In the Room Where It Happens: Characterizing Local Communication and Threats in Smart Homes

Aniketh Girish*
IMDEA Networks Institute / Universidad Carlos III de Madrid

Tianrui Hu*
Northeastern University

Vijay Prakash
New York University

Daniel J. Dubois
Northeastern University

Srdjan Matic
IMDEA Software Institute

Danny Yuxing Huang
New York University

Serge Egelman
ICSI / University of California, Berkeley

Joel Reardon
University of Calgary / AppCensus

Juan Tapiador
Universidad Carlos III de Madrid

David Choffnes
Northeastern University

Narseo Vallina-Rodriguez
IMDEA Networks Institute / AppCensus

ABSTRACT

The network communication between Internet of Things (IoT) devices on the same local network has significant implications for platform and device interoperability, security, privacy, and correctness. Yet, the analysis of local home Wi-Fi network traffic and its associated security and privacy threats have been largely ignored by prior literature, which typically focuses on studying the communication between IoT devices and cloud end-points, or detecting vulnerable IoT devices exposed to the Internet. In this paper, we present a comprehensive and empirical measurement study to shed light on the local communication within a smart home deployment and its threats. We use a unique combination of passive network traffic captures, protocol honeypots, dynamic mobile app analysis, and crowdsourced IoT data from participants to identify and analyze a wide range of device activities on the local network. We then analyze these datasets to characterize local network protocols, security and privacy threats associated with them. Our analysis reveals vulnerable devices, insecure use of network protocols, and sensitive data exposure by IoT devices. We provide evidence of how this information is exfiltrated to remote servers by mobile apps and third-party SDKs, potentially for household fingerprinting, surveillance and cross-device tracking. We make our datasets and analysis publicly available to support further research in this area.

CCS CONCEPTS

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

KEYWORDS

Smart Home, IoT, Local Communication, Security, Privacy, Side Channels, Household Fingerprinting

ACM Reference Format:
Aniketh Girish*, Tianrui Hu*, Vijay Prakash, Daniel J. Dubois, Srdjan Matic, Danny Yuxing Huang, Serge Egelman, Joel Reardon, Juan Tapiador, David Choffnes, and Narseo Vallina-Rodriguez. 2023. In the Room Where It Happens: Characterizing Local Communication and Threats in Smart Homes. In *Proceedings of the 2023 ACM Internet Measurement Conference (IMC '23), October 24–26, 2023, Montreal, QC, Canada.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3618257.3624830

*The two lead authors contributed equally to this work.

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

ACM ISBN 979-8-4007-0382-9/23/10.
https://doi.org/10.1145/3618257.3624830

1 INTRODUCTION

Internet of Things (IoT) devices are increasingly pervasive in home environments thanks to the many advantages and services that they offer to users, with an estimated household penetration of 43.8% in 2022 [113]. However, IoT devices may have undetected security and privacy threats due to design and development errors, privacy-intrusive business models, supply-chain vulnerabilities, or even an incorrect use by end-users [34, 39, 54, 74, 89, 104, 119].

Prior research has mostly focused on understanding how IoT devices interact with cloud services [34, 50, 54, 65], or identifying vulnerabilities, often using mobile companion applications (apps) as gateways to the devices [34, 90, 115, 118] or highlighting instances of information leaks to the cloud by these Internet-facing devices [89, 105]. Despite these important research contributions, our understanding of the security and privacy threats caused by the continuous and seamless interaction among IoT devices and mobile apps in the *local* home network is still limited.

In this paper, we aim to fill this gap by conducting a systematic analysis of the network protocols used by IoT devices to interact with each other in the home network, and the unexpected implications for security, privacy, and device functionality. Concerns range from standard security issues such as vulnerable services running on open ports, to privacy issues related to sensitive information broadcasted by devices in the local network, thus enabling household and device fingerprinting. In our threat model, we consider IoT devices or software with access to a home network (i.e., behind

the firewall) that exploit device vulnerabilities or local network protocols to gather privacy- or security-sensitive data from other devices in the same local network, an attack that would not be possible from the Internet.

All these threats are indeed real. Surreptitious device discovery and device metadata disseminated by IoT devices in the local network may be equally (ab)used by advertising companies, spyware, or state-level surveillance. The Snowden revelations showed how apparently harmless data such as device models and device MAC addresses facilitate user tracking and household profiling [52]. Similarly, Reardon [102] and the Wall Street Journal [117] discovered a spyware SDK embedded that sent broadcast messages to the whole LAN to fingerprint other devices. This SDK—allegedly disseminated by a U.S. defense contractor through mobile apps present in the US Play Store—exfiltrated this data to the cloud along with clipboard contents, various device identifiers, and location data. This same malicious SDK was observed embedded in deceptively-advertised encryption software released by a sister company, which *was* also a root Certificate Authority [83].

This anecdotal evidence highlights the need to understand how protocols are used on the LAN, what are the associated threats, and to what extent are they potentially exploited in order to inform the design and deployment of future standards and mitigations. Thus, this paper focuses on answering the following research questions:

- **RQ1:** What are the protocols that smart home IoT devices support and use within the local network to interact with other devices, including mobile apps?
- **RQ2:** What are the security and privacy threats associated with the use of these protocols in the local network?
- **RQ3:** Are network and device information been gathered by advertising and tracking services? How amenable is this information to household fingerprinting and cross-device tracking?

Contributions. Through a combination of rigorous experiments, both automated and manual, both controlled and uncontrolled, this paper presents a comprehensive analysis of local network communications of a diverse range of consumer smart IoT devices and mobile apps, and their associated threats. We study IoT local traffic in different settings and scenarios and using complementary analysis methods. Specifically, we combine (*i*) passive traffic captures and active scans in an IoT lab consisting of 93 different IoT devices, (*ii*) the traffic interactions between 2,335 mobile apps—including companion IoT apps and regular apps—with those 93 devices, and (*iii*) organic local network interactions from 13,487 IoT devices crowdsourced from 3,800 households. We consider only IP-based consumer smart home IoT devices, their companion mobile apps, and their wireless (Wi-Fi/IEEE 802.11) or wired (Ethernet) traffic in the local network. Analyses of other wireless interfaces like Bluetooth are outside the scope of this paper.

The key contributions of this work are:

- We conduct the first rigorous analysis of smart home IoT traffic in the local network using a combination of controlled traffic analysis, active scans and runtime mobile app analysis. We find a diverse use of protocols—both standard and proprietary—for device discovery and data exchange. We reveal analogous protocol support for devices offering similar services or from the same vendor, for service interoperability. Yet, the purpose of some protocols remains unknown due to the lack of accurate traffic classification methods for local IoT traffic and documentation.
- We identify concerning security and privacy threats associated with local network traffic. We observe uncontrolled dissemination of sensitive and identifiable information, such as MAC addresses, device models, UUIDs, and geolocation. These can be harvested by malicious actors in the home network (*e.g.,* spyware or privacy-intrusive advertising companies). We also identify devices running outdated and vulnerable services.
- We provide evidence of Android mobile apps and third-party SDKs (*e.g.,* innoSDK and AppDynamics) harvesting local network information potentially for advertising and tracking purposes. In order to access this data, they abuse user-space discovery protocols (*e.g.,* UPnP and mDNS) to bypass the Android permissions that control the access to sensitive information such as the MAC address of the Wi-Fi Access Point. Further, using IoT Inspector's crowdsourced data, we demonstrate that local network information is a valuable asset for privacy-intrusive practices like household fingerprinting and cross-device tracking.

In summary, the threats we consider in this paper are important and unaddressed by existing mitigations. As we show, many devices are vulnerable to the threats described above, and these issues are actively exploited by mobile apps, malware, and third-party SDKs. We discuss potential mitigations that may contribute to minimize the impact of the threats presented in this paper in § 7.

Responsible Disclosure. We responsibly disclosed all security and privacy risks to the respective vendors, along with proof-of-concept details and an assessment of their security and privacy impact. Interactions with affected vendors are still ongoing at the time of writing, and we report on details in Appendix §A. We note that Google is engaging with us to explore mitigations that could be implemented via the Android OS, app review processes, and general IoT standardization efforts.

Research Artifacts. To ensure reproducibility and facilitate follow-up research, we have made our data and analysis artifacts with the exception of commercial products, publicly available at https://github.com/Android-Observatory/IoT-LAN/.

Ethical Considerations. All data collected from human subjects was conducted with participant consent and IRB approval at respective institutions. No personal identifiers were used for our analysis nor will be disclosed with our dataset releases. § 3.4 provides a detailed discussion of the ethical considerations of this work.

2 BACKGROUND

IoT devices not only communicate over the Internet, but also with other devices and software services running on the local network. This is facilitated by popular IoT platforms such as SmartThings or Google Home that provide support and protocols for discovering, connecting, and managing IoT devices, thus enabling interoperability across IoT vendors, platforms, and devices. While standard protocol specifications (*e.g.,* UPnP, mDNS, DHCP) may be well known, their actual use and behavior in home networks is not. Not to mention the little attention that proprietary protocols (*e.g.,* TP-Link Smart Home protocol [28], and Tuya protocol [27]) and new standards (*e.g.,* Matter [22]) have received in previous research.

While these local network protocols are vital for seamless integration and device communication in smart home systems (*i.e.,* enabling device discovery, network management, and device pairing), their potential for misuse and abuse is concerning. Malicious actors can manipulate these protocols to bypass permission models and siphon information through side channels. Such exploitation targets IoT devices or software (e.g., mobile apps) that utilize IoT protocols' vulnerabilities, enabling them to scan or collect sensitive data from other devices in the local network—attacks that would not be possible from the Internet.

Security threats. We consider IoT devices using insecure network protocols that expose sensitive data to on-path observers, such as home routers or recipients of broadcast messages; or devices that integrate vulnerable networking services and protocols that can be abused by malicious actors from within the home network. Compromised devices under the control of an attacker (*e.g.,* through malware running on a mobile device like spyware) are included in our model, since they may be able to scan devices and monitor traffic to/from other devices on the local network.

Privacy threats. Protocols such as UPnP and mDNS can be abused by IoT devices and applications (*e.g.,* mobile and smart TV apps) with access to networking APIs to actively scan the network or to passively gather sensitive network-, user-, and device-related data from other devices in the network. Prior work demonstrated that router MAC addresses can be used to infer device (and user) locations with street-level precision [106] or for surveillance [52]. This is possible because Wi-Fi hotspots tend to have fixed locations and MAC addresses that uniquely identify them. Thus, developers and tracking services can use this data to query users' geolocation from online geocoding services like Wigle [29]. Further, access to network and device metadata can be used to infer household social structures and socio-economic level [46, 114], while also facilitating cross-device tracking and household fingerprinting [41, 76].

2.1 Home Network Scanning by Mobile Apps

It is for these reasons that mobile platforms consider local network data like the Wi-Fi Access Point (AP) SSID and BSSID to be sensitive, requiring elevated privileges to access it through the official APIs [37, 63]. Yet, a recent article published at the Wall Street Journal revealed that Google removed apps with third-party code by Measurement Systems—a company with ties to a U.S. defense contractor [117]—from the Play Store. In some apps, this code was soliciting local network information with neither user notice nor consent, by directly opening up network sockets to send broadcast messages (in lieu of requesting the information through official permission-protected APIs).

To mitigate these risks and control access to nearby devices over Wi-Fi, both iOS (v14) and Android (v13) introduced changes in their permission models [10, 19]. Since Android 13, developers must request access to the `NEARBY_WIFI_DEVICES` permission to read the SSID of the Wi-Fi AP. Previously, starting in Android 9 SSID/BSSID access required either the `ACCESS_COARSE_LOCATION` or `ACCESS_FINE_LOCATION` runtime permissions [13]. Yet, these Android permission model changes do not prevent mobile apps from scanning the local IP address space using mDNS and SSDP/UPnP, for which Android even provides native support through the NsdManager (Network Service Discovery Manager) interface [61].

We confirm this behavior through a proof-of-concept app built for Android 13; the app can discover other devices on the local network using these protocols while holding only the `INTERNET` permission to access the hostname of the local device, and the `CHANGE_WIFI_MULTICAST_STATE` permission to get the multicast socket lock. Neither permission is considered "dangerous," and therefore does not require the user's explicit consent [62]. Moreover, third-party SDKs inherit the same privileges as the host app [53], so they can scan the home network and gather local network information without the awareness of the app developer.

In iOS, local network traffic requires developers to acquire the `com.apple.developer.networking.multicast` entitlement—whose use must be explicitly approved by Apple [38]—and include the `NSLocalNetworkUsageDescription` in the app manifest to communicate (either with unicast of multicast connections) to other hosts in the local network [36]; a permission that also requires explicit user consent to proceed. We implemented a PoC iOS (v16.7) app to confirm the multicast entitlement and permission requirements. However, it is possible that this attack can be performed in iOS once this entitlement is granted, or using other side-channels we are not yet aware of. Similarly, third-party smart TV apps could potentially exploit programmatic and unprotected access to local network sockets to perform fingerprinting attacks. However, the analysis of the Smart TV ecosystem is left for future work.

3 DATASETS AND METHODOLOGY

Our study relies on three methods to gain a comprehensive view of devices' local network activity and its risks: (*i*) the *MonIoTr Lab Dataset* (§ 3.1) contains data collected from our smart-home IoT testbed using both passive traffic collection techniques and active scans, enabling detailed insights into the traffic IoT devices generate on the LAN; (*ii*) the *Mobile App Dataset* (§ 3.2) reveals information about local network communication when smartphone apps are deployed into our smart-home testbed; and (*iii*) the *Crowdsourced IoT Dataset* (§ 3.3) offers IoT traffic gathered from large numbers of volunteer participants' home networks, providing insight into local IoT traffic "in the wild." We conclude this section with a description of our traffic classification method (§3.5) and limitations (§3.6).

3.1 MonIoTr Lab Testbed

The MonIoTr Lab testbed [23] consists of 93 IP-based devices representing 78 unique device models. The devices are deployed in a "living lab" that resembles a studio apartment that is connected to the Internet via a Wi-Fi AP that captures all network traffic utilizing *tcpdump*. The captured traffic is stored in separate files for each MAC address, enabling us to distinguish traffic from individual devices. In our testbed, we do not attempt to decrypt any encrypted flows or modify any devices to obtain privileged access. Table 3 (Appendix) lists the devices in our testbed, which are in the following categories: cameras, smart speakers, smart TVs, home automation, sensors, appliances, smart hubs, and smart health devices.

To capture more instances of local IoT traffic between devices, similar to prior work [80, 105], from 11/2022 to 12/2022, we capture

five consecutive days of traffic without human interaction [1], as well as traffic generated from 7,191 interactions when we manually or automatically interact with the different IoT devices in our testbed. During our experiments without human interaction, no one was allowed to enter the lab. The interactions considered for the experiments are triggered by (*i*) IoT companion apps running on a Google Pixel 3 and an iPhone 7 connected to the same network as the IoT device; or (*ii*) voice commands to activate different voice assistants, which subsequently interact with the corresponding device according to the voice command. Specifically, we use either the Echo Spot or Echo Show 5 (powered by Alexa), or the Google Home Mini or Google Nest Hub (powered by Google Assistant). We refer to the generated datasets as MonIoTr Lab passive dataset.

IoT Honeypots. Passive traffic collection can be incomplete when flows require specific input or responses from endpoints that we do not control. To address this limitation, we deploy various honeypot within the same network as our IoT devices. These honeypots capture network scans from IoT devices and issues authentic responses to requests, mimicking real-world device interactions. They support protocols such as SSDP, mDNS, UPnP, HTTP(S), and telnet. Given our control over these responses, the honeypots give us the ability to track how information propagates through the IoT devices.

Active Scans. Active scans provide information about open services of smart devices that may not be observed passively. Informed by previous work [34, 77], we perform active scans using *nmap* [25]. We run TCP SYN scans on all ports (1-65535), UDP scans on popular ports (1-1024), and IP-level protocol scans. We picked well-known UDP ports as UDP scanning is generally slower and more challenging than TCP scanning. Note that only 54 and 20 devices responded to TCP SYN and UDP scans, respectively, and 58 to IP protocol scans. We also use *Nessus* scanner [94] to detect potential vulnerabilities in running services, and annotate their security states and associated CVE (Common Vulnerabilities and Exposures).

3.2 Instrumented Mobile Devices

In addition to our Pixel and iPhone devices running companion apps to control IoT devices (§3.1), we use a Google Pixel 3a smartphone running a licensed version of AppCensus' runtime analysis technology [14]. Specifically, we use a system-level instrumented variant of Android 9.0 (Pie), combined with Frida [16] scripts, that allows us to track runtime resource access by mobile apps (*e.g.,* access to permission-protected Android APIs) and log all network traffic in plaintext (*i.e.,* using MITM approaches to decrypt TLS traffic and other encodings). We use AppCensus' visibility into app runtime behavior to detect potential instances of cross-device tracking and network fingerprinting, and to track how sensitive data is transmitted from IoT devices to companion apps, and then to remote servers. We pair the testing smartphone to each IoT device, then we use Android's Application Exerciser Monkey [48] to generate synthetic user inputs for approximately five minutes. The reason why we focus our analysis on the Android ecosystem is due to the technical impediments caused by the closed nature of the iOS platform, which prevent us from dynamically analyzing iOS apps with the same detail as Android apps.

App Dataset. We crawled the Google Play Store [17] in May 2023 to download 2,335 apps with varying popularity and user ratings. These include 987 IoT-specific apps (*e.g.,* companion apps) and 1,348 "regular" apps (*e.g.,* social network apps). Analyzing both types of apps allows comparisons of behavior in the local network across app categories, and reveals whether local network activity and local network data collection is exclusive to IoT companion apps. To collect mobile companion apps for smart home IoT devices, we use the Google Play API to search for similar apps, starting with a seed of 35 well-known IoT companion apps. To create the regular app dataset, we randomly selected 1.5K unique package names from Androzoo [33]. Then, we employed a custom-built Google Play Store crawler to fetch the most recent version of each app, yielding a total of 1,348 downloaded apps.

3.3 Crowdsourced IoT Traffic Data

We collected the IoT Inspector dataset [69] and analyzed a subset of the data for this paper. This subset includes only passive local network traffic captured using ARP spoofing [69] of 13,487 Internet-connected devices (3,893 users or households), along with their mDNS and SSDP responses. The data was collected with the consent of participants from April 12, 2019 through July 15, 2022. We consider only traffic whose source and destination IP addresses are in ranges reserved for private networks [45].

The IoT Inspector data that we use in this paper contains the following data: source and destination IP addresses and ports, device IDs,[2] and the number of bytes sent and received by each inspected device over five-second windows, along with the timestamps of these windows. IoT Inspector does not collect any packet payload with two exceptions: (i) the hostname fields in DHCP Request and DNS Response packets, and (ii) the full mDNS and SSDP responses [69]. We collect these for analyses related to entropy (§ 6.3).

IoT Inspector data does not directly collect vendor and product information for devices. Instead, we rely on device metadata (*e.g.,* the first three octets of a MAC address) and user-provided device labels (*e.g.,* vendor and model) to infer the vendor and product information of devices, as described in prior work [69] and in Appendix E. In total, using this method, the 13,487 devices in the dataset are associated with 199 vendors, and 323 products (i.e., vendor-category combinations).

3.4 Ethics

The MonIoTr Lab testbed is part of an IRB-approved user study where our participants with informed consent use the space and the IoT devices as they see fit. When we conduct controlled experiments to produce labeled traffic traces and datasets of this project, participants are not allowed in the room. The active scans and app analysis are fully automated and performed when the lab is closed to prevent collecting any data from users.

IoT Inspector's data collection is approved by New York University's IRB. We follow industry-standard security and privacy practices to safeguard the data and limit its access, including storing

[1] We chose to collect data over five consecutive days to ensure that we capture background local traffic, including those occurring very infrequently (*e.g.,* firmware update checks).

[2] Computed by HMAC, which uses SHA-256 to hash the original device MAC addresses with a secret salt persistent per user [69].

the dataset on a secure server in our institution and restricting its access to only co-authors of this paper, as approved by our IRB. For details on how IoT Inspector protects user privacy, refer to the original paper [69], the IRB protocol [6], and the FAQ page [9].

3.5 Traffic Classification Methodology

While traffic classification is a well-trodden research area, we found that existing and widely-used tools like *tshark* [30] or *nDPI* [47] are insufficient for the IoT context. Specifically, *nDPI* utilizes signature- and behavioral-based detection, and heuristic techniques, whereas *tshark* relies on packet header and payload information to identify application-layer protocols using predefined specifications.

To identify the most accurate and effective tool for classifying local IoT traffic, we compare the outputs of *nDPI* (v4.7.0), and *tshark* (v3.6.2). We refer the reader to Appendix C.2 for details about this cross-validation. Informed by the experimental results, we selected *nDPI* to classify the captured IoT traffic and augmented it with manually-defined rules informed by our manual evaluation, thus allowing us to handle errors and coverage limitations. For protocols below the transport layer, we used the type field in Ethernet frame headers to identify the protocol, determining if traffic was non-IP (*e.g.,* ARP) or IP. For IP traffic, we extracted the protocol field in IP packet headers to identify transport layer protocols.

As mentioned in § 3.1, we use *nmap* to scan for open ports on devices in our testbed. The tool, however, primarily relies on port numbers and packet responses to infer the protocol behind an open service. We find these inferences to be incorrect in many cases, *e.g.,* due to similar scan responses provided by different services or the use of non-standard port numbers. We manually validated and corrected *nmap* labels to ensure accurate service identification.

3.6 Limitations

Incompleteness. Despite extensive measurements, we cannot guarantee that we observe all possible instances of IoT local communications. Similarly, the app behaviors we observe are a lower bound since our automated mobile app instrumentation may miss some local interactions (*e.g.,* those that require logging in).

Traffic characterization. The prevalent use of custom protocols and non-standard ports by IoT devices poses a significant challenge for accurately classifying network protocols. Despite our use of widely-used tools such as nDPI, tshark, nmap, and manual verification and domain expertise, our protocol classification may still be incorrect in some cases.

Limited payload visibility. The adoption of non-standard protocols and encryption in IoT traffic sources constrains our visibility over payloads using passive traffic captures. This limitation impedes our exhaustive exploration of the threat landscape. While modifying IoT devices might provide visibility to encrypted payloads, this approach is not scalable for a large number of devices and is often infeasible for many of them. For this reason, we complement our passive traffic captures with the honeypots and AppCensus' mobile app testing capabilities, so that we can get visibility into some of the encrypted flows generated by IoT devices.

Device Update. Although many devices in our testbed like smart speakers receive automatic updates from the vendor, others require us to manually update their firmware after receiving notifications on the companion apps or on-screen prompts. However, we did not manually check for updates for all 93 devices before each experiment so there is a possibility that some devices might not be up-to-date while performing some experiments. Our study, therefore, reflects the state of IoT devices at the time of experiments.

4 LOCAL IOT TRAFFIC OVERVIEW

This section answers **RQ1** by providing a high-level overview of the network protocols used (and supported) by the IoT devices in our lab's local network. We use passively gathered data to characterize which devices communicate with each other on the local network, and what protocols they use if they do (§4.1). We complement this analysis with the results of our active scans (§4.2) and with the analysis of protocols used by mobile app analysis to communicate with IoT devices (§4.3).

4.1 Passive Traffic Captures

Figure 1 depicts how smart home devices communicate with each other as a network graph, where nodes are devices and edges are connections via UDP (dashed lines), TCP (solid lines), or both (thick solid lines). Nearly half (43/93) of the devices in our lab contact at least another device in the local network using TCP or UDP unicast traffic. This graph-based representation reveals clusters of devices that communicate with each other, either because they are from the same vendor or because they offer interoperable features.

At the protocol level, we find a substantial and diverse range of local network protocols (both standard and proprietary) and communication paradigms (unicast, multicast, and broadcast) across devices. Figure 2 represents this as a bar graph, with the x-axis representing protocols and the left y-axis representing the fraction of devices observed using each protocol. The color map shows the method used to infer the set of protocols supported by each device. Focusing on the blue bars, which represent passively observed traffic, we identify 21 protocols for local network communication according to our traffic classification method. We observe that an average IoT device in our lab supports 8 different protocols for local communication, with devices like Google's Nest Hub supporting up to 16. The most commonly used protocols are local network management ones like ARP (92%), DHCP (92%), and EAPOL (84%), followed by ICMP (78%) and IGMP (56%). All devices used IPv4 but only 59% supported IPv6. We also identify the newly-released IPv6-based Matter [22] traffic from Amazon Echo smart speakers.

Device discovery is key for platform interoperability, and accordingly we find that 93% of devices used broadcast-based protocols like ARP, XID/LLC, DHCP, while 73% of devices used multicast ones like mDNS, ICMPv6, SSDP, DHCPv6, IGMPv2/v3, and CoAP. We also identify several unidentified UDP-based protocols using both multicast and broadcast IPs.

At the application layer, the most popular protocols are standardized ones like HTTP (40%), SSDP (35%), and TLS (35%), followed by proprietary ones like TPLINK-SHP[28] (26%) and TuyaLP[27] (5%). The latter two also offer discovery capabilities. RTP (Real-time Transport Protocol) is also widely used (10%) for real-time data exchanges and device synchronization [109]. For example, Amazon Echo uses RTP over UDP:55444 for a multi-room music feature [24].

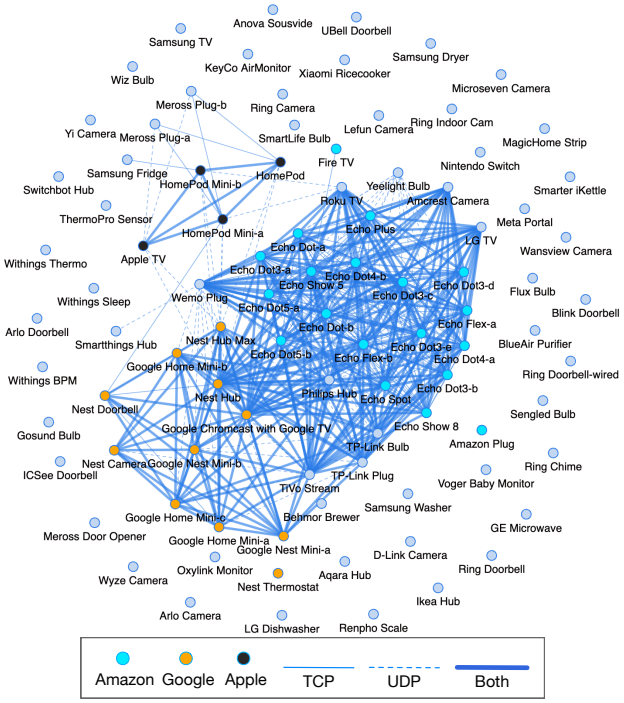


Figure 1: Transport layer device-to-device communication generated from the MonIoTr Lab passive dataset. For clarity, we show neither multicast- and broadcast-discovery protocols nor their interactions with smartphone apps (§4.3). See Figure 4 in the Appendix for isolated depictions of vendor-specific clusters.

We notice protocol usage differences between vendors, platforms, and device types. For example, Google and Amazon Echo smart speakers communicate with each other using TLSv1.2 (zoomed-in TCP graphs for Google in Figure 4a and Amazon Echo in Figure 4b in the Appendix), and unidentified UDP protocols. We refer the reader to the UDP graphs for Google in Figure 4d and Amazon Echo, with a clear coordinator in Figure 4e. In contrast, Apple devices use TLSv1.3, and other unidentified UDP and TCP protocols, which are possibly part of the HomeKit protocol [18]. We also observe inter-manufacturer communication for device interoperability. The reason is that either they are connected to a corresponding platform (*e.g.,* Alexa, Homekit) or they support functions that need local communication such as ChromeCast. Additionally, certain devices (*e.g.,* Philips Hue Hub, three smart plugs and bulbs, Amcrest Camera, and three smart TVs), facilitate platform interoperability through protocols like SSDP or TPLINK-SHP with open local APIs.

4.2 Active Scans

The orange bars in Fig. 2 (left y-axis) show the most relevant open services identified with active scans. Due to space limitations in Fig. 2, we group many protocols in the long tail as *Other-UDP* and *Other-TCP*. Many of these open services and protocols were not captured passively. In fact, the diversity of actions triggered on the devices during the experiments or the presence of at least two devices supporting this protocol limit the range of protocols captured using passive methods. For example, while 33% of IoT devices run an HTTP server on port 80, only 15% of them generated this traffic in our passive dataset. We find 178 unique open TCP ports and 115 unique open UDP ports on 61 devices which are associated with protocols such as HTTP(S), Telnet, NetBIOS, Socks5, or even DNS servers, some of which are known for being exploitable by other programs running on the local network [75, 84], a topic we explore later in the paper.

TCP ports 55442, 55443, and 4070 are open by 20% of the devices. These ports are primarily used by Amazon Echo devices for audio caching (HTTP), and device control (HTTPS) [1]. For UDP, port 68 (DHCP) is open in 7% of devices, followed by DNS port 53 (5%), and PTP port 320 (5%). As in passive traffic captures, we note that devices from the same vendor or offering the same functionality keep similar port combinations open.

4.3 App-to-IoT Local Traffic

The 2,335 mobile apps in our dataset (both companion and non-IoT apps) use 18 unique protocols for network discovery and local communication with the 93 IoT devices in the MonIoTr Lab. Using our mobile traffic instrumentation, we observe background ARP and ICMP traffic in the majority of the app tests. However, this traffic may be generated by the OS as raw packet access requires root privileges [4]. We note that our traffic attribution mechanism only applies to transport-layer protocols so we cannot confirm which component generates ARP and ICMP traffic.

Only a small fraction of analyzed apps use mDNS (6%), and SSDP (4%) for local network scanning and device discovery. Yet, we note that the NetBIOS protocol—designed for local network communication—is also used to scan the local network by 0.5% of the apps. For example, Device Finder [99] and Network Scanner [107] are two examples that employ NetBIOS to scan the local network and list all devices, likely intended for diagnosing home network issues. We find that 25% of the apps use TLS to interact with IoT devices in the local network once paired. However, IoT companion apps tend to use IoT-related custom protocols like TPLINK-SHP than regular ones. We also found that the Tuya and Chromecast companion apps already use the Matter standard to advertise their availability via mDNS [22].

4.4 Takeaways

By combining complementary traffic analysis methods, we find a substantial, diverse, and previously unreported use of network protocols for local communication among IoT devices. These comprise a large mix of protocols (*i.e.,* standard and proprietary) and communication paradigms (*i.e.,* unicast, multicast and broadcast). The use of these protocols extends to mobile apps, including both IoT companion apps and regular ones. We also note similar behaviors and protocol support for devices offering akin features or from the same manufacturer, mostly for interoperability and control. We leave further analysis of local communication patterns as future work. Next, we explore the security and privacy implications of this diverse range of local network communication protocols.

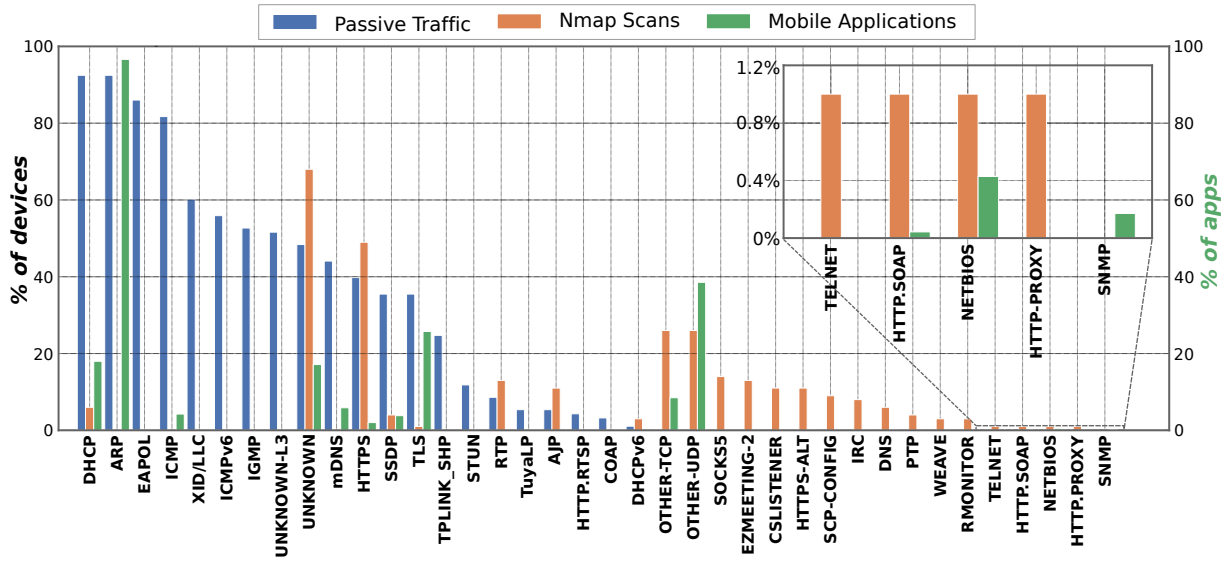


Figure 2: Percentage of protocols observed across the 93 devices deployed in our IoT devices, passively and with the active scans. We report the protocols observed when integrating 2,335 IoT and regular mobile apps. The y-axis values for the mobile app category refer to the number of tested apps observed using these protocols (N=2,335), rather than the number of IoT devices (N=93). The inset zooms into the long tail of the distribution.

5 THREAT ANALYSIS

This section answers **RQ2** by analyzing the threats arising from: (*i*) potentially vulnerable services and components (*e.g.,* outdated firmware versions and deprecated protocol versions), (*ii*) user actions (*e.g.,* open streaming activities), and (*iii*) dissemination of device and network information (*e.g.,* unique identifiers) that facilitate fingerprinting and cross-device tracking. We study how an attacker with access to the local network (*e.g.,* IoT manufacturers, compromised Wi-Fi APs, or mobile and smart TV applications) can exploit these issues following the threat model described in § 2. For this analysis, we combine our MonIoTr Lab passive dataset and honeypot with the output of a *Nessus* scanner deployed in the local network. We present the protocols in a bottom-up approach following the TCP/IP protocol stack.

	MAC	Device/Model	OS Version	Display name	UUIDs	GWid Prod. Key	OEM id	Geolocation	Outdated OS/SW
ARP	✓								
DHCP	✓	✓	✓	✓					✓
mDNS	✓	✓		✓					
SSDP		✓	✓	✓	✓				✓
TuyaLP						✓			
TPLINK	✓	✓		✓			✓	✓	

Table 1: Observed information exposure in the local network by IoT devices per discovery protocol.

5.1 Discovery Protocols

We now describe how IoT devices use discovery protocols like UPnP and mDNS to scan the local network, and the associated risks. Table 1 summarizes potentially sensitive data that we observe exposed by IoT devices via these protocols in our IoT lab. At the end of this section, we investigate the frequency of discovery messages and which devices respond to them, because high frequency scans can reveal further information about the household. Appendix §D explains how we analyze the scanning frequency.

ARP. ARP is a standard Layer 3 protocol to discover MAC addresses associated with IP addresses in local networks. Prior to conducting the analysis, our working hypothesis was that ARP traffic would be used only for resolutions between a device and the gateway, or between two devices that identified each other using other discovery protocols. However, ARP may be abused to harvest the MAC addresses of all the devices in the local network, which can be valuable persistent device IDs that enable geolocation, fingerprinting, and cross-device tracking. In fact, we find that devices perform ARP scans in different ways. Amazon Echo devices perform daily broadcast ARP scanning of the entire local IP space, and also send targeted unicast ARP messages to 83% of other devices[3]. Interestingly, while only 58% of devices in our testbed respond to Echo's broadcast ARP scans, all of them reply to the unicast ones, thus revealing their presence. Six devices also send requests for public IPs, which may be an intentional behavior to identify device and network misconfigurations [59].

ICMPv6. This protocol is used by 55% of the devices for multicast discovery, adhering to the SLAAC standard [91], indicating IPv6 networking support. ICMPv6 queries can include the MAC addresses of the sender as part of the IPv6 Neighbor Advertisement

[3]Note that we found no discussion of local network communication or local scanning in Amazon's privacy policy.

standard [111], which could be collected by other devices on the local network. It is worth noting that we observed the Nest Hub sending multicast ICMPv6 requests to 2,597 distinct addresses.

DHCP. While our testbed uses DHCP [49] by default for IP addressing, 86 devices actively request 30 different data types from other devices using DHCP. This includes network-relevant information (e.g., Subnet Mask, Router address, and DNS server address) but also unexpected requests associated with deprecated standards (*e.g., SMTP Server*, *Name Server*, and *Root Path*). Unfortunately, devices carelessly respond and expose sensitive information about themselves such as their hostname, and DHCP client name and version. We identified hostnames for 67% of devices, and 16 unique DHCP client versions from 40% of devices. We find that 37 devices—including Amazon Echo and Google ones—use old or custom DHCP client versions, potentially leaving them open to vulnerabilities (*e.g.,* [2]). Interestingly, we observe a variety of hostname naming methods across devices. For example, a combination of device names and MAC address (Ring Chime), their device model name (*e.g.,* Ring Cameras), or a combination of model or vendor name with partial MAC address (*e.g.,* Tuya devices). Other devices expose user-defined display names (*e.g.,* Google and Apple smart speakers) that could reveal sensitive data like the name of the room or owner (*e.g.,* "Jane Doe's Kitchen Homepod"). A small number of vendors, however, follow more secure approaches that minimize data exposure. For example, GE Microwave and TiVo Stream obfuscate their hostnames and names, respectively, using variable random bytes for each request.

mDNS. We find that 44% of the IoT devices in our testbed use mDNS [43] for device discovery, name resolution, and service advertisement within the local network. Among them, 90% actively send queries and nearly 98% send multicast responses, while a relatively small fraction, about 20%, also send unicast responses. mDNS queries and responses reveal hostnames representing the services supported by the device, such as casting (*e.g.,* Viziocast), printing (*e.g.,* IPP), platform-specific services (*e.g.,* Alexa), commercial streaming services (*e.g.,* Spotify), IoT standards (*e.g.,* Matter, Thread), and networking protocols (*e.g.,* Bonjour Sleep Proxy). Importantly, mDNS hostnames are often constructed by appending unique identifiers such as MAC addresses, device IDs, serial numbers, or other device-specific tags, thus creating distinct hostnames that facilitate device discovery and communication in a network, but also cross-device tracking and fingerprinting. For example, Philips Hub reveals MAC address in its mDNS hostnames; worse, the `.local` URL of Spotify Connect devices is composed of MAC address, device ID and special UUIDs—perhaps session IDs—, depending on the service being advertised with the ZeroConf API [112]. For a detailed illustration, readers can refer to the example provided in Table 5 in the Appendix.

SSDP/UPnP. SSDP/UPnP protocol [32] is a UDP-based protocol used by 32% of the devices in our testbed. [4] All Amazon and Google smart speakers and hubs support SSDP. Unlike mDNS, SSDP allows both active service discovery or passive service broadcasting. In active SSDP, devices send `M-SEARCH` queries to IP multicast groups, while in passive SSDP, devices send multicast `NOTIFY` messages. In our testbed, 26 out of 30 devices using SSDP send M-SEARCH messages and 7 of the 30 devices send NOTIFY ones. Only 9 devices respond to SSDP multicast messages, including 4 smart TVs and two Nest hubs with Chromecast. We observe that Amazon smart speakers search for 2 generic network services (`ssdp:all` and `upnp:rootdevice`), while Google products search for specific ones. We also observe how 8 devices expose device information such as UUID, OS version, UPnP implementation name and version. The use of deprecated UPnP versions or its incorrect use can also have security implications. Fifteen years after the release of UPnP 1.1, 9 IoT devices still use UPnP version 1.0, which has been demonstrated to be exploitable [34]. Roku TV sends SSDP requests related to the IGD (Internet Gateway Device) protocol, which can be exploited by malware [7]. Some vendors have reported that their newer firmware versions no longer run deprecated versions. We also notice possible misconfigurations. For example, the Fire TV sends NOTIFY messages to announce a /16 local IP address which is not supported on the LAN; and LG TV's requests are sent by three different firmware versions: WebOS TV/Version 0.9, WebOS/1.5, and WebOS/4.1.0.

Other discovery protocols. We now discuss unusual or platform-specific protocols used for both discovery and communication.

- **TPLINK-SHP.** Google Home and Amazon Echo smart devices—and their companion apps—use the TPLINK-SHP protocol over UDP broadcast to discover and extract information from TP-Link devices. These respond to the queries with their system information, including the device latitude and longitude in plain text as shown in Table 5. The disclosure of this information allows an eavesdropper in the local network to get the geolocation of the home along device metadata such as device name, Original Equipment Manufacturer (OEM) ID, and device status. Other smart devices also use TPLINK-SHP over TCP to control TP-Link devices. Therefore, a local attacker could control TP-Link devices via this protocol without authentication, as shown by prior work [28, 67].
- **TuyaLP.** Tuya devices (and companion apps) use the custom TuyaLP protocol on UDP ports 6666 or 6667 to broadcast discovery messages. Tuya devices do not respond to requests unless they come from their companion apps. The Jinvoo Bulb sends its GWid and Product key in plaintext.
- **CoAP.** Three devices (Samsung fridge, and two Homepod Minis) use the CoAP standard [110], a web transfer protocol for constrained devices with discovery capabilities. Specifically, the fridge requests a URI corresponding to IoTivity, a software framework for local communication developed by Samsung [72]. We were unable to decode the payloads from Homepod Mini traffic.
- **Unidentified traffic.** We note that 45 devices generate traffic that we were unable to classify with our methods. However, the periodicity of some of this traffic suggests that they may be related to device discovery, network scanning, or synchronization. For example, Echo devices periodically send a broadcast packet to UDP port 56700 every 2 hours, which seems to be used by Lifx [20], a smart device manufacturer not represented in our testbed. While most of this traffic is encoded, some plaintext cases contain strings with unknown meaning, preventing us from drawing conclusions about their purpose and risks.

[4] We note that SSDP is the foundation of the discovery protocol UPnP.

Discovery Intervals. The ability to scan the entire IP space at a high frequency allows identifying temporal patterns, such as smartphones entering and leaving the households. Therefore, a high scanning interval can reveal the social network of the household, and perform for cross-device and mobility tracking. Amazon Echo, Google, and Apple devices periodically send most mDNS queries every 20s-100s depending on the hostname. Similarly for SSDP, we find that Google smart speakers and hubs send active SSDP requests to IP multicast groups every 20s, whereas Echo smart speakers send active SSDP scans every 2-3 hours. We also note that short scanning intervals can increase network congestion and energy consumption in resource-constrained or battery-powered devices (*e.g.,* smart cameras, sensors, and smart locks).

5.2 Other Protocols

We now analyze those protocols used for exchange of data and commands. Unlike discovery messages, which often follow standard protocols and are—by design—readable by any device on the network, these non-discovery protocols may be proprietary and encrypted (or encoded). As we discussed in § 3, this limits our visibility to analyze this type of traffic passively and infer their purpose. For this reason, we only discuss those protocols and devices for which we have identified a potential threat.

Plaintext HTTP. Combining the insights of our passive and active analysis, 33 devices in our testbed communicate using plaintext HTTP traffic, 26 of which appear only as clients and 5 as servers. The LG TV and Nest Hub send both HTTP requests and responses. By inspecting the metadata of HTTP requests, we detect 17 devices related to SSDP/UPnP services, which offer control such as multi-screen casting, and could reveal user activities within the home. The only devices that include a User-Agent string in the HTTP headers are Google products and the LG TV. Google devices (Chromecast OS) include the OS version, the device type, and the Chromecast firmware version in the HTTP headers, while LG TV (LG WebOS) reports the OS and the supported UPnP versions.

Nessus collects service banners to identify the web server and the exact version deployed on 6 devices. This knowledge can be leveraged to seek exploits specific to a given software version. For example, the Lefun camera runs an HTTP server that allows accessing backup files. *Nessus* retrieved content containing server configuration details. Similarly, the Microseven camera runs JQuery 1.2, known to have multiple XSS vulnerabilities [87, 88]. This device also runs a remote service that allows unauthenticated users to view camera snapshots. *Nessus* acquired one snapshot from the remote camera using the specific ONVIF requests [98]. Moreover, our scanner also listed all user accounts on the device, and identified the directory where camera recordings get stored.

TLS. We find that 32 devices (from Google, Amazon Echo, and Apple ecosystems) predominantly use TLS to communicate with each other in the local network. In fact, most TLS traffic on the local network uses recent protocol versions (1.2 and 1.3), and the devices use custom public key infrastructures (due to the fact that devices on a local network generally cannot obtain globally unique DNS names for validating the subject name). As a result, the devices already supporting TLS are likely robust against eavesdroppers, yet there are noticeable differences across them:

- Google smart speakers, Nest hubs, Google Chromecast (with Google TV), and TiVo TV (based on Android TV) communicate using TLSv1.2 and use their own internal PKI with root certificates not in any trust store (according to Censys) and with leaf certificates that expire in 20 years. *Nessus* scanner detected one high-severity issue across all these devices that run TLS on port 8009 due to the small size of the encryption key (64-122 bits). This facilitates birthday attacks to obtain cleartext data in the presence of a long-duration encrypted sessions [85]. We did not verified whether this vulnerability is exploitable.
- Amazon Echo devices, also employing TLSv1.2, use self-signed TLS certificates issued by each device with a validity of three months. The Issuer and Subject's Common Name of each certificate are an IP address in the *192.168.0.0/16* prefix or *0.0.0.0*. This set of devices uses two-way TLS authentication, where both client and server send their respective certificates.
- Apple TV and smart speakers are the only ones communicating with each other using TLSv1.3, with certificates encrypted in their handshakes, similar to their device-to-cloud traffic [100].
- Through active scans, we find that D-Link camera, SmartThings and Philips Hue hubs differ from the previous vendors, since they all use self-signed certificates that last from 20 up to 28 years. However, we cannot infer for which purpose these certificates are used since as we do not passively observe any local communication using them.

DNS. Apple HomePod Mini and the WeMo plug run a DNS server susceptible to DNS server cache snooping and remote information disclosure [93]. This vulnerability potentially enables malicious actors within the local network to track user activity by discovering recently-resolved domains, thus exposing visited hosts. However, we did not validate the feasibility of this attack. In the case of Apple HomePod Mini, *Nessus* identified the DNS server as SheerDNS 1.0.0, an old version with known security flaws [92]. Additionally, when querying for device hostname using Nessus, the DNS services on all three of these devices revealed the testbed's remote host name and private IP of the DNS server.

5.3 Takeaways

This section discovers and analyzes a prevalent issue in smart IoT devices within the home: the exposure of sensitive device and network information through discovery protocols, including MAC address, UUIDs, device model name and user-defined display names, OS version, outdated or potentially vulnerable software on devices, and geolocation. As we analyze in the next section, this information can inadvertently reveal sensitive information about users and households, allowing targeted attacks and/or household fingerprinting for advertising or surveillance purposes. Our analysis also reveals potential risks associated with the use of unencrypted HTTP in the network, the presence of exploitable services and potential developer misconfigurations and errors.

6 CASE STUDY: HOUSEHOLD FINGERPRINTING

This section aims to answer **RQ3**. It provides evidence of how the uncontrolled dissemination of sensitive device and network

information in the local network is abused by mobile apps (§6.1) and third-party libraries (§6.2). It then leverages IoT Inspector's crowdsourced data to measure the entropy of the information disseminated by IoT devices in 3,860 households (i.e., users) (§6.3). Our results demonstrate that harvesting this information is a valuable asset for organizations performing privacy-intrusive practices such as household fingerprinting, advertising, and cross-device tracking.

6.1 Device Data Dissemination Beyond the Local Network

Using our instrumented Android smartphones, we trace the dissemination of information collected from discovery protocols to endpoints hosted in the cloud. Out of the 2,335 analyzed mobile apps, 9% of them use at least one of the following protocols to scan the home network: mDNS (6.0% of apps), SSDP/UPnP (4.0% of apps), and NetBIOS (0.5% of apps). In most cases, the use of these discovery protocols is required to deliver their service and control the device, as in the case of mobile companion apps. Note that 10 mobile apps—only 2 of them falling in the IoT app category—use NetBIOS, a protocol that was not observed in the passive traffic captures (§4). However, the use of these protocols can be potentially misused as described below.

MAC Addresses. Six IoT apps relay MAC addresses of IoT devices to the cloud, with recipients being either first-party domains (e.g., Alexa) or third-party providers like Tuya, a China-based IoT platform provider, and Amplitude, an analytics service. For example, Amazon Alexa's companion app collects through Amazon Echo (*i*) the MAC address of different devices configured on Alexa—such as the smart microwave, and (*ii*) the Philips Bridge ID, a unique device identifier [70]. However, this app also shares the MAC address of other devices in the network like the Meross smart plug, despite it not having been configured to work with Alexa. The Nest Hub behaves similarly, sharing the Wi-Fi AP MAC address to the Chromecast app, even when the app and the devices are not paired.

Fire TV and Roku TV expose their own MAC address to 10 and 31 apps (such as casting, streaming or peripheral controlling apps) using SSDP/UPnP, respectively. Some apps also enrich device MAC addresses with other sensitive information obtained using Android APIs. For example, the Blueair purifier companion app uploads to its servers the purifier's MAC address along with the mobile device's coarse geolocation and its Android Advertising ID (AAID) (a resettable ID for advertising purposes [60]). This practice has direct privacy consequences: by linking persistent IDs such as MAC addresses with pseudo-resettable ones and accurate geolocation data, the developer defeats any user attempt to reset these values. This behavior is in violation of best practices for user ID collection [60]. Additionally, organizations collecting that information can use it to feed geocoding services.

Interestingly, we also observe dissemination in the downlink traffic. Specifically, 13 companion apps associated with IoT devices in the testbed (*e.g.,* Amazon Echo) receive MAC addresses from other devices in the local network via Tuya machines or AWS instances. Unfortunately, we cannot tell how and when this information was obtained for the first time, but we believe that this may have happened at the initial pairing stage.

Router Information. As we discuss in §2, access to Wi-Fi Access Point's MAC address and its BSSID are protected by a system permission in Android devices. However, we note that multiple apps in our dataset collect and upload them to the cloud: the router MAC addresses (28 apps), *i.e.,* the bridge device described in § 3, the Router SSID (36 apps), and the Wi-Fi MAC address (15 apps). We also find non-IoT apps from the same developer actively scanning the local network to identify all nearby Wi-Fi MAC addresses and BSSIDs, subsequently transmitting this data to MyTracker [12], a Russian analytics and attribution SDK. We note that this data dissemination occurs without requesting the necessary Android permissions to access these data types as discussed in § 2. Additionally, FireTV, RokuTV, and Nest Hub relay the Router SSID to eight different apps, including casting apps and streaming apps like YouTube.

TPLINK-SHP Identifiers. Certain identifiers transmitted in TPLINK-SHP broadcasts, such as bulb device ID, hardware ID, OEM ID, and plug device ID, are relayed to the cloud by TP-Link supporting devices. The TP-Link companion app uploads this information to its own servers, along with the geolocation of the plug and mobile device. This information could potentially be used to track user movements and fingerprint the household. The Amazon Alexa companion app collects and uploads to its own servers the device IDs for both the TP-Link bulb and plug, along with the TP-Link OEM ID, collected from TPLINK-SHP.

6.2 SDK libraries

Third-party SDKs associated with advertising and tracking companies can leverage their presence in thousands of mobile apps to scan and obtain device information in the local network, often without app developer awareness. We discuss three particularly brazen cases of third-party libraries present in Android apps gathering local network information:

Umlaut InsightCore SDK. The app Simple Speedcheck [71] includes a monetization library called "Umlaut insightCore SDK." This library conducts SSDP discovery operations, specifically targeting the UPnP Internet Gateway Device (IGD) service active on local devices. Through such services, the library can potentially control and manage network settings, such as configuring port forwarding or managing network connections, thus potentially exposing these devices to unauthorized external access. Its privacy policy indicates that it uploads system and network information such as the list of connected devices in the local network and geolocation [5]. According to their privacy policy, this data is collected *with the purposes of supporting the development and improvement of communication networks as well as usage-related analyses.*

AppDynamics. In version 6.18.3 of the CNN app [44], users can cast content to a smart TV on their local network by leveraging SSDP/UPnP. The app incorporates a Cisco-owned app performance analytics and profiling SDK called AppDynamics [15], which tracks UI and network event. By offering code instrumentation and wrapper APIs for common network library callbacks (*e.g.,* okhttp), the AppDynamics SDK allows developers to track each network request. As a result, AppDynamics not only monitors communication over the Internet but also local traffic activity such as the file descriptor of the local URL of the UPnP/SSDP XML file. In the case of CNN app,

when searching for devices to cast content, it receives device information through UPnP. Intriguingly, AppDynamics accesses this information arbitrarily. This potentially enables the profiling of users' home networks and their activities. This is particularly concerning, as UPnP device descriptors contain device information, supported UPnP services, and their MAC addresses. Additionally, AppDynamics actively tracks requests to events.claspws.tv/v1/event, with URL parameters that include base64 encoded Wi-Fi AP SSID, Android device ID, Identifier for Advertising (IDFA), and list of devices with screens in the local network via UPnP/SSDP; these details are arbitrarily collected by AppDynamics through a side channel [86]. The latest version of the CNN app (7.23.1) at the time of writing has removed the casting feature, thereby preventing AppDynamics from collecting SSDP device information. Nonetheless, the request tracking functionality remains integrated into the app, thus continuing to pose potential privacy risks.

innosdk. Ten apps conduct NetBIOS scans, three of which utilize ARP either by implementing it natively or using the `libarp.so` library to collect MAC addresses and subsequently send targeted NetBIOS requests with the payload structure as shown in Table 5 in the Appendix. The transmission of this string is an attempt to do an enumeration of any available NetBIOS shares on the local network. The "Lucky Time - Win Rewards Every Day" app [21] sends a UDP datagram to every IP in the `192.168.0.0/24` prefix, regardless of whether there was a machine assigned to that IP. This scan is performed by the third-party library "innosdk," which connects to the endpoint `gw.innotechworld.com`. It is also noteworthy that this network transmission is algorithmically generated by the SDK instead of stored as a constant, perhaps to avoid being detected as obvious malware. This app has been removed from the Google Play Store since our analysis.

6.3 Household Fingerprintability: Entropy Analysis

The previous sections identify that large numbers of unique device identifiers and metadata are exposed in the local network, and this can have privacy and security implications as Snowden revealed [52]. We now show how such identifiers and metadata can lead to a fingerprinting attack, *i.e.,* where a client can uniquely identify a household based on observed local network traffic.

To do so, we rely on IoT Inspector's crowdsourced data, which captures the organic behavior of 12,669 IoT devices deployed in 3,860 households, associated with 264 different products from 165 vendors. We quantify the extent to which unique household fingerprints can be generated by measuring the entropy of the information gathered by experiment participants. Specifically, we analyze the information present in plaintext mDNS and SSDP responses [69]. From every mDNS and SSDP payload, we extract what appears to be unique identifiers, including names, UUIDs, and MAC addresses:

(1) **Names.** For example, a device in the dataset has the following value under the `name` field in its SSDP response: "Roku 3 - `REDACTED`'s Room". Using regular expressions, we search for responses where an English word is followed by an apostrophe, "s", space, and another word.
(2) **UUIDs.** We search for standard UUID patterns [79].

#	Pdt	Vdr	Dev	Σ Hse	Identifier(s)	Hse	Ent
0	154	107	4,175	1,811	N/A	N/A	N/A
1	160	100	6,915	3,007	name	2 (50.0%)	3.4
					UUID	2,814 (94.2%)	8.9
					MAC	572 (94.4%)	7.8
2	76	59	1,577	1,201	name, UUID	22 (81.8%)	12.3
					UUID, MAC	1,182 (95.6%)	16.7
3	1	1	2	2	name, UUID, MAC	2 (100.0%)	20.1

Table 2: Information exposed via mDNS and SSDP. "#" counts identifier types exposed, including first names, UUIDs, and MAC addresses. "Pdt" counts distinct products exposing this information, "Vdr" counts vendors across these products, "Dev" counts distinct devices, and "Σ Hse" counts households for these devices. The "Identifier(s)" column shows which identifier(s) are exposed over how many households ("Hse"), with the percentages of households that can be uniquely identified in the parentheses. "Ent" shows the entropy.

(3) **MAC Addresses.** We search for the standard MAC address formats, with and without ":" and "-". To reduce false positives, we compare each potential MAC address with the OUI (the first three bytes of MAC addresses) that IoT Inspector collects for each device [69], and filter out cases where the OUI is not a substring of the regular expression's result.

To estimate household fingerprintability, we count the number of unique products, vendors, devices, and households where we observe the identifier for each type of exposed identifier. Then, we compute the entropy: $-log_2(1/N)$, where N denotes the number of distinct values for each type of exposed identifier, as proposed by the EFF tool "Cover Your Tracks" to measure web fingerprintability [51].

Table 2 reports the results of the entropy analysis. The first row shows 154 products (associated with 107 vendors, covering 4,175 devices in 1,811 households) exposing none of the three types of identifiers—name, UUID, or MAC address— in the collected mDNS or SSDP responses. The second row shows products that exposed at least one type of identifier; *e.g.,*, 2,814 households exposed UUIDs only. The table also shows cases where a combination of identifier types was exposed; for instance, some 1,182 households exposed both the UUIDs and MAC addresses. The last row shows 2 households exposing all three identifier types; both households have Roku TVs, which exposed the user's name ("`REDACTED`'s Roku Express"), UUIDs, and the MAC addresses (which are a part of the UUIDs).

The three types of identifiers could potentially be abused to fingerprint households by third-party apps, SDKs or other co-located IoT devices. We illustrate the fingerprintability in the "Ent" (entropy) column in Table 2. We obtain entropy values over 12.3 in households with at least 2 devices. As a reference point, HTTP User Agent strings have an entropy of 10.5 if used to track web users [51]. In addition to entropy analysis, we also measure how many households can be uniquely identified, as shown in the parentheses in the "Hse" column. For example, 2,814 households exposed UUIDs; the unique UUIDs could identify 94.2% of these households. If a third-party tracker were to combine UUID and MAC addresses (exposed in 1,182 households), the unique combinations of UUID and MAC addresses could identify 95.6% of these households. That being said, we do not know whether these values remain unique per

device *over time* (e.g., over days and months), as the median time of data collection per household is less than one hour [69]. Still, these results further confirm that the three types of identifiers are mostly unique across households and could lead to fingerprintability.

We note that a typical smart home in IoT Inspector's dataset is smaller than the setup present in the IoT Lab. A regular household in the IoT Inspector dataset has a median of 3 different IoT devices that often communicate with each other over TCP and UDP connections. Hence, the results presented in this analysis are a lower-bound estimation of the fingerprintability of these households.

6.4 Takeaways

Multicast protocols like mDNS and SSDP expose sensitive identifiers, such as first names, UUIDs, and MAC addresses that have been already collected by mobile applications and analytics SDKs. Most of these identifiers and other device metadata are unique per device so they enable accurate household fingerprinting and user tracking across devices and networks. Our study predominantly monitors these information relays and side channels discussed in section 6.1 and 6.2 through mobile companion apps. It is crucial to understand that similar vulnerabilities might exist in direct device-to-device interactions. However, due to the opaque or "black-box" nature of many IoT devices and our limited visibility into their encrypted traffic, our capacity to directly observe and diagnose device-to-device channels is constrained. Thus, our observations via mobile apps serve as a proxy, suggesting broader implications for IoT security and privacy beyond just smartphone interactions.

7 DISCUSSION

We argue that local networks must be considered a zero-trust environment. While ARP, UPnP, and mDNS protocols are essential for seamless integration and device communication in smart home systems, they introduce privacy and security risks that must be considered. As we demonstrate in §6, mobile apps, spyware, and third-party SDKs already harvest this information for commercial and malicious purposes. Informed by our empirical results and the conversations with IoT vendors triggered by our responsible disclosure, this section discusses possible mitigations to these risks.

Usable Security and Privacy Controls. Transparency in the IoT space is generally deficient, and the visibility of local interactions is not an exception. Many IoT devices are designed to connect to networks or other devices without requiring any user intervention or authorization. In fact, many IoT platforms provide APIs to third-party app developers to integrate these capabilities in their software. While this may reduce friction for device operation, it can also lead to a total lack of user awareness about which devices are connected to their home networks and what data is being shared. The lack of transparency and controls can leave users vulnerable to security and privacy threats. To mitigate these issues, mobile and IoT platforms must offer usable solutions to force third-party apps to obtain explicit consent from the user at runtime before scanning and connecting with other devices in the local network. We believe that the model introduced by iOS to prevent local network scanning is a move in the right direction. However, incorporating these changes in mobile and IoT platforms may take a long time, as they may require profound architectural changes that can introduce unexpected compatibility issues. These measures can be complemented by the vetting processes that platforms have to control the dissemination of malicious and deceptive programs in their stores.

Secure-by-design Firmware and Timely Updates. IoT devices have been historically shipped with inadequate default security settings and unnecessary capabilities that can be abused to enable attacks on other systems. Our findings in §5 confirm that many home IoT devices run outdated services and use protocols with known vulnerabilities. While we acknowledge that tracking newly discovered vulnerabilities and shipping security patches in a timely manner is challenging, vendors cannot dismiss the crucial need to keep their products updated throughout their life cycle.

Supply Chain Analysis and Hardening. Several devices in our setting, such as smart TVs, run services with different versions of UPnP, some of which are vulnerable. These products are built in a (possibly complex) supply chain involving different teams or even external organizations, each with its own dependencies and development practices. The presence of a potentially large number of actors in the supply chain makes it hard to achieve goals such as secure-by-design and routine security updates. One key complication is the lack of knowledge about the supply chain itself. The use of Software Bill of Materials (SBOMs) has recently gained momentum as a measure to address some of these issues.

Standardization Efforts. The results of our §6 experiments demonstrate that network and device data broadly available in the local network can enable cross-device tracking and household fingerprinting. Standardization methods can be an effective mitigation, by promoting the adoption of privacy- and security-by-design principles across vendors like data exposure minimization or ID randomization, ultimately reducing consumer risks. One prominent example of a new protocol standard is Matter [22], a protocol already being adopted by major IoT vendors. However, while Matter is positioned as an alternative cross-platform method for local device communication, it does not address most of the threats we identified in this paper (yet). In fact, Matter still considers the local network as a trusted environment and exposes MAC addresses in mDNS discovery as well [64]. Recent initiatives such as ETSI's "Cyber Security for Consumer Internet of Things: Baseline Requirement" [3] standard already have considerations for data processing and collection in consumer IoT devices. However, they are very generic and do not consider the risks associated with the dissemination of sensitive device and network information within the local network that enable fingerprinting attacks and cross-device tracking. Other efforts such as OWASP's Mobile Application Security group can also incorporate and discuss threats such as the fingerprinting attacks described in this paper [26].

Regulation and Certification. A recent wave of regulations in the US and the EU aim to improve the security of the supply chain for connected products. Both the EU Cyber Resilience Act (CRA) [8] and the US National Cybersecurity Strategy [11] seek to ensure that connected digital products are manufactured according to best cybersecurity practices and provided with the latest security updates throughout their life cycle. In the case of the CRA, non-adherence to a baseline of essential cybersecurity requirements will translate into fines, and some critical devices will need to comply

with stricter standards and possibly go through third-party audits. Standardization bodies like NIST [96], and industry-led efforts such as IoXt [73], aim to facilitate these audits through IoT certification programs and providing security and privacy guidelines for vendors. Though still in a preliminary stage, the combination of these regulations and auditing frameworks with more privacy-oriented IoT standards have the potential to shape market forces to adopt best security and privacy practices.

8 RELATED WORK

IoT Traffic Characterization. Prior work conducted general studies to characterize the behavior of smart homes in the wild, thus capturing device behavior with real user stimuli [39, 68, 69, 77, 80, 105, 108, 115]. Typically, researchers rely on instrumented IoT gateways to monitor device network traffic [81, 105, 108]. While this approach revealed the dissemination of personal data (*e.g.,* MAC addresses and geolocation) to the cloud using unencrypted network protocols [105], they offer little insight into the local network communication behavior of IoT devices. Several studies have highlighted security and privacy risks associated with IoT local network deployment and protocols such as mDNS and UPnP [58, 66, 75, 78, 101]. However, the inherent risks and the potential for cross-device data leakage via local smart home communication to third parties have remained unexplored in the research literature. While Hakim et al. [66] proposed honeypots to identify malicious UPnP activities, our honeypots emulate real smart devices to monitor data dissemination. Könings *et al.* [78] investigated zero configuration networking, emphasizing the need for improved user awareness and changes in device naming conventions to enhance privacy, aligning with our findings.

Vulnerability Analysis. In the wake of the Mirai botnet attack [35], the research community has intensely scrutinized the security of IoT devices, uncovering a range of vulnerabilities and attack vectors [34, 35, 54, 119, 120]. While IoT vendors have strived to respond effectively by adopting various best practices, several core device services can still be prone to side-channel data leakage [97]. Consequently, IoT platforms like smart speakers, hubs, and smart TVs have gradually moved towards implementing permission-based models to enhance user security and privacy. Yet, Fernandes *et al.* [55–57] statically analyzed smart apps to detect side-channels that allow circumventing the permission model of these platforms.

Application Analysis. The research community has proposed various methods to detect harmful and privacy-invasive behaviors across mobile and smart IoT platforms [31, 89, 95, 103, 106, 116, 119]. However, the opaque nature of IoT devices presents considerable challenges to fully understanding the privacy and security risks when multiple devices are co-located in a smart home system, particularly their traffic payloads [100]. Yet, prior efforts analyzed the companion apps of smart devices to indirectly infer security vulnerabilities of smart devices [42, 104] and data exposure from the companion apps [40, 90, 118]. IoTProfiler [90] used NLP, ML, and static program analysis to detect potential data leaks in companion apps. Their findings that over 70% of IoT devices rely on local connections with mobile apps for initial data processing aligns with our observations. In contrast, we observe that any regular app can inadvertently integrate SDKs that scan the network and expose to remote servers the data made available by devices in the same smart home ecosystem, thus exploiting side-channels to circumvent the Android permission model [103].

9 CONCLUSION

This paper presented the first analysis of local network interactions observed in an IoT testbed comprising 93 devices, as well as their interactions with mobile apps. Through a rigorous combination of complementary methodological approaches, we demonstrated the concerning prevalence of vulnerable services and components in smart home IoT devices, and how the misuse of local network protocols exposes sensitive network-, user-, and device-specific data to other devices and third-party mobile apps and SDKs running on the local network.

We provided evidence that such data is collected by mobile apps and third-party SDKs and transmitted to remote servers. Additionally, we performed the first analysis on the entropy of data that is exposed over discovery protocols such as mDNS and SSDP, demonstrating that this information is highly effective for performing household fingerprinting and cross-device tracking. The threats we consider in this paper are real, important, and unaddressed by Android's permission model.

As we showcased in our analysis, many devices are still vulnerable to the threats described above, and these issues are actively exploited by mobile apps and SDKs. We discussed several potential mitigations, from technical solutions to standardization and regulatory actions, and we encourage device manufacturers, software developers, and policymakers to take our findings into account to improve privacy and security for home IoT deployments and enhance transparency in the IoT space. We note that Google is engaging with us to explore mitigations that could be implemented via the Android OS, app review processes, and general IoT standardization efforts.

ACKNOWLEDGEMENTS

We thank our shepherd and the anonymous reviewers for their valuable comments and suggestions. We also thank Nipuna Weerasekara (IMDEA Networks) for his help developing the PoC apps. The project is partially funded by the NSF ProperData award (SaTC-1955227). IMDEA Networks and UC3M are funded by the EU H2020 grant TRUST aWARE (101021377). The IoT Inspector work is funded in part by National Science Foundation Award CNS-2219867. Srdjan Matic was partially supported by the Atracciòn de Talento grant (Ref. 2020-T2/TIC-20184) funded by Madrid regional government, the PRODIGY Project (TED2021-132464B-I00) funded by MCIN/AEI/10.13039/501100011033 the European Union NextGenerationEU/PRTR, and the grant PID2022-142290OB-I00, funded by MCIN/AEI/10.13039/501100011033 and by the ESF+. Narseo Vallina-Rodriguez has been appointed as a 2019 Ramon y Cajal fellow (RYC2020-030316-I) by the Ministry of Science of Spain. The opinions, findings, and conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the views of any of the funding bodies.

REFERENCES

[1] 2018. Breaking Smart Speakers We are Listening to You. https://www.youtube.com/watch?v=3sLC0XaqvMg. Accessed on May 26, 2023.

[2] 2019. CVE-2019-11766. Available from MITRE, CVE-ID CVE-2019-11766.. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-11766

[3] 2020. ETSI EN 303 645 V2.1.1. Cyber Security for Consumer Internet of Things: Baseline Requirements. https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf.

[4] 2020. Google Issue Tracker: Allow binding privileged ports or creating raw sockets. https://issuetracker.google.com/issues/156966374. Accessed on May 21, 2023.

[5] 2020. umlaut insightCoreSDK Data Privacy Policy. https://web.archive.org/web/20220514011406/https://tacs.c0nnectthed0ts.com/policy1/data_privacy.html. Accessed on May 26 2023.

[6] 2021. IRB Approval with Protocol. https://inspector.engineering.nyu.edu/irb. Accessed on Sept 11, 2023.

[7] 2021. This sneaky malware will cause headaches even after it is deleted from your PC. https://web.archive.org/web/20210126173325/http://www.zdnet.com/article/this-sneaky-malware-will-cause\-headaches-even-after-it-is-deleted-from-your-pc/. Accessed on May 11, 2023.

[8] 2022. EU Cyber Resilience Act. https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act. Accessed on Sept 21, 2023.

[9] 2023. Frequently Asked Questions. https://github.com/nyu-mlab/iot-inspector-client/wiki/Frequently-Asked-Questions. Accessed on Sept 11, 2023.

[10] 2023. If an app would like to connect to devices on your local network. https://support.apple.com/en-us/HT211870. Accessed on Sept 11, 2023.

[11] 2023. USA National Cybersecurity Strategy. https://www.whitehouse.gov/wp-content/uploads/2023/03/National-Cybersecurity-Strategy-2023.pdf. Accessed on Sept 21, 2023.

[12] [n. d.]. Analytics and attribution system for mobile apps and websites. https://tracker.my.com. Accessed on May 11, 2023.

[13] [n. d.]. Android API Permission Research. https://evolving-android.cpsc.ucalgary.ca/index.html. Accessed on May 11, 2023.

[14] [n. d.]. AppCensus. https://www.appcensus.io/. Accessed on May 21, 2023.

[15] [n. d.]. AppDynamics. https://www.appdynamics.com/. Accessed on May 11, 2023.

[16] [n. d.]. Frida.re. https://frida.re. Accessed on May 11, 2023.

[17] [n. d.]. Google Playstore. https://play.google.com/store. Accessed on May 11, 2023.

[18] [n. d.]. HomeKit by Apple. https://www.apple.com/ios/home/. Accessed on May 11, 2023.

[19] [n. d.]. If an app would like to connect to devices on your local network. https://developer.android.com/guide/topics/connectivity/wifi-permissions. Accessed on Sept 11, 2023.

[20] [n. d.]. LIFX. https://www.lifx.com. Accessed on May 21, 2023.

[21] [n. d.]. Lucky Time - Win Rewards Every Day. https://play.google.com/store/apps/details?id=com.luckyapp.winner. Accessed on May 26 2023.

[22] [n. d.]. Matter standard. https://csa-iot.org/all-solutions/matter/. Accessed on May 11, 2023.

[23] [n. d.]. MonIoTr Lab. https://moniotrlab.khoury.northeastern.edu.

[24] [n. d.]. Multi-room Music and Alexa Home Theater. https://www.amazon.com/alexa-multi-room-audio/b?ie=UTF8&node=21480962011. Accessed on May 11, 2023.

[25] [n. d.]. Nmap: The network mapper. https://nmap.org/. Accessed on May 21, 2023.

[26] [n. d.]. OWASP Mobile Application Security. https://mas.owasp.org.

[27] [n. d.]. TinyTuya. https://pypi.org/project/tinytuya/. Accessed on May 21, 2023.

[28] [n. d.]. TP-Link WiFi SmartPlug Client and Wireshark Dissector. https://github.com/softScheck/tplink-smartplug. Accessed on May 11, 2023.

[29] [n. d.]. WIGLE. https://www.wigle.net/. Accessed on May 21, 2023.

[30] [n. d.]. The world's most popular network protocol analyzer. https://www.wireshark.org. Accessed on May 21, 2023.

[31] Razaghpanah Abbas, Nithyanand Rishab, Vallina-Rodriguez Narseo, Sundaresan Srikanth, Allman Mark, Kreibich Christian, and Gill Phillipa. 2018. Apps, Trackers, Privacy, and Regulators A Global Study of the Mobile Tracking Ecosystem. *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

[32] Shivaun Albright, Paul J. Leach, Ye Gu, Yaron Y. Goland, and Ting Cai. 1999. *Simple Service Discovery Protocol/1.0.* Internet-Draft draft-cai-ssdp-v1-03. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-cai-ssdp-v1/03/ Work in Progress.

[33] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2016. AndroZoo: Collecting Millions of Android Apps for the Research Community. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. 468–471.

[34] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. Sok: Security evaluation of home-based IoT deployments. In *2019 IEEE symposium on security and privacy (sp)*. IEEE, 1362–1380.

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

[36] Apple, Inc. [n. d.]. com.apple.developer.networking.multicast. Developer Documentation. https://developer.apple.com/documentation/bundleresources/information_property_list/nslocalnetworkusagedescription.

[37] Apple, Inc. 2021. Get the name of the Wi-Fi network to which the device is currently associated. Developer Forums. https://developer.apple.com/forums/thread/679038.

[38] Apple, Inc. Accessed on Sept 21, 2023. com.apple.developer.networking.multicast. Developer Documentation. https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_developer_networking_multicast.

[39] Noah Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. 2019. Keeping the Smart Home Private with Smart(er) IoT Traffic Shaping. *Proceedings on Privacy Enhancing Technologies (PoPETs)* (2019).

[40] Leonardo Babun, Z. Berkay Celik, Patrick D. McDaniel, and A. Selcuk Uluagac. 2021. Real-time Analysis of Privacy-(un)aware IoT Applications. *Proceedings on Privacy Enhancing Technologies (PoPETs)*.

[41] Justin Brookman, Phoebe Rouge, Aaron Alva, and Christina Yeung. 2017. Cross-Device Tracking: Measurement and Disclosures. *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2017, 2 (2017), 133–148.

[42] Jiongyi Chen, Wenrui Diao, Qingchuan Zhao, Chaoshun Zuo, Zhiqiang Lin, Xiaofeng Wang, W. Lau, Menghan Sun, Ronghai Yang, and Kehuan Zhang. 2018. IoTFuzzer: Discovering Memory Corruptions in IoT Through App-based Fuzzing. In *Network and Distributed System Security Symposium*.

[43] Stuart Cheshire and Marc Krochmal. 2013. Multicast DNS. RFC 6762. https://doi.org/10.17487/RFC6762

[44] CNN. [n. d.]. CNN Breaking US & World News. https://play.google.com/store/apps/details?id=com.cnn.mobile.android.phone. Accessed on May 11, 2023.

[45] Michelle Cotton, Leo Vegoda, Ron Bonica, and Brian Haberman. 2013. Special-Purpose IP Address Registries. RFC 6890. https://doi.org/10.17487/RFC6890

[46] Mathieu Cunche, Mohamed Ali Kaafar, and Roksana Boreli. 2012. I know who you will meet this evening! linking wireless devices using wi-fi probe requests. In *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 1–9.

[47] Luca Deri, Maurizio Martinelli, Tomasz Bujlow, and Alfredo Cardigliano. 2014. nDPI: Open-source high-speed deep packet inspection. In *IWCMC*.

[48] Android Developers. [n. d.]. UI/Application Exerciser Monkey. https://developer.android.com/studio/test/other-testing-tools/monkey. Accessed on May 11, 2023.

[49] Ralph Droms. 1997. Dynamic Host Configuration Protocol. RFC 2131. https://doi.org/10.17487/RFC2131

[50] Jide S Edu, Xavier Ferrer-Aran, Jose M Such, and Guillermo Suarez-Tangi. 2021. SkillVet: Automated Traceability Analysis of Amazon Alexa Skills. http://arxiv.org/abs/2103.02637

[51] EFF. [n. d.]. Cover your Tracks. https://coveryourtracks.eff.org/. Accessed on May 22, 2023.

[52] Stephen Farrell, Farzaneh Badiei, Bruce Schneier, and Steven M. Bellovin. 2023. *Reflections on Ten Years Past The Snowden Revelations.* Internet-Draft draft-farrell-tenyearsafter-00. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-farrell-tenyearsafter/00/ Work in Progress.

[53] A Feal, Julien Gamba, Juan Tapiador, Primal Wijesekera, Joel Reardon, Serge Egelman, and Narseo Vallina-Rodriguez. 2021. Don't accept candy from strangers: An analysis of third-party mobile sdks. *Data Protection and Privacy, Volume 13: Data Protection and Artificial Intelligence* 13 (2021), 1.

[54] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Securisty analysis of emerging smart home application. In *IEEE Symposium on Security and Privacy (S&P)*.

[55] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security analysis of emerging smart home application.

[56] Earlence Fernandes, Justin Paupore, Amir Rahmati, Daniel Simionato, Mauro Conti, and Atul Prakash. 2016. FlowFence: Practical Data Protection for Emerging IoT Application Frameworks. In *25th USENIX Security Symposium (USENIX Security 16)*.

[57] Earlence Fernandes, Amir Rahmati, Jaeyeon Jung, and Atul Prakash. 2017. Security Implications of Permission Models in Smart-Home Application Frameworks. In *IEEE Symposium on Security and Privacy (S&P)*.

[58] Dimitris Geneiatakis, Ioannis Kounelis, Ricardo Neisse, Igor Nai-Fovino, Gary Steri, and Gianmarco Baldini. 2017. Security and privacy issues for an IoT based smart home. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*.

[59] Google. 2005. Using a test query to determine whether a network device suffers from a software bug or design flaw. https://patents.google.com/patent/EP1867141B1/en. Accessed on Sept 11, 2023.

[60] Google. 2006. Best practices for unique identifiers. https://developer.android.com/training/articles/user-data-ids.

[61] Google. 2022. NsdManager. Android Developers. https://developer.android.com/reference/android/net/nsd/NsdManager.

[62] Google. 2022. Runtime Permissions. Android Developers. https://source.android.com/docs/core/permissions/runtime_perms.

[63] Google. 2023. WifiInfo. Android Developers. https://developer.android.com/reference/android/net/wifi/WifiInfo.

[64] Google. [n. d.]. Matter. Comissionable and Operational Discovery. https://developers.home.google.com/matter/primer/commissionable-and-operational-discovery. Accessed on May 26, 2023.

[65] Zhixiu Guo, Zijin Lin, Pan Li, and Kai Chen. 2020. SkillExplorer: Understanding the Behavior of Skills in Large Scale. In *Proceedings of the USENIX Security Symposium*.

[66] Muhammad A. Hakim, Hidayet Aksu, A. Selcuk Uluagac, and Kemal Akkaya. 2018. U-PoT: A Honeypot Framework for UPnP-Based IoT Devices. In *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*.

[67] Andrew Halterman. 2019. Storming the Kasa? Security analysis of TP-Link Kasa smart home devices. (2019).

[68] Tianrui Hu, Daniel J. Dubois, and David Choffnes. 2023. BehavIoT: Measuring Smart Home IoT Behavior Using Network-Inferred Behavior Models. In *Proceedings of the Internet Measurement Conference (IMC)*.

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

[70] Philips Hue. [n. d.]. How to develop for Hue? https://developers.meethue.com/develop/get-started-2/. Accessed on Sept 11, 2023.

[71] Etrality Internet Speed Test. [n. d.]. Simple Speedcheck. https://play.google.com/store/apps/details?id=org.speedspot.speedspotspeedtest. Accessed on May 11, 2023.

[72] IoTivity. [n. d.]. IoTivity. http://iotivity.org. Accessed on May 26, 2023.

[73] ioXt. [n. d.]. The ioXt Security Pledge. https://www.ioxtalliance.org/the-pledge. Accessed on May 11, 2023.

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

[75] Golam Kayas, Mahmud Hossain, Jamie Payton, and S. M. Riazul Islam. 2020. An Overview of UPnP-based IoT Security: Threats, Vulnerabilities, and Prospective Solutions. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*.

[76] Aleksandra Korolova and Vinod Sharma. 2018. Cross-app tracking via nearby bluetooth low energy devices. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. 43–52.

[77] Deepak Kumar, Kelly Shen, Benton Case, Deepali Garg, Galina Alperovich, Dmitry Kuznetsov, Rajarshi Gupta, and Zakir Durumeric. 2019. All Things Considered: An Analysis of IoT Devices on Home Networks.. In *USENIX Security Symposium*. 1169–1185.

[78] Bastian Könings, Christoph Bachmaier, Florian Schaub, and Michael Weber. 2013. Device Names in the Wild: Investigating Privacy Risks of Zero Configuration Networking. In *2013 IEEE 14th International Conference on Mobile Data Management*.

[79] Paul J. Leach, Rich Salz, and Michael H. Mealling. 2005. A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122. https://doi.org/10.17487/RFC4122

[80] Anna Maria Mandalari, Daniel J Dubois, Roman Kolcun, Muhammad Talha Paracha, Hamed Haddadi, and David Choffnes. 2021. Blocking without breaking: Identification and mitigation of non-essential iot traffic. *Proceedings on Privacy Enhancing Technologies* 2021, 4 (2021), 369–388.

[81] Anna Maria Mandalari, Hamed Haddadi, Daniel J. Dubois, and David Choffnes. 2023. Protected or Porous: A Comparative Analysis of Threat Detection Capability of IoT Safeguards. In *Proc. of the 44th IEEE Symposium on Security and Privacy (Oakland 2023)*.

[82] Iljitsch van Beijnum Marcelo Bagnulo, Philip Matthews. 2011. Stateful NAT64: Network address and protocol translation from IPv6 clients to IPv4 servers. https://www.rfc-editor.org/rfc/rfc6146.

[83] Joseph Menn. 2022. Mysterious company with government ties plays key internet role. The Washington Post. https://www.washingtonpost.com/technology/2022/11/08/trustcor-internet-addresses-government-connections/.

[84] Lionel Metongnon and Ramin Sadre. 2018. Beyond Telnet: Prevalence of IoT Protocols in Telescope and Honeypot Measurements *(WTMC '18)*.

[85] MITRE. 2016. CVE-2016-2183 Detail - NVD. https://nvd.nist.gov/vuln/detail/CVE-2016-2183. Accessed on May 11, 2023.

[86] MITRE. 2020. CVE-2020-0454. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-0454. Accessed on May 11, 2023.

[87] MITRE. 2020. CVE-2020-11022 Detail - NVD. https://nvd.nist.gov/vuln/detail/CVE-2020-11022. Accessed on May 11, 2023.

[88] MITRE. 2021. CVE-2021-11023 Detail - NVD. https://nvd.nist.gov/vuln/detail/CVE-2020-11023. Accessed on May 11, 2023.

[89] Hooman Mohajeri Moghaddam, Gunes Acar, Ben Burgess, Arunesh Mathur, Danny Yuxing Huang, Nick Feamster, Edward W. Felten, Prateek Mittal, and Arvind Narayanan. 2019. Watching You Watch: The Tracking Ecosystem of Over-the-Top TV Streaming Devices. In *Conference on Computer and Communications Security (CCS)*.

[90] Yuhong Nan, Luyi Xing Xueqiang Wang, Ruoyu Wu Xiaojing Liao, Yifan Zhang Jianliang Wu, and XiaoFeng Wang. 2023. Are You Spying on Me? Large-Scale Analysis on IoT Data Exposure through Companion Apps. In *USENIX Security*.

[91] Dr. Thomas Narten, Tatsuya Jinmei, and Dr. Susan Thomson. 2007. IPv6 Stateless Address Autoconfiguration. RFC 4862. https://doi.org/10.17487/RFC4862

[92] Nessus. 2003. SheerDNS < 1.0.1 Multiple Vulnerabilities. https://www.tenable.com/plugins/nessus/11535. Accessed on May 26, 2023.

[93] Nessus. 2004. DNS Server Cache Snooping Remote Information Disclosure. https://www.tenable.com/plugins/nessus/12217. Accessed on May 26, 2023.

[94] Nessus. 2005. Exposure Management Meets Tenable Security Center. https://www.tenable.com. Accessed on May 26, 2023.

[95] TJ OConnor, Dylan Jessee, and Daniel Campos. 2021. Through the Spyglass: Towards IoT Companion App Man-in-the-Middle Attacks. In *Cyber Security Experimentation and Test Workshop (CSET)*.

[96] National Institute of Standards and Technology (NIST). 2021. DRAFT Baseline Security Criteria for Consumer IoT Devices. https://www.nist.gov/system/files/documents/2021/08/31/IoT%20White%20Paper%20-%20Final%202021-08-31.pdf. Accessed on May 11, 2023.

[97] O'Flynn. 2016. A LIGHTBULB WORM? https://www.blackhat.com/docs/us-16/materials/us-16-OFlynn-A-Lightbulb-Worm-wp.pdf. Accessed on May 11, 2023.

[98] ONVIF. [n. d.]. Home - ONVIF. https://www.onvif.org/. Accessed on May 11, 2023.

[99] Zoltán Pallagi. [n. d.]. Network Scanner, Device Finder. https://play.google.com/store/apps/details?id=com.pzolee.networkscanner&hl=en&gl=US. Accessed on May 11, 2023.

[100] Muhammad Talha Paracha, Daniel J. Dubois, Narseo Vallina-Rodriguez, and David R. Choffnes. 2021. IoTLS: Understanding TLS Usage in Consumer IoT Devices. In *Proceedings of the Internet Measurement Conference (IMC)*.

[101] Vesa Pehkonen and Juha Koivisto. 2010. Secure Universal Plug and Play network. (2010). https://doi.org/10.1109/ISIAS.2010.5604189

[102] Joel Reardon. 2022. The Curious Case of Coulus Coelib. The AppCensus Blog. https://blog.appcensus.io/2022/04/06/the-curious-case-of-coulus-coelib/.

[103] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge Egelman. 2019. 50 Ways to Leak Your Data: An Exploration of Apps' Circumvention of the Android Permissions System. *Proceedings of the USENIX Security Symposium*.

[104] Nilo Redini, Andrea Continella, Dipanjan Das, Giulio De Pasquale, Noah Spahn, Aravind Machiry, Antonio Bianchi, Christopher Kruegel, and Giovanni Vigna. 2021. DIANE: Identifying Fuzzing Triggers in Apps to Generate Under-constrained Inputs for IoT Devices. In *IEEE Symposium on Security and Privacy (S&P)*.

[105] 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 *Proceedings of the Internet Measurement Conference (IMC)*. 267–279.

[106] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, Serge Egelman, et al. 2018. "Won't somebody think of the children?" examining COPPA compliance at scale. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*.

[107] First Row. [n. d.]. Network Scanner. https://play.google.com/store/apps/details?id=com.myprog.netscan&hl=en&gl=US. Accessed on May 11, 2023.

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

[109] Henning Schulzrinne, Stephen L. Casner, Ron Frederick, and Van Jacobson. 2003. RTP: A Transport Protocol for Real-Time Applications. RFC 3550. https://doi.org/10.17487/RFC3550

[110] Zach Shelby, Klaus Hartke, and Carsten Bormann. 2014. The Constrained Application Protocol (CoAP). RFC 7252. https://doi.org/10.17487/RFC7252

[111] W. A. Simpson, Dr. Thomas Narten, Erik Nordmark, and Hesham Soliman. 2007. Neighbor Discovery for IP version 6 (IPv6). https://datatracker.ietf.org/doc/rfc4861/. Accessed on May 11, 2023.

[112] Spotify. [n. d.]. ZeroConf API. https://developer.spotify.com/documentation/commercial-hardware/implementation/guides/zeroconf. Accessed on May 21, 2023.

[113] Statista. 2022. Smart Home - United States: Statista Market Forecast. https://www.statista.com/outlook/dmo/smart-home/united-states. Accessed on May

11, 2023.
[114] Milan Stute, David Kreitschmann, and Matthias Hollick. 2018. One billion apples' secret sauce: Recipe for the apple wireless direct link Ad hoc protocol. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 529–543.
[115] Rahmadi Trimananda, Janus Varmarken, Athina Markopoulou, and Brian Demsky. 2020. Packet-level signatures for smart home devices. In *Network and Distributed Systems Security (NDSS) Symposium*, Vol. 2020.
[116] Janus Varmarken, Hieu Le, Anastasia Shuba, Athina Markopoulou, and Zubair Shafiq. 2020. The TV is Smart and Full of Trackers: Measuring Smart TV Advertising and Tracking. *Proceedings on Privacy Enhancing Technologies (PoPETs)*.
[117] Wall Street Journal. 2022. Google Bans Apps With Hidden Data-Harvesting Software. https://www.wsj.com/articles/apps-with-hidden-data-harvesting-software-are-banned-by-google-11649261181. Accessed on May 11, 2023.
[118] Xueqiang Wang, Yuqiong Sum, Susanta Nada, and XiaoFeng Wang. 2019. Looking from the Mirror: Evaluating IoT Device Security through Mobile Companion Apps. In *Proceedings of the USENIX Security Symposium*.
[119] Wei Zhou, Yan Jia, Yao Yao, Lipeng Zhu, Le Guan, Yuhang Mao, Peng Liu, and Yuqing Zhang. 2019. Discovering and Understanding the Security Hazards in the Interactions between IoT Devices, Mobile Apps, and Clouds on Smart Home Platforms. In *Proceedings of the USENIX Security Symposium*.
[120] Wei Zhou, Yan Jia, Yao Yao, Lipeng Zhu, Le Guan, Yuhang Mao, Peng Liu, and Yuqing Zhang. 2019. Discovering and Understanding the Security Hazards in the Interactions between IoT Devices, Mobile Apps, and Clouds on Smart Home Platforms. In *28th USENIX Security Symposium (USENIX Security 19)*.

A DISCLOSURE PROCESS

Our work identified numerous security and privacy issues affecting software running on mobile and IoT devices. In this section, we provide details about our vulnerability disclosure process, principles followed, vendor interactions, and status of remediations. We note that interactions with responsible parties are ongoing at the time of writing, so we expect further remediations to occur in the future.

Our approach was to follow responsible disclosure principles by privately informing responsible parties through their vulnerability disclosure programs or customer contacts before publication. There is no standard for time given to address these issues; we gave vendors 30 days notice given timing constraints for publication. While all parties involved would have liked a longer embargo period, this was unfortunately not possible given available time and resources.

We sent disclosures to 19 IoT vendors and manufacturers, as well as reported to Google a list of Android mobile apps and SDKs that we discovered collecting sensitive local network information. At the time of writing, we have received responses from 11 of these parties, and are actively collaborating with them to address the identified vulnerabilities. In addition, we are in the process of disclosing the potential privacy violations found in this study to regulators in relevant jurisdictions.

B IOT DEVICES

Table 3 lists the IoT devices that comprise our MonIoTr Lab IoT testbed.

C TRAFFIC PROCESSING

C.1 Local Traffic Filter

If the local IP range is 192.168.10.0/24, the rule in WireShark/tshark (version >= 3.7.0) consists of the following conditions:

```
(ip.dst===192.168.10.0/24
and ip.src===192.168.10.0/24)
or (eth.dst.ig==1)
or (eth.dst.ig==0&&!ip)
```

- Local IP unicast traffic (ip.dst===192.168.10.0/24 and ip.src===192.168.10.0/24): Both source and destination IP addresses belong to the 192.168.10.0/24 subnet.
- Multicast/broadcast traffic (eth.dst.ig==1): The destination Ethernet address is a multicast or broadcast address .
- Non-IP unicast traffic (eth.dst.ig==0&&!ip): The traffic is unicast but does not have an IP layer.

C.2 Protocol Classification

In order to identify the most accurate traffic classification method, we perform a cross-comparison of the outputs reported by *tshark* and *nDPI*. We apply these tools to 366K local network packets and flows captured in the IoT lab without any user interaction. We follow RFC 6146 [82] to define UDP and TCP flows, i.e., a chronologically ordered set of TCP segments/UDP datagrams with the same 5-tuple combination (source IP, source port, destination IP, destination port, transport protocol).

We found that *tshark* reported 35 labels for 76% of flows, whereas *nDPI* provided 18 labels for 74% of them. The tools provided different labels for 16% of the connections, and neither reported a label for 7.5% of the connections (which mostly corresponded to layer 3 traffic). Figure 3 shows the labeling inconsistencies between *nDPI* and *tshark*.

We manually examined the flows in which they disagree, concluding that they belonged to 19 different application layer protocols. Out of those 16% flows in which they disagreed, 95% of them were misclassified by *tshark* as generic "transport-layer traffic" or TP-Link's custom protocol, while *nDPI* correctly identified most of them as SSDP flows. On the other hand, *nDPI* incorrectly identified a small fraction of SSDP flows as CiscoVPN traffic, and other local traffic generated by the Nintendo Switch (EAPOL layer 2 traffic) as AmazonAWS. Another case is RTP traffic, which is often misclassified by *nDPI* and *tshark* because RTP does not specify a standard port number and its payload is not plaintext. Through a combination of controlled experiments and informed speculation, we created tools to partially identify some RTP traffic used by smart devices. Similarly, we observed Google devices frequently communicating using UDP over ports within the range 10000–10010. This traffic was initially classified as STUN by both *nDPI* and *tshark*. However, experiments similar to those conducted with Echo devices indicated that this is a misclassification of RTP traffic and that is likely used for control and synchronization purposes.

D PROTOCOL AND TRAFFIC ANALYSIS

D.1 Frequency Analysis

The frequency of the discovery traffic can also provide information on the purpose and privacy implication of local communication. To check the periodicity of the traffic, we use an approach[68] that combines Discrete Fourier Transformation (DFT) and autocorrelation. We check periodicity for traffic from each unique (destination, protocol) tuple. We do not consider port here as the randomization of port number is prevalent on IoT devices. We find that 88% of discovery protocol flows are periodic, and we identify a total of 580 different periodic groups (destination, protocol) across our IoT devices, averaging approximately 6.2 groups per device. We leave further in-depth analysis as future work.

Game Console	Generic IoT	Home Appliance	Home Automation	Media/TV	Surveillance	Voice Assistant
Nintendo (1)	Keyco (1)	Anova (1)	Amazon (1)	Amazon (1)	Amcrest (1)	Amazon (17)
	Oxylink (1)	Behmor (1)	Aqara (1)	Apple (1)	Arlo (2)	Apple (3)
	Renpho (1)	Blueair (1)	Google (1)	Google (1)	Blink (1)	Meta (1)
	Tuya (1)	GE (1)	IKEA (1)	LG (1)	D-Link (1)	Google (7)
	Withings (3)	LG (1)	MagicHome (1)	Roku (1)	Google (2)	
		Samsung (3)	Meross (3)	Samsung (1)	ICSee (1)	
		Smarter (1)	Philips (1)	Tivostream (1)	Lefun (1)	
		Xiaomi (1)	Ring (1)		Microseven (1)	
			Sengled (1)		Ring (4)	
			SmartThings (1)		Tuya (1)	
			SwitchBot (1)		Ubell (1)	
			TP-Link (2)		Wansview (1)	
			Tuya (3)		Wyze (1)	
			WeMo (1)		Yi (1)	
			Wiz (1)			
			Yeelight (1)			

Table 3: IoT devices under test categorized by device type. The number in the parentheses indicates the number of devices.

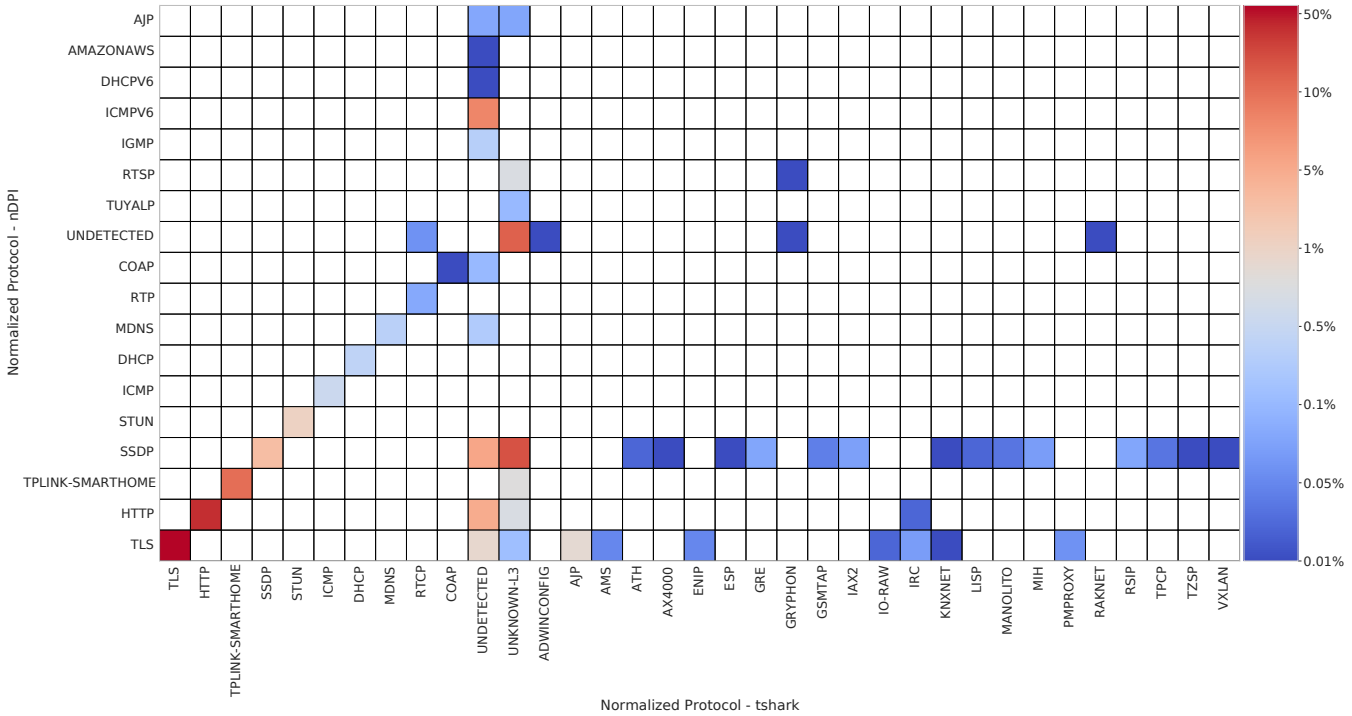
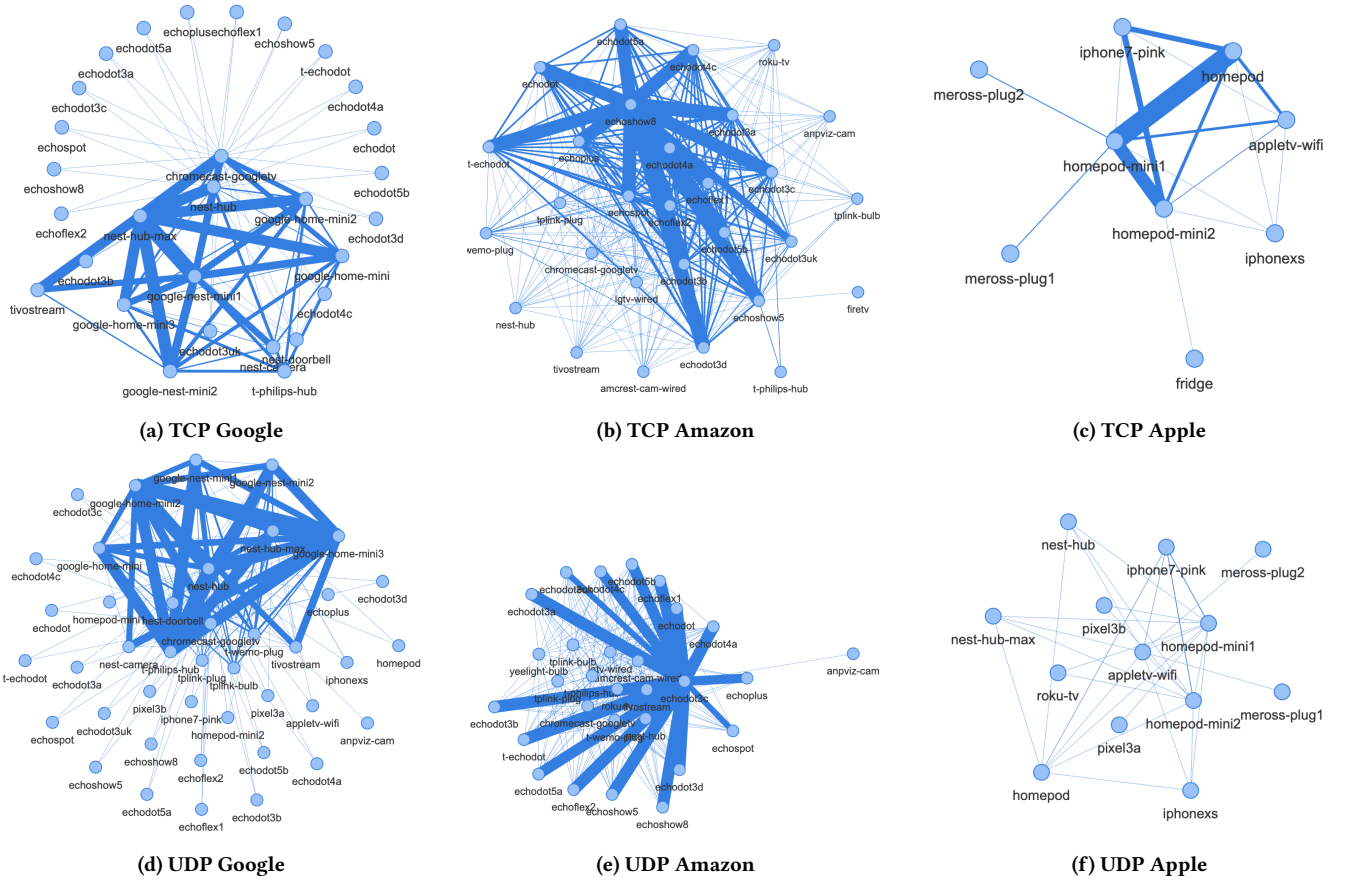


Figure 3: Heatmap of Tshark vs nDPI Normalized Protocols.

D.2 Discovery Protocol Responses

We correlate multicast and broadcast discoveries with their responses by inspecting unicast inbound traffic to the devices that initiate the discoveries. We search for traffic employing the same transport layer protocol and port number within a short time period (empirically set as 3 seconds in this study) following the discovery requests. A response could also be multicast traffic such as QM mDNS, which we plan to explore in future work.

We present the number of these protocols used by each device grouped by device category in Table 4. Excluding ARP, DHCP, and ICMPv4/v6 which are used by most of the devices, we observe that 32 devices receive responses from other devices using these



(a) TCP Google (b) TCP Amazon (c) TCP Apple

(d) UDP Google (e) UDP Amazon (f) UDP Apple

Figure 4: Zoom-in TCP and UDP graphs for the Apple, Amazon and Google clusters.

Device Group	# Discovery Protocols	# Protocols with Response	# Devices Responded to
Amazon Echo	3.65	1.82	9.47
Google&Nest	4.0	3.0	5.14
Apple	1.0	1.0	5.0
Tuya	1.0	0.0	0.0
TVs	1.4	1.0	2.0
Cameras	1.17	1.0	1.5
Hubs	1.5	0.0	0.0
Home Auto	1.0	1.0	1.0
Appliances	2.0	0.0	0.0

Table 4: Number of discovery protocols (exclude ARP, DHCP, ICMP/v6 as they are used by most devices) used grouped by category. Number of protocol with at least one response per category. Number of devices responded to each device grouped by category.

protocols. Amazon Echo devices, on average, receive responses from 9.47 other smart devices as shown in Table 4.

D.3 Vendor-specific Clusters

We present the isolated depictions of vendor-specific device-to-device communication clusters in Figure 4. These figures illustrate extracted clusters for Google's, Amazon's, and Apple's IoT platforms. The thinkness of the edges corresponds to the volume of the traffic.

D.4 Payload examples

Table 5 provides several examples of SSDP, mDNS and NetBIOS payloads exposing device information.

E IDENTIFICATION OF DEVICES FOR IOT INSPECTOR

As explained in the original paper [69], IoT Inspector collects metadata of network traffic (e.g., source/destination IP addresses and ports), some payloads (e.g., DNS), and crowdsourced user labels. The dataset does not explicitly identify vendors, models, products of devices. In this section, we discuss how we *infer* identities of devices, including the vendor and category information, based on this dataset. We say "infer" because we do not have the ground

SSDP	HTTP/1.1 200 OK SERVER: Linux, UPnP/1.0, Private UPnP SDK ... <?xml version="1.0" ?> <friendlyName>AMC020SC43PJ749D66</friendlyName> <serialNumber>9c:8e:cd:0a:33:1b</serialNumber> <UDN>uuid:device_3_0-AMC020SC43PJ749D66</UDN> <serviceList> <service>
mDNS	Ethernet II, Src: PhilipsL_68:5f:61 (00:17:88:68:5f:61), Dst: IPv4mcast_fb (01:00:5e:00:00:fb) ... Multicast Domain Name System (response) Philips Hue - 685F61._hue._tcp.local: type TXT, class IN, cache flush _hue._tcp.local: type PTR, class IN, Philips Hue - 685F61._hue._tcp.local 1.6.F.5.8.6.E.F.F.F.8.8.7.1.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.E.F.ip6.arpa: type PTR
NetBIOS	00 01 00 00 00 00 00 00 20 43 4b 41 41 41 41 41 ..... CKAAAAA 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA 41 41 41 41 41 41 41 41 41 00 00 21 00 01 AAAAAAAAA..!..
TPLINK-SHP	{"system":{"get_sysinfo ... "deviceId":"8006E8E9017F556D283C850B4E29BC1F185334E5", "hwId":"60FF6B258734EA6880E186F8C96DDC61" ... oemId":"FFF22CFF774A0B89F7624BFC6F50D5DE "alias":"TP-Link Plug","dev_name":"Wi-Fi Smart Plug With Energy Monitoring" ... "latitude":42.337681,"longitude":-71.087036
Co-located devices leaking data to the cloud	HTTP/1.1 200 OK {"entity":{"entityId":"SKILL_eyJza2lsbElkIjoiYW16bjEuYXNrLnNraWxsLmI0YmYyYjRkLT -> 8012A5191D2CB6983983DB807412997E18990EFF> –> Light bulb deviceId ","entityType":"CLOUD_DISCOVERED_DEVICE"},"capabilityStates": ["{\"namespace\":\"Alexa.BrightnessController\",\"name\":\"brightness\",\"value\":100,

Table 5: Examples of payload exposing device information such as MAC addresses within the local network and cloud.

truth due to the crowdsourced nature; we can only validate our findings across different internal data sources and/or through manual inspection.

Overview. We obtained a subset of IoT network traffic from IoT Inspector's authors. For each device, we make sure that at least two pieces of the following metadata are available: OUI (the first three bytes of a MAC address), DHCP hostname, mDNS/SSDP responses, hostnames contacted, and the user labels. The user labels are what IoT Inspector users have optionally named their IoT devices in terms of the vendor, product name, and product type; these labels can be selected from a pre-filled drop-down list in the IoT Inspector UI, or can be free-form text [69]. The entire IoT Inspector dataset includes 216,824 devices, of which 25,033 devices have at least two pieces of the metadata above. In the next few steps, we will infer the identities, including the vendor and categories, for these 25,033 devices.

Inferences with ChatGPT. Using OpenAI's TextCompletion API,[5], we develop prompt to infer device vendors and categories based on a device's DHCP hostname, mDNS/SSDP responses, and user labels. We use this API because it is trained on Internet-scale data, which likely includes public knowledge on various IoT devices. Also, we pick these three pieces of metadata because, based on our manual sampling, they are likely to contain identifying information (albeit imperfect), explicitly (i.e., substring) or implicitly. For example, user labels are crowdsourced and sometimes include incorrect spellings [69]; mDNS/SSDP responses often include the vendor and product information, although the exact formats could differ across vendors; and DHCP hostnames may be indicative of the product identity (e.g., the string "cast" often appearing in the DHCP hostnames of Google Chromecast devices). We treat all these metadata as unstructured natural languages—especially given the diversity of IoT devices—and feed them into the TextCompletion API.

By iteratively testing different prompts on a small subset of known devices, we develop the following prompts. To infer the vendor names, we ask: "I have an IoT device named '`[metadata]`'. What is the company that makes this IoT device? Output the companyś name only." For device type, we use this prompt instead:

[5]https://platform.openai.com/docs/guides/completion

"I have an IoT device named '`[metadata]`'. What type of IoT device is this? Output the name of the device type only." We replace `[metadata]` with user labels, DHCP hostnames, or mDNS/SSDP responses, separately, extracted from the IoT Inspector dataset. We apply these prompts to the 25,033 devices with the TextCompletion API. After removing empty or unknown responses, we have the API responses for 24,998 of the devices. At the time of writing, the API cost was approximately $70 USD in total.

F GENERATIVE AI ACKNOWLEDGMENTS

As explained in Section E, we use ChatGPT (OpenAI's TextCompletion API) to infer device identities in IoT Inspector data. ChatGPT was also utilized to automatically generate LaTeX format tables.