecords

[19] H. Habibi Gharakheili and V. Sivaraman, "Cloud assisted home networks," in *Proc. ACM CAN*, Incheon, Republic of Korea, Dec 2017.

[20] R. Hofstede *et al.*, "Towards Real-time Intrusion Detection for NetFlow and IPFIX," in *Proc. CNSM*, Zurich, Switzerland, Oct 2013.

[21] J. J. Davis, "Machine Learning and Feature Engineering for Computer Network Security," Ph.D. dissertation, QUT, QLD, Australia, 2017. [Online]. Available: https://eprints.qut.edu.au/106914/1/Jonathan_Davis_Thesis.pdf

[22] P. Aitken, B. Claise, and B. Trammell, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," RFC 7011, Sep. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc7011.txt

[23] NetSA CERT, "YAF," 2006. [Online]. Available: https://tools.netsa.cert.org/yaf/yaf.html

[24] ——, "Super Mediator," 2012. [Online]. Available: https://tools.netsa.cert.org/super_mediator

[25] R. Moskowitz, D. Karrenberg, Y. Rekhter, E. Lear, and G. J. de Groot, "Address Allocation for Private Internets," RFC 1918, Feb. 1996. [Online]. Available: https://rfc-editor.org/rfc/rfc1918.txt

[26] IETF, "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)," 2008. [Online]. Available: https://tools.ietf.org/html/rfc5103

[27] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine Learning for Internet of Things Data Analysis: a Survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, Aug 2018.