

# IoT Inspector: Crowdsourcing Labeled Network Traffic from Smart Home Devices at Scale

ANONYMOUS

The proliferation of smart home devices has created new opportunities for empirical research in ubiquitous computing, ranging from security and privacy to personal health. Yet, data from smart home deployments are hard to come by, and existing empirical studies of smart home devices typically involve only a small number of devices in lab settings. To contribute to data-driven smart home research, we crowdsource the largest known dataset of labeled network traffic from smart home devices from within real-world home networks. To do so, we developed and released IoT Inspector, an open-source tool that allows users to observe the traffic from smart home devices on their own home networks. Since April 2019, 5,404 users have installed IoT Inspector, allowing us to collect *labeled* network traffic from 54,094 smart home devices. We demonstrate how this data enables new research into smart homes through two case studies focused on security and privacy. First, we find that many device vendors, including Amazon and Google, use outdated TLS versions and send unencrypted traffic, sometimes to advertising and tracking services. Second, we discover that smart TVs from at least 10 vendors communicated with advertising and tracking services. Finally, we find widespread cross-border communications, sometimes unencrypted, between devices and Internet services that are located in countries with potentially poor privacy practices. To facilitate future reproducible research in smart homes, we will release the IoT Inspector data to the public.

CCS Concepts: • **Networks** → *Home networks*.

Additional Key Words and Phrases: smart home, Internet-of-Things, network measurement, security, privacy

## ACM Reference Format:

Anonymous. 2019. IoT Inspector: Crowdsourcing Labeled Network Traffic from Smart Home Devices at Scale. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 37, 4, Article 111 (August 2019), 20 pages. <https://doi.org/10.1145/1122445.1122456>

## 1 INTRODUCTION

Internet-connected consumer devices, also known as smart home or Internet of things (IoT) devices, have seen increasingly widespread adoption in recent years. These new technologies create new challenges and research opportunities for ubiquitous computing. Conventional challenges include security (e.g., distributed denial-of-service attacks by IoT botnets [1]); privacy (e.g., toys transmitting sensitive information about children to third parties [2]); and device inventory and management (e.g., determining what devices are connected to a network [3]). Ultimately, data about smart home devices—and the usage of these devices—holds tremendous opportunities for understanding how people use IoT technologies and for designing new ubiquitous computing applications that rely on the collection or analysis of data from these devices.

However, this research needs large amounts of labeled data from smart home devices, which is challenging to obtain at scale for several reasons:

**(i) Scaling challenges.** According to one estimate [4], there are more than 8 billion Internet-connected devices in the world. Many of these devices are on private home networks [5]. Yet, analysis of smart home devices often

---

Author's address: Anonymous.

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2019 Association for Computing Machinery.

2474-9567/2019/8-ART111 \$15.00

<https://doi.org/10.1145/1122445.1122456>

requires either physical or local network access to the devices themselves; as a result, much of the existing work operates in small-scale lab environments [6, 7]. Researchers have occasionally deployed custom hardware in consumer homes to gather information about devices in homes [8, 9], but these types of deployments often require significant effort, since they require users to install a (sometimes costly) physical device on their home networks. Another approach is to scan the Internet for exposed devices [1]. However, this approach omits devices behind gateway routers that act as network address translators (NATs).

**(ii) Labeling challenges.** Absent a large corpus of ground-truth device labels, researchers often can infer the identities of only a limited set of devices [10]. Researchers have previously published analyses of proprietary data from Internet-wide scans [5], but these datasets are not public and typically do not have specific or reliable device labels.

These limitations make it difficult to carry out empirical ubiquitous computing research based on data from real smart homes, ranging from measurements of security/privacy violations in the wild [6, 7] to training machine learning algorithms for modeling device behaviors [11, 12] or inferring device identities [3, 13].

**Our solution.** We have constructed the largest known dataset of labeled smart home network traffic by developing and releasing an open-source application, *IoT Inspector*, that crowdsources the data from within home networks in a way that respects user privacy (Section 3). Our Institutional Review Board (IRB) has approved the study. Since we released the software on April 10, 2019, *IoT Inspector* has collected network traffic from 5,404 global users and 54,094 devices, 12,993 of which have user-provided labels. We have validated the correctness of these labels against external information; we discuss the challenges of label validation and our validation approach in Section 5. The data are also still growing, as users are actively downloading and using *IoT Inspector* at the time of writing.

To demonstrate the potential of the *IoT Inspector* dataset for ubiquitous computing research, we analyzed a subset of the data collected between April 10, 2019 and January 21, 2020 to study the pervasiveness of two security & privacy issues plaguing smart homes: incorrect encryption of network traffic and communication with third-party advertisers and trackers. During this period, 50% of *IoT Inspector* users collected at least 35.0 minutes of traffic each, and 25% of these active users collected at least 3.2 hours of traffic each. Additionally, 2,465 users manually labeled the identities of 12,993 devices (Section 4). Using this dataset, we find that many device vendors, including Amazon and Google, use outdated TLS versions and send unencrypted traffic, sometimes to advertising and tracking services (Section 6.1). Also, we discover that smart TVs from at least 10 vendors communicated with advertising and tracking services. Finally, we find widespread cross-border communications, sometimes unencrypted, between devices and Internet services that are located in countries with potentially poor privacy practices (Section 6.2).

This unique dataset will enable many types of research that have otherwise suffered from limited scale and labels. Similar to how ImageNet [14] advanced the field of computer vision, we hope to contribute to smart home research by providing our data to expand the scope of empirical analyses and develop more generalizable or realistic models. Since we released *IoT Inspector*, seven research groups have contacted us to ask about using the data in a wide variety of ways (Section 4.4).

In summary, to crowdsource smart home IoT traffic and labels at scale, we developed *IoT Inspector* to be a usable tool to incentivize usage across a large population. In doing so, we have made the following contributions:

- ❖ We have crowdsourced the largest known dataset of labeled, real-world, smart home network traffic and device labels using *IoT Inspector*, an open-source tool that we developed to help gather this data at scale. We have also developed algorithms to automatically standardize and validate these crowdsourced labels.
- ❖ Through an initial analysis of the dataset, we discovered widespread security and privacy with smart home devices, including insecure TLS implementation and pervasive use of tracking and advertising services.

Such information is uniquely available in the IoT Inspector dataset. In addition to these preliminary case studies, we highlight other types of ubiquitous computing research that can use the dataset.

- ❖ **We make the dataset available to interested researchers** (Section 4.4). This includes the anonymized network traffic (in the form of ⟨device identifier, timestamp, remote IP or hostname, remote port, protocol⟩) and device labels (in the form of ⟨device identifier, category, vendor⟩).

## 2 RELATED WORK

We first discuss existing techniques to obtain large, labeled traffic datasets and their relation to IoT Inspector (Section 2.1). We then describe previous and ongoing smart home studies that could benefit from a large-scale, labeled dataset such as the one IoT Inspector has collected (Section 2.2).

### 2.1 Crowdsourcing labeled traffic datasets at scale

Existing techniques to obtain labeled network traffic at scale face multiple challenges. In particular, lab studies are restricted to a small set of devices [6, 7], while Internet-scanning omits devices behind NATs and often produces limited device labels [1, 10].

*Hardware-based approaches.* We design IoT Inspector to crowdsource the network traffic and labels of smart home devices from a large user population, following in the footsteps of a number of previous crowdsourcing studies. For example, multiple researchers have deployed custom routers to collect the participants' home traffic: Chetty et al. [15] developed Kermit, a router-based tool, to help users diagnose slow network connectivity. BISmark [8, 16] collected network performance characteristics through deploying routers in home networks. NetMicroscope [9] analyzed the quality of video streaming services through custom routers in participants' home networks. However, unlike IoT Inspector, the hardware-based approaches used in these studies are difficult to scale to more users due to the cost of hardware and shipment.

*Software-based approaches.* We are not aware of any software tools other than IoT Inspector that collect smart home traffic at scale. Netalyzer [17] was a web-based application that helped users analyze home network performance and also gathered network statistics from 99,000 different public IP addresses. DiCioccio et al. [18] developed HomeNet Profiler [19] to explore how effectively UPnP could be used to measure home networks. While software tools are typically easier to deploy than hardware routers, most such tools have actively probed the home network (e.g., by performing a “scan”) rather than passively collecting traffic.

IoT Inspector combines the benefits of hardware and software data collection platforms. By designing IoT Inspector as a software tool, we avoid some of the deployment barriers that router-based studies face. We also develop IoT Inspector to behave like a router and intercept network traffic via ARP spoofing (Section 3.1), thereby building a dataset of smart home network traffic at scale. Furthermore, we draw inspiration from Netalyzer [17] and design IoT Inspector to benefit users, with the goal of promoting participation and user engagement (Section 3.4). At the same time, we make user privacy our first-order concern (Section 3.3) much as in previous work [8].

### 2.2 Smart home research that could benefit from IoT Inspector data

The increasing prevalence of smart home devices has spurred researchers to investigate a variety of problems using empirical methods. These studies have typically relied on either small-scale laboratory-based data, or proprietary datasets.

*Discovering security and privacy violations.* Past work has explored security and privacy issues of a small set of smart home devices in lab settings. Chu et al. and Sasha et al. [2, 20] found a variety of security flaws in IoT children's toys; Wood et al. [7] found cleartext health information in home IoT medical device communications; and Acar et al. [21] presented web-based attacks on smart home devices that host local web servers, demonstrating

their real-world applications on seven home IoT devices (e.g., Google Home and Chromecast). A larger dataset of labeled network traffic enable the study the problems across a much wider array of devices and vendors.

Other studies have relied on data from actively “scanning” devices on the Internet or in home networks. Antonakakis et al. [1] scanned the Internet and identified public-facing devices compromised by the Mirai botnet; Kumar et al. [5] used a proprietary dataset from an antivirus company to discover vulnerable devices within home networks. Despite the scale of these studies, researchers do not have reliable labels of device types and vendors; rather, they could only *infer* the device identities using a variety of signals (e.g., based on HTTP response headers [10], default passwords [1], or proprietary rules [5]). In contrast, lab studies permit knowledge of device types and vendors but are limited to much smaller scale. IoT Inspector allows the collection of a large, labeled dataset of network traffic from devices that are deployed in real networks.

*Modeling device activities.* Other past work has applied machine learning identify unknown devices [3, 13] and detect anomalous behaviors [11, 12]. These studies used a small number of devices in lab settings for training and validation. It is unclear, however, whether the models would be equally effective if tested in real-world settings, with a larger set of devices as being used by real humans.

### 3 CROWDSOURCING SMART HOME NETWORK TRAFFIC AT SCALE

In this section, we describe the design and implementation of IoT Inspector to crowdsource labeled network data at scale. We developed IoT Inspector, an open-source tool that consumers can download on their computers at home to analyze the network activities of their smart home devices. To attract users to participate in this crowdsourcing effort, we designed IoT Inspector in a way that makes setup easy; our goal was to make the application as close to “one click” as possible. Users can run IoT Inspector on macOS- and Linux-based computers<sup>1</sup> without dedicated hardware such as custom routers. Furthermore, we designed IoT Inspector to promote user engagement by showing a real-time analysis of their smart home network traffic on the user interface, which allows users to identify potential security and privacy problems. With user consent, IoT Inspector anonymizes and uploads the network data and device labels to our server, where we preprocess the data for researchers to analyze.

#### 3.1 Designing a software tool to capture traffic

Many home network measurement platforms [8, 9] require participants to first obtain custom routers to collect network traffic. This requirement, however, limits the scale of such studies due to the cost of hardware and shipment.

To minimize the setup cost and facilitate deployment at scale, we design IoT Inspector to be a software-based data collection tool that users can install in a relatively small number of steps. First, the user needs a macOS- or Linux-based computer that is connected to the smart home network. From IoT Inspector’s website, the user can download the precompiled executable or the source code.<sup>2</sup> The executable includes all necessary platform-dependent libraries, so that the user can launch IoT Inspector without needing to install additional packages. When a user runs IoT Inspector for the first time, it displays a consent form—approved by our university’s IRB—that details what data IoT Inspector collects and that IoT Inspector poses no more than minimal risk to users.

Upon the user’s consent, IoT Inspector automatically discovers devices on the network and captures traffic from select devices, as outlined below:

<sup>1</sup>The Windows version is still under development at the time of writing.

<sup>2</sup>Users have to accept macOS’s warning that the app is not from the official AppStore. We cannot submit the app to the AppStore because it does ARP spoofing.

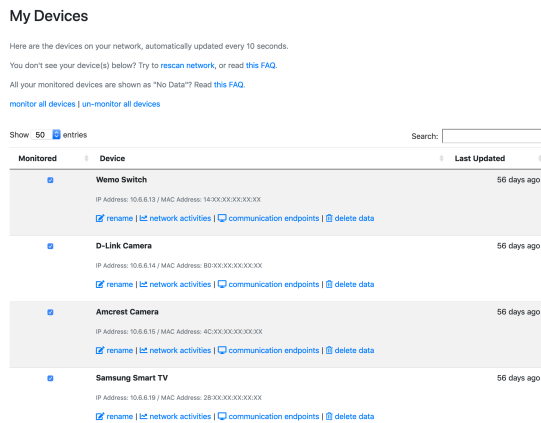


Fig. 1. A screenshot of IoT Inspector’s user interface that shows a list of devices on the network.

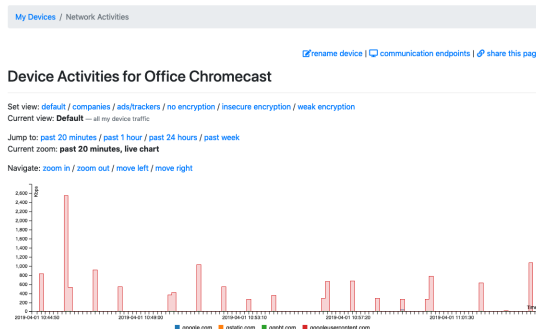


Fig. 2. A screenshot of bandwidth usage for individual endpoints on a Chromecast device.

*Discovering devices via ARP scanning.* Upon launch, IoT Inspector automatically sends out ARP packets to all IP addresses in the local subnet to discover devices. At the same time, IoT Inspector opens the user interface (UI) in a browser window that shows a list of device IP addresses and MAC addresses currently on the network. We show an example screenshot of this UI in **Figure 1**. From this UI, users have to explicitly indicate which of the listed devices IoT Inspector is allowed to monitor (i.e., collect traffic). To help users choose what devices to monitor, IoT Inspector also displays the *likely* identities of individual devices, using external data sources such as the IEEE Organizationally Unique Identifier (OUI) database (which shows the manufacturers of network chips [22]) along with mDNS and SSDP announcements that may include a device’s identity (as collected by Netdisco [23]).

*Capturing traffic via ARP spoofing.* By default, IoT Inspector only ARP scans the network to discover devices. For IoT Inspector to capture any device traffic, a user would have to explicitly indicate which device(s) to monitor from the device list (**Figure 1**).

For each monitored device, IoT Inspector continuously sends two ARP spoofing packets every two seconds, similar to Debian’s arpspoof utility [24]: one packet is sent to the monitored device using the IP address of the router as the source, and one packet is sent to the router using the IP address of the monitored device as the source. In this way, IoT Inspector can intercept all traffic between the monitored device and the router.

The ARP spoofing packets are unlikely to consume significant bandwidth, although network latency is likely to be increased due to packets taking extra hops to go through IoT Inspector. Each ARP packet is typically 42 bytes. If there are  $N$  monitored devices (excluding the router) on the local network, then IoT Inspector would need to send out  $2N$  packets every two seconds, or  $42N$  bytes per second. In a home network of 50 devices (which is the upper limit for IoT Inspector by default), for instance, the bandwidth overhead would be 2.1 Kilobytes/second.<sup>3</sup>

Through a combination of ARP scanning and spoofing, IoT Inspector is able to discover devices and capture their traffic in a way that requires minimal user engagement and no dedicated hardware. Using this captured traffic, we can generate a dataset for research (Section 3.2) and promoting user engagement (Section 3.4).

*Limitations.* The current implementation of ARP spoofing only captures traffic between monitored devices and the router. However, smart home devices could be communicating with one another. In future versions, IoT Inspector will send out spoofed ARP packets to not only the targeted device and the router, but also to

<sup>3</sup>We tried running two instances of IoT Inspector in the lab to inspect separate sets of devices; both instances collected traffic normally.

the targeted device and any other devices being monitored. Excluding the router, if a user monitors  $N$  devices,  $D_1, D_2, \dots, D_N$ , IoT Inspector will send out 2 spoofed packets between  $D_i$  and the router for  $i = 1, \dots, N$ , followed by 2 spoofed packets between  $D_i$  and  $D_j$  for  $i = 1, \dots, N$  and  $i < j \leq N$ . In this way, IoT Inspector will send out a total of  $2(N + (N - 1) + (N - 2) + \dots + 1) = (N + 1)N$  packets every two seconds and result in a bandwidth overhead of 53.6 Kilobytes/second for  $N = 50$ .

### 3.2 Collecting network traffic and device labels

IoT Inspector collects two types of data: network traffic and device labels.

*Network traffic.* Users choose which devices to *monitor* (**Figure 1**), such that packets to and from the monitored devices are captured by the computer that runs IoT Inspector. IoT Inspector parses the relevant fields from the captured packets using the Scapy Python library, removes sensitive information, and uploads the resulting data to the database server at five-second intervals. Specifically, IoT Inspector collects the following data:

- ❖ SHA-256 hashes of device MAC addresses, using a secret salt<sup>4</sup> that IoT Inspector randomly generates upon first run.
- ❖ Manufacturer of the device’s network chipset, based on the first 3 octets of the MAC address (i.e., OUI).
- ❖ DNS requests and responses.
- ❖ Remote IP addresses and ports.
- ❖ Aggregated flow statistics, such as the number of bytes sent/received over five-second windows.
- ❖ Data related to device identities, including SSDP/mDNS/UPnP messages, HTTP User-Agent strings, and hostnames from DHCP Request packets, that are useful for validating device identity labels entered by users (Section 5).
- ❖ TLS Client Hello messages.
- ❖ Timezone of the computer running IoT Inspector

*Device labels.* Recorded network traffic alone is typically insufficient for research, as it is often necessary to characterize network activities that are specific to certain models or types of devices. IoT Inspector therefore asks users to voluntarily label their devices’ identities. From IoT Inspector’s UI, users can enter the name (e.g., “Roku TV”), category (e.g., “TV”), and vendor (e.g., “Roku”) for one or more of their devices. IoT Inspector provides dropdown textboxes with auto-complete capabilities, so that users can select from a list of known labels. If, on the other hand, the desired labels are not in the dropdown lists, users can enter free text.

IoT Inspector uploads the device labels along with the network traffic data to a central database hosted at our institution. We use this dataset to investigate two security and privacy issues of smart home devices within and across device categories and vendors (Section 6).

### 3.3 Protecting privacy of others in household

The design of IoT Inspector, along with our data collection, storage, and retention policies/practices, has been approved by our university’s IRB. We follow industry-standard security and privacy practices. For example, each instance of IoT Inspector uploads the captured network data to our central server via HTTPS, and we store this data on a secure, fully updated server hosted on our institution’s network. IoT Inspector only collects the data outlined in Section 3.2.

Nonetheless, the data collected may inadvertently contain sensitive information. As such, we designed IoT Inspector to allow a user to retroactively remove select data. For example, a device could be involved in sensitive

<sup>4</sup>IoT Inspector does not share the secret salt with us.



activities, or a user may have accidentally monitored a medical device; in this case, the user can delete all data associated with this device from our server directly through IoT Inspector’s UI. Additionally, collected DHCP or SSDP messages may include a user’s identity (e.g., “Joe’s TV”); in this case, the user can have IoT Inspector remove all DHCP and/or SSDP messages from a specific device from our server.

Furthermore, IoT Inspector may pose privacy risks to other people on the same network who do not use IoT Inspector. Although ARP spoofing makes it easy for IoT Inspector to start capturing traffic, this design could also potentially make it easy for a user to analyze sensitive activities of other people on the same network.

To increase the barrier of such malicious activities, we design IoT Inspector such that it does not upload any traffic from devices that show signs of being general-purpose computing devices, such as phones, tablets, or computers. We make this determination based on two data sources: (i) the HTTP User Agent string (which is often missing due to reduced adoption of unencrypted HTTP); and (ii) the FingerBank API [25], a proprietary service that takes as input the first 3 octets of a device’s MAC address, along with a sample of five domains contacted, and outputs the device’s likely identity. By parsing this output and looking for specific keywords such as “phone,” “macOS,” “Android,” or “Windows,” we infer whether the device is potentially a smart home device or a general purpose computer.

It is possible that IoT Inspector may mistake an actual smart home device for a computer (e.g., due to false positives of FingerBank’s API). Users can manually correct this mistake by following the instructions on the IoT Inspector UI and entering the device’s MAC address (e.g., often printed on the device itself, or displayed on the settings menu if the device has a screen), thereby demonstrating that the user likely has physical access to the device. We admit, however, that this design merely increases the barrier for a malicious user; it does not completely prevent an advanced user from ARP-scanning the network, obtaining the MAC address of a targeted device, and bypassing this protection. At the same time, it is worth noting that a user who is advanced enough to bypass IoT Inspector’s protections is likely sophisticated to perform network traffic capture and monitoring *without the help of IoT Inspector in the first place*.

### 3.4 Keeping users engaged

Our goal is to not only make this dataset useful to researchers; it should also provide users with insight on their smart home networks. This draws from Netalyzr’s experience that providing benefits to users also boosted user count and engagement [26].

To this end, we set up an automated script on the server to preprocess the data collected, produce tables and charts in real time, and push these visualizations back to the front end for users to discover potential security and privacy issues.

*Preprocessing data.* The data preprocessing pipeline involves two steps: (1) aggregating the collected packets into flows (i.e., same source and destination IP addresses, ports, and protocol) at 5-second intervals; and (2) identifying the remote endpoints that communicate with the monitored devices.

We identify remote endpoints because, by default, each packet collected only shows the communication between a monitored device and some remote IP address. An average user, however, may not know which real-world entity is associated with the IP address. As such, our automated script attempts to first find the hostname that corresponds to the remote IP address based on past DNS responses or the SNI field from previous TLS Client Hello messages. It is possible that one or both of DNS and SNI might be missing in the dataset; for example, a user could have started running IoT Inspector after the relevant DNS and SNI packets were sent/received and thus IoT Inspector would fail to capture these packets. In the case where we do not observe DNS or SNI data for a particular IP address, we infer the hostname based on passive DNS [27] or reverse DNS (i.e., PTR records).

It is possible that the hostnames themselves may not always be indicative of the endpoint’s identity. For instance, an average user may not know that `fbcdn.net` is a Facebook domain. To help users learn about the identities of

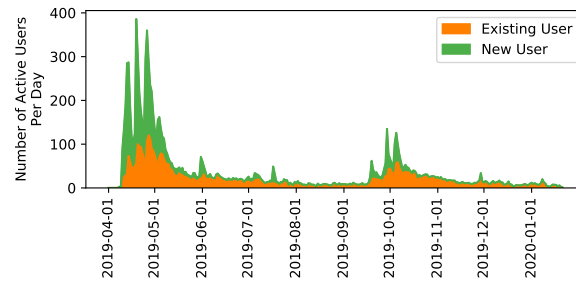


Fig. 3. A stacked area chart showing the number of active users every day. All dates are based on UTC.

endpoints, we use the webXray Domain Owner List to turn hostnames into human-readable company names [28]. We also use the Disconnect list to label hostnames that are known to track users or serve advertisements [29]. We further complement this information with a manually compiled database of common ports; for instance, if a device contacts an IP address with destination port 123, IoT Inspector shows the user that the remote service is likely an NTP time server.

*Presenting data.* After the automated script labels each remote endpoint, it generates tables and charts for the UI in real time. Each table shows the hostnames and/or companies that a device has communicated with since the user launched IoT Inspector, e.g., **Figure ??**. In contrast, each chart shows the bandwidth usage by connections with individual endpoints from a given device over time, e.g., **Figure 2**.

Our primary goal is for users to learn new insights about their devices, such as what third parties a device communicates with and when devices send and receive data. Our hope is that these insights also encourage more users to install IoT Inspector and keep running IoT Inspector to monitor more devices—effectively generating a larger dataset for research.

## 4 DATASET

On April 10, 2019, we announced the release of IoT Inspector with the help of social media. In particular, we first set up a website where the public would be able to download IoT Inspector’s executable or source code, along with documentation on how the software works, how it collects the data, and how we use the data. We host the website on a subdomain under our academic institution (i.e., in the form of <https://iot-inspector.institution.edu>). We also published Twitter posts that linked to the website. Subsequently, we observed articles about IoT Inspector from a number of media outlets, including three US-based technology websites (i.e., Gizmodo, TechCrunch, and HackerNews) and the Canadian Broadcasting Corporation.

As of January 21, 2020, IoT Inspector’s dataset includes 10,559 users who have scanned 182,630 devices. Some 5,404 of these users (i.e., *active users*, based on the definition in Section 4.1) have allowed IoT Inspector to capture network traffic from 54,094 of the devices. These numbers are still growing, as IoT Inspector is actively gathering labeled traffic data.

For this paper, we analyze a subset of the data collected between April 10, 2019 and **January 21, 2020**. This section describes aggregate statistics about the users and devices during this study period.

### 4.1 User statistics

*All users.* Every time IoT Inspector is installed, it generates and writes to disk a random User ID that persists across computer reboots. During our study period, IoT Inspector collected 10,559 such unique IDs, which we take as the number of IoT Inspector users.



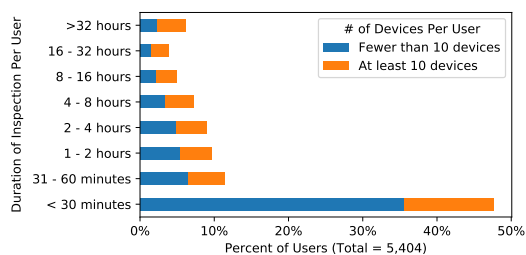


Fig. 4. A stacked bar chart showing the duration of inspection per user for two different user groups: one group where each user inspected fewer than 10 devices, and one group where each user inspected at least 10 devices.

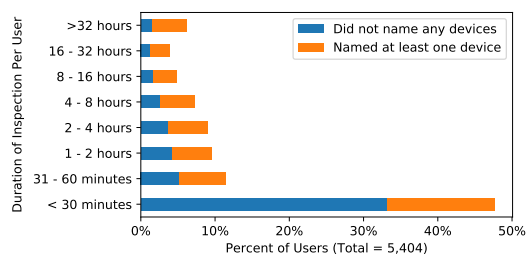


Fig. 5. The same stacked bar chart for two different user groups: one group where each user labeled at least one device, and one group where each user did not enter any labels.

*Active users.* Not every instance of IoT Inspector uploaded network traffic data to our collection server. In fact, only 5,404 users (51.2% of all users) uploaded some network traffic; we consider these users as *active users*. We fail to observe any network traffic from the non-active users because some home routers may have dropped ARP-spoofing traffic, our data collection server may have been temporarily out of service due to high load,<sup>5</sup> or the users may not have consented to data collection. For the rest of the paper, we only look at active users and devices from which these active users captured network traffic.

We observe a steady number of active users every day. During the first two months of release, there was an average of 125.0 active users every day. We show the distribution of the number of daily active users in **Figure 3**; in this stacked area chart, a “New User” on Day  $t$  means the user started using IoT Inspector on Day  $t$ , whereas an “Existing User” on Day  $t$  suggests that the user had started using IoT Inspector prior to Day  $t$ . We note that the number of active users tended to be higher around weekends (e.g., Saturdays April 13, 20, and 27). The local maxima values in the chart (e.g., around June 1, 2019 and Oct 1, 2019) correlated with media coverage of IoT Inspector.

Many of these active users ran IoT Inspector for a limited duration. Half of the active users collected at least 35.0 minutes of network traffic, and 25% of the active users at least 3.2 hours of traffic. We show a distribution of the duration of inspection for each each in **Figure 4**.

These users are likely from around the world. Even though IoT Inspector does not collect users’ public IP addresses, we can still infer their geographical distribution based on each user’s timezone. In particular, the timezones for 62.5% of the users are between UTC -07:00 and -04:00 (e.g., between San Francisco and New York), while for 27.8% of the users the timezones are between UTC 00:00 and 03:00 (between London and Moscow).

Although we do not collect the identities of users, a small number of users have identified themselves, including journalists from the Washington Post, New York Times, and National Public Radio, who independently used IoT Inspector to investigate the security and privacy of smart home IoT products. A government agency, New York City Cyber-Command, notified us that their staffers were also using IoT Inspector.

## 4.2 Device statistics

*All devices.* Upon launch, IoT Inspector scans the local network and presents the user with a list of devices on the network. From April 10 to Jan 21, 2020, IoT Inspector discovered 182,630 devices in total.<sup>6</sup> For each of these devices, IoT Inspector only collects the OUI and mDNS (if available) data.

<sup>5</sup>IoT Inspector stops collecting data if it is unable to contact the remote data collection server.

<sup>6</sup>For performance reasons, IoT Inspector discovers at most 50 devices per user.

*Devices from which IoT Inspector collected traffic.* By default, IoT Inspector does not collect network traffic from any device unless the user explicitly chooses to monitor it (**Figure 1**). As such, IoT Inspector only collected network traffic from 54,094 of the discovered devices during the study period across the 5,404 active users. On average, each active user inspected 10.0 devices. In the 25th, 50th, and 75th percentile, an active user inspected 3.0, 7.0, and 14.0 devices respectively. Users who each inspected 10 or more devices were associated with longer durations of inspection compared with users who inspected fewer than 10 devices, as illustrated in **Figure 4**.

*Devices with traffic and labels.* For the majority of the monitored devices, we do not have any user-entered device name, category, and vendor labels. In total, only 12,993 devices (7.1% of all devices, 24.0% of all monitored devices) have such labels, as entered by 2,465 users (45.6% of the 5,404 active users). As illustrated in **Figure 5**, users who entered the device name, category, and vendor labels were associated with longer durations of inspection compared with users who did not enter any such labels.

### 4.3 Interaction data

IoT Inspector’s web interface allows users to view the network activities of inspected devices, either in the form of a table showing the domains contacted or a chart showing bandwidth usage over time (e.g., **Figure 2**). In both cases, users can filter the view of network activities (e.g., as exemplified in the “Set view” row of buttons in **Figure 2**) in the following ways: “default,” which is the default setting; “companies,” which shows network activities labeled with known companies (e.g., traffic to “fbcdn.net” labeled as Facebook traffic); “ads/trackers” for third-party advertising and tracking services; “no encryption,” for unencrypted HTTP traffic on port 80; “insecure encryption” for outdated TLS versions (e.g., SSL 3.0 or TLS 1.0); and “weak encryption” for TLS connections with weak ciphersuites (e.g., RC4) in the TLS ClientHello messages. In each of these views, the web interface automatically updates the corresponding chart or table every 15 seconds through an asynchronous Javascript call.

By examining such asynchronous Javascript calls in the HTTP server log, we estimate the total number of hours that users spent in each view. The “default” view, for example, is associated with 29,168 hours of user activities, whereas the non-default views were associated with only 273 hours. These statistics suggest that in most cases users opted for the default view.

For the 273 hours of non-default view, users spent the most time (89 hours) in the “ads/trackers” view, followed by the “no encryption” view (72 hours). Users spent the least time (23 hours) in the “insecure encryption” view. These values suggest that users appear to be the most interested in the advertising and tracking services that communicated with their devices followed by insecure connections.

### 4.4 Data release

Interested researchers can contact us to access the IoT Inspector dataset as the following CSV files:

- ❖ *Device\_labels.csv*. Columns: device identifier, category, and vendor
- ❖ *Network\_flows.csv*. Columns: device identifier, timestamp of first packet, remote IP/hostname, remote port, protocol (i.e., TCP or UDP), number of bytes sent in a five-second window, and number of bytes received in the window.
- ❖ *TLS\_client\_hello.csv*. Columns: device identifier, timestamp, TLS version, cipher suites, and TLS fingerprint (Section 6.1).

To date, research groups from seven research institutions have contacted us to request access to the dataset in a variety of ways, including: training machine learning models for device identification and anomaly detection; measuring the security risks of IoT messaging systems such as MQTT [30]; identifying third-party services in the smart home ecosystem; and inferring and labeling human activities to understand the privacy risks of devices.

## 5 LABELING SMART HOME DEVICES

As with crowdsourced data, the user-entered device name, category, and vendor labels are likely inconsistent and may be prone to mistakes. In this section, we first *standardize* these labels by resolving inconsistencies such as typographical errors or different names for the same device (Section 5.1). We then describe how we validate the correctness of the standardized labels (Section 5.2).

### 5.1 Standardizing device labels

As described in Section 3.2, users can manually label a device with its name, vendor, and category. Before we can check the correctness of these labels, we note that the labels could be inconsistent. First, a user could mis-spell certain words in the labels; in our dataset, for example, 292 devices are labeled with the vendor name “philips” (correct spelling), but 85 devices are labeled as “phillips” and 7 devices as “philip’s.” Another source of inconsistencies comes from multiple ways of naming the same type of devices. For example, Google Home has multiple models, such as Google Home and Google Home Mini, which users could consider as either a smart speaker or voice assistant. Similarly, a vendor could also be labeled in different ways. For example, Ring, a surveillance camera vendor, was acquired by Amazon, so Ring devices are actually Amazon devices. Also, we have seen instances of Logitech Harmony Remote Controllers labeled as both “logitech” and “logitech harmony” as the vendor, while in reality the vendor should be “logitech” alone.

To mitigate these inconsistencies, we first run the following algorithm to standardize the device vendor labels. **(1)** We conduct an initial clean-up of the user-entered vendor labels by removing extra spaces and non-ASCII characters and by transforming into lower cases. **(2)** We merge possibly equivalent vendor labels, which likely are a result of mis-spellings (e.g., “phillips” and “philips”) or being too specific (e.g., “logitech harmony” instead of “logitech”). To do so, we first sort the vendor labels by the number of devices where each label appears in descending order. For a label at position  $i$ , we compare it with each label at position  $j$  for all  $j > i$  (i.e., which appears in fewer number of devices). We rename the label at  $j$  as the label at  $i$  if either of the two conditions is satisfied: The Levenshtein distance between the two labels is one, or the label at  $i$  is a substring of the label at  $j$ . We assume that the label at  $i$  is the correct label since more devices are labeled as such. **(3)** We manually merge equivalent vendor labels based on our human knowledge, e.g., renaming all instances of “ring” as “amazon”. **(4)** We manually remove vendors that are not known to manufacture smart home IoT products, such as Asus, a computer maker.

After we standardize the vendor labels, we run another algorithm to standardize the device names for each vendor. **(1)** For each device, we concatenate the device name and category labels into a single label, because the name alone may not help us understand what type of devices in our subsequent analysis. For each of these concatenated labels, we remove extra spaces and non-ASCII characters and we transform it into lower cases. **(2)** We remove common stop words such as “an” and “the.” We also remove IoT-specific stop words, such as “smart” and “device.” **(3)** We merge possibly equivalent device name labels using the same method as the second step above (where we automatically merge vendor labels). **(4)** We manually merge equivalent device names based on our human knowledge, e.g., renaming all instances of “outlet” as “switch” and “bulb” as “light”. **(5)** We manually remove devices whose names suggest they are not smart home IoT devices, e.g., names containing words such as “computer,” “phone,” or “tablet.” Although IoT Inspector automatically deters users from inspecting what appears to be non-IoT devices (as we described in Section 3.3), a user could override this action by entering the MAC address of the device of interest, so it is possible that our dataset includes network traffic associated with computers or phones.

We apply these algorithms to 12,993 devices that users have inspected, and labeled by assigning device names, categories, and vendors. This produces a list of 7,942 devices across 92 vendors whose device name and vendor labels have been standardized.

<i>Method</i>	<i>Validated</i>	<i>Not Validated</i>	<i>No Data</i>
Common Top Domain	76.0%	21.3%	2.7%
Vendor-in-Domain	66.4%	30.9%	2.7%
Netdisco	38.7%	11.3%	49.9%
OUI	22.6%	69.6%	7.8%

Table 1. Outcomes of validation for the 7,942 devices (across 92 vendors) whose name and vendor labels have been standardized in Section 5.1. Each row sums to 100.0%.

## 5.2 Validating device labels

The standardization process above is based on the original name, category, and vendor labels as entered by users. Still, these labels can be completely incorrect; for instance, we have seen a device being labeled as a Belkin smart switch, but it communicated with a large number of endpoints that are commonly associated with Android devices, rather than a small number of endpoints as we expect based on our observation in the lab and on other Belkin devices in the dataset.

In this section, we describe a method to validate the standardized device vendor labels. We leave the validation of device name labels to future work, because most of the analyses in this paper focus on aggregate behaviors of devices across vendors. Moreover, different types of devices of the same vendor may communicate with similar Internet endpoints, such as Google Chromecast and Google Home based on our observation in the lab. While we could analyze the characteristics of network traffic over time (e.g., number of bytes sent/received) to distinguish among different types of devices of the same vendors, such analysis is beyond the scope of this paper.

*Validation methods.* We develop a method to automatically validate a device’s vendor label with the following four methods, each of which produces three outcomes of validation: Validated, Not Validated, and No Data.

**Common Top Domain:** For vendors with  $N$  devices (where  $N \geq 10$ ), we extract the *registered domains* that each device contacted based on our dataset; for instance, the registered domain for “wiwo.orvibo.com” is “orvibo.com.” We count the number of devices where each of the registered domains appears. For the registered domain that appears in the most number of devices, if it appears in more than  $N/2$  devices, we call it the *common domain*.<sup>7</sup> If a device of a given vendor communicated with the common domain, we consider the vendor label as Validated. If we do not observe any registered domains or if the vendor is associated with fewer than 10 devices, we denote the validation outcome as No Data.

**Vendor-in-Domain** For a device whose standardized vendor is  $V$ , for each registered domain  $D$  that it communicated with, if  $V$  is a substring of  $D$ , then we consider  $V$  as validated. For example, Philips devices tend to communicate with “philips.com,” based on our in-lab observations. One caveat to this approach is that not all vendor-specific domains include the name of the vendor; for instance, Belkin smart switches communicate with “xbcs.net,” based on our in-lab observations. While the Belkin vendor label will not be considered as validated under this method, it is considered as validated under the Common Domain method because “xbcs.net” appears in the network traffic of the most number of Belkin devices. If we do not observe any registered domains for a given device, we denote the validation outcome as No Data.

**Netdisco:** IoT Inspector regularly scans the user’s network with Netdisco, an open-source library that discovers smart home devices on the local network using SSDP, mDNS, and UPnP protocols [23]. The library parses any subsequent responses into JSON strings. These strings may include a device’s self-advertised name (e.g., “google\_home”). If a device’s vendor label is a substring of Netdisco’s scan result for this device, we consider the vendor as Validated. If the device does not respond to Netdisco’s scans, we denote the validation outcome as No Data.

<sup>7</sup>We exclude “ntp.org” from this method because many devices of different vendors communicate with ntp.org to presumably synchronize their clocks.

<i>Device Vendors</i>	<i># of Devices</i>	<i># of Users</i>	<i>Most Common Names</i>
google	1,272	644	google-home (46.5%), chromecast (30.0%)
amazon	1,076	632	echo (65.6%), fire (17.0%)
sonos	432	198	speaker (71.3%), play (15.7%)
philips	355	355	hue (98.3%), hub (0.8%)
belkin	247	123	switch (98.4%), netcam (0.8%)
wyze	216	108	camera (96.3%), light (1.4%)
roku	191	155	streamer (49.2%), stick (14.7%)
logitech	168	154	harmony (79.2%), camera (13.7%)
hp	154	152	printer (95.5%), officejet (1.9%)
ecobee	147	122	thermostat (95.2%), switch (4.1%)

Table 2. Vendors with the most number of devices, along with two examples of device names under each vendor. In total, there are 6,776 devices across 81 vendors.

**OUI:** We extract the OUI from the first three octets of a device’s MAC address. We translate OUIs into English names of manufacturers based on the IEEE OUI database [22]. If a device’s vendor label is a substring of the OUI-based manufacturer, then we consider the vendor as Validated. If the IEEE OUI database does not have a corresponding entry for the OUI, we denote the validation outcome as No Data.

*Outcomes of validation.* For the 7,942 devices (across 92 vendors) whose names and vendors we have standardized in Section 5.1, we validate their labels using the four methods above. We show a distribution of the validation outcomes in **Table 1**. Of the 7,942 devices, 85.5% of the devices are marked as Validated using at least 1 method, 72.3% at least 2 methods, 30.1% at least 3 methods, and 15.7% all four methods. Examples of devices that are validated using all four methods include 75.7% of Sonos devices (i.e., smart speakers), and 65.6% of all Roku devices (i.e., smart TVs and TV sticks). In contrast, all 9 Koogee smart light strips are marked as Not Validated in any of the methods.

One key takeaway from **Table 1** is that there are trade-offs between the availability of a validation method and its effectiveness. For example, the Netdisco method is available on fewer devices than both Domain methods, but Netdisco is able to validate a higher percentage of devices where data is available. This result suggests that although Netdisco data is less prevalent than domain data, Netdisco is more effective for validating device labels.

Despite their availability now, domains may not be the most prevalent data source for device identity validation in the near future, because domain names will likely be encrypted. In particular, DNS over HTTPS or over TLS is gaining popularity, making it difficult for an on-path observer to record the domains contacted by a device. Moreover, the SNI field—which includes the domain name—in TLS Client Hello messages may be encrypted in the future [31]. These technological trends will likely require future device identification techniques to be less reliant on domain information.

In total, 6,776 devices across 81 vendors have their vendor labels validated using at least one of the methods. For the rest of the paper, we will examine these 6,776 devices only. We show examples of these devices in **Table 2** and **Table 3**.

## 6 NEW POSSIBILITIES WITH LABELED TRAFFIC DATA AT SCALE

IoT Inspector’s device traffic and labels dataset – the largest of its kind that we are aware of – can create new research opportunities in diverse areas. In this section, we show two examples of questions that we can answer using our data: whether smart devices encrypt their traffic using up-to-date TLS implementations; and what endpoints they communicate with.

<i>Device Names</i>	<i># of Devices</i>	<i># of Vendors</i>	<i># of Users</i>
echo	707	2	445
google-home	592	1	360
switch	582	15	325
camera	441	19	243
chromecast	382	1	293
speaker	349	12	183
hue	349	1	349
printer	216	6	213
thermostat	197	5	169
nest	195	1	151

Table 3. Device names associated with the most number of devices. In total, there are 6,776 devices across 81 vendors.

<i>Device Names</i>	<i>% of Devices to Non-trackers</i>	<i>% of Devices to Trackers</i>	<i># of Devices</i>
ihome	92.7%	0.0%	41
lutron	92.5%	0.0%	53
nvidia	92.1%	7.9%	38
tivo	90.9%	3.9%	77
amazon	88.3%	10.4%	1,076
philips	85.4%	0.3%	355
logitech	85.1%	0.0%	168
broadlink	81.5%	0.0%	27
roku	75.9%	18.3%	191
qnap	75.0%	1.9%	52

Table 4. Top ten vendors with the highest fraction of devices that communicated with the Internet over unencrypted HTTP on port 80, along with the percentage of devices from each vendor that communicated with non-tracking and tracking domains over port 80.

### 6.1 Which devices encrypt their traffic?

Major browsers such as Google Chrome encourage websites to adopt HTTPS by labelling plain HTTP websites as “insecure” [32]. However, such an effort to push for encrypted communication is yet to be seen across smart home device vendors. Our goal is to understand if smart home devices encrypt their traffic and whether the encryption follows secure practices, such as using the latest TLS version and not advertising weak ciphers.

*Which devices communicated over potentially unencrypted traffic.* Given that IoT Inspector does not capture packet payloads, one way for us to infer whether some network traffic is encrypted is to examine the protocol. In particular, we make the assumption that traffic over port 80 is likely unencrypted HTTP traffic. Of the 6,776 devices we obtained in Section 5.2, 3,462 devices (51.1%) communicated with an Internet endpoint over port 80. These devices are associated with 69 vendors, out of a total of 81 validated vendors.

In **Table 4**, we show top ten vendors with the highest fraction of devices that communicated with the Internet over unencrypted HTTP on port 80. To further understand what type of endpoints with which these devices communicated, we show the percentage of devices that communicated with non-tracking and tracking domains. A tracking domain is a domain that is associated with an advertising and/or tracking service, such as Google Analytics. To identify ads/tracking domains, we use Disconnect list, which includes registered domains that are known to be trackers and advertisers. Disconnect list is commonly used by popular privacy tools and Firefox for its Enhanced Tracking Protection [29, 33]. Our analysis show that of the 191 Roku streaming sticks and smart TVs,



75.9% communicated with a non-tracking domain over port 80, and 18.3% communicated with a tracking domain over port 80. Similarly, all Nvidia devices that communicated with tracking domains are also smart TVs. We highlight unencrypted communications with tracking domains here, because the traffic could potentially include persistent information that could identify users (especially for smart TVs [34]), and any network providers on path could obtain this information.

*Which devices used TLS.* We next investigate TLS usage across the devices in our dataset. Even though the literature abounds with large TLS measurements of websites [35] and mobile apps [36], much of the existing work on smart home TLS is restricted to a small number of devices in lab settings [37]. By using the IoT Inspector dataset, we provide the first and largest analysis on TLS usage for smart home devices in the wild.

In particular, we analyze TLS ClientHello messages in the dataset. Out of the 6,776 validated devices, we are able to extract valid ClientHello messages from 3,844 devices (56.7%) across 69 vendors (out of a total of 81 validated vendors). However, 375 devices across 26 vendors (e.g., smart TVs from Amazon, TCL, and Panasonic, along with Neato vacuum cleaners) used outdated TLS versions, i.e., SSL 3.0 and TLS 1.0. While the latest TLS version is 1.3, National Institute of Standards and Technology (NIST) guidelines requires at least TLS 1.2 [38] and versions below 1.1 are vulnerable to attacks such as BEAST and POODLE [39, 40]. Although we are not aware of any actual TLS-related attacks on smart home devices, we argue that any outdated TLS versions are potential targets for attacks in the future.

## 6.2 What endpoints do devices communicate with?

In this section, we investigate how many endpoints devices communicated with, whether these endpoints are associated with advertising and tracking services, and in which geographical regions these endpoints are located. Although there is much existing work on such third parties on the web [41] and mobile devices [42], we are the first to study this problem in the smart home ecosystem at scale.

*Number of endpoints.* How many endpoints a device communicates with may reflect the number of entities—including both first- and third-party services—that potentially have collected user data. To calculate the number of endpoints per device, we count the number of distinct registered domains that the device communicated with. Across the 81 vendors, devices of 22 vendors, most of which appear to manufacture devices such as smart switches and lightbulbs, each communicated to 1.0 registered domain in the median case. Furthermore, devices of 42 vendors each communicated with 2.0 or fewer registered domains in the median case, while devices of 78 vendors each communicated with 10.0 or fewer registered domains.

For vendors whose devices communicated with more than 5 registered domains, we generally observe two broad categories of devices. One category includes devices that appear to support Internet contents, such as smart TVs (e.g. Nvidia Shield and Roku TVs, which respectively communicated with 18.0 and 9.0 registered domains in the median case) where users could potentially install third-party apps. Another category includes devices that appear to support peer-to-peer protocols, such as networked storage devices (e.g., Qnap and Synology, whose devices respectively communicated with 18.5 and 8.0 registered domains in the median case).

*Tracking endpoints.* For each vendor, we calculate the fraction of devices that made at least one connection to domains in the Disconnect list [29]. Devices from 55 of the 81 vendors did not appear to communicate with any tracking domains. The top 10 vendors with the highest fraction of devices that communicated with at least one tracking domain are all associated with smart TVs. In particular, 57.9% (highest fraction across the vendors) of Nvidia devices (which correspond to Nvidia Shield TV) communicated with tracking domains, and 47.1% (next highest fraction) of Roku devices communicated with tracking domains. The remaining 8 vendors in the top 10 include TCL, Samsung, Sony, and LG, among others. We note that although previous work has studied the

User Time Zones Between SF and NYC (79 vendors)			User Time Zones Between London and Moscow (58 vendors)		
<i>Countries</i>	<i>Any Port</i>	<i>Port 80 Only</i>	<i>Countries</i>	<i>Any Port</i>	<i>Port 80 Only</i>
United States	98.7%	74.7%	United States	86.2%	56.9%
Canada	50.6%	30.4%	Germany	63.8%	41.4%
Australia	48.1%	3.8%	Ireland	50.0%	27.6%
Ireland	44.3%	22.8%	United Kingdom	39.7%	27.6%
Netherlands	31.6%	15.2%	Netherlands	36.2%	25.9%
China	30.4%	11.4%	France	32.8%	19.0%
Germany	30.4%	15.2%	China	31.0%	19.0%
France	30.4%	12.7%	Japan	31.0%	10.3%
United Kingdom	26.6%	10.1%	Australia	29.3%	1.7%
Japan	21.5%	7.6%	Singapore	24.1%	3.4%

Table 5. Percentage of vendors in each group that communicated with endpoints in each country over Any Port and over unencrypted HTTP on Port 80 Only.

smart TV tracking ecosystem on Roku and Amazon in the lab setting [34], our analysis includes more smart TV vendors and examines devices in the wild.

*Potentially tracking endpoints.* So far, we identify advertising or tracking domains based on the Disconnect list [29], which is specifically used to block such domains on the web and mobile platforms. We are not aware of similar tracker blacklists compiled for smart home devices. To address this gap, we look for third-party services that could potentially aggregate data across different types of devices. One example of such services is *device control platforms*. Device control platforms coordinate device control via mobile apps, collecting device status updates and allowing users control their smart home devices through their phones. While some vendors use first-party device control platforms, e.g., Samsung Camera uses its own XMPP server on [xmpp.samsungsmartcam.com](http://xmpp.samsungsmartcam.com), other vendors may choose to use third-party platforms. These platforms may be able to observe changes in device state and infer users' behaviors and lifestyles. For example, merely keeping track of when a user turns on/off a smart plug may reveal sensitive information on a user's life habits (e.g., when they are asleep vs awake, or whether a user is at home) [43]. Although we do not have evidence whether such platforms keep and/or analyze this data, by being the first to study these platforms, we are hoping to raise the awareness of the potential privacy concerns.

Examples of third-party device control platforms include TuYa (an MQTT [30] platform whose company is headquartered in China) or PubNub (headquartered in California). In particular, 57 devices across 9 vendors used TuYa's service. These devices include 15 Smart Life switches, a Wyze surveillance camera, and 2 Philips Hue lights. Moreover, 168 devices across 11 vendors used PubNub's service.

*Geographic distribution of endpoints.* Finally, we show that devices may communicate with endpoints that are located beyond the users' geographic region. Recall from Section 4.1 that 62.5% of users are located in time zones between San Francisco and New York City, and 27.8% of users are located in time zones between London and Moscow. For both of these user groups, we show the top 10 countries associated with the highest fraction of vendors whose devices communicated with these countries.

As shown in **Table 5**, for devices located between San Francisco and New York City time zones, 98.7% of the 79 device vendors communicated with endpoints in the US; however, 74.7% of the 79 vendors also communicated over port 80, which is likely unencrypted HTTP traffic. For devices located between London and Moscow time zones, 86.2% of the 58 vendors communicated with endpoints in the US, and 56.9% also communicated over port 80. Given the diverse geographic distribution of endpoints, if a vendor sends sensitive information to an endpoint in a foreign country that has a poor track record of privacy protection, user privacy could be at stake.

## 7 FUTURE WORK

IoT Inspector offers an unprecedented look into the network behavior of smart home devices in the wild. Although certain design decisions limit the scope of data that IoT Inspector can collect, the IoT Inspector dataset enables a variety of follow-up research beyond just the two examples in Section 6. This section presents these limitations and opportunities for future work using the IoT Inspector dataset and other smart home device analysis methods.

*Understanding user behaviors.* The fact that 6,069 users downloaded IoT Inspector within the first 26 days of release demonstrates widespread interest and concern about smart home security and privacy. It also shows that many users, even those with security/privacy concerns, trust academic institutions enough to deploy research software in their homes. However, it is difficult to have users run IoT Inspector over an extended period of time; the median duration of traffic collected from the monitored devices is only 35.0 minutes (Section 4). Furthermore, for every 1 hour spent on the main user interface, the same user only spent 11 seconds using a special filter that allows the users to examine all the tracking/advertising domains contacted (Section 4.3). Even with the HTTP log data we have, it is not clear why the user engagement is not higher.

To improve user engagement, we need a qualitative understanding of user behaviors, such as why they decided to use IoT Inspector, as well as whether, how, and why IoT Inspector satisfied their needs, because the current design of IoT Inspector does not allow us to gather any user feedback. Future work could involve a more in-depth design exercise for IoT Inspector’s interface and functionality, such as conducting qualitative user studies in the lab.

Furthermore, while IoT Inspector helps users identify potential security and privacy issues in their smart homes, the ultimate goal is for users to take actions, e.g., upgrading their device firmware, isolating devices on a separate network, or even unplugging devices when not in use. The current dataset collected does not produce any insight on whether, how, and why users changed their behaviors after using IoT Inspector, and whether they are comfortable letting IoT Inspector influence their behaviors (as opposed to using commercially available products). Again, our future work includes conducting qualitative user studies, where we pose these questions to the users to understand what actions users would take as a result of IoT Inspector.

*Validating crowdsourced data.* Our current method of validating device labels (Section 5) may be limited, as our four validation methods may introduce false negatives, e.g., correct vendor labels but not being marked as Validated because of missing data). These methods may also introduce false positives; for instance, a device that a user labeled as Vendor A but which is actually associated with Vendor B may be correctly validated, if devices of both Vendors A and B happen to share the same common top domain, “amazonaws.com,” because both vendors happen to use Amazon Web Services for their IoT infrastructure. However, absent of ground truth labels of devices, we are unable to estimate the false negative and false positive rates for our methods. Our future work includes asking users to submit photos of their devices as ground truth labels.

We plan to explore automatic device identification in future work. Existing literature on device identification has used a variety of machine learning methods with features from network traffic rates and packet headers [44–46] as well as acquisitional rule-based techniques [10]. We plan to extend these studies, train machine learning models on IoT Inspector’s dataset, and develop new methods that would automatically infer device identities in the wild. For example, we plan to analyze the network activities over time to further distinguish between different vendors and/or across different types of devices of the same vendor. After all, a surveillance camera is likely to send a constant stream of bytes, whereas a smart switch would typically experience a spike in network activities only when it is turned on/off.

## 8 CONCLUSION

In response to the proliferation of smart home devices and the corresponding lack of data enabling ubiquitous computing research in this area, we crowdsourced a dataset of smart home network traffic and device labels from 44,956 devices across 4,322 users with IoT Inspector, an open-source software tool that we designed to enable large-scale, unobtrusive data collection from within smart home networks. To our knowledge, this dataset is the largest (and perhaps only) of its kind. To demonstrate the potential of this dataset to shed new insights into smart homes, we used the data to study questions related to smart home security and privacy. In particular, the IoT Inspector dataset enabled us to discover the transmission of unencrypted traffic by 2,159 smart home devices across 44 vendors and to identify insecure encryption practices even by Google and Amazon devices; we also identified third-party trackers and data aggregators on smart TVs and a large variety of other smart devices. These insights are the tip of the iceberg in what this large—and growing—dataset can offer for ubiquitous computing research across a wide range of areas from security and privacy to human behavior.

## REFERENCES

- [1] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, “Understanding the Mirai Botnet,” in *USENIX Security Symposium*, 2017.
- [2] G. Chu, N. Apthorpe, and N. Feamster, “Security and Privacy Analyses of Internet of Things Children’s Toys,” *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 978–985, 2019.
- [3] J. Ortiz, C. Crawford, and F. Le, “Devicemien: Network device behavior modeling for identifying unknown iot devices,” in *Proceedings of the International Conference on Internet of Things Design and Implementation*, ser. IoTDI ’19. New York, NY, USA: ACM, 2019, pp. 106–117. [Online]. Available: <http://doi.acm.org/10.1145/3302505.3310073>
- [4] Rob van der Meulen. (2017) Gartner Says 8.4 Billion Connected “Things” Will Be in Use in 2017, Up 31 Percent From 2016 . [Online]. Available: <https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016>
- [5] D. Kumar, K. Shen, B. Case, D. Garg, G. Alperovich, D. Kuznetsov, R. Gupta, and Z. Durumeric, “All Things Considered: An Analysis of IoT Devices on Home Networks,” in *USENIX Security Symposium (to appear)*, 2019.
- [6] N. Apthorpe, D. Y. Huang, D. Reisman, A. Narayanan, and N. Feamster, “Keeping the smart home private with smart (er) iot traffic shaping,” *arXiv preprint arXiv:1812.00955*, 2018.
- [7] D. Wood, N. Apthorpe, and N. Feamster, “Cleartext Data Transmissions in Consumer IoT Medical Devices,” in *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*. ACM, 2017, pp. 7–12.
- [8] S. Sundaresan, S. Burnett, N. Feamster, and W. De Donato, “BISmark: A Testbed for Deploying Measurements and Applications in Broadband Access Networks,” in *USENIX Annual Technical Conference (ATC)*, 2014.
- [9] P. Schmitt, F. Bronzino, R. Teixeira, T. Chattopadhyay, and N. Feamster, “Enhancing transparency: Internet video quality inference from network traffic,” *Research Conference on Communications, Information and Internet Policy*, 2018.
- [10] X. Feng, Q. Li, H. Wang, and L. Sun, “Acquisitional rule-based engine for discovering internet-of-things devices,” in *USENIX Security Symposium*, 2018, pp. 327–341.
- [11] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, “Kitsune: an ensemble of autoencoders for online network intrusion detection,” *Network and Distributed Systems Security Symposium (NDSS)*, 2018.
- [12] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. Sadeghi, and S. Tarkoma, “Iot sentinel: Automated device-type identification for security enforcement in iot,” in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 2177–2184.
- [13] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, “Profiliot: A machine learning approach for iot device identification based on network traffic analysis,” in *Proceedings of the Symposium on Applied Computing*, ser. SAC ’17. New York, NY, USA: ACM, 2017, pp. 506–509. [Online]. Available: <http://doi.acm.org/10.1145/3019612.3019878>
- [14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [15] M. Chetty, D. Haslem, A. Baird, U. Ofoha, B. Sumner, and R. Grinter, “Why is My Internet Slow?: Making Network Speeds Visible,” in *SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2011.
- [16] S. Grover, M. S. Park, S. Sundaresan, S. Burnett, H. Kim, B. Ravi, and N. Feamster, “Peeking Behind the NAT: An Empirical Study of Home Networks,” in *Internet Measurement Conference (IMC)*, 2013.

- [17] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, "Netalyzer: Illuminating the Edge Network," in *ACM Internet Measurement Conference (IMC)*, 2010.
- [18] L. DiCioccio, R. Teixeira, M. May, and C. Kreibich, "Probe and Pray: Using UPnP for Home Network Measurements," in *International Conference on Passive and Active Measurement (PAM)*, 2012.
- [19] L. DiCioccio, R. Teixeira, and C. Rosenberg, "Measuring Home Networks with HomeNet Profiler," in *International Conference on Passive and Active Measurement (PAM)*, 2013.
- [20] S. Shasha, M. Mahmoud, M. Mannan, and A. Youssef, "Playing With Danger: A Taxonomy and Evaluation of Threats to Smart Toys," *IEEE Internet of Things Journal*, 2018.
- [21] G. Acar, D. Huang, F. Li, A. Narayanan, and N. Feamster, "Web-based Attacks to Discover and Control Local IoT Devices," in *ACM SIGCOMM Workshop on IoT Security and Privacy (IoT S&P)*, 2018.
- [22] IEEE. Organizationally unique identifier. [Online]. Available: <http://standards-oui.ieee.org/oui.txt>
- [23] Netdisco. Netdisco. [Online]. Available: <https://github.com/home-assistant/netdisco>
- [24] Debian. Arpspoof - intercept packets on a switched lan. [Online]. Available: <https://manpages.debian.org/jessie/dsniff/arpspoof.8.en.html>
- [25] inverse.ca. Fingerbank. [Online]. Available: <https://fingerbank.org/>
- [26] C. Kreibich, N. Weaver, G. Maier, B. Nechaev, and V. Paxson, "Experiences from netalyzer with engaging users in end-system measurement," in *Proceedings of the First ACM SIGCOMM Workshop on Measurements Up the Stack*, ser. W-MUST '11. New York, NY, USA: ACM, 2011, pp. 25–30. [Online]. Available: <http://doi.acm.org/10.1145/2018602.2018609>
- [27] F. Security. Farsight security passive dns faq. [Online]. Available: <https://www.farsightsecurity.com/technical/passive-dns/passive-dns-faq/>
- [28] T. Libert. (2019) webxray domain owner list. [Online]. Available: [https://github.com/timlib/webXray\\_Domain\\_Owner\\_List](https://github.com/timlib/webXray_Domain_Owner_List)
- [29] DisconnectMe. (2019) Disconnect tracking protection. [Online]. Available: <https://github.com/disconnectme/disconnect-tracking-protection/>
- [30] MQTT. (2019) Message Queuing Telemetry Transport. [Online]. Available: <http://mqtt.org/>
- [31] Seth Schoen. (2019) ESNI: A Privacy-Protecting Upgrade to HTTPS. [Online]. Available: <https://www.eff.org/deeplinks/2018/09/esni-privacy-protecting-upgrade-https>
- [32] Emily Schechter. (2018) A milestone for Chrome security: marking HTTP as "not secure". [Online]. Available: <https://www.blog.google/products/chrome/milestone-chrome-security-marking-http-not-secure/>
- [33] Marissa Wood. (2019) Today's Firefox Blocks Third-Party Tracking Cookies and Cryptomining by Default - The Mozilla Blog . [Online]. Available: <https://blog.mozilla.org/blog/2019/09/03/todays-firefox-blocks-third-party-tracking-cookies-and-cryptomining-by-default>
- [34] H. Mohajeri Moghaddam, G. Acar, B. Burgess, A. Mathur, D. Y. Huang, N. Feamster, E. W. Felten, P. Mittal, and A. Narayanan, "Watching you watch: The tracking ecosystem of over-the-top tv streaming devices," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. ACM, 2019.
- [35] Z. Durumeric, Z. Ma, D. Springall, R. Barnes, N. Sullivan, E. Bursztein, M. Bailey, J. A. Halderman, and V. Paxson, "The security impact of https interception," in *NDSS*, 2017.
- [36] A. Razaghpanah, A. A. Niaki, N. Vallina-Rodriguez, S. Sundaresan, J. Amann, and P. Gill, "Studying TLS Usage in Android Apps," in *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '17. New York, NY, USA: ACM, 2017, pp. 350–362. [Online]. Available: <http://doi.acm.org/10.1145/3143361.3143400>
- [37] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "SoK: Security Evaluation of Home-Based IoT Deployments," in *IEEE Symposium on Security and Privacy (S&P)*, 2019.
- [38] K. McKay and D. Cooper. (2019, Aug) Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations.
- [39] Chromium Bugs. (2014) Issue 436391: Add info on end of life of SSLVersionFallbackMin & SSLVersionMin policy in documentation. [Online]. Available: <https://bugs.chromium.org/p/chromium/issues/detail?id=436391>
- [40] B. Möller, T. Duong, and K. Kotowicz, "This poodle bites: exploiting the ssl 3.0 fallback," *Security Advisory*, 2014.
- [41] S. Englehardt and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1388–1401.
- [42] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill, "Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem," 2018.
- [43] N. Apthorpe, D. Y. Huang, D. Reisman, A. Narayanan, and N. Feamster, "Keeping the Smart Home Private with Smart(er) IoT Traffic Shaping," in *Proceedings on Privacy Enhancing Technologies Symposium (PETS)*, 2019.
- [44] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "Profilot: a machine learning approach for iot device identification based on network traffic analysis," in *Proceedings of the symposium on applied computing*. ACM, 2017, pp. 506–509.
- [45] J. Ortiz, C. Crawford, and F. Le, "DeviceMien: network device behavior modeling for identifying unknown IoT devices," in *Proceedings of the International Conference on Internet of Things Design and Implementation*. ACM, 2019, pp. 106–117.

- [46] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "Iot sentinel: Automated device-type identification for security enforcement in iot," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 2177–2184.