BehavIoT: Measuring Smart Home IoT Behavior Using Network-Inferred Behavior Models

Tianrui Hu
Northeastern University
Boston, Massachusetts, USA
hu.tian@northeastern.edu

Daniel J. Dubois
Northeastern University
Boston, Massachusetts, USA
d.dubois@northeastern.edu

David Choffnes
Northeastern University
Boston, Massachusetts, USA
choffnes@ccs.neu.edu

ABSTRACT

Smart home IoT platforms are typically closed systems, meaning that there is poor visibility into device behavior. Understanding device behavior is important not only for determining whether devices are functioning as expected, but also can reveal implications for privacy (*e.g.,* surreptitious audio/video recording), security (*e.g.,* device compromise), and safety (*e.g.,* denial of service on a baby monitor). While there has been some work on identifying devices and a handful of activities, an open question is what is the extent to which we can automatically *model* the entire behavior of an IoT deployment, and how it *changes* over time, without any privileged access to IoT devices or platform messages.

In this work, we demonstrate that the vast majority of IoT behavior can indeed be modeled, using a novel multi-dimensional approach that relies only on the (often encrypted) network traffic exchanged by IoT devices. Our key insight is that IoT behavior (including cross-device interactions) can often be captured using relatively simple models such as timers (for periodic behavior) and probabilistic state-machines (for user-initiated behavior and devices interactions) during a limited observation phase. We then propose deviation metrics that can identify when the behavior of an IoT device or an IoT system changes over time. Our models and metrics successfully identify several notable changes in our IoT deployment, including a camera that changed locations, network outages that impact connectivity, and device malfunctions.

CCS CONCEPTS

• **Networks → Home networks**; **Network monitoring**; • **Security and privacy**;

KEYWORDS

Smart Home, IoT, Measurement Techniques, Behavior Modeling

ACM Reference Format:
Tianrui Hu, Daniel J. Dubois, and David Choffnes. 2023. BehavIoT: Measuring Smart Home IoT Behavior Using Network-Inferred Behavior Models. In *Proceedings of the 2023 ACM Internet Measurement Conference (IMC '23), October 24–26, 2023, Montreal, QC, Canada.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3618257.3624829

1 INTRODUCTION

Smart home Internet of Things (IoT) systems are closed systems, meaning that we know little about whether a device (or set of devices) is behaving in ways that might violate expectations such as privacy, security, and correctness. Prior work has shown that these devices can indeed pose significant security [14, 15, 34, 47, 60, 75, 77], privacy [19, 29, 40, 49, 59], and safety [20, 21, 27, 28, 74] risks. Such diverse types of undesired behavior are difficult to detect and mitigate because we have a poor understanding of what is normal device behavior and how it changes over time.

A key question, then, is whether it is possible to accurately model the behavior of IoT systems, and use these models to identify deviations of interest as the behavior changes over time. Specifically, we focus on whether we can develop models that (*i*) do not require the modification of devices or privileged access to closed APIs; (*ii*) can abstract the behavior of an individual device (*device behavior*), as well as the behavior of a system composed of many IoT devices (*system behavior*); (*iii*) can account for the changes of such behavior models, *i.e.,* by providing metrics to quantify model deviations (*deviation metrics*). Several prior efforts focus on pieces of this solution [13, 21, 25, 26, 28, 33, 41, 42, 49, 53, 59, 67, 72–74], but fall short of addressing it entirely.

In this paper, we propose a measurement approach that addresses the above issues by modeling the per-device and system-wide behavior of an IoT system using network-inferred behavior models (from now, simply *behavior models*), *i.e.,* models that are built using information inferred from the IP traffic produced by an IoT system. Because our approach relies only on network traffic, it is inherently platform-agnostic, requires no privileged access to devices or APIs, and is easy to deploy at routers or gateways. Our behavior models rely on the observation that most consumer devices are relatively simple, having a limited set of functions and states [33, 35, 48, 53, 59, 74], and their network traffic, although mostly encrypted, typically exhibits predictable patterns that are either periodic or that correlate with the actual functions being used [13, 50–53, 67, 76].

More specifically, we first create *device behavior models* by inferring from *all* IP network traffic collected during an observation period a list of events generated by individual IoT devices (*e.g.,* toggling a switch, periodic maintenance tasks, heartbeat, etc.). One key insight is that such events can be accurately modeled by separating them into user events, periodic events, and aperiodic events. In real IoT deployments, however, individual device behavior does not tell the whole story because many devices interact with each other. To model these interactions, we observe that we can model a collection of cross-device interaction events as a probabilistic state-machine, which serves as our *system behavior model*. Finally, we measure the

IMC '23, October 24–26, 2023, Montreal, QC, Canada

ACM ISBN 979-8-4007-0382-9/23/10…$15.00
https://doi.org/10.1145/3618257.3624829

behavior deviation by defining several *deviation metrics* to quantify under multiple dimensions how much IoT events are consistent with generated device and system behaviors models over time.

We evaluate our approach on several real datasets produced in an IoT testbed (comprising 49 devices) where we run controlled and uncontrolled experiments. Specifically, we identify testbed device and system behavior models during an observation period, then analyze the changes in behavior using our deviation metrics first on a synthetic dataset, where we perturb the traffic of our IoT testbed, and then on real traffic produced as part of a user study where 40 study participants are allowed to use the devices as they see fit (data collected with consent and IRB approval, see §3). Based on the user study data, we identified several IoT behavior deviations due to user activity (user participants relocating devices), service outages, and device malfunctions.

We believe that the ability to model IoT behavior and measure behavior deviations at different levels of abstraction can help IoT safeguards (*e.g.,* [6, 7, 11]) identify situations were IoT devices behave unexpectedly, can assist with developing profiles of IoT systems to fill the gap left by the lack of deployed MUD profiles [44], and can help audit IoT system behavior with respect to regulatory and privacy policy compliance.

Summarizing, our main contributions are:

(1) New datasets from controlled experiments using 49 devices, and from IRB-approved uncontrolled experiments involving 40 participants for three months (§3).
(2) A platform-independent measurement approach for inferring device behavior models, system behavior models, and behavior deviations from network traffic (§4).
(3) An evaluation of our approach and a measurement study of IoT behavior in our new datasets (§5 and §6).

To facilitate follow-up research, we publicly released our dataset from controlled experiments and software [38].

2 GOALS, ASSUMPTIONS, AND SCOPE

Goals. The goal of this paper is to answer the following research questions in the context of a smart home IoT system (*i.e.,* a set of consumer IoT devices in the same smart home):

RQ1. Can we measure and characterize the behavior of an IoT system from (even encrypted) network traffic? Previous work explored the problem of modeling per-device IoT behavior from network traffic in terms of user events as those associated with user activities (*e.g.,* turning on a light) [13, 35, 53, 59, 67, 72], typically to assess privacy threats where an attacker tries to profile IoT user activity; however, their models only cover a minority of all IoT traffic—in fact, we observe that the vast majority of IoT traffic (overall average 98.5%, median 98.631%) is due to non-user background activities (*e.g.,* heartbeats) in our testbed. Having a way to infer non-user events can enrich device behavior models significantly since such events occur *constantly and independently* of user activity. The challenge in inferring non-user events is the lack of ground truth, which makes existing user event inference approaches unsuitable for the task. We address this challenge using an unsupervised approach to infer a model of periodic behavior for classifying periodic events.

To model the system IoT behavior, we rely on the insight that a user interaction with the IoT system may cause correlated sequences of user events from different IoT devices (*e.g.,* a user triggering multiple sensors at once every time they enter a room). Based on this insight, we use automatic model inference algorithms on sequences of user events to abstract such correlations, thus generating a system behavior model that is more abstract and amenable for analysis than the raw event sequence itself.

RQ2. Can we measure and characterize behavior deviations of an IoT system? After inferring the behavior models of the IoT system, we investigate how to measure their deviations, defined as changes in such models. Having a way to measure behavior changes can help identify any new or unexpected behavior of the IoT system (*e.g.,* due to firmware upgrades, to changes in user utilization patterns, to service outages, or even malicious IoT behavior). The main challenge in doing this is that our IoT behavior models are multidimensional, in the sense that they model several characteristics of the IoT systems (*e.g.,* periodicity of device behavior and different temporal aspects of system behavior) so it is not obvious *a priori* how to capture different changes in behavior that might occur. To address this, we first propose several deviation metrics, each one capturing one different aspect of the IoT system behavior, that quantify the amount of behavior change between the behavior model and a new sequence of events inferred from the network traffic; then we show on our datasets how we can use our deviation metrics to identify notable cases of behavior deviation.

Assumptions and scope. To answer our research questions, we assume we can observe (but not decrypt) the IP traffic that traverses the gateway serving the IoT devices. We do not consider traffic that does not traverse the gateway (*e.g.,* Thread and non-IP traffic such as Zigbee). We also do not consider on-device apps as the nature of apps traffic is different from that of IoT devices with no apps and already considered in previous work [69]. For devices offering apps, we either include them without using any on-device apps, or we exclude them, in case they offer no significant functionality without relying on apps (*e.g.,* smart TVs).

3 DATA COLLECTION

This section presents our IoT testbed and the datasets produced by our experiments (available for download [38]).

3.1 Testbed

To collect data for this measurement study, we build a testbed that consists of 49 IoT devices that we deployed in a lab that resembles a studio apartment. We connected the devices in the testbed to the Internet via a gateway that captures all network traffic, separated by device. We do not perform any device modification or traffic decryption in this work. We selected the devices (listed in Table 1) from a wide range of categories that were deemed popular according to Amazon search results at the time of purchase.

3.2 Dataset from Controlled Experiments

We conduct controlled experiments from 8/2021 to 10/2021.

Activity dataset for user event behaviors (30 devices). We use this set of experiments and their data to obtain ground-truth labels for inferring *user-action models* from the network traffic, *i.e.,*

Category	Camera (11)	Smart Speaker (11)	Home Automation & Sensor (16)	Appliance (5)	Hub (6)
Devices (49 total)	D-Link Camera iCSee Doorbell LeFun Cam Microseven Camera Ring Camera Ring Doorbell Tuya Camera Ubell Doorbell Wansview Cam Wyze Cam Yi Camera	Echo Dot Echo Dot3 Echo Dot4 Echo Flex Echo Plus Echo Show5 Echo Spot Google Home Mini Google Nest Mini Homepod Mini Homepod	Amazon Plug D-Link Sensor Govee Bulb Meross Dooropener Nest Thermostat Smartlife Bulb TPLink Bulb Keyco Air Sensor Jinvoo Bulb Gosund Bulb Magichome Strip Philips Bulb Ring Chime Wemo Plug TPLink Plug Thermopro Sensor	Behmor Brewer Samsung Fridge Smarter iKettle GE Microwave Anova Sousvide	Aqara Hub IKEA Hub SmartThings Hub SwitchBot Hub Philips Hub Wink Hub2
Interactions	Move in front of camera, watch remotely, record video, take picture, voice intercom, ring	Voice command, change volume, turn on/off	Turn on/off, change brightness/color, set modes, move in front of sensor	Turn on/off	Turn on/off

Table 1: IoT devices under test per category and the interaction experiments we performed (if available).

models for identifying *user events* in network traffic. Similar to prior work [49, 59, 67], we conduct experiments where we interact with devices in our testbed. Most interaction experiments that involve the use of a companion app or a voice command are automated and repeated at least 30 times for each activity. Based on prior work [49], we use screenshot-based validation to determine if interactions were successfully executed. We assign a unique label to each interaction (*e.g.,* "lightbulb on").

Idle dataset for non-user event behaviors (49 devices). As we discuss later, the vast majority of network traffic in our experiments is unrelated to user actions. To model this behavior, we require a dataset where we are certain that no user interactions occur. To meet this need, we run "idle" experiments to capture the traffic of an IoT device when it is isolated from any interactions. The idle dataset contains 5 consecutive days of network traffic from 49 devices.

Routine dataset (18 devices). To simulate a real smart home environment that includes a mix of idle periods and user actions, and to measure system IoT behaviors (across multiple devices), we conduct one-week-long controlled experiments that involve multi-device interactions. Inspired by the popularity of automation platforms such as Alexa [5], SmartThings [12], and IFTTT [8], we use trigger-actions routines/automations to execute a sequence of user events (action), in response to another event (trigger). For this dataset, we considered a subset of 18 devices that supported trigger-actions in our testbed. We consider the size of our subset sufficiently representative since the average number of IoT devices per home is 7 according to crowdsourced data from IoT Inspector [39].

To generate the dataset, we manually created automations that involve various types of activities and/or devices, using both the Alexa and IFTTT platforms. The functionalities of the included devices encompass on/off, color, and dimming features for smart bulbs; turning smart plugs on or off, door opening, hub interactions, thermostat changes, and smart kettle on/off; motion, video, and ring functions for smart cameras/doorbells; and voice control for smart speakers. These automations draw from: (1) popular routines [5], automations [12], and applets [8] found online using the popular automations in each platform's corresponding online marketplace (*e.g.,* "if doorbell rings, blink the light"); (2) knowledge about how the devices are typically used [48] (*e.g.,* "turn on all lights using voice assistant"); and (3) similar automations created by prior work [25, 33] (*e.g.,* "if motion detected, turn on the light"). Appendix A provides the full list of devices and automations we use. In addition to pre-programmed automations, we also directly trigger device functionality by voice commands and companion apps during the experiment, to better simulate a real smart home environment with arbitrary user behavior.

3.3 Dataset from Uncontrolled Experiments

To measure the changes of device behavior over time, we conduct uncontrolled experiments from 12/2021 to 2/2022.

Uncontrolled dataset (47 devices). For 87 days, 40 participants were allowed to use the testbed as they saw fit. By entering the testbed facility, they triggered our pre-generated routines and other aspects of device functionality—either intentionally or unintentionally. We did not conduct any controlled experiments during this three-month data collection period. Note that due to various reasons, not all devices stayed online during the entire three months, a topic we will revisit when we analyze behavior deviations later in this paper.

Ethics. The testbed is part of an IRB-approved user study where consenting participants use the space and the IoT devices as they see fit, commonly entering the room to use the fridge, microwave, *etc.* When we conduct controlled experiments to produce labeled traffic traces, participants are not allowed in the room. Per the terms of our approved research protocol, we do not share any data from user interactions with unauthorized individuals.

4 BEHAVIOR MODELING

Our behavior modeling approach relies on three steps (Fig. 1): (*i*) the *device behavior inference step* (§4.1) analyzes the network traffic to identify and model events generated by IoT devices (*e.g.,* "turning on a light" or periodic "heartbeats"); (*ii*) the *system behavior inference step* (§4.2) combines events from IoT devices in an IoT system into a unified model that captures emergent behavior of the whole system; (*iii*) the *behavior deviation inference step* (§4.3) uses deviation metrics to identify significant changes in behavior compared to previously generated device and system models.

4.1 Modeling Device Behavior

We now describe how we use measurements to build device behavior models, *i.e.,* models that capture the behavior of a single IoT device. We consider the following two device behavior models: (*i*)

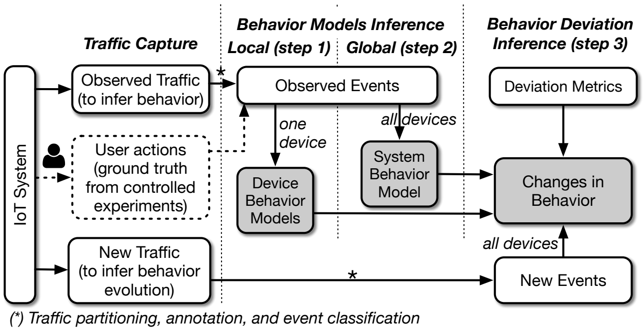

Traffic Capture
Behavior Models Inference
Local (step 1)
Global (step 2)
Behavior Deviation Inference (step 3)
IoT System
Observed Traffic (to infer behavior)
User actions (ground truth from controlled experiments)
New Traffic (to infer behavior evolution)
Observed Events
one device
all devices
Device Behavior Models
System Behavior Model
Deviation Metrics
Changes in Behavior
all devices
New Events
*
(*) Traffic partitioning, annotation, and event classification

Figure 1: *Overview of our approach.* The diagram shows the three steps of our approach, with gray boxes representing the output of each step.

periodic models, which capture periodic behavior (*e.g.,* keep-alive heartbeat), and (*ii*) user-action models, which capture behavior due to user actions or triggers (*e.g.,* turning on a light).

To infer device behavior models we first partition (typically encrypted [59]) network traffic into discrete labeled events using ground-truth information. Specifically, network traffic generated by an IoT device will be labeled as either user events (caused by user actions or triggers), periodic events (caused by non-user periodic traffic), or aperiodic events (the remaining cases). We then use the periods of periodic events to inform *periodic models* and machine learning classifiers to inform *user-action models.*

Prior approaches [13, 53, 58, 59, 67, 72] already offer user-event classification and a certain degree of background-traffic recognition, but they offer neither a suitable approach for identifying non-user events, nor a way to disjointly partition the network traffic into user and non-user events, which is needed to inform our device behavior models. To fill this gap, we first introduce a novel way to infer discrete non-user events and to separate them from user events. Then, we show how we can use such events to build periodic and user-action behavior models.

Traffic partitioning and annotation. Given a sequence of network packets transmitted to/from an IoT device, we first assemble the packets into *flows* and *bursts* (analogous to related work on encrypted traffic for IoT [53, 76], and mobile security [66, 68]). We define a flow as a chronologically ordered set of TCP segments/UDP datagrams with the same 5-tuple (source IP, source port, destination IP, destination port, transport protocol). Because flows can last hours or even days, we may need to divide flows into smaller chunks of data for the purpose of event extraction. Like prior work, we do so using the notion of bursts. A flow burst is a consecutive chunk of packets from the same flow in which the interval between any two consecutive packets is less than a threshold, which we set at 1 second, as suggested by prior work [66, 76]. *We refer to* ***flow burst*** *as* ***flow*** *in the rest of the paper for simplicity.*

After identifying flows, we annotate them with the start time, duration, protocol, destination and other 21 features related to packet and flow sizes, and timings (see Table 8 in Appendix B for the full list of features) for the purpose of event inference. Note that our techniques do not rely on the contents of packet payloads, and as such we make no attempt to decrypt any traffic. In addition to limiting the potential privacy concerns for such traffic analysis, this approach also means that the analysis can be conducted at the Internet gateway (*e.g.,* home router). We derive all the annotations directly from the headers and timing of each flow, except for the destination domain name. We extract the domain name using DNS responses and/or TLS handshakes (via the Server Name Indication field). If these methods do not reveal a name for the destination IP in a flow (*e.g.,* when DNS/SNI information is encrypted or appeared outside of our data collection period), we rely on reverse DNS lookups [9]. If none of the above approaches yields a domain name, we leave the IP's domain name blank.

Inferring periodic models. Based on our analysis of network traffic from IoT devices during idle periods (*i.e.,* when the devices are not used), we find that most flows with the same destination domain name and protocol appear at regular time intervals. To capture this behavior, we use periodic models, *i.e.,* models that can capture the traffic patterns of periodic events.

Specifically, we separate traffic for each unique (destination domain, protocol) tuple into different traffic groups, then check if the traffic in each of the groups has periodicity, using an unsupervised approach that combines Discrete Fourier Transformation (DFT) and autocorrelation [36, 46, 71]. First, we use DFT to extract candidate periods for a group by identifying the frequencies that carry significant power in spectral density. Then, we use autocorrelation to validate the candidate periods and identify the most likely period for each pattern. The periods that have a significant autocorrelation score are chosen as the final periods of the signal. We finally define a traffic group with periodicity as our *periodic model.*

Once we have inferred our periodic models, we need a way to use them for classification purposes to recognize if future unlabeled traffic flows are periodic events. The simplest way to achieve this would be using a timer-based approach to label flows with inferred periods. However, we found that many non-deterministic factors such as network congestion substantially reduce the accuracy of this approach. To address this, we rely on the observation that periodic traffic patterns are relatively static, so unsupervised clustering is amenable to labeling such traffic: first, we use a timer to label periodic traffic with clearly identifiable periods, then, for the remainder of the traffic, we use DBSCAN [30], which does not require specifying the number of clusters (something that is unknown *a priori*), to group each periodic flow into the clusters trained using idle traffic. Traffic flows belonging to any clusters are labeled as periodic events.[1]

Inferring user-action models. To infer user-action models, *i.e.,* models that capture the traffic patterns of user events, we use an approach similar to prior work [13, 53, 59], *i.e.,* we use controlled experiments as ground truth to train supervised Random Forest classifiers [18], which serve as our user-action models. Once trained, these user-action models can classify future unlabeled flows related to user activities. As the focus and novelty of this paper is not on user event classification, we provide details only in Appendix B.

[1] We confirmed, via manual analysis from our controlled experiments, that these observed periodic events are always unrelated to user events.

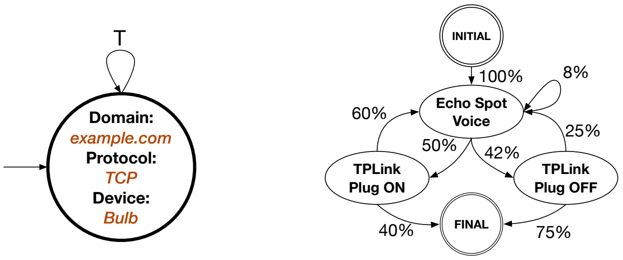


Figure 2: Behavior Models. The periodic model on the left represents the periodic TCP traffic to 'example.com' from device 'Bulb' with period T. The PFSM on the right captures the relationships among three events (states), where edges represent transition probabilities.

4.2 Modeling System Behavior

We now consider the question of how to capture the behavior of a system with multiple devices that can interact with each other. Our key insight is that system-wide behavior can be modeled as a finite state machine, where sequences of user events correspond to traversals of the state machine. Based on this, we propose an approach that (*i*) combines temporally correlated user events into *user event traces*, and (*ii*) repurposes a model inference algorithm to transform a set of user event traces into a probabilistic finite state machine (PFSM) that models the process generating the traces.

Inferring user event traces. To abstract the system behavior, we need to consider multiple user events from different IoT devices and their timing information. Specifically, we first use user-action models to extract event sequences from the network traffic, and then split such user events sequences into temporally correlated *event traces*. If the time interval of any two consecutive user events in a sequence is larger than a certain threshold,[2] we partition them into different event traces. Each of these newly created event traces can be considered as individual "logs" of events in the IoT system.

From event traces to PFSM. A naive approach to modeling system behavior is to combine event traces into a directed acyclic sequence graph of events. However, this approach results in large event-sequence models that provide little insight into system invariants (*e.g.,* a motion sensor always turning on a light)—much like the difference between a program containing multiple repetitions of the same statement, and one that uses a concise `for` loop that produces the same output. Our goal is to compactly model such invariants from user-event sequences. To do so, we rely on the observation that IoT systems behave like finite state machines (FSMs), and these FSMs can be automatically generated by a tool originally designed for distributed system debugging (namely, Synoptic [17]). Specifically, Synoptic is a tool that automatically generates PFSM models from a set of system execution traces.

Given user event traces as input, Synoptic outputs a PFSM representing our system behavior model, where each state models a user activity and the transitions model the probability that one activity leads to another activity. The probability of a transition from an activity state to another in a PFSM depends only on the current state. Using transitions, we can capture the temporal and causal relations between states and their likelihood of appearance in event traces. In the simple example in Fig. 2, we show how a PFSM is able to capture key relationships between events: in this example, it is easy to see that the TP-Link plug is triggered only by the Echo Spot and the plug's "On" event is triggered more often than "Off."

Note that Synoptic is not the only tool for generating models from log files: we have chosen it due to its scalability with respect to trace sizes and the presence of probabilistic information. Like any modeling system, Synoptic has limitations. For example, it cannot model states that are not in the logs, its invariant inferences may not be perfect, and it cannot model algebraic or logical relationships. Despite these limitations, we empirically found Synoptic to be sufficient for our study and leave the evaluation of other models to future work.

Leveraging probabilistic information of a PFSM. Both our PFSM behavior model and a deterministic behavior model only consisting of raw user event traces (the naive approach above) can recognize deterministic lists of events (*i.e.,* same events always in the same order) as instances of the same system behavior. However, in reality, IoT systems are typically non-deterministic, especially as the number of IoT devices and types of user events grow. The reason is that different user events from different devices may happen simultaneously or in a slightly different order, and some sequences of user events may happen more frequently than others.

Our PFSM models account for this by using transition probabilities and by combining similar states together, allowing the PFSM to model sequences of events that have never been encountered before, which in turn can account for some amount of (inconsequential) non-determinism in an IoT system. A more detailed explanation of why our models exhibit this desirable property is explained in [17], where the authors show that Synoptic PFSM models are generative: they may accept traces not present in the log. Finally, the PFSM's transition probabilities can inform the likelihood that an observed user event trace corresponds to the behavior modeled by the PFSM: we leverage this to build system deviation metrics, discussed in the next section.

4.3 Measuring Behavior Deviations

Once we have models that capture behavior in an IoT system, a key question is how do these behaviors change over time? To answer this question, we use our behavior models as a baseline to measure the extent to which new events produced by the IoT system *deviate* from the behavior observed in the past, *i.e.,* during the observation period in which we establish its behavior models. Specifically, we use one per-device deviation metric, based on the periodic model, and two system-wide deviation metrics based on the PFSM.

For the ***periodic-event deviation metric***, we look for cases where the observed traffic patterns have different timings than expected based on the modeled period. Periodic events that strictly follow their periods have, by definition, zero deviation. For periodic events that do not follow their periods, we use count-up timers to get the elapsed time T_0 (time difference between the current time and the last time the event occurred).

Formally, we define the *periodic-event deviation metric* $M_p \in [0, +\infty)$ as $M_p = \log(\frac{|T_0 - T|}{T} + 1)$, which measures the difference

[2] 1 minute in our approach; chosen empirically (as in prior work [33, 66, 76]) to provide a good trade-off between the number of traces and trace size.

between the elapsed time T_0 and the inferred period T, normalized by the period T, for periodic traffic group p. We use the log transformation to mitigate skewed scores for easier interpretation.

For the ***short-term deviation metric***, we leverage the probabilistic information of the PFSM to quantify the similarity of new user event traces with respect to the ones seen in the past. Intuitively, this metric assigns large values to traces that lead to non-existent states (*i.e.,* new events) or to states reachable via low-probability transitions (*i.e.,* unlikely event sequences). Because this metric is defined over individual traces and those traces have limited duration, it identifies short-term deviations in event sequences.

Formally, we define this metric (for a trace T) as $A_T = 1 - \log(P_T)$, where $A_T \in [1, +\infty]$ and $P_T \in [0, 1]$ is the probability that the trace is consistent with the PFSM. We use the log transformation to account for the wide range of small probability values in P_T, while we use the negative inverse to have a score that is 1 when there is no deviation. We calculate P_T by traversing a path corresponding to the trace in the PFSM (a path is a route from s_0 to s_f) and multiplying each transition probability along the path: $P_T = P(s_0) \prod_{i=1}^{n} P(s_n | s_{n-1})$.[3]

For the ***long-term deviation metric***, we also leverage the state-transition probabilities in the PFSM, but here we consider the compound effect of multiple traces instead of just one. This metric captures long-term deviations from the PFSM in terms of event-transition probabilities, including cases like a smart speaker constantly misactivating and recording [1]—a normal state for the device (and thus not detected by other metrics), but is an important deviation because its transition frequency is higher than normal.

Formally, we define this metric $Z \in [0, +\infty)$ as $Z = |z|$, where z is the z-score that measures how far a sample data is from the mean. We use the absolute value to ensure a zero lower bound and a score that becomes larger as the deviation increases. This metric can be interpreted as the statistical significance of the deviation from the previously observed transition probability in a snapshot. Given a transition from s_i to s_j, with transition probability $Ps_j | s_i = p$ having Binomial distribution, we compute the z-score as $z = \frac{p - p_0}{\sqrt{p_0(1-p_0)/n}}$, where n is the number of occurrences of s_i and p_0 denotes the transition probability over n transitions from s_i to s_j.

5 BEHAVIOR MODELS EVALUATION

In this section we evaluate our approach for modeling device behavior (§5.1) and system behavior (§5.2) (RQ1), and for determining behavior deviations (§5.3) (RQ2).

5.1 Device Behavior Modeling

We now evaluate our methodology for inferring periodic and user-action behavior models, demonstrating how we accurately identify the corresponding events.

Periodic models. To infer periodic models we use a method based on discrete Fourier transform (DFT) and autocorrelation. We evaluate its accuracy by generating 100 periodic sequences with varying periods and 100 aperiodic sequences by applying random permutations to these periodic sequences. Additionally, we generate 100 periodic sequences with noise by combining generated periodic sequences and aperiodic sequences. In all of these cases, our approach correctly infers periods and accurately classifies sequences as aperiodic when appropriate 100% of the time, demonstrating the effectiveness of our method.

	Home Auto	**Camera**	**Smart Speakers**	**Hub**	**Appliance**	**Total**
Periodic Coverage	99.9%	99.9%	99.7%	99.4%	99.9%	99.8%
Periodic Event Acc.	99.9%	99.9%	99.7%	98.0%	99.6%	99.2%
User Event Acc.	99.1%	98.9%	96.5%	100.0%	100.0%	98.9%
Aperiodic %	0.16%	0.05%	1.55%	1.95%	0.04%	0.52%

Table 2: Event inference per IoT device category. *Periodic Coverage* indicates how many flows exhibit periodicity in idle dataset. *Periodic Event Acc.* and *User Event Acc.* shows how correctly periodic events and user events are inferred. *Aperiodic ratio* indicates the portion of aperiodic flows in idle and activity datasets.

We next evaluate how many non-user flows can be modeled as periodic in the idle dataset (containing only background traffic). While we have no ground truth about the purpose of background traffic due to the closed nature of IoT systems, our hypothesis is that most background IoT traffic is related to periodic activities (*e.g.,* heartbeats, status synchronization, *etc.*). After running our approach, we find that 99.8% (Table 2) of the flows in the idle dataset exhibits periodicity, lending support to our hypothesis. Finally, we evaluate the ability of our periodic models to identify flows as periodic events. We split our idle dataset into training and testing set, and train our inference model on the traffic that exhibits periodicity in the training set. After testing on the idle dataset, we find that our approach identifies more than 99.2% of inferred periodic flows as periodic events (Table 2). The remaining 0.8% of cases are primarily due to variations in background traffic caused by non-deterministic factors such as network congestion, or by imperfect classifiers.

User-action models. We use machine-learning classifiers to model user actions, and build our own models instead of using existing ones [13, 53, 67] due to their limitations; namely, lack of support for UDP (12.8% of idle flows and 48.4% of activity flows are UDP in our dataset), limited device support, and no ability to classify non-user traffic (see §4.1).

We find that our user-action models (*i.e.,* our user event classifiers) meet or exceed the accuracy of a recent state-of-the-art *user event* classifier, PingPong [67] (a signature-based IoT user event classification method). For the six overlapping devices between our two studies, we have identical (100%) accuracy for half and better accuracy for the other half, as shown in Table 3. Extending beyond the PingPong dataset, we measure an overall 98.9% user-action model classification accuracy across 30 different devices in the activity dataset (Table 2).

Furthermore, we evaluate our event inference using the false negative rate (FNR) and false positive rate (FPR) metrics.

[3] If a trace includes a new transition with $P = 0$, the P_T would be zero. However, due to training data limitations (*e.g.,* rare events missing from the training set), the zero score may be too sensitive for discovering behavior changes. We address this by additive smoothing [63].

Device Name	*BehavIoT* accuracy	PingPong accuracy
Amazon Plug	100%	98%
Wemo Plug	100%	100%
TP-Link Bulb	96.15%	83.3%
TP-Link Plug	100%	100%
Nest Thermostat	94.74%	93%
Smartlife Bulb	100%	100%

Table 3: User event classification accuracy comparison between *BehavIoT* and PingPong [67].

False negative rate. We define FNR as the ratio of false negative user events (*i.e.,* user events incorrectly classified as non-user events) to the total number of user events in the activity dataset. With respect to FNR, our event inference approach performs quite well: 19 out of 30 devices have zero false negatives. The FNR for the remaining eleven devices is 5.84%. The SmartThings Hub is responsible for all of the false negatives in the Hub category. It has a high FNR (71.88%) because the events generated by its low-bandwidth user activities (turning on/off all connected Zigbee devices) are often indistinguishable from background events that share the same TCP connection. Note that we did not trigger other activities in our controlled experiments on the SmartThings Hub, since the SmartThings platform is not our primary focus. For other cases of false negatives, we could not find a precise explanation, but we speculate that either the device or its companion app did not work properly during the automated activity experiments, similar to observations from our previous work [49].

False positive rate. We compute the false positive rate (FPR) as the number of false positive events (*i.e.,* events in the idle dataset that are misclassified as user events), divided by the total number of events in the idle dataset. We find that only 0.09% of the events in the idle dataset are misclassified as user events. Of those, nearly 80% are caused by Echo Show 5, which has many flows in its idle dataset that exhibit traffic patterns that are similar to user events.

5.2 System Behavior Modeling

In this section, we evaluate how well our system modeling approach (*i.e.,* the PFSM model) captures system properties. We first evaluate the scalability of our model, and then evaluate other properties reported in previous work.

PFSM model scalability. One advantage to the PFSM approach is that it can relatively compactly represent the behavior of an IoT system, making it easier to interpret and analyze. The reason for the compactness is that similar states in the PFSM are collapsed into a single state, while they would stay distinct in state-transition chains. An alternative simple model-building strategy is to combine all the traces as parallel event sequences. To evaluate which approach to use, we turn to Fig. 3, which shows how the number of nodes and edges grows as new devices are added. The figure shows that the PFSM abstracts a compact and scalable model with nodes and edges that grow much more slowly with increasing devices when compared to the alternative approach of using event sequences. More specifically, the PFSM constructed from the *routine dataset* has *35* nodes and *211* transitions (edges) for 18 devices, 209 traces, and 701 activity events (on average, 39 events per device). Using traces from the same dataset, such a model has 710 nodes and 910 transitions, thus confirming that the PFSM provides a model that is more scalable due to substantially fewer nodes and edges.

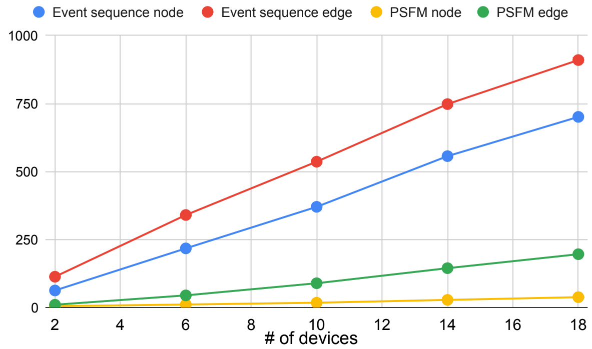


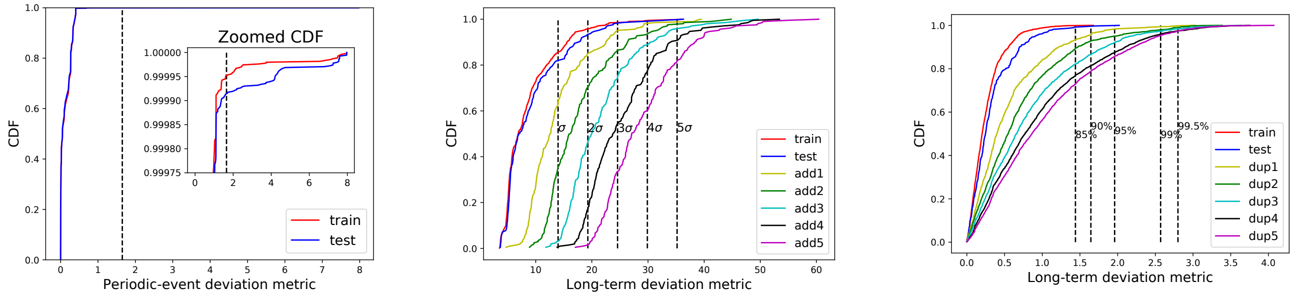
Figure 3: Complexity of models generated by different approaches, showing the advantage of using PFSM.

PFSM properties. Since we generate PFSM using Synoptic [17] (see §4.2), we expect that the two main abstraction properties of Synoptic models also hold for our PFSM: (*i*) the ability to accept every event trace used to generate the model (*i.e.,* each user event trace maps to a valid path in the model), (*ii*) the ability to accept event traces not used to generate the model (but that are similar to ones that generated the model). To confirm the first property, we successfully verified that 100% of the event traces we used to build the model are accepted by the PFSM. For the second property, we verified the presence of user event traces in our datasets that we did not use to build the model that are also accepted by PFSM. This indicates that our PFSM is able to recognize traces that have never been encountered before. We manually inspected a sample of these traces and verified that, although different, they were related to user activities used to generate the model. Specifically, they are a combination, or permutation, of previously seen traces.

5.3 Deviation Inference Evaluation

Deviation metrics. Our deviation metrics are intended to reveal changes in IoT system behavior that are significant, and we use thresholds to capture this significance from a statistical point of view. To this end, we use 5-fold cross-validation on our datasets from controlled experiments, divided into training data (used to infer our behavior models) and testing data (*not* used to infer our behavior models) plus artificially perturbed versions of them, to see how our deviation metrics behave.

The *periodic-event deviation metric* captures how much the timing of a periodic event deviates from the expected timing. To evaluate the metric, we compare its value for periodic events in both the training and the testing partitions of the idle dataset. Fig. 4a shows the CDF of this metric on periodic events from such dataset partitions. The distributions of the testing and training sets overlap, indicating that there are few changes to periodic traffic between the two datasets. In fact, more than 99% of periodic traffic flows in the idle training set are consistent with the previously inferred periods (periodic model) and generate a zero deviation metric.



(a) Full and zoomed CDFs of the *periodic-event deviation metric* from the idle training and testing sets and vertical lines corresponding to the threshold.

(b) CDFs of the *short-term deviation metric* for event traces in routine training and testing sets combined from 5 folds and CDFs from synthetic datasets. The y-axis represents the percentage of event traces.

(c) CDFs of the *long-term deviation metric* for all event transitions in each trace from the routine training and testing sets combined from 5 folds and CDFs from synthetic datasets. The y-axis represents the percentage of event transitions.

Figure 4: CDFs of deviation metrics from the controlled experiments datasets.

The *short-term deviation metric* captures how much a given user-event trace deviates from the PFSM representing the system behavior model. To evaluate the metric, we demonstrate that the value increases if we artificially increase the amount of user-event deviations compared to baseline traces. To show this, we generate five synthetic datasets, all based on the testing routine dataset, by adding to each event trace between one and five user events that produce new transitions. Fig. 4b shows the CDF of the metric for each dataset.[4] In particular, we find that the distributions of the metric's values on five synthetic datasets shift to the right as the amount of introduced deviation increases, showing that our metric tends to become larger with respect to the amount of previously unseen event transitions added to the system.

The *long-term deviation metric* measures long-term deviations in event-occurrence frequency, determining if any user-event sequences are occurring more or less frequently compared to the system behavior model. To show how the metric changes with respect to long-term deviations, we synthesize five datasets by duplicating traces in a testing set. The duplicated trace simulates changes in user-event-sequence frequency (*e.g.*, sending audio frequently from a smart speaker). Fig. 4c shows a clear trend of the metric distributions shifting right as the introduced deviation increases, which is what the metric was designed to do.

Significant deviation thresholds. The deviation metrics quantify changes in device behavior; we now focus on how to use the magnitude of the deviations to distinguish small changes in behavior from more significant changes in behavior that may require more attention. Specifically, we define, for each deviation metric, a threshold for statistical significance of the deviation.

For the *periodic-event deviation metric*, we empirically choose 1.61 (when $T_0 = 5T$) as the deviation threshold, a value determined by identifying the knee of the zoomed CDF as shown in Figure 4a. Given the vast number of periodic events (see §6.1), a higher threshold ensures we avoid marking an excessive number of deviations for manual review.

For the *short-term deviation metric*, we use the threshold $\rho = \mu + n\sigma$, calculated based on statistical properties of the events used to create the system behavior model. Based on our observations from Figure 4b, we set n as 3 for having the best tradeoff between capturing significant deviations and not flagging too many deviations as to become unwieldy.

For the *long-term deviation metric*, we consider the confidence interval *CI* (see Fig. 4c): scores outside the interval are considered statistically significant deviations. We use the commonly used value of *CI* = 95% to inform our threshold.

Deviation inference test cases. To assess whether deviation inference identifies notable changes in IoT system behavior, we test it using network traffic containing changes in behavior inspired by various real-world examples from prior work [14, 28, 29, 33, 37] and news articles [1–4]. The types of deviation we consider are: (*i*) unusual sequences of user events; (*ii*) changes in non-user events. We synthesize this traffic by modifying the event sequences in the routine dataset. Our approach is able to detect all generated cases as significant deviations, and we provide explanations for each case.

Deviations due to new event sequences. Changes in system behavior can lead to new combinations of user events, due to factors such as firmware updates or device compromise. We emulate this deviation by first injecting user events that cause new transitions in the PFSM, and then measuring their presence using the *short-term deviation metric* and the *long-term deviation metric*. For example, we inject a trace with several user events from an iKettle and Echo Spot after we turn off all lights and plugs, and then open the Meross (garage) Door-opener to leave home. Our approach identifies this as a significant deviation since we never trigger these user activities after we leave home in our controlled experiments.

[4] Each curve represents data combined from evaluating the deviation metric on 5 folds (*i.e.*, 5 distinct partitions of training and testing sets) to avoid bias from particularities of any one fold.

Device	Ave # of Periodic Models	Highest #
Home Auto	4.06	Nest Thermo: 8
Camera	5.82	ICSee Doorbell: 10
Smart Speaker	23.36	Echo Show5: 31
Hub	6.00	Philips Hub: 15
Appliance	6.40	Samsung Fridge: 22
Total	9.27	Echo Show5: 31

Table 4: Observed periodic models by device category.

Deviations due to event loss. Missing events can result in changes of behavior that can happen, for example, when a device is malfunctioning, or it is experiencing a service outage. We simulate this deviation by removing events from our controlled experiment datasets. Specifically, we remove events from an automation-introduced routine between Ring camera and Gosund Bulb, *i.e.,* we simulated the Gosund Bulb being offline and its events discarded. This change in behavior is detected by the *short-term deviation metric* and the *long-term deviation metric* since it causes statistically significant changes in user event traces. The *periodic-event deviation metric*s also detect this deviation since periodic activity is affected as well.

Deviations due to device misactivations. Unauthorized Device activations can result in problematic situations such as exposure of sensitive information (*e.g.,* from smart speakers [1, 29]), battery draining [65, 70], *etc.* We synthesize misactivations by inserting events simulating frequent device activations (*e.g.,* Echo Spot activating nine times in a row). Our approach detects synthetic misactivations as significant deviations since the *long-term deviation metric* or the *short-term deviation metric* exceed the threshold.

6 BEHAVIOR CHARACTERIZATION

In this section we use our behavior modeling approach and our deviation metrics in a real-world environment. Specifically, we characterize the IoT behavior observed during controlled and uncontrolled experiments in our IoT testbed.

6.1 Behavior Models Characterization

First, we provide a general characterization of our behavior models, then we provide some insights on unclassified traffic, and finally analyze the destinations by event type.

Periodic models. To measure the fraction of periodic traffic in realistic settings with user interactions, we combine the idle, activity, and routine datasets. Using our periodic modeling approach, we identify that an average 97.8% of the events in the combined dataset exhibit periodicity (see Table 9 in Appendix C). On the one hand, this is not surprising—periodic behavior is a hallmark of long-running online systems. On the other hand, this lends credence to our hypothesis that periodic traffic can often be classified due to its regular behavior, and thus provides a solid foundation for detecting deviations from such behavior.

From our 49 devices, we build 454 periodic models, corresponding to 9.27 (5) models per device on average (median). The difference in mean and median implies a skew toward a relatively small number of devices with large numbers of models. We summarize observed periodic models per device category in Table 4, where we show the average number of periodic models per device in each category (2nd column) and the name of the device with the largest number of periodic models, along with the number (3rd column). Echo Show 5 exhibits 31 periodic models, followed by Echo Spot and HomePod Mini, both with 27. While a periodic model can represent features such as heartbeats (*e.g.,* traffic to `device-metrics-us.amazon.com`), there are cases where we cannot map models to device features due to generic domain names (*e.g.,* `*.cloudfront.net`) and encrypted traffic obfuscating the purpose of the traffic.

We note that the complexity of a device correlates with the number of different periodic background traffic behaviors it generates. Devices with more functionality (*e.g.,* smart speakers and app-supporting devices like the smart fridge) generate more periodic models in comparison to less complex devices like home automation devices. For instance, the Echo Show 5 has more periodic models than Echo Dots, presumably because the former supports functions like Prime video streaming, image display, and purchase authentication, which are not supported on Echo Dots due to the absence of a screen. Conversely, devices with a small set of features such as the TP-Link Plug (with only one periodic model directed to the TP-Link cloud destination, excluding DNS and NTP communication) and other home automation devices exhibit small sets of periodic behavior.

We find that IoT devices with similar or identical functionality from the same vendor (*e.g.,* Amazon or Tuya) often have a similar set of periodic models in terms of periods and destinations, potentially due to shared software components. Interestingly, we also notice variations among devices from the same vendor. For instance, the TP-Link Bulb and Plug, despite contacting the same destination, exhibit periodic models with different periods. Similarly, smart light bulbs from Tuya have different periods and third-party destination domains. This could be attributed to the use of different software components, different versions of these components, or diverse configurations. While we have no ground truth to explain these differences, one possible explanation is that they use a different software supply chain. Such variations, if due to supply-chain issues, could make it more difficult to maintain software over time, potentially posing security risks.

User-action models. We build 57 user-action models from 141 distinct activities for 27 devices. During our experiments, user event classification achieves high accuracy, largely because simple devices typically exhibit easily identifiable network traffic patterns. In fact, we achieve perfect classification performance for most of these simple devices (see Table 3).

However, there are some types of user events that prove extremely challenging, if not impossible, to classify accurately. This occurs when different activities generate similar traffic patterns, making them indistinguishable by classifiers. Upon further inspection, we identify that some events correspond to identical message sizes, but with different data fields in their encrypted payload, which typically come in pairs, such as on/off, active/inactive. For instance, although the *'COLOR'* and *'ON'* activities of the Jinvoo Bulb demonstrate minor differences, our classifier differentiates them with 100% accuracy. Conversely, the *'ON'* and *'OFF'* activities from devices that are both from the same vendor and same category are identical in most features like packet size, and our user-action models aggregate them together as *on/off*. We find that the binary state activities, such as *on* and *off* from 13 devices out of 18, are indistinguishable. We also observe that different types of video processing events are difficult to distinguish, due to being implemented in similar ways.

PFSM system models. The PFSM model abstracts complex interactions among different devices due to both *programmed behaviors*, *i.e.,* deterministic user event sequences induced by automation or applets, as well as *non-programmed behaviors* introduced by human interactions and correlated events from devices that sense the same environment. For instance, we find that the Ring Camera states in the PFSM are always followed by Gosund Bulb's 'on' or 'off' states, which is consistent with what we specify in an automation, *i.e.,* a *programmed behavior*. Similarly, we observed cases in which *non-programmed behaviors* we were not aware of were also modeled by our testbed's PFSM. For example, we observe high-probability transitions between "movement detected" states of two smart cameras. After investigating the reason, we discovered that they were next to each other and triggered together when we generated the PFSM.

Unmodeled traffic (aperiodic events). The traffic flows that cannot be classified by periodic and user-action events are aperiodic events. We find that only 0.675% of the flows from idle, activity, and routine datasets are left unclassified and labeled as aperiodic events. These flows are sent to 283 distinct destinations across all 49 devices, on average, 5.78 per device (see Table 9 in Appendix C for more details).

While investigating the causes of the 0.2% of flows that do not exhibit periodicity in the idle dataset, we identified that the majority of them are from smart speakers or hubs. There are several reasons why this occurs: (i) our classifiers are not perfect and may fail to find labels for user event traffic; (ii) hubs act as gateways for many other IoT devices and process events for them, which can in turn add noise to traffic patterns in ways that confound classifiers.

In some cases, the traffic may have a period that is so long, *e.g.,* one day, that it cannot confidently be detected by our approach with a 5-day idle dataset. For example, Amazon Echo devices perform update checks every 24 hours or more. In other cases, the traffic is expected not to be periodic, such as firmware or other updates that are not released on regular schedules. Last, smart speakers tend to run more complex software (*e.g.,* a full Android/FireOS system, along with skills or apps) that can lead to irregular background traffic patterns. For example, `mas-sdk.amazon.com`, used for advertising physical and digital products that Amazon sells [10] on Echo Show 5. We leave a deeper analysis of aperiodic events as a topic of future work.

Event destination analysis. We characterize the destinations that each modeled event entails, which can reveal how information is exposed to other parties over the Internet for each type of event. For each event's destination, we identify the organization name for a second-level domain or an IP using WHOIS data or commonsense matching rules (*e.g.,* 'Amazon' corresponds to alexa.com). If the organization associated with IP aligns with the name, manufacturer, or an affiliate of the IoT device, we categorize it as a first party. Services such as cloud or CDN providers are labeled as support parties. All other entities are considered third parties.

Event	Device	First Party	Support Party	Third Party
Periodic Event	Home Auto	27	18	5
	Camera	13	23	18
	Smart Speakers	206	26	17
	Hubs	6	8	15
	Appliance	12	7	8
	Total	**264**	**82**	**63**
User Event	Home Auto	12	2	1
	Camera	7	11	2
	Smart Speakers	7	0	0
	Hubs	2	2	0
	Appliance	0	1	0
	Total	**28**	**16**	**3**
Aperiodic Event	Home Auto	36	3	7
	Camera	7	8	3
	Smart Speakers	178	7	6
	Hubs	7	1	0
	Appliance	10	2	8
	Total	**238**	**21**	**24**

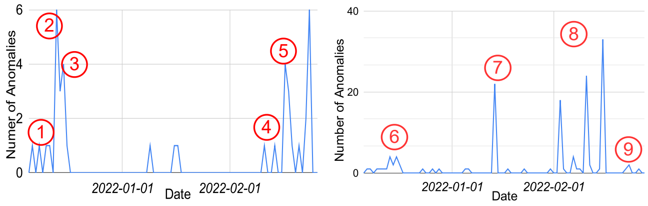
Table 5: Destination party per event type from idle, activity, and routine dataset.

Table 5 summarizes the result of this mapping, broken down by event type and device category. We find that 15.0% of destinations associated with periodic events are third-party ones, significantly more than the fraction of third-party user-event destinations (6.4%) and aperiodic event destinations (8.5%). This is expected, as user events tend to invoke a device's primary functionality (and thus does not require third parties), and many aperiodic events are related to updates or skill activities that also use first party communication [45]. In contrast, more periodic activities are associated with third parties. Particularly, we find that 6 devices periodically send requests to Google's DNS server though our DHCP server does not specify it as the default DNS server. Similarly, our smart devices periodically sync up with 17 distinct NTP servers including those from third parties such as Google, Apple, Amazon, and servers outside of US such as in Germany, Greece, or China. While it is perhaps interesting that such distant NTP servers were selected, this global spread of servers may alone constitute no additional privacy risk given the limited privacy implications of the NTP protocol. However, if these device NTP interactions contribute to establishing reliable fingerprints of devices or device activities, they could contribute to privacy risks.

The percentage of support parties for user events (34.0%) is higher than others (20.0% for periodic events and 7.4% for aperiodic events) since one third of the devices we test in our activity dataset rely on cloud services such as AWS for device control and communication. One important implication is that Amazon has substantial visibility into network traffic from IoT devices made by other vendors, who often are Amazon competitors in the IoT marketplace.

Non-essential destination analysis. A previous study [49] investigated whether communication with a destination (domain name) is required for smart device functionality, and produced a list of non-essential destinations that could be blocked without impairing device functionality, and a list of essential destinations. We now investigate how our observed events correlate with this list, to understand which events are essential or not.

We first search for matches in the non-essential list, finding that 22 of our observed destinations are labeled as non-essential. Of



(a) Behavior deviations due to user events. **(b) Behavior deviations due to periodic events.**

Figure 5: Deviations in uncontrolled experiments.

these, 16 are destinations associated with periodic events, and 6 with aperiodic events. This means that, in our dataset, periodic and aperiodic event destinations tend to be non-essential compared with user event destination.

We then compare the essential list, finding that 55 destinations are labeled as essential. Among them, we found that out of 18 device models that overlapped between our device and theirs, 15 devices have destinations associated with periodic events that are essential, meaning that these periodic models are required to support the functionality of the device. As for user events, 15 of their devices overlap with the devices we interacted with to generate labels in our activity dataset. We found that 13 out of 15 devices have destinations associated with user-action models that are essential. For the remaining two devices, we manually confirmed that their destinations originated from the same cloud provider but differed from the domains in the essential list. This motivates the need to revisit such domain classifications over time as device behavior changes. In terms of aperiodic events, 17 destinations are labeled as essential, all from Amazon, Samsung, and Google smart speakers and hubs. We confirm that these destinations are shared with either periodic or user event destinations, but exhibit different traffic patterns, thus labeled as aperiodic events. This raises concerns about the shared use of destinations for different purposes, potentially evading destination-based privacy-enhancing tools like IoTrim [49].

6.2 Behavior Deviation Characterization

We now analyze behavior deviations in the uncontrolled datasets. Using our deviation metrics and thresholds, we detect a total of 177 significant behavior deviations (2 per day on average). In the following paragraphs we describe how these deviations reveal important changes in behavior.

6.2.1 Behavior deviations due to user events. Figure 5a shows the 40 behavior deviations identified via the PFSM by the *short-term deviation metric* (4 of 40) and by the *long-term deviation metric* (36 of 40) over three months (0.46 per day). We now explain the labeled deviations due to user events.

Case 1, 4, and 5: Two behavior deviations due to unusual user events detected by the *long-term deviation metric* occurred near the beginning of the experiment period. Upon manual inspection, we found that unexpected activations of the Wyze Camera triggered these deviations and we determined the cause to be the relocation of Wyze Camera to a location where it is more sensitive to motion in the room. Case 4 and 5 were due to a similar reason. This kind of deviation is important because moving a camera can have significant implications for privacy (*e.g.,* if users are unexpectedly monitored as a result of camera movement) and/or security (*e.g.,* if cameras are moved to prevent detection of thieves). Interestingly, we can detect this type of change in behavior even though it was never observed in our training set (in fact, we detect it because of this) and even though the model was not designed specifically to detect this.

Case 2: Many behavior deviations around Dec 13 were detected by the *long-term deviation metric* and the *short-term deviation metric*, indicating unusual user event frequency and combinations. After looking at the traces responsible for the deviations, we confirmed that they were due to experiments from another project in our lab, *e.g.,* 50 consecutive voice activations on Echo Spot within 30 minutes. This demonstrates that unusual activity in the IoT system, even if it corresponds to previously seen individual events, can still be detected as significant deviations using our model. In this example, the detected Echo Spot behavior potentially exhibits the same risks as the case where the Google Home Mini erroneously and constantly recorded audio [1].

Case 3: We identified eight behavior deviations via the *long-term deviation metric* on Dec 15 due to many repeating events from the SmartLife Bulb and SwitchBot Hub, which were caused by network issues and incorrect configuration of the devices after these two devices were reset by researchers as part of an experiment for another research project. In this case, our behavior models help identify changes in behavior due to misactivation, which can be helpful for users who want to ensure the correct operation of their IoT devices.

6.2.2 Behavior deviations due to periodic events. Fig. 5b shows the detection of 137 behavior deviations via the *periodic-event deviation metric* over three months, with at least one occurring on 31 of the 87 days. We now explain some interesting cases.

Cases 6–8: All these cases were caused by documented network outages or from devices being temporarily removed from our testbed for other experiments. Our approach flags them as behavior deviations due to the total absence of non-user events. While there are certainly many other ways to detect such outages, it is nonetheless interesting that our behavior model detects it without being designed explicitly to do so.

Case 9: Several instances of periodic behavior deviation were triggered by the SwitchBot Hub frequently being turned off for minutes or hours. Upon manual investigation, we found that these interruptions were caused by device malfunctions, though the exact reasons for them remain unknown to us.

6.2.3 Takeaways. By leveraging our behavior models and deviation metrics, we can effectively identify a range of behavior deviations. While our measurement approach is not designed to determine the actual root causes of such deviations, our behavior models and inferred events provide the context for determining them. For example, the cases above show how our approach can help detect or verify outages, device failures, and device relocations.

7 DISCUSSION

7.1 Summary of Findings

To summarize the key takeaways from our modeling and characterization of IoT systems, we find that:

(*i*) The vast majority (97%) of IoT traffic is periodic, with a small portion (2.325%) due to user actions, and an even smaller amount (0.675%) left unclassified. The fact that so much network traffic is classifiable into periodic and user events is good for purposes like anomaly detection, but also problematic when it comes to privacy, given that network observers (*e.g.,* network providers, home routers) can use this to identify devices and activities in a home. Further aggravating privacy concerns, we find that a substantial portion (15%) of periodic events entail third parties.

(*ii*) Most unclassified traffic (aperiodic events) is not essential, *i.e.,* it may be blocked without losing device functionality. This motivates the need for better information and disclosures regarding the purpose of network activities in IoT systems, to assist in understanding the impact of such unnecessary information exposure.

(*iii*) IoT system behavior was relatively stable during a three-month longitudinal study, with only few statistically significant deviations per day in our uncontrolled environment. The deviations we identified covered a range of important changes in behavior, including a camera being moved, network outages, and device misconfigurations/failures—even though our models were not specifically designed to detect any of these types of issues.

7.2 Other Behavior Model Applications

In this study, we primarily developed an approach to analyze the behavior of an IoT system and to apply it to our IoT testbed to shed light on the changes of its behavior over a period of three months. Our modeling approach can serve additional purposes as follows.

Informing IoT profiles. The creation of IoT profiles, defined as devices specifications and intended communication patterns, is an important problem partially addressed by the IETF in March 2019 with the Manufacturer Usage Description standard (MUD — RFC 8520 [44]). However, even four years after standardization, the MUD standard was not adopted yet by any of the devices in our testbed. We believe that our approach can assist with automatically creating such profiles based on our behavior models, and can also help with automatically verifying compliance with existing profiles. For example, we identified that TP-Link Plug exhibited the following models — PFSM (states): on and off; periodic models (protocol-destination-period in second): TCP-*.tplinkcloud.com-236, DNS-*.neu.edu-3603, NTP-*.pool.ntp.org-3603. The network traffic corresponding to these models could become a MUD profile, and made available to the manufacturer for validation. Once validated, any network traffic from the device that deviated from these models could be flagged a non-compliant.

Regulatory and privacy policy compliance. IoT devices usually must comply with local regulations (*e.g.,* GDPR [56] and CCPA [54]) and their published privacy policies, which may impose restrictions on the data, their purpose, and their destinations. Because our behavior models capture many of these aspects of IoT device behavior, we believe our approach can help with regulatory and privacy policy compliance analysis. Specifically, our approach can flag behavior deviations as events for further investigation. For example, we found that aperiodic events not matching our periodic and user-action models are typically not essential and may be blocked without affecting the device functionality. If, upon further investigation, we find that the destination of the traffic is a known tracker and we can block such traffic without affecting the device functionality, it means that the device may be exposing data unnecessarily, which could violate the data minimization principle of article 5c of GDPR.

Anomaly detection. We believe that existing anomaly detection systems can leverage our behavior modeling approach to establish baseline normal behavior, and leverage our deviation metrics and thresholds to detect behavior deviations as anomalies. Our approach provides names/descriptions of the IoT devices responsible for the deviation and the reasons behind its detection, including information such as the specific deviation score that triggered the detection, the events involved, and the extent of the deviation. We have proposed thresholds that measure statistical significance; however, in an anomaly detection system those can be modified to find the desired trade-off between sensitivity and specificity. Furthermore, the fact that our approach abstracts events and traces can help labeling the detected anomalies with the actual events and traces responsible for them, thus aiding the system administrator in triaging such anomalies. While we believe the information we provide is sufficient for experts and IoT enthusiasts to understand and use, we have not conducted any user studies, nor have we explored how to convey this information to lay users. We leave these topics to future work; it is our hope that at least some anomalies detected by our system can be presented to typical home users in ways that are understandable and actionable (*e.g.,* notifying them that a camera is recording video when it should not, and perhaps should be removed).

Any anomaly detection approach can raise privacy concerns due to the information gleaned from traffic analysis. To mitigate such risks, our system can run in a local network (*e.g.,* at the home gateway) without any external dependencies (*e.g.,* cloud servers). Our approach does not require data to be collected from users; rather, models based on lab experiments can be pushed into home-network-based deployments. An interesting avenue for future work is to incorporate privacy-preserving techniques (*e.g.,* differential privacy, privacy-preserving federated learning) that enable monitoring from outside the local network and without loss of privacy.

7.3 Limitations and Future Work

Ground-truth limitations. Our work assumes the availability of ground-truth labels to create the user-action models, as also assumed by previous work [13, 59, 67]. However, there may be situations where such ground-truth is not available, incomplete, or obsolete (*e.g.,* devices events changing significantly after a firmware update, or devices offering too many functions), resulting in inaccurate user-action models and user event inference. This limitation can be addressed by using user-action models built using unsupervised clustering methods and by periodically retraining the model. By combining our dataset and the results in Kolcun *et al.* [43], we find that most IoT devices exhibit relatively static network traffic behavior. That said, small changes that we observe over time mean that periodically updating models will result in better long-term detection performance. We leave this last topic as future work.

Modeling limitations. We model only behavior that can be inferred from IP network traffic traversing our gateway to Internet destinations. Our approach could incorporate local and non-IP traffic using wireless sniffers, perform a similar event inference approach from related work [64], and integrate these events into our models. However, we focus only on wide-area IP traffic in our work, since capturing it requires no special-purpose hardware.

Two different events with identical IP-traffic characteristics are indistinguishable in our system. These limitations can be mitigated by integrating with event logs extracted from the companion apps, platforms, or APIs (*e.g.,* SmartThings APIs); however, in this work we focused on the case where only network traffic is available.

Finally, our system behavior model does not account for two or more unrelated user events often occurring at the same time. We believe it is possible to address this if we can identify that the events otherwise always occur separately, but this is left as an optimization for an uncommon case.

8 RELATED WORK

IoT behavior analysis. Many prior works in the context of IoT behavior analysis [14, 29, 34, 40, 43, 55, 59] focus on IoT measurement studies to reveal privacy, security, or safety issues of IoT devices. Other studies characterize attacks on IoT devices [15, 52, 60, 61]. Finally, other papers consider IoT application security and privacy [16, 19, 31, 76] and design approaches to improve IoT systems and/or data privacy [22, 23, 25, 32, 42, 49]. These studies do not model or characterize IoT behaviors by event types, and in particular neglect non-user events that account for the vast majority of the IoT traffic. Moreover, they do not consider emergent system behavior and do not measure behavior changes.

IoT event inference. Numerous prior studies fingerprint IoT events by analyzing network traffic [13, 53, 67]. Some of them [13, 67] consider only user-event traffic. HomeSnitch [53] detects non-user (background) traffic, but does not distinguish periodic traffic, does not consider UDP traffic, and requires a manual process for identifying and labeling traffic; thus, their approach is not suitable to fully inform our behavior models. Finally, several studies [24, 50, 52] infer periodic features from network traffic for specific purposes (*e.g.,* device identification or attacks), but they do not use these features to partition the traffic into periodic events as we do, and therefore these approaches are not suitable to inform our behavior models or to filter the traffic to classify aperiodic events; hence, the only comparable approaches to our work are the ones that include user-event classification since they use models that are comparable to our user-action models. In §5.1 we discuss and compare our user-action models with [67].

9 CONCLUSION

In this paper, we proposed a novel measurement approach for modeling and characterizing device and system behavior, and how behavior changes in a real IoT deployment. Our evaluation showed that our models capture system properties not previously captured by previous work, such as modeling periodic and aperiodic events, and capturing emergent system behavior. Moreover, our analysis of these models in a real-world setting sheds light on what is happening behind the scenes in today's IoT deployments; for example, we identified the large prevalence of periodic traffic, the fact that periodic and aperiodic non-user traffic tends to be non-essential for functionality, and important deviations in behavior due to unforeseen causes (*e.g.,* changing the location of a device).

We expect our approach to be useful not only for follow-up research, but also for other applications such as anomaly detection, generation of IoT profiles, and regulatory compliance. To facilitate follow-up research, and/or for applying our approach in different contexts, we released the code and data from our experiments [38].

ACKNOWLEDGMENTS

We thank our shepherd and the anonymous reviewers for their constructive feedback. This research was supported by the NSF (BehavIoT CNS-1909020, ProperData SaTC-1955227). The opinions, findings, conclusions, and recommendations expressed are those of the authors and do not necessarily reflect the views of any of the funding bodies.

REFERENCES

[1] 2017. Google admits its new smart speaker was eavesdropping on users. http://money.cnn.com/2017/10/11/technology/google-home-mini-security-flaw. Accessed on May 26, 2023.

[2] 2019. Massive Google Outage Turned Smart Homes Into Zombies. https://www.thedailybeast.com/massive-google-outage-turned-smart-homes-into-zombies. Accessed on May 26, 2023.

[3] 2020. It's time for smart home devices to have local failover options during cloud outages. https://staceyoniot.com/smart-home-devices-cloud-outage-vs-local/. Accessed on May 26, 2023.

[4] 2021. First-world-problems-Amazon-outage-left-tens-thousands-without-Alexa. https://www.dailymail.co.uk/news/article-10291367/First-world-problems-Amazon-outage-left-tens-thousands-without\-Alexa-Roombas-Ring-apps.html. Accessed on May 26, 2023.

[5] [n. d.]. Alexa Routines. https://www.wikipedia.org/. Accessed on May 26, 2023.

[6] [n. d.]. Bitdefender Box 2. https://www.bitdefender.com/smart-home/. Accessed on May 26, 2023.

[7] [n. d.]. Fingbox. https://www.fing.com/products/fingbox. Accessed on May 26, 2023.

[8] [n. d.]. IFTTT. https://ifttt.com/. Accessed on May 26, 2023.

[9] [n. d.]. Internet Systems Consortium BIND(dig). https://www.isc.org/bind/. Accessed on May 26, 2023.

[10] [n. d.]. Mobile Associates API Overview. https://developer.amazon.com/docs/mobile-associates/mas-overview.html. Accessed on May 26, 2023.

[11] [n. d.]. RATtrap. https://www.myrattrap.com/. Accessed on May 26, 2023.

[12] [n. d.]. SmartThings. https://www.smartthings.com/. Accessed on May 26, 2023.

[13] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and Selcuk Uluagac. 2020. Peek-a-Boo: I see your smart home activities, even encrypted!. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 207–218.

[14] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. Sok: Security evaluation of home-based iot deployments. In *2019 IEEE Symposium on Security and Privacy*. IEEE, 1362–1380.

[15] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. 2017. Understanding the mirai botnet. In *26th USENIX Security Symposium (USENIX Security 17)*.

[16] Leonardo Babun, Z Berkay Celik, Patrick McDaniel, and A Selcuk Uluagac. 2021. Real-time analysis of privacy-(un) aware iot applications. *Proceedings on Privacy Enhancing Technologies* 2021, 1 (2021), 145–166.

[17] Ivan Beschastnikh, Yuriy Brun, Sigurd Schneider, Michael Sloan, and Michael D Ernst. 2011. Leveraging existing instrumentation to automatically infer invariant-constrained models. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*. 267–277.

[18] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[19] Z Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A Selcuk Uluagac. 2018. Sensitive Information Tracking in Commodity IoT. In *27th USENIX Security Symposium (USENIX Security 18)*.

[20] Z Berkay Celik, Patrick McDaniel, and Gang Tan. 2018. Soteria: Automated IoT Safety and Security Analysis. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. 147–158.

[21] Z Berkay Celik, Gang Tan, and Patrick D McDaniel. 2019. IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT.. In *Network and Distributed System Security Symposium, NDSS*.

[22] Yunang Chen, Mohannad Alhanahnah, Andrei Sabelfeld, Rahul Chatterjee, and Earlence Fernandes. 2022. Practical Data Access Minimization in Trigger-Action Platforms. (2022).

[23] Yunang Chen, Amrita Roy Chowdhury, Ruizhe Wang, Andrei Sabelfeld, Rahul Chatterjee, and Earlence Fernandes. 2021. Data Privacy in Trigger-Action Systems. In *2021 IEEE Symposium on Security and Privacy*. IEEE, 501–518.

[24] Haotian Chi, Chenglong Fu, Qiang Zeng, and Xiaojiang Du. 2022. Delay Wreaks Havoc on Your Smart Home: Delay-based: Automation Interference Attacks. In *2022 IEEE Symposium on Security and Privacy*. IEEE, 1575–1575.

[25] Haotian Chi, Qiang Zeng, Xiaojiang Du, and Lannan Luo. 2021. PFIREWALL: Semantics-Aware Customizable Data Flow Control for Smart Home Privacy Protection. *Network and Distributed System Security Symposium, NDSS* (2021).

[26] Adrien Cosson, Amit Kumar Sikder, Leonardo Babun, Z Berkay Celik, Patrick McDaniel, and A Selcuk Uluagac. 2021. Sentinel: A Robust Intrusion Detection System for IoT Networks Using Kernel-Level System Information. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*. 53–66.

[27] Wenbo Ding and Hongxin Hu. 2018. On the safety of iot device physical interaction control. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 832–846.

[28] Wenbo Ding, Hongxin Hu, and Long Cheng. 2021. IOTSAFE: Enforcing Safety and Security Policy with Real IoT Physical Interaction Discovery. In *Network and Distributed System Security Symposium, NDSS*.

[29] Daniel J Dubois, Roman Kolcun, Anna Maria Mandalari, Muhammad Talha Paracha, David Choffnes, and Hamed Haddadi. 2020. When speakers are all ears: Characterizing misactivations of iot smart speakers. *Proceedings on Privacy Enhancing Technologies* 2020, 4 (2020), 255–276.

[30] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *kdd*, Vol. 96. 226–231.

[31] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security analysis of emerging smart home applications. In *2016 IEEE symposium on security and privacy*. IEEE, 636–654.

[32] Earlence Fernandes, Justin Paupore, Amir Rahmati, Daniel Simionato, Mauro Conti, and Atul Prakash. 2016. Flowfence: Practical data protection for emerging iot application frameworks. In *25th USENIX Security Symposium (USENIX Security 16)*.

[33] Chenglong Fu, Qiang Zeng, and Xiaojiang Du. 2021. Hawatcher: Semantics-aware anomaly detection for appified smart homes. In *30th USENIX Security Symposium (USENIX Security 21)*.

[34] Aniketh Girish, Tianrui Hu, Vijay Prakash, Daniel J. Dubois, Srdjan Matic, Danny Yuxing, Serge Egelman, Joel Reardon, Juan Tapiador, David Choffnes, and Narseo Vallina-Rodriguez. 2023. In the Room Where It Happens: Characterizing Local Communication and Threats in Smart Homes. In *Proc. of the Internet Measurement Conference (IMC'23)*.

[35] Tianbo Gu, Zheng Fang, Allaukik Abhishek, Hao Fu, Pengfei Hu, and Prasant Mohapatra. 2020. Iotgaze: Iot security enforcement via wireless context analysis. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 884–893.

[36] Jiawei Han, Jian Pei, and Micheline Kamber. 2011. *Data mining: concepts and techniques*. Elsevier.

[37] Weijia He, Valerie Zhao, Olivia Morkved, Sabeeka Siddiqui, Earlence Fernandes, Josiah Hester, and Blase Ur. 2021. SoK: Context sensing for access control in the adversarial home IoT. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 37–53.

[38] Tianrui Hu, Daniel J. Dubois, and David Choffnes. 2023. BehavIoT Dataset and Software. https://moniotrlab.khoury.northeastern.edu/publications/behaviot-imc23.

[39] Danny Yuxing Huang, Noah Apthorpe, Frank Li, Gunes Acar, and Nick Feamster. 2020. Iot inspector: Crowdsourcing labeled network traffic from smart home devices at scale. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 2 (2020), 1–21.

[40] Umar Iqbal, Pouneh N Bahrami, Rahmadi Trimananda, Hao Cui, Alexander Gamero-Garrido, Daniel J. Dubois, David Choffnes, Athina Markopoulou, Franziska Roesner, and Zubair Shafiq. 2023. Tracking, Profiling, and Ad Targeting in the Alexa Echo Smart Speaker Ecosystem. In *Proc. of the Internet Measurement Conference (IMC'23)*.

[41] Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, Amir Rahmati, Earlence Fernandes, Zhuoqing Morley Mao, Atul Prakash, and SJ Unviersity. 2017. ContexIoT: Towards Providing Contextual Integrity to Appified IoT Platforms.. In *Network and Distributed System Security Symposium, NDSS*, Vol. 2. San Diego, 2–2.

[42] Haojian Jin, Gram Liu, David Hwang, Swarun Kumar, Yuvraj Agarwal, and Jason I Hong. 2022. Peekaboo: A Hub-Based Approach to Enable Transparency in Data Processing within Smart Homes. In *2022 IEEE Symposium on Security and Privacy (S&P'22)*.

[43] Roman Kolcun, Diana Andreea Popescu, Vadim Safronov, Poonam Yadav, Anna Maria Mandalari, Yiming Xie, Richard Mortier, and Hamed Haddadi. 2020. The Case for Retraining of ML Models for IoT Device Identification at the Edge. *arXiv preprint arXiv:2011.08605* (2020).

[44] Eliot Lear, Ralph Droms, and Dan Romascanu. 2019. Manufacturer Usage Description Specification. RFC 8520. https://doi.org/10.17487/RFC8520

[45] Christopher Lentzsch, Sheel Jayesh Shah, Benjamin Andow, Martin Degeling, Anupam Das, and William Enck. 2021. Hey Alexa, is this skill safe?: Taking a closer look at the Alexa skill ecosystem. *Network and Distributed Systems Security (NDSS) Symposium2021* (2021).

[46] Zhenhui Li, Bolin Ding, Jiawei Han, Roland Kays, and Peter Nye. 2010. Mining periodic behaviors for moving objects. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1099–1108.

[47] Haoyu Liu, Tom Spink, and Paul Patras. 2019. Uncovering security vulnerabilities in the Belkin WeMo home automation ecosystem. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 894–899.

[48] Sunil Manandhar, Kevin Moran, Kaushal Kafle, Ruhao Tang, Denys Poshyvanyk, and Adwait Nadkarni. 2020. Towards a natural perspective of smart homes for practical security and safety analyses. In *2020 IEEE Symposium on Security and Privacy*. IEEE, 482–499.

[49] Anna Maria Mandalari, Roman Dubois, Daniel J.and Kolcun, Muhammad Talha Paracha, Hamed Haddadi, and David Choffnes. 2021. Blocking Without Breaking: Identification and Mitigation of Non-Essential IoT Traffic. In *Proc. of the Privacy Enhancing Technologies Symposium (PETS)*.

[50] Samuel Marchal, Markus Miettinen, Thien Duc Nguyen, Ahmad-Reza Sadeghi, and N Asokan. 2019. Audi: Toward autonomous iot device-type identification using periodic communication. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1402–1412.

[51] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N Asokan, and Ahmad-Reza Sadeghi. 2019. DÏoT: A federated self-learning anomaly detection system for IoT. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 756–767.

[52] TJ OConnor, William Enck, and Bradley Reaves. 2019. Blinded and confused: uncovering systemic flaws in device telemetry for smart-home internet of things. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. 140–150.

[53] TJ OConnor, Reham Mohamed, Markus Miettinen, William Enck, Bradley Reaves, and Ahmad-Reza Sadeghi. 2019. HomeSnitch: behavior transparency and control for smart home IoT devices. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. 128–138.

[54] State of California Department of Justice. 2018. California Consumer Privacy Act of 2018 (CCPA). https://oag.ca.gov/privacy/ccpa. Accessed on May 26, 2023.

[55] Muhammad Talha Paracha, Daniel J Dubois, Narseo Vallina-Rodriguez, and David Choffnes. 2021. IoTLS: understanding TLS usage in consumer IoT devices. In *Proceedings of the 21st ACM Internet Measurement Conference*. 165–178.

[56] European Parliament. 2016. Regulation (EU) 2016/679 (General Data Protection Regulation). https://gdpr-info.eu/. Accessed on May 26, 2023.

[57] Roberto Perdisci, Thomas Papastergiou, Omar Alrawi, and Manos Antonakakis. 2020. IoTFinder: Efficient Large-Scale Identification of IoT Devices via Passive DNS Traffic Analysis. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 474–489.

[58] Antônio J Pinheiro, Jeandro de M Bezerra, Caio AP Burgardt, and Divanilson R Campelo. 2019. Identifying IoT devices and events based on packet length from encrypted traffic. *Computer Communications* 144 (2019), 8–17.

[59] Jingjing Ren, Daniel J. Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. 2019. Information Exposure for Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach. In *Proc. of the Internet Measurement Conference (IMC'19)*.

[60] Eyal Ronen and Adi Shamir. 2016. Extended functionality attacks on IoT devices: The case of smart lights. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 3–12.

[61] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. 2017. IoT goes nuclear: Creating a ZigBee chain reaction. In *2017 IEEE Symposium on Security and Privacy*. IEEE, 195–212.

[62] Said Jawad Saidi, Anna Maria Mandalari, Roman Kolcun, Hamed Haddadi, Daniel J Dubois, David Choffnes, Georgios Smaragdakis, and Anja Feldmann. 2020. A Haystack Full of Needles: Scalable Detection of IoT Devices in the Wild. In *Proceedings of the ACM Internet Measurement Conference*. 87–100.

[63] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Vol. 39. Cambridge University Press Cambridge.

[64] Narmeen Shafqat, Daniel J Dubois, David Choffnes, Aaron Schulman, Dinesh Bharadia, and Aanjhan Ranganathan. 2021. ZLeaks: Passive Inference Attacks

on Zigbee based Smart Homes. *arXiv preprint arXiv:2107.10830* (2021).

[65] Ryan Smith, Daniel Palin, Philokypros P Ioulianou, Vassilios G Vassilakis, and Siamak F Shahandashti. 2020. Battery draining attacks against edge computing nodes in IoT networks. *Cyber-Physical Systems* 6, 2 (2020), 96–116.

[66] Vincent F Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. 2016. Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 439–454.

[67] Rahmadi Trimananda, Janus Varmarken, Athina Markopoulou, and Brian Demsky. 2020. Packet-level signatures for smart home devices. In *Network and Distributed System Security Symposium, NDSS*, Vol. 2020.

[68] Thijs van Ede, Riccardo Bortolameotti, Andrea Continella, Jingjing Ren, Daniel J Dubois, Martina Lindorfer, David Choffnes, Maarten van Steen, and Andreas Peter. 2020. FLOWPRINT: Semi-Supervised Mobile-App Fingerprinting on Encrypted Network Traffic. In *Network and Distributed System Security Symposium, NDSS*. Internet Society.

[69] Janus Varmarken, JA Aaraj, Rahmadi Trimananda, and Athina Markopoulou. 2022. FingerprinTV: Fingerprinting Smart TV Apps. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, Vol. 2022. 606–629.

[70] Eugene Y Vasserman and Nicholas Hopper. 2011. Vampire attacks: draining life from wireless ad hoc sensor networks. *IEEE transactions on mobile computing* 12, 2 (2011), 318–332.

[71] Michail Vlachos, Philip Yu, and Vittorio Castelli. 2005. On periodicity detection and structural periodic similarity. In *Proceedings of the 2005 SIAM international conference on data mining*. SIAM, 449–460.

[72] Yinxin Wan, Kuai Xu, Guoliang Xue, and Feng Wang. 2020. IoTArgos: A multi-layer security monitoring system for Internet-of-Things in smart homes. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 874–883.

[73] Fei Wang, Jianliang Wu, Yuhong Nan, Yousra Aafer, Xiangyu Zhang, Dongyan Xu, and Mathias Payer. 2022. PROFACTORY: Improving IoT Security via Formalized Protocol Customization. *31th USENIX Security Symposium (USENIX Security 22)* (2022).

[74] Qi Wang, Pubali Datta, Wei Yang, Si Liu, Adam Bates, and Carl A Gunter. 2019. Charting the attack surface of trigger-action IoT platforms. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 1439–1453.

[75] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 103–117.

[76] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. 2018. Homonit: Monitoring smart home apps from encrypted traffic. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1074–1088.

[77] Wei Zhou, Yan Jia, Yao Yao, Lipeng Zhu, Le Guan, Yuhang Mao, Peng Liu, and Yuqing Zhang. 2019. Discovering and Understanding the Security Hazards in the Interactions between IoT Devices, Mobile Apps, and Clouds on Smart Home Platforms. In *28th USENIX Security Symposium (USENIX Security 19)*.

A TESTBED DEPLOYMENT

The list of devices we deployed and the functionalities we triggered while collecting the routine dataset are shown in Table 6. The list of automations we enabled is shown in Table 7.

B ADDITIONAL DETAILS OF EVENT INFERENCE

Features. Table 8 lists the features we selected. Features fall into these three categories: (i) Packet features, (ii) Timing features, (iii) Flow features. Note that we do not use IP address or port numbers for classification because they are highly dynamic. However, destination domain name and protocol do not change often [49, 57, 59, 62], and proved to be important features.

User event classification. We choose a Random Forest classifier [18] because it is lightweight and easy to deploy on an edge device (*e.g.,* a home router), and worked well with limited training samples. Specifically, we use a separate binary classifier for each possible user activity instead of one multi-class classifier for each device. The input for each classifier is the set of all the flows in our training set, each of which has a boolean label which is true when the flow corresponds to the activity and false otherwise. When predicting the event for a given flow, we select the prediction of the binary classifier with the highest confidence of being positive. If none of the classifiers gives a positive result, the flow is labeled as an aperiodic event.

Device	activity
Ring Doorbell	motion, ring, video
Ring Camera	motion, video
D-Link Camera	motion, video
Wyze Camera	motion, video
TPLink Plug	on, off
Wemo Plug	on, off
Amazon Plug	on, off
TPLink Bulb	on, off, color, dim
Smartlife Bulb	on, off, color, dim
Jinvoo Bulb	on, off, color, dim
MagicHome Strip	on, off, color, dim
Gosund Bulb	on, off, color, dim
Govee Bulb	on, off, color, dim
Meross Dooropener	on, off
Nest Thermostat	set, on, off
SwitchBot Hub	on, off
iKettle	on
Echo Spot	voice

Table 6: 18 devices and corresponding user activities used in automation experiments (routine dataset). On and off mean turning on and off. Color and dim represent changing the color and brightness of the device.

Index	Automations
R1 Alexa & IFTTT	says 'open/close garage', then open/close the Meross Dooropener.
R2 Alexa	All light on routine: says 'turn on all lights'
R3 Alexa	All light off routine: says 'turn off all lights'
R4 Alexa	says 'turn on TV' (using SwitchBot Hub) and then turn off Magichome Strip
R5 Alexa	says 'turn off TV' (using SwitchBot Hub) and then turn on Magichome Strip
R6 Alexa	If Ring Doorbell rings, turn on Wemo Plug and weather reports on Echo Spot, and then turn off Wemo Plug after 5s
R7 Alexa	If Ring Doorbell detects a motion, then blink Smartlife Bulb by turning it on for 5s then turn off, and set the color of Jinvoo Bulb to red
R8 Alexa	If Ring Camera motion, then turn on Gosund Bulb
R9 Alexa	If D-Link Camera detects a motion, then turn on TPLink Bulb
R10 APP	Turn on Nest Thermostat on 6 AM and turn off on 10 PM
R11 Alexa	says 'I am leaving', change Nest Thermostat temperature to 72, open garage (R1). wait 5 min, close garage(R1).
R12 IFTTT	If Wyze Camera motion, then turn on TPLink Plug, clip Wyze Camera, turn off TPLink Plug
R13 IFTTT	Morning routine: says 'good morning', then boil the iKettle to 100 and turn on Govee Bulb
R14 IFTTT	Good night routine: says 'good night', then turn off Govee Bulb
R15 IFTTT	If Meross Dooropener opens, then turn on TPLink Bulb and change its color to maroon
R16 IFTTT	If Meross Dooropener closes, then turn off TPLink Plug and set TPLink Bulb color to green

Table 7: The list of automations we set up in our testbed.

Device	Periodic event %	Aperiodic event %
Amazon Plug	98.765%	0.229%
Aqara Hub	98.696%	0.345%
Behmor Brewer	99.855%	0.145%
Jinvoo Bulb	94.571%	0.030%
D-Link Camera	97.098%	0.00%
D-Link Motion Sensor	99.961%	0.039%
Echo Dot	99.751%	0.249%
Echo Dot3	85.948%	14.052%
Echo Dot4	99.902%	0.098%
Echo Flex	98.086%	1.914%
Echo Plus	98.014%	1.986%
Echo Show5	94.014%	4.036%
Echo Spot	98.505%	0.778%
Samsung Fridge	99.797%	0.203%
Google Home Mini	99.846%	0.253%
Google Nest Mini	99.846%	0.154%
Gosund Bulb	95.213%	0.000%
Govee Bulb	88.797%	0.696%
Homepod Mini	98.071%	1.929%
Homepod	98.490%	1.510%
iCSee Doorbell	95.789%	0.048%
IKEA Hub	99.568%	0.062%
iKettle	99.822%	0.059%
Keyco Air Sensor	99.888%	0.112%
LeFun Camera	95.681%	0.198%
MagicHome Strip	93.786%	0.026%
Meross Dooropener	99.788%	0.00%
Microseven Camera	98.969%	0.00%
LG Microwave	98.967%	1.033%
Nest Thermostat	91.830%	1.275%
Philips Bulb	99.972%	0.028%
Ring Camera	76.127%	1.270%
Ring Chime	99.941%	0.059%
Ring Doorbell	89.130%	0.062%
Smartlife Bulb	84.543%	0.120%
SmartThings Hub	97.286%	2.628%
Anova Sousvide	99.965%	0.00%
SwitchBot Hub	96.842%	0.00%
Thermopro Sensor	99.964%	0.036%
Philips Hub	99.704%	0.296%
Wemo Plug	96.227%	1.302%
TPLink Bulb	96.484%	0.580%
TPLink Plug	96.643%	0.134%
Tuya Camera	90.976%	0.056%
Ubell Doorbell	99.435%	0.00%
Wansview Camera	99.614%	0.011%
Wink Hub2	100.000%	0.00%
Wyze Camera	97.294%	0.127%
Yi Camera	99.507%	0.00%
ALL	97.798%	0.675%

Table 9: The fraction of periodic and aperiodic events in idle, activity, and routine datasets.

Feature	Description
meanBytes	Average bytes in a flow.
minBytes	The lowest number of bytes in a flow
maxBytes	The highest number of bytes in a flow
medAbsDev	The median absolute deviation of number of bytes in a flow
skewLength	The skewness of the number of bytes in a flow
kurtosisLength	The kurtosis of the number of bytes in a flow
meanTBP	The average of time differences between consecutive packets
varTBP	The variance of time differences between consecutive packets
medianTBP	The median of time differences between consecutive packets
kurtosisTBP	The kurtosis of time differences between consecutive packets
skewTBP	The skew of the time differences between consecutive packets
network_out_external	The number of packets sent to server
network_in_external	The number of packets received from server
network_external	The number of packets transmit to/from server
network_local	The number of packets transmit between local devices
network_out_local	The number of packets sent to other device in the local network
network_in_local	The number of packets received from other device
meanBytes_out_external	Average bytes sent to servers per packet
meanBytes_in_external	Average bytes received from servers per packet
meanBytes_out_local	Average bytes sent to other device in the local network per packet
meanBytes_in_local	Average bytes received from other device in the local network per packet

Table 8: List of selected features used for event inference.

C ADDITIONAL EVALUATION OF EVENT INFERENCE

Table 9 shows the fraction of periodic and aperiodic events in our controlled datasets for each device.

D ACKNOWLEDGMENTS

ChatGPT was used to generate LaTeX format tables, and revise BibTeX format for certain citations of this work.