Towards Labeling On-Demand IoT Traffic

Daniel Campos
Florida Institute of Technology
Melbourne, FL, USA
dcampos2015@my.fit.edu

TJ OConnor
Florida Institute of Technology
Melbourne, FL, USA
toconnor@fit.edu

ABSTRACT

A lack of transparency has accompanied the rapid proliferation of Internet of Things (IoT) devices. To this end, a growing body of work exists to classify IoT device traffic to identify unexpected or surreptitious device activity. However, this work requires fine-grained labeled datasets of device activity. This paper proposes a holistic approach for IoT device traffic collection and automated event labeling. Our work paves the way for future research by thoroughly examining different techniques for synthesizing and labeling on-demand traffic from IoT sensors and actuators. To demonstrate this approach, we instrumented a smart home environment consisting of 57 IoT devices spanning cameras, doorbells, locks, alarm systems, lights, plugs, environmental sensors, and hubs. We publish an open-source dataset consisting of 16,686 labeled events over 468,933 network flows. Our results indicate that vendor APIs, trigger-action frameworks, and companion notifications can be used to generate scientifically valuable labeled datasets of IoT traffic.

CCS CONCEPTS

• **Information systems → Traffic analysis**; • **Networks → Mobile and wireless security**; • **Human-centered computing → Ubiquitous and mobile devices**.

ACM Reference Format:
Daniel Campos and TJ OConnor. 2021. Towards Labeling On-Demand IoT Traffic. In *Cyber Security Experimentation and Test Workshop (CSET '21), August 9, 2021, Virtual, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3474718.3474727

1 INTRODUCTION

The pervasive use of Internet-of-Things (IoT) devices has accelerated in the last two years, with 21.5 billion devices actively connected worldwide [16, 26]. IoT actuators and sensors connect us to our homes through a variety of devices, including wireless doorbells, security cameras, locks, movement sensors, thermostats, and security alarms. To accompany this growth, vendors have begun offering IoT-related products and services in this domain by leveraging turnkey solutions to the always-on and always-connected cloud. However, security and privacy issues have plagued this rapid growth with a lack of established standards and protocols [4, 6, 9, 14, 15, 20, 30, 34].

CSET '21, August 9, 2021, Virtual, CA, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-9065-1/21/08…$15.00
https://doi.org/10.1145/3474718.3474727

The lack of transparency and control in IoT devices has complicated user security and privacy. A recent class-action lawsuit argued that Ring Inc. allowed attackers to access users' cameras covertly and surreptitiously [23]. In a separate incident, an ADT technician pled guilty to remotely spying on customers over 9,600 times in a four-year period [8]. Due to the limited interfaces on IoT devices, victims are left unaware of their devices' actions or how to implement access controls [12]. IoT devices occupy our most personal spaces with cameras and microphones. Unfortunately, this lack of transparency and control has enabled intimate partner violence by using IoT devices to spy and harass victims [2, 8, 9]. Labeling and identifying traffic flows by purpose (i.e., camera streaming) offers the promise of providing greater transparency to users.

Encrypted communication protocols and device resource constraints complicate transparency and the examination of these devices' activities. IoT devices' limited interfaces have left consumers blinded without the ability to interact directly with operating systems, resulting in a lack of awareness of the data collected and sent by IoT sensors and actuators. Largely, consumers are left to trust the vendor to control the processing and storage of the massive data produced by these devices. In addition to malicious actors, vendors have monetized users' data without clear transparency to the end-users. As an example, Ring partnered with over 1,300 law enforcement agencies to share camera recordings [5, 10].

Previous works have examined collecting IoT device communication datasets [13, 18, 22, 24, 25, 28, 31]. Previous dataset approaches include logging plaintext protocols such as DHCP and DNS requests [13] or full network captures [17, 24]. However, previous datasets lack fine-grained labels to determine the ground truth for device activities (e.g., when a motion detection camera is transmitting video.) They are often broadly focused on creating a binary division between benign traffic and attack traffic (e.g., a device controlled by the Mirai botnet versus a device in normal operation) [19, 22]. Or they contain coarse-grained labels identifying specific device models [13, 17]. Coarse-grained approaches lack labels that identify devices' on-demand event activities (e.g., a camera streaming video). Datasets that include fine-grained labels are created manually, recording the actions of a lab assistant performing these actions repeatedly [18]. This approach cannot scale to support the sample demands for supervised learning algorithms. There is a growing need to create fine-grained labeled and scalable datasets to enable future research on classifying encrypted IoT traffic.

A key insight of our work is the use of application programming interfaces, trigger-action frameworks, and companion notifications to automatically label events from network traces. This insight allows us to construct fine-grained labels of device behaviors. Further, we leverage trigger action frameworks and vendor APIs to remotely trigger events, generating an appropriate scale of events for machine learning.

Contributions: In this paper, we present an approach for collecting a labeled dataset of fine-grained activities for IoT device network traffic. Our approach overcomes the limitations of previous works by including detailed device histories with network traces. We make the following contributions in this paper:

- We examine the trade-offs of application programming interfaces, trigger-action frameworks, and companion application notifications to automatically label events from network traces and produce rich histories of events.
- We publish an open-source dataset using our methodology, which contains 16,686 labeled events over 468,933 network flows from 57 devices. To facilitate our experiment at larger scales, we publish our dataset at https://research.fit.edu/iot.

Organization: Section 2 motivates the need for a labeled IoT dataset and discusses relevant background about the structure of fragmented IoT device ecosystems. Section 3 provides an overview and high-level goals for our approach for capturing network traces and labeling device histories. Section 4 explains our detailed approach for our experiment to collect and label traffic. Section 5 discusses limitations and presents opportunities for future research. Section 6 examines related work. Section 7 concludes.

2 BACKGROUND

The following section motivates our work and provides an overview of the structure of IoT ecosystems.

2.1 Motivating Example

We motivate the need for a labeled dataset by examining the criminal actions of an ADT Corporation employee. We hypothesize that this crime is indicative of a greater problem of IoT's role in personal privacy abuse [2, 9]. In 2020, an ADT employee pled guilty to accessing the security cameras of 220 women over 9,600 times during a four-year period [8]. The security cameras' flawed access control allowed the attacker to add his account without user approval and access the victims' recordings and device streams. Investigators only discovered the intrusion after a customer identified the attacker's e-mail on their account access page. Subsequently, ADT performed an internal investigation that uncovered 219 other compromised accounts. The investigators' findings reported that the attacker had accessed accounts of predominately attractive younger women. Companion applications restrict users to limited transparency. In this instance, a lack of historical audit logs allowed the attacker to access and stream users' cameras without detection for years.

This privacy compromise demonstrates the systemic lack of transparency and control in IoT devices. Workstations, laptops, and mobile phones provide users with strong mechanisms to enforce access control and audit events. The Android and iPhone operating systems both enforce fine-grained access control on a per-app basis. This tailored approach enables users to grant or deny access to sensitive objects like the camera, microphone, or GPS. Mobile phones also record access attempts, providing transparency to the end-users. As demonstrated in our example, IoT devices lack this same transparency and control. The victims had no knowledge they were recorded or how to prevent access. IoT devices' resource constraints prevent users from controlling or auditing sensitive events (e.g., enabling a camera, turning on a microphone, or unlocking a door.) Further, the pervasive use of encryption prevents any network-level forensics of device activity [24, 25, 31]. This lack of transparency relies on vendors, such as ADT, to honestly broker users' data. However, ADT failed over the four-year period when it did not identify that a single attacker accessed 220 unique cameras.

Previous works have hypothesized that machine learning approaches can identify and classify network traffic on IoT devices to overcome transparency issues [1, 21, 27, 31, 32]. Machine learning can predict on-demand events (e.g., a camera detecting motion or transmitting video) by analyzing the statistics of encrypted network flows. However, the precision of such approaches requires a training dataset of known IoT events and traffic. As we discuss in Section 6, current datasets lack the fine-grained detail and scale to train algorithms effectively. Our work provides a dataset capable of enabling the further study of algorithmic approaches for identifying when IoT device network traffic contains sensitive events. Before we propose our solution, we examine IoT devices' purposes, scattered ecosystems, and integration frameworks.

2.2 IoT Sensors and Actuators

IoT devices are broadly categorized on two purposes: sensing and actuating [7]. Sensors report environmental and physical stimuli in the world. Actuators enable remote control of IoT devices. We review these devices' purposes to present our labeling strategy.

Sensors: Sensors observe and report environmental and physical changes in IoT devices. Examples include microphones that listen to sound, infrared sensors that detect motion, humidity sensors that detect moisture, or thermostats that measure temperature. IoT devices also sense and report changes to their physical state. For example, smart-locks sense and report when an owner physically unlocks a door. These always-on sensors provide an always-current state of their device through perpetual connections to internet-servers.

Actuators: Actuators allow owners to change or modify the state of a device. For example, a smart-lock owner can remotely unlock the front door for package delivery. Or an owner can remotely trigger an electronic relay to change the color of smart-lights. In industrial IoT solutions, actuators exist in the form of remote valve controls, where users can modify the flow rate of a water or gas line or even remotely activate electrical breakers. To enable the always-availability of actuators, IoT devices periodically poll internet-connected servers for owner requests and commands.

The combination of sensing and actuating offers the convenience of IoT. An internet-connected thermostat that detects a change in temperature can trigger an actuator to turn on the air-conditioning unit. However, to broker these connections requires an IoT ecosystem of hybrid servers. IoT ecosystems bridge the gap between perpetual sensor reports and periodic actuator polling.

2.3 IoT Ecosystems

As depicted in Figure 1, IoT ecosystems broker connections between perpetually-reporting sensors, periodically-polling actuators, and device owners. These architectures rely on cloud-based

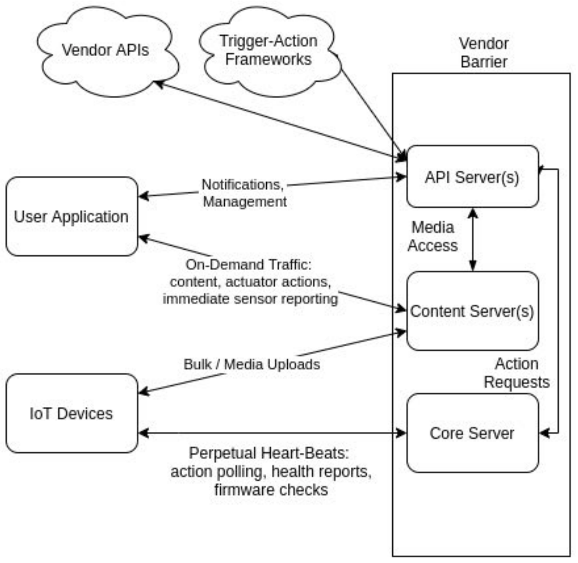
Vendor APIs
Trigger-Action Frameworks
Vendor Barrier
API Server(s)
Notifications, Management
User Application
Media Access
On-Demand Traffic: content, actuator actions, immediate sensor reporting
Content Server(s)
Bulk / Media Uploads
Action Requests
IoT Devices
Core Server
Perpetual Heart-Beats: action polling, health reports, firmware checks

Figure 1: Trigger-action frameworks, vendor APIs, and companion applications enable overcoming IoT vendor barriers that abstract transparency and control away from users.

solutions [34]. As depicted in Figure 1, a scattered solution of cloud servers enables an owner to sense and control a device. These vendor-dependent solutions consist of core servers, API gateways, and content distribution networks. Vendor barriers abstract transparency and control away from users by offloading processing and storage to cloud-based servers. We broadly categorize these servers on their intended purpose: maintaining core connections, providing content delivery, and exposing application programming interface.

Core-Servers: IoT core servers (e.g., Amazon Web Services IoT, Azure IoT, and Google Cloud IoT) offer the promise of scalability by securely terminating IoT traffic at provision-less endpoints [33]. In IoT ecosystems, these servers handle the idle or *heartbeat* messages that offer the always-on, always-available aspect of IoT. The communication to these servers adopts varying degrees of encryption based on the maturity of the solution [25, 29].

Application Programming Interface (API) Gateways: IoT API Gateways expose the functionality of IoT devices to the users and applications. Gateways perform essential functions, including authentication, registration, device management, and provisioning. They bridge the communication between users and the perpetually connected core servers by relaying user commands (e.g., begin recording video). Further, APIs connect users to the IoT-produced media on content distribution networks (e.g., recorded video files).

Content Distribution Networks (CDNs): IoT device architectures leverage heavily optimized content distribution networks to store and transmit large data. In managed IoT environments, these networks store bulk media (e.g., audio/video recordings, sensor logs, device histories.) By offloading storage and processing to the cloud, IoT vendors can rapidly scale their infrastructure [3]. CDNs generally store attributeless media and implement access control by generating unique identifiers and access tokens. Intra-server communication between CDNs and API gateways relay these controls to users and applications.

2.4 Trigger-Action Frameworks

Users enjoy the convenience of IoT sensors and actuators interacting autonomously based on a set of pre-defined rules. For example, a proximity sensor can trigger a wireless garage door to open and power on the outdoor lights. This cross-vendor interoperability presents a unique challenge for users with a heterogeneity of device vendors. Third-party trigger-action frameworks, such as Alexa and If-This-Then-That (IFTTT), overcome this challenge by creating centrally managed locations to define rules and automate actions. Third-party trigger-action frameworks rely on a set of pre-defined rules that automate API calls across different vendors' managed IoT architectures. In the previous example, an owner may grant IFTTT access to make API calls across different vendors, including their SmartThings proximity sensor, MyQ garage door opener, and Philips Hue Smart Lights. Instead of the API calls happening directly to the device - they occur across the managed cloud environments for the SmartThings, MyQ, and Philips Hue vendors.

While third-party frameworks automate convenience for the owner, they also provide us an opportunity to automate actions for labeling device activities. For example, we can set a time-based rule for the IFTTT trigger-action framework to record a 30-second video clip on an Arlo camera. Third-party trigger-action frameworks and time-based triggers present an opportunity to create a scalable dataset of IoT device behaviors. For example, we remotely locked and unlocked a smart lock 176 times in our experiment in Section 4. This solution allows us to overcome the scalability limitations of previous IoT datasets created by manual triggers [18]. We discuss our approach for leveraging trigger-action frameworks in further detail in Section 4. In the following subsections, we examine the type of traffic we can observe in network traces.

2.5 IoT Traffic Types

We distinguish IoT device traffic into two main categories: perpetual heartbeats and on-demand activity. Our work focuses on detecting and labeling spontaneous on-demand activity.

Perpetual Heartbeats: IoT devices perpetually connect to core servers to provide availability for users. As maintaining long-running connections are costly, IoT devices achieve perpetual connections by periodically polling core servers at fixed intervals. In addition to regularly polling for instructions, IoT devices may regularly push health checks such as battery state or request firmware updates. We call this traffic *heartbeats* due to the periodic nature that connects it to device management. Previous works have observed that this type of traffic is predictable with fixed signatures [11, 20]. Previous work also refer to this traffic as idling [25, 31]. However, we find this description problematic as it fails to capture the traffic's purpose. We use the term heartbeat to describe connections necessary to the overall health of the always-available aspect of IoT. As this traffic happens periodically regardless of user interaction, we can gather samples without any external triggers. As Huang examined, DNS data is often sufficient to label heartbeat traffic [13].

On-Demand Activity: On-demand traffic occurs when users or applications request sensor data or actuator actions. The convenience of IoT is provided by rapidly supporting these on-demand requests. For example, a user may request that a camera upload streaming video to a content distribution network. Other on-demand traffic examples including performing specific actuators' actions (e.g., locking a door) or immediately sensing and reporting (e.g., checking the temperature). While users generally force on-demand traffic, we can also use the previously discussed trigger-action frameworks to trigger on-demand traffic. Labeling on-demand traffic enables us to identify traffic anomalies in network traces that match the characteristic of on-demand traffic but do not appear in device histories or logs (e.g., a Ring doorbell transmitting video to police servers).

3 OVERVIEW

Our work proposes a solution to automatically generate and label on-demand events from IoT devices across unique vendors and types. This section discusses the challenges to this problem and explores the ideas behind triggering and labeling events.

3.1 Challenges

Several challenges accompany this problem. In particular, we examine the challenges of applying fine-grained labels that describe device activity in the presence of pervasive encryption. Further, we address the need to generate enough samples to facilitate further machine learning research. Our work addresses these challenges by integrating a hybrid of solutions discussed in Section 4 by using trigger-action frameworks, interfacing with vendor APIs, and intercepting companion application notifications.

C1) Pervasive Encryption: The pervasive use of encryption in IoT ecosystems complicates inferring device activity from network traces. Previous work has measured the use of encryption in IoT ecosystems [25]. We echo their finding that unencrypted protocols correspond to the maturity of vendors. To label device histories of mature vendors' traffic requires overcoming encryption limitations by extracting device activity labels from trigger-action frameworks, vendor APIs, and application notifications.

C2) Fine-Grained On-Demand Event Labels: Identifying unintended device activity requires fine-grained labels that clearly distinguish the different activities of IoT devices (e.g., remotely toggling power, detecting motion, locking a door.) Previous work has addressed this challenge by relying on the IFTTT framework to trigger events [31]. However, relying solely on IFTTT proves problematic as it is unsupported by several popular vendors [31]. We overcome this challenge by directly interfacing with vendor APIs and intercepting companion application notifications.

C3) Scale of On-Demand Samples: To classify and identify unintended device activity requires a dataset with an appropriate number of labeled samples. Generating labeled samples through manual interaction proves problematic as it is laborious to reproduce [18]. Previous work has addressed this by leveraging the IFTTT trigger action framework [31]. We complement this approach by describing the use of IFTTT time-based trigger recipes to remotely schedule events and generate an appropriate set of labeled samples.

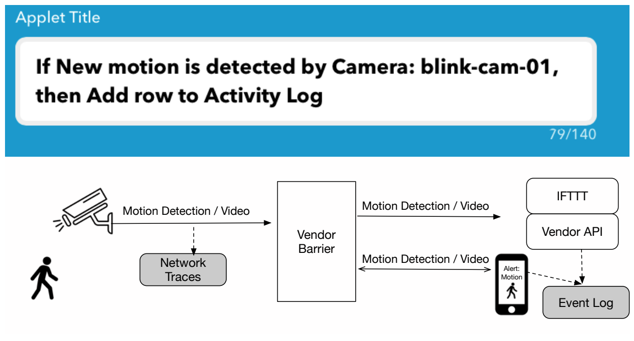

Figure 2: IFTTT Recipes, Vendor APIs, and Companion Applications all provide opportunities to capture and precisely label manually triggered events.

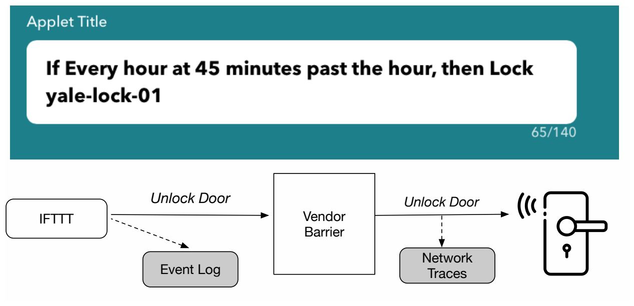

Figure 3: Scheduling repeated IFTTT recipes enables us to automatically trigger and label device activities.

3.2 Key Ideas

The key insight of our work is exploring a hybrid model of trigger-action frameworks, vendor APIs, and companion application notifications to generate and label on-demand events. In this section, we provide an overview of how we generate and label events.

Triggering Actions: We generate (or trigger) on-demand actions either manually or by leveraging vendor APIs and trigger-action frameworks. As depicted in Figure 2, manually interacting with a device (such as walking by a motion detection camera) will generate network traces as the camera signals motion detection and uploads video to a content distribution network. Manually triggering events is problematic for reproducibility and scale. Previous work identified IFTTT trigger action framework offers an opportunity to repeatedly generate on-demand events [31]. We improve this effort by taking advantage of a recent improvement to the IFTTT framework. As depicted in Figure 3, IFTTT recently enabled support to schedule on-demand events repeatedly on an hourly basis. This addition greatly simplified our approach, generating 24 samples daily for 14 devices in our dataset across eight unique device activities. This included two locks as depicted in Figure 4.

Logging Events: Repeatedly scheduled triggers proves helpful in generating a scalable label event history. However, it is necessary to capture and label the noise, motion, and vendor-specific events that occur manually within the lab environment. We leverage IFTTT

Figure 4: When supported, we leveraged IFTTT triggers to remotely cause on-demand events such as locking and unlocking two of the seven door locks in our dataset, which produced 706 events.

trigger-action frameworks, exposed vendor APIs, and capture companion application notifications to label manual events. This hybrid approach allowed us to overcome limitations exposed by vendor implementations. Using IFTTT recipes, we captured manually triggered events with IFTTT by writing the event to a log file. However, as all devices do not support IFTTT [31], our approach considered other labeling methods for manually triggered events. Vendor APIs, such as the Samsung SmartThings API, maintain a forensically rich event history. In our experiment in Section 4, we queried the Samsung SmartThings API to discover the detailed logs of seven devices not exposed by IFTTT. IoT companion applications, hosted on mobile phones and tablets, often provide the only transparency to IoT end-users. We captured the companion applications' notifications by hooking the push notification system of a mobile phone paired to our IoT devices. This approach discovered the detailed event histories of 21 devices not exposed by IFTTT or vendor APIs. Our experiments, detailed in Section 4, demonstrate the benefit of this heterogeneous approach to labeling events.

4 DATASET CAPTURE EXPERIMENT

The following section discusses the structure and execution of an experiment to capture and label a dataset of IoT traffic. We examine the devices in our dataset, our capture and label methodologies.

4.1 IoT Lab Environment

IoT Devices: As illustrated in Table 1, our dataset consists of a representative set of 57 smart home IoT devices for consumers, available from well-known US retailers including Best Buy, Walmart, Target, and Amazon. We focus our collection on devices within smarthome automation categories, including cameras, doorbells, locks, alarm systems, lights, plugs, environmental sensors, and hubs. Of the 57 devices, 26 connect through smart hubs, and 31 connect directly to the internet over a wired or wireless connection.

Capture Hardware: Our capture hardware consists of a Linksys WRT1900AC router running OpenWRT, a Dell PowerConnect 6248P hardware switch, and a pfSense Server. The Linksys router offers the physical layer connection (either wired or wireless for devices), mirroring the routers commonly used in smarthomes. The Linksys router operates in AP Mode, forwarding the traffic over the hardware switch to the pfSense server for network layer IP addressing and routing. To ease future classification efforts, we reserved static

```
{"package":"com.tplink.kasa_android","title":"kasa−cam−01
detected motion","text":"","date":"March␣12,␣2021␣at␣01:46:05PM"}
{"package":"com.tplink.kasa_android","title":"kasa−cam−01
detected sound","text":"","date":"March␣12,␣2021␣at␣02:01:45PM"}
```

Figure 5: In our work, we hook the Android Notification Service to read and log IoT device notifications, delivering JSON formatted logs that provide transparency of IoT device events.

IP addresses for all IoT devices. We leverage the switch's port mirroring technology to mirror both outbound and inbound traffic from the router's outgoing port into a second port on the server, which performs the capture. Further, the captures are temporarily written to an in-memory filesystem, preventing packet loss during bursty traffic. We then rotate and move captures to a physical disk on an hourly basis for long-term storage.

Capture Experiment: We recorded 468,933 network flows and 16,686 unique events from March 08, 2021 at 1:30 PM GMT to March 15, 2021 at 9:16 PM GMT. We generated labels by remotely triggering events using IFTTT recipes and capturing notifications and logs from companion applications, IFTTT, and vendor APIs. We remotely triggered 4,944 events (or 29.63% of labeled events in our dataset). We further labeled 11,742 events by manually interacting with devices (or the remaining 70.37% events in our dataset.) To manually trigger events, a researcher visited the lab daily and interacted with devices by unlocking doors, ringing doorbells, walking by nearby motion detection cameras, and speaking loudly. Manually triggered events include any event generated by an environmental change, which can have an automatic source. The Philips Hue Lights, for example, indirectly trigger CV-dependent motion sensors when the lights change state. Several actuators also triggered audio sensors when the lock physically switched states.

4.2 Labeling Implementation

Companion Application Notifications: We capture companion application notifications from 27 unique devices across 10 vendors to label events. Capturing companion notifications allows us to label events from IoT vendors who do not integrate with trigger-action frameworks or permit API access. Companion applications, implemented on phones and tablets often serve as the only transparency for IoT devices. Vendors often achieve this transparency through the use of push notifications. For example, these notifications inform users that their smart doorbell has detected audio or motion. While vendor implementations vary, they ultimately deliver notifications to a standardized phone or tablet notification subsystem API. Notifications are implemented pervasively across IoT companion applications to implement the always-on, always-available aspects of IoT. In our work, we hook the Android Notification Service to read and log IoT device notifications. Figure 5 depicts an example of the resulting hooked logs, showing motion and audio detection on our TP-Link Kasa Camera. This approach enabled us to record 6,366 total events that would have otherwise been unobservable with the same precision. We implemented this functionality in a Python script and published it to provide reproducibility.

Table 1: Our dataset contains 16,686 labeled events over 468,933 network flows from 57 devices focused on smarthome automation categories, including cameras, doorbells, locks, alarm systems, lights, plugs, environmental sensors, and hubs

Device	Total Packets	Traffic (In / Out)	Flows (TCP / UDP)	Total Events	Labeled Activities	Label Source
Arlo Base Station	815,751	41.19 MiB / 454.19 MiB	3,703 / 4,569	353	arm,disarm	IFTTT Triggers
⟶Arlo Pro Smart Security Camera 1				214	motion,record	IFTTT Logs,IFTTT Triggers
⟶Arlo Pro Smart Security Camera 2				222	motion,record	IFTTT Logs,IFTTT Triggers
August Connect Wi-Fi Bridge	295,033	12.01 MiB / 17.68 MiB	175 / 2	0		
⟶August Smart Lock Pro (3rd Gen)				61	lock,unlock	IFTTT Logs
August Connect Wi-Fi Bridge	294,712	11.97 MiB / 17.56 MiB	192 / 2	0		
⟶Yale Real Living Assure Lock Deadbolt				21	lock,unlock	IFTTT Logs
Blink Sync Module	203,619	7.67 MiB / 8.02 MiB	61 / 739	353	arm,disarm	IFTTT Triggers
⟶Blink Indoor Home Security Camera 1	23,861	596.10 KiB / 19.63 MiB	37 / 39	39	motion	IFTTT Logs
⟶Blink Indoor Home Security Camera 2	22,835	611.67 KiB / 17.87 MiB	39 / 41	42	motion	IFTTT Logs
⟶Blink Indoor Home Security Camera 3	25,434	654.57 KiB / 20.37 MiB	42 / 44	45	motion	IFTTT Logs
Geeni Aware Surveillance Camera 1	407,569	12.88 MiB / 69.77 MiB	19,381 / 4,727	486	motion	Companion App
Geeni Aware Surveillance Camera 2	415,853	13.28 MiB / 75.79 MiB	19,418 / 4,638	529	motion	Companion App
Geeni Doorpeek Smart Wi-Fi Doorbell	263,380	6.88 MiB / 41.38 MiB	902 / 102	232	motion,ring	Companion App
Geeni Doorscreen Smart Wi-Fi Doorbell	2,382,015	117.12 MiB / 144.34 MiB	627 / 676	96	ring,motion	Companion App
Geeni Look Indoor Smart Security Camera 2	252,425	6.80 MiB / 38.70 MiB	819 / 83	601	motion,sound	Companion App
Geeni Sentinel Smart Camera	1,674,494	169.98 MiB / 249.50 MiB	1,525 / 5,749	981	motion	Companion App
Merkury Innovations Smart Wifi Camera	204,070	5.24 MiB / 33.66 MiB	773 / 109	83	motion	Companion App
Merkury Smart Wi-Fi Doorbell Camera	256,006	7.03 MiB / 46.42 MiB	859 / 191	110	motion,ring	Companion App
Nest Cam IQ Indoor	14,905,585	494.64 MiB / 2.69 GiB	52 / 25	194	person,motion,sound	Companion App
Nest Hello Smart Wi-Fi Video Doorbell	780,899	26.53 MiB / 115.66 MiB	83 / 2	60	motion,person,ring,sound,	Companion App
Night Owl Doorbell 1	4,010,530	781.92 MiB / 272.51 MiB	124,227 / 27,891	76	motion,ring	Companion App
Night Owl Doorbell 2	4,044,239	787.20 MiB / 276.69 MiB	124,462 / 27,910	110	motion,ring	Companion App
Philips Hue Bridge	212,863	4.05 MiB / 34.31 MiB	1,052 / 2,702	0		
⟶Philips Hue White and color				353	off,on	IFTTT Triggers
⟶Hue Ambiance Smart LED Bar Light 1				353	off,on	IFTTT Triggers
⟶Hue Ambiance Smart LED Bar Light 2				352	off,on	IFTTT Triggers
⟶Hue Ambiance Smart LED Bar Light 3				352	off,on	IFTTT Triggers
⟶Hue Ambiance Smart LED Bar Light 4				352	off,on	IFTTT Triggers
Ring Alarm Base Station 1st Generation	190,361	10.87 MiB / 18.00 MiB	416 / 3,000	0		
⟶Ring Alarm Contact Sensor 1				61	open,batt,close	Companion App
⟶Ring Alarm Contact Sensor 2				69	open,close	Companion App
⟶Ring Alarm Contact Sensor 3				82	open,close	Companion App
⟶Ring Alarm Motion Detector 1				76	motion	Companion App
⟶Ring Alarm Motion Detector 2				74	motion	Companion App
⟶Ring Alarm Motion Detector 3				69	motion	Companion App
Ring Spotlight Cam	3,064,828	114.44 MiB / 2.32 GiB	698 / 874	117	motion	Companion App
Ring Video Doorbell 1	654,039	34.42 MiB / 384.91 MiB	399 / 814	182	motion,ring	Companion App
Ring Video Doorbell 2	617,518	31.83 MiB / 359.76 MiB	355 / 718	166	motion,ring	Companion App
Ring Video Doorbell Pro	7,620,669	228.94 MiB / 6.24 GiB	1,071 / 1,308	341	motion,ring	Companion App
Schlage Encode Smart WiFi Deadbolt	8,603	319.50 KiB / 883.65 KiB	15 / 13	37	lock,unlock,batt	Companion App
Sifely Lock Gateway G2	19,427	612.15 KiB / 796.64 KiB	466 / 3	0		
⟶Sifely Keyless Entry Door Lock				4	unlock	Companion App
SimpliCam HD Indoor Security Camera	328,044	7.29 MiB / 244.23 MiB	20 / 4,509	14	motion	Companion App
Simplisafe Video Doorbell Pro	595,814	12.68 MiB / 494.38 MiB	39 / 6,615	291	motion,ring	Companion App
SmartThings Hub	535,465	54.41 MiB / 32.25 MiB	11,989 / 9,755	0		
⟶Kwikset Electronic Deadbolt				49	unlock,lock	Smartthings Logs
⟶SmartThings Motion Sensor				931	motionstart,motionend,temp	Smartthings Logs
⟶SmartThings Multipurpose Sensor				657	temp,move,open,close,	Smartthings Logs
⟶Samsung Smartthings Outlet				343	off,on	Smartthings Logs
⟶SmartThings Water Leak Sensor				603	temp,waterstart,waterend	Smartthings Logs
⟶Yale Touchscreen Deadbolt				379	unlock,lock,timeoutunlock	Smartthings Logs
SmartThings Smart Camera	494,353	26.05 MiB / 217.58 MiB	2,514 / 3,762	2,598	rssi,lqi,soundend,personend, soundstart,motionstart,personstart,record, motionend	Smartthings Logs
TP-Link Kasa Spot Camera	4,091,964	103.10 MiB / 3.39 GiB	2,213 / 15,731	1,546	sound,motion,off,on,	Companion App,IFTTT Triggers
Ultraloq Bridge WiFi Adapter	38,040	1.49 MiB / 1.83 MiB	52 / 60	0		
⟶Ultraloq U-Bolt Smart Deadbolt				357	unlock,lock	IFTTT Logs,IFTTT Triggers
Wyze Cam v2	1,658,118	53.04 MiB / 716.46 MiB	10,743 / 12,157	970	motion,sound,record,restart, off,on	IFTTT Logs,IFTTT Triggers
Totals	51,408,416	3.11 GiB/18.96 GiB	329,396 / 139,537	16,686		

Vendor-Specific APIs: IoT vendors often expose their devices' control and logging through application programming interfaces (APIs.) This transparent approach enables broad adoption of their products into other vendor ecosystems. Further, offering this unified set of capabilities supports enables cross-vendor integration into their ecosystems. As an illustration of these APIs, the widely popular SmartThings platform exposes a broad array of device capabilities and logs. Figure 6 depicts a partial history of SmartThings devices in our lab. In our work, we leverage these event-driven logs to produce fine-grained labels of the device history. Although IFTTT supports limited SmartThings integration, directly interfacing with the API exposes a fuller set of logged activity. As an example, our SmartThings water-leak sensor identifies and reports two events: the initial moisture detection and when the sensor no longer detects moisture. Only the initial moisture detection is available through IFTTT, whereas the complete picture is available in the vendor API. In observation, vendor APIs offered better fidelity of their device histories than what is observable through IFTTT. Three environmental sensors that observed temperature (a multi-sensor, a water sensor, and a motion sensor) reported 1,618 events (or 9.70% of the aggregate of all labeled events in our dataset.)

```
"March␣15,␣2021␣at␣04:31:41AM","smartthings−cam−01","event\_soundstart"
"March␣15,␣2021␣at␣04:31:53AM","smartthings−cam−01","event_soundend"
"March␣15,␣2021␣at␣04:44:13AM","smartthings−outlet−01","event_off"
"March␣15,␣2021␣at␣04:59:05AM","smartthings−outlet−01","event_on"}
"March␣15,␣2021␣at␣04:30:12AM","yale−lock−01","event_unlock"}
"March␣15,␣2021␣at␣04:53:11AM","smartthings−multi−01","event_temp"
```

Figure 6: We leverage event-driven logs from vendor application programming interfaces (APIs), including the SmartThings platform, to produce fine-grained labels of the device history.

Trigger Action Frameworks: We leveraged the IFTTT trigger action framework to trigger and log events, constructing 77 IFTTT recipes for 20 unique devices. We separated our approach into two IFTTT recipe types: recipes that logged manual events and recipes that automatically triggered events. The aggregate of our 77 IFTTT recipes generated roughly a third of all labeled activity in our dataset. Manual IFTTT recipes logged four different events (lock, unlock, motion, sound) across 10 IoT devices in our dataset, recording only 613 device events (or only 3.67% of total events.) While IFTTT was capable of manually logging more devices, we sourced labels from companion app notifications and vendor APIs with more precise time labels. Unfortunately, IFTTT only provides minute-level precision in timestamps, making it an unattractive source for manual logs. Automatic time-based trigger recipes generated eight events (arm, disarm, lock, unlock, turn off, turn on, record, restart) for 14 devices in our dataset, triggering 4,944 labeled events (29.63% of our labeled events.) We leveraged IFTTT triggers in devices that controlled lighting and actuators to collect activities for environment sensors, like smart lighting and door locks. Both devices trigger significant environmental changes, which can be detected by CV-based motion detectors and audio sensors, allowing us to collect data from [environment] sensors without requiring physical interaction.

5 DISCUSSION

In this section, we consider the limitations of our experiment and offer insight for future works.

5.1 Limitations

IFTTT Precision: IoT vendors integrate their devices into IFTTT by supporting periodic polling and queries of their vendor APIs. The vendor sets the rate and threshold of this polling. The rate varies among vendors from three seconds to an entire minute. Further, IFTTT date and time recipe ingredients only support minute-level precision. While vendors may support triggering events that complete within seconds, the final log only produces minute-level accuracy. Future work may examine the use of IFTTT webhook's to trigger on-demand requests through a web request. This approach could replace the minute-level accuracy created by IFTT recipes with second-level precision of the webhook script's execution time. However, the precision would still rely on the individual vendor's periodic polling rate. This approach requires further investigation to ensure that webhooks execute timely and do not violate the respective vendor thresholds.

Cross-Referencing Multiple Sources: When remotely triggering events through IFTTT recipes, we observed some recipes stayed active despite successfully executing. For a small number of samples, this implementation flaw caused multiple events to occur at the same time. IFTTT will execute an action based on a device's identifier and incorrectly match the response to multiple rules, including hidden or archived applets. IFTTT documents this behavior on their support page. However, we only discovered this limitation only after examining the device logs that support both SmartThings API and IFTTT trigger action framework. Reviewing the SmartThings logs, we uncovered that certain events appeared once, but IFTTT reported multiple identical events in the same time frame. In this case, we established the SmartThings logs as the definitive source of an event versus the less precise IFTTT platform. Conversely, we discovered that SmartThings reported multiple, identical events in the same timeframe but with unique ids. This appeared to only occur with the SmartThings camera sound, person, and motion events. Future work should always examine multiple sources of events per device to identify similar implementation challenges.

5.2 Future Work

Labeling Heart-Beat (Idle) Traffic: Our work examines labeling on-demand events. We approach our solution to foster future research into identifying unintended device activity through machine learning approaches. However, we do not label the *heart-beat* or idle traffic in our dataset. Previous work has identified that heart-beat labels can be accurately generating by examining per-packet characteristics [11, 20, 31]. We consider labeling heart-beat traffic as necessary but orthogonal research and reserve it for future work.

Additional Event Sources: Our work paves the way for future classification research by proposing a holistic methodology for generating and labeling events. As we discovered that IFTTT only supported a subset of our devices, we reserve future work to examine additional sources of labels, including leveraging IFTTT Webhooks, the Alexa Skills trigger action framework, and the Apple Home ecosystem.

Dataset Accuracy: Our work examines a novel approach to generating large-scale datasets. However, this work does not measure the accuracy of the dataset past initial packet flow labels. Events generated through the companion application are guaranteed to generate packet flows since notification generation is dependent on a successful transmission from the device. Events generated remotely can fail when IFTTT itself experiences downtime. Our experiment observed a noticeable lack of both labels and flows during the final 6 hours. Only events reported through the companion application and SmartThings hub were logged. After investigating, we discovered that IFTTT reported downtime during the period of data loss. We offset the lost data by truncating the dataset end time to a point before the IFTTT downtime, but we reserve future work to automatically identify API and trigger-action framework failures by observing a lack of predictable traffic patterns.

6 RELATED WORK

A growing body of work has constructed datasets for analyzing and classifying IoT device traffic [18, 22, 24, 25, 28, 31]. We observe two critical limitations using existing datasets to classify IoT on-demand event traffic and identify unexpected device activity. First, these datasets limit the scope of labeled events or devices. Second, existing datasets broadly label traffic on coarse-grained divisions.

Ren et al. [25] is the closest approach to our work. Their work constructed a dataset of 81 IoT devices to analyze information exposure. Their work examined broadly labeling three unique on-demand events (power, voice, and video) to uncover the geolocation of destination servers for these on-demand activities. Their work is promising as it also attempts to examine the difference between idle (heartbeat traffic) and on-demand events. They make several interesting findings regarding unexpected behavior during idle traffic. Our work improves upon the set of unique labels, examining 28 unique labels across 16,686 samples over 57 devices. Further, five devices in our dataset overlap with their dataset, enabling future work to examine machine learning classifiers across unique datasets. We also discover key insights into vendor and provider dependencies over multiple vendors, an insight that parallels Ren et al.'s work into discovering privacy and jurisdiction boundaries on IoT data flows. Miettinen et al. [18] labeled manual interaction with 31 IoT devices, focusing their collection scope to the initial device setup phase. In contrast, our work examines labeling the on-demand action performed by device actuators and sensors (e.g., lock/unlocking doors, detecting motion, transmitting video.) after devices have been configured.

Trimanada et al. [31] also leveraged the IFTTT trigger-framework to generate fine-grained labeled traffic for 13 out of 18 IoT devices in their dataset. As Trimanada et al. identify, narrowing the scope to only IFTTT supported devices excludes several popular smart home devices. In Section 4, our work proposed overcoming this limitation by capturing notifications from the companion Android applications, resulting in labeling an additional 21 devices in our dataset that were unsupported by IFTTT. Perdisci et al. [24] constructed a dataset consisting of 65 devices, including multiple voice assistants, cameras, smart speaker, IoT hubs, lights, TVs, game consoles, and smart appliances. However, their work focused classification on the specific device using DNS traffic and did not attempt to label the device activity. In a similar approach, Sivanathan et al. [28] constructed a dataset of the traffic of 28 different IoT devices spanning cameras, lights, plugs, motion sensors, appliances, and health monitors over a six month period. Their work, which spanned a six-month period, provides promise about uncovering features for identifying specific devices but does not address classifying on-demand events. The Aposemat IoT-23 dataset [22], captured by the Stratosphere Laboratory, contains the network captures of three IoT devices. However, the traffic is broadly labeled as benign or malicious to support research identifying IoT botnet traffic. This approach is important but differs from our dataset, where the intent is to provide a dataset to support research classifying on-demand events to identify unexpected device activity.

7 CONCLUSION

A growing body of work exists to classify IoT device activities to identify unintended activities [1, 21, 27, 31, 32]. However, this work requires fine-grained labeled network traces with device histories. Our work is the first to present a systemic approach for labeling network traces. We examine the tradeoffs in producing device histories from vendor APIs, trigger action frameworks, and application notifications. To demonstrate this approach, we instrumented a smart home environment consisting of 57 IoT devices spanning cameras, doorbells, locks, alarm systems, lights, plugs, sensors, and hubs. We publish an open-source dataset consisting of 16,686 labeled events over 468,993 network flows. Our results indicate that vendor APIs, trigger-action frameworks, and application notifications can be used to generate scientifically valuable datasets of IoT traffic.

ACKNOWLEDGMENTS

This material is based upon work supported in whole or in part with funding from the Office of Naval Research (ONR) contract #N00014-20-1-2798. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the ONR and/or any agency or entity of the United States Government.

REFERENCES

[1] 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 *Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, Lintz, Austria, 207–218.

[2] Ahmed Alshehri, Malek Ben Salem, and Lei Ding. 2020. Are Smart Home Devices Abandoning IPV Victims?. In *Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, Guangzhoo, China, 1368–1375.

[3] Amazon Web Services. 2021. Cloud Object Storage | Store & Retrieve Data Anywhere | Amazon Simple Storage Service (S3). https://aws.amazon.com/s3/

[4] 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 *USENIX Security*. USENIX, Vancouver, Canada, 1093–1110.

[5] Dell Cameron and Dhruv Mehrotra. 2021. Ring's Neighbors Data Let Us Map Amazon's Home Surveillance Network. https://gizmodo.com/ring-s-hidden-data-let-us-map-amazons-sprawling-home-su-1840312279

[6] Jinchun Choi, Afsah Anwar, Hisham Alasmary, Jeffrey Spaulding, DaeHun Nyang, and Aziz Mohaisen. 2019. Iot malware ecosystem in the wild: a glimpse into analysis and exposures. In *Symposium on Edge Computing*. ACM, Bellevue, WA, 413–418.

[7] Salvatore Distefano, Giovanni Merlino, and Antonio Puliafito. 2012. Sensing and actuation as a service: A new development for clouds. In *International Symposium on Network Computing and Applications*. IEEE, Cambridge, MA, 272–275.

[8] Erin Dooley. 2021. ADT Technician Pleads Guilty to Hacking Home Security Footage. https://www.justice.gov/usao-ndtx/pr/adt-technician-pleads-guilty-hacking-home-security-footage

[9] Diana Freed, Jackeline Palmer, Diana Minchala, Karen Levy, Thomas Ristenpart, and Nicola Dell. 2018. "A Stalker's Paradise" How Intimate Partner Abusers Exploit Technology. In *CHI Conference on Human Factors in Computing Systems*. ACM, Montreal, Canada, 1–13.

[10] Jason Kelley and Matthew Guariglia. 2020. Amazon Ring Must End Its Dangerous Partnerships With Police. https://www.eff.org/deeplinks/2020/06/amazon-ring-must-end-its-dangerous-partnerships-police

[11] Ali Hariri, Nicolas Giannelos, and Budi Arief. 2019. Selective Forwarding Attack on IoT Home Security Kits. In *European Symposium on Research in Computer Security*. Springer, Luxembourg, September, 360–373.

[12] Weijia He, Maximilian Golla, Roshni Padhi, Jordan Ofek, Markus Dürmuth, Earlence Fernandes, and Blase Ur. 2018. Rethinking access control and authentication for the home internet of things (iot). In *USENIX Security*. USENIX, Baltimore, MD, 255–272.

[13] 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. *Interactive, Mobile, Wearable and Ubiquitous Technologies* 4 (June 2020), 46:1–46:21. https://doi.org/10.1145/3397333

[14] Blake Janes, Heather Crawford, and TJ OConnor. 2020. Never Ending Story: Authentication and Access Control Design Flaws in Shared IoT Devices. In *Security and Privacy Workshops (SPW)*. IEEE, San Francisco, CA, 104–109.

[15] Josephine Lau, Benjamin Zimmerman, and Florian Schaub. 2018. Alexa, are you listening? privacy perceptions, concerns and privacy-seeking behaviors with

smart speakers. *Human-Computer Interaction* 2 (2018), 1–31.

[16] Knud Lueth. 2020. State of the IoT 2020: 12 billion IoT connections, surpassing non-IoT for the first time. https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/

[17] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. 2017. Iot sentinel: Automated device-type identification for security enforcement in iot. In *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, Atlanta, GA, 2177–2184.

[18] M. Miettinen, S. Marchal, I. Hafeez, T. Frassetto, N. Asokan, A. Sadeghi, and S. Tarkoma. 2017. IoT Sentinel Demo: Automated Device-Type Identification for Security Enforcement in IoT. In *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, Atlanta, GA, 2511–2514. https://doi.org/10.1109/ICDCS.2017.284 ISSN: 1063-6927.

[19] Anand Mudgerikar, Puneet Sharma, and Elisa Bertino. 2019. E-Spion: A System-Level Intrusion Detection System for IoT Devices. In *Asia Conference on Computer and Communications Security (Asia CCS '19)*. ACM, New York, NY, USA, 493–500. https://doi.org/10.1145/3321705.3329857

[20] TJ OConnor, William Enck, and Bradley. Reaves. 2019. Blinded and Confused: Uncovering Systemic Flaws in Device Telemetry for Smart-Home Internet of Things. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*. ACM, Miami,FL, 140–150.

[21] 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 *Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '19)*. ACM, New York, NY, USA, 128–138.

[22] A Parmisano, S Garcia, and MJ Erquiaga. 2020. A Labeled Dataset with Malicious and Benign IoT Network Traffic.

[23] Kari Paul. 2020. Dozens sue Amazonś Ring after camera hack leads to threats and racial slurs. http://www.theguardian.com/technology/2020/dec/23/amazon-ring-camera-hack-lawsuit-threats Section: Technology.

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

[25] Jingjing Ren, Daniel J Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. 2019. Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach. In *Internet Measurement Conference*. ACM, Amsterdam, Netherlands, 267–279.

[26] Helen Rebecca Schindler, Jonathan Cave, Neil Robinson, Veronika Horvath, Petal Hackett, Salil Gunashekar, Maarten Botterman, Simon Forge, and Hans Graux. 2013. *Europe's policy options for a dynamic and trustworthy development of the Internet of Things: SMART 2012/0053*. RAND Corporation, Santa Monica, CA. https://doi.org/10.7249/RR356

[27] Aman Singh, Shashank Murali, Lalka Rieger, Ruoyu Li, Stefan Hommes, Radu State, Gaston Ormazabal, and Henning Schulzrinne. 2018. HANZO: Collaborative Network Defense for Connected Things. In *2018 Principles, Systems and Applications of IP Telecommunications (IPTComm)*. IEEE, Chicago, IL, 1–8.

[28] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2018. Classifying IoT devices in smart environments using network traffic characteristics. *Transactions on Mobile Computing* 18 (2018), 1745–1759.

[29] Vijay Sivaraman, Hassan Habibi Gharakheili, Clinton Fernandes, Narelle Clark, and Tanya Karliychuk. 2018. Smart IoT devices in the home: Security and privacy implications. , 71–79 pages.

[30] Ioannis Stellios, Panayiotis Kotzanikolaou, Mihalis Psarakis, Cristina Alcaraz, and Javier Lopez. 2018. A survey of iot-enabled cyberattacks: Assessing attack paths to critical infrastructures and services. *Communications Surveys & Tutorials* 20, 4 (2018), 3453–3495.

[31] Rahmadi Trimananda, Janus Varmarken, Athina Markopoulou, and Brian Demsky. 2020. Packet-Level Signatures for Smart Home Devices. In *Network and Distributed Systems Security Symposium*. Internet Society, San Diego, CA, 1–18.

[32] Chenggang Wang, Sean Kennedy, Haipeng Li, King Hudson, Gowtham Atluri, Xuetao Wei, Wenhai Sun, and Boyang Wang. 2020. Fingerprinting encrypted voice traffic on smart speakers with deep learning. In *Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, Linz, Austria, 254–265.

[33] Jin-Yong Yu and Young-Gab Kim. 2019. Analysis of IoT platform security: A survey. In *International Conference on Platform Technology and Service (PlatCon)*. IEEE, Jeju, Korea, 1–5.

[34] Bin Yuan, Yan Jia, Luyi Xing, Dongfang Zhao, XiaoFeng Wang, and Yuqing Zhang. 2020. Shattered Chain of Trust: Understanding Security Risks in Cross-Cloud IoT Access Delegation. In *USENIX Security*. USENIX, Virtual Event, 1183–1200.