Can Allowlists Capture the Variability of Home IoT Device Network Behavior?

Weijia He
Dartmouth College
weijia.he@dartmouth.edu

Kevin Bryson
University of Chicago
kbryson@uchicago.edu

Ricardo Calderon
University of Chicago
ricardocalderon@uchicago.edu

Vijay Prakash
New York University
vp2179@nyu.edu

Nick Feamster
University of Chicago
feamster@uchicago.edu

Danny Yuxing Huang
New York University
dhuang@nyu.edu

Blase Ur
University of Chicago
blase@uchicago.edu

***Abstract*—Home Internet of Things (IoT) devices can be difficult for users to secure. Prior work has suggested measuring these devices' network behaviors and using these characterizations to create allowlists of permitted endpoints. Unfortunately, previous studies have typically been conducted in controlled lab settings, with one or two devices per product. In this paper, we examine whether popular home IoT products' network behaviors generalize via both in-lab experiments of 24 devices and a large, crowdsourced dataset of IoT devices in the wild. We find that observing traffic from one device in one lab is often insufficient to fully characterize an IoT product's network behaviors. For example, specifying which endpoints a device may contact based on initial measurements in our lab led 25% of products to stop functioning later, and even more when using a VPN. We then used the crowdsourced dataset to better understand this traffic's heterogeneity and pinpoint how to create more generalizable allowlists. We identified causes of failure, such as regionalization, CDN usage, third-party integrations, and API changes. Finally, we used the crowdsourced data in numerous configurations to specify which endpoints each product in our lab could contact. We found that domain-level allowlists enabled the majority of devices to function in our lab using data collected years in the past. For the remaining devices, we characterize how to mitigate the failures observed and pave the way to creating more generalizable allowlists.**

1. Introduction

Many homes now contain Internet of Things (**IoT**) devices, such as smart light bulbs, cameras, and thermostats [30]. Unfortunately, home IoT devices have suffered from numerous security issues [1]. Attackers have exploited vulnerabilities in devices' software, protocols, and default settings [6], creating botnets like Mirai [3] and Hajime [16]. The range of vendors creating IoT devices, the difficulty of deploying patches to devices without screens or traditional user interfaces, and a lack of standardization contribute to these security challenges [34].

Rather than relying only on potentially unresponsive vendors to patch devices, households could control the network traffic of their own IoT devices. The Manufacturer Usage Description (**MUD**), standardized in an IETF RFC, has been proposed to help combat security threats to home IoT devices [19]. The idea is to specify devices' expected behaviors, particularly expected network traffic destinations (endpoints), in a MUD file. A homeowner could thus use a MUD file to create network policies for their home IoT devices. Any behavior that deviates from those specified would be blocked by default. In this way, MUD files function as an **allowlist**, enumerating the only external endpoints that the home IoT device can contact. In other words, MUD-like allowlists restrict outgoing traffic to a predefined set of endpoints.

Because allowlists have a small attack surface, they are very attractive from a security perspective. For instance, whereas botnets like Mirai [3] were used to perform a distributed denial of service (DDoS) attack using compromised home IoT devices, a MUD-like allowlist would greatly mitigate such DDoS attacks. First of all, only endpoints that are listed on any common device's allowlist would be a potential DDoS victim. Furthermore, the allowlists for different devices would likely have minimal overlap, minimizing the total amount of traffic that could be directed at any single target even if it did appear on some allowlist. A MUD-like allowlist could also make it harder for an attacker to exfiltrate private data from the home, requiring that they establish a sink on an endpoint listed in a home's allowlists. If implemented correctly (i.e., all listed endpoints are legitimate), such allowlists can make stronger security guarantees than alternative approaches like blocklists or anomaly detection.

While allowlists thus have powerful security properties, they have rarely been used in practice and manufacturers to date mostly have not created them. In response, in this paper we essentially investigate the possibility of crowdsourcing them. For general-purpose computing devices, enumerating the destinations that general should be able to contact is typically intractable. However, because many home IoT devices perform a very limited set of actions and thus would presumably contact only a few endpoints, intuition suggests that allowlists may be practical for securing home IoT devices at the network level, leading to the MUD standard.

The critical challenge is that MUD files must *fully specify a product's expected behaviors*. An incomplete MUD file, or incomplete training data for an anomaly detection model, could cause devices to cease functioning correctly, destroying the user experience. Compared to blocklists or anomaly detection, completeness is essential

for allowlist functionality; otherwise users might quickly abandon allowlists, harming security. Because manufacturers have been slow to release MUD files for their own devices, leaving open the question of how MUD files should be populated. Specifically, would simple observation of these devices (e.g., in a lab) suffice?

Measurement studies are a potential solution for identifying home IoT devices' expected network behaviors [25], [27]. By observing and recording different home IoT products' traffic in the lab, one may be able to learn the endpoints the device contacts and use those to create the MUD file, as proposed in prior work [14], [15]. The disadvantage of this approach is that it assumes an observation made from one or two devices in the lab (the approach taken in prior work) generalizes to the behavior of the *same* product in deployment, regardless of various factors, ranging from the network environment to geography. As we show in this paper, measurements made in the lab may fail to capture the variability of home IoT products' network behaviors across environments.

Toward this goal, **we measure the variability of 24 home IoT devices' network traffic using both a large-scale, crowdsourced dataset and an in-lab testbed, providing a more general picture of home IoT devices' network behaviors than prior work.** Studying a single device in a controlled environment, as in past work, misses key sources of variability across devices, regions, network architectures, and time. Without accounting for such variability, observations of traffic destinations may not transfer across instances of an IoT product, requiring each IoT device owner to take the onerous step of exercising all of a device's functionalities to make informed decisions about what security policies to deploy. Further, if the owner observes unseen behaviors from their devices, there is no way to tell if this is a normal variation or actual threat.

As our first contribution, we performed an in-lab study of 24 popular home IoT products. We show how variability in a product's network behaviors can break device functionality if measurements of a single device bootstrap a MUD. Specifically, we exercised these 24 products' key functions, recording their network traffic in our lab. Using these measurements, we enforced a MUD that only allows the device to contact endpoints observed in the initial measurements. We matched previously observed traffic to future endpoints at three levels of abstraction, ranging from second-level domain to hostname (i.e., fully qualified domain name). We then attempted to exercise the same functionalities. We ran variants of these experiments with the device using the same network in our US-based lab, as well as over a VPN network connected to servers in Germany with proper registration information. The experiments varying the effective geography were inspired by prior work [25] comparing network measurements in a US lab and a UK lab. However, that work did not attempt to restrict future traffic based on these measurements and thus did not measure the resultant MUD file's impact on device functionality.

In the experiments using network traffic measured in the US to enforce a simulated MUD for that same device connected over a VPN to a German network, 11 of the 24 devices stopped working. Some of these issues were caused by the use of regional identifiers (e.g., `avs-alexa-6-na.amazon.com` vs. `avs-alexa-7-eu.amazon.com`). Even when both the initial measurement and subsequent experiment were conducted in our lab, 6 of the 24 devices stopped working due to variations in the specific endpoints contacted.

As our second contribution, we measured sources of variability in the network behaviors of distinct devices of the same make and model ("product") more broadly using a large, crowdsourced dataset. Specifically, we searched for the same 24 popular home IoT products in the IoT Inspector [17] dataset to compare and contrast a given product's network behaviors across households. We focused on the hostnames and ports with which each device communicates. We expected to see the variations observed in our lab study, as well as variations based on network configurations (e.g., DNS and NTP). We indeed observed these variations. To our surprise, we also observed more substantial variation. Specifically, to observe 95% of the hostnames contacted by all devices in our dataset of a given product required a sample of over 60% of the devices of that product. Despite the variability that we observed, many hostnames in this "long tail" were related to each other. For example, some hostnames had small variations (e.g., `oculus2975-us1.dropcam.com` vs. `oculus1802-us1.dropcam.com`), presumably to support load balancing. Thus, observations made from one device are unlikely to transfer to other devices of that same product without modification.

As our third contribution, we characterized the degree to which a device's network behaviors can be transferred to other devices of the same product based on this crowdsourced data. Transferability refers to how much network traffic observed from one device may apply to another. High transferability is necessary for creating shareable MUD files. As part of this analysis, we compared different representations of endpoints. Some of the 24 products contacted a huge number of different hostnames that is difficult to capture even in a large dataset. For these products, only second-level domains are shared among instances of the device. To make potential MUDs more robust in the presence of a few mislabeled or compromised devices, we also explored requiring that a host be contacted by multiple instances of a product for it to be considered. We found that frequently one or two dozen instances are needed to capture this variability. In short, this analysis of the IoT Inspector dataset enabled us to identify parameters (representation, sample size, threshold, regions, lifespan) needed to create allowlists and estimate their performance.

Finally, we measured whether the same 24 products continued to function normally in our lab when we enforced MUD-like allowlists created from the crowdsourced dataset that was collected years prior. In other words, we use the (years-old) IoT Inspector dataset to create allowlists we actually tested in our lab to verify their impact on device functionality. It was common for the products in our lab to contact different hostnames (fully qualified domain names) than those seen in the crowdsourced data, causing many devices to stop working when using our most restrictive representation of endpoints for our MUD-like allowlists. However, when we represented endpoints using the second-level domain, 17 of the 24 products continued to function normally, 5 were mostly functional, and only 2 stopped functioning entirely.

While rare actions (e.g., reinitialization) likely had not been captured in IoT Inspector, they usually shared the same endpoints with other functionalities collected in the dataset. This result suggests that domain-level characterizations may retain utility for years.

Among products that lost partial or full functionality, apparent API changes (e.g., completely changing the domain the device typically contacts) or new features were most often to blame. We highlight how relatively straightforward updates to the MUD-like allowlists would have restored functionality to many of these products, highlighting that domain-level allowlists are mostly feasible for home IoT devices. The key exception was media-streaming devices, whose endpoints vary so significantly across users that allowlists are unlikely to be feasible. At the same time, we also highlight how many products rely on public cloud services (e.g., AWS), minimizing some of the security guarantees allowlists provide.

2. Problem Formulation

We aim to measure and characterize the variability and transferability of home IoT devices' network traffic. More specifically, we want to understand whether the observations we made through one set of devices (e.g., from a public dataset or in a lab) can be transferred to another set of devices (e.g., the devices one owns).

2.1. Definition

A home IoT device can be viewed in terms of different levels of abstractions. For example, a Wyze camera can be seen as a physical device or a general product. To clarify our discussion, we use the following terms:

- **Vendor** refers to a company (e.g., Amazon) that makes the home IoT product.
- **Product** refers to the collection of devices sold under a specific make and model name (e.g., Amazon Ring), potentially encompassing multiple versions.
- **Device** refers to a single physical instance of a home IoT product (e.g., a single Amazon Ring in a home).
- **Type** refers to the category of multiple products with similar purposes (e.g., both the Amazon Ring and Wyze Camera are a "home IoT camera").

2.2. Variables in Measurement

Suppose a user has several home IoT devices connected to their home network, over which they have total control, including inspecting, allowing, or blocking network traffic. They may want to better understand if their home IoT devices are behaving normally or are contacting servers they are not supposed to contact. Because vendors normally do not release a list of endpoints a product should contact, the user may want a public specification (e.g., MUD files based on home IoT datasets) to learn how other home IoT devices behave, comparing them to their own devices. Users face several challenges transferring knowledge from a public dataset to their own devices:

Representativeness: Prior work found that home IoT devices contact hosts around the world [25], [27]. However, it remains unclear how regional differences impact device functionality. Mandalari et al. showed that many endpoints home IoT devices contact do not relate to device functionality [21]. If regional differences in traffic affect device functionality, an unrepresentative dataset may cause the user to think essential traffic is illegitimate.

Host representations: Although one could assume different device instances of the same product generally behave similarly, it remains unclear at what level of abstraction these similarities generalize. For example, some products may contact a small number of endpoints, which means the fully qualified domain name (**FQDN**) may be shared across all device instances. Other products with a more complicated backend infrastructure may only share second-level domains among devices. Products could also use the same set of IP addresses across all devices.

Dataset reliability: Crowdsourced home IoT datasets can sometimes contain "dirty" data, such as traffic from a compromised device. Thus, verifying a product's network behaviors through multiple device instances is important. More devices contacting a particular host may suggest that host is a legitimate endpoint for the product. We set *thresholds* for considering an observed endpoint valid. Specifically, for a threshold of n, a user will only consider a host valid if the host appears in the traffic of at least n devices in the sample.

Sample size: When using a public dataset to study device behavior, it is important to use a dataset large enough to account for potential discrepancies. If the sample size is too small, one may miss key variations, leading a user to falsely believe their device is compromised. In contrast, larger datasets can be costly to collect and maintain.

Stability over time: A dataset is a snapshot in time, showcasing how home IoT devices behaved in the past. A device's legitimate endpoints may have changed since data collection, raising questions about a dataset's lifetime.

2.3. Background on Allowlists

A firewall is one of the most popular network security systems. It governs the incoming and outgoing traffic between a protected network (e.g., a home network) and an unprotected one (e.g., the Internet). In homes, firewalls are often built into consumer network routers. When people connect their IoT devices to their home network, these IoT devices will then naturally be protected by this firewall.

There are two main security strategies for firewalls: allowlists and blocklists. Allowlists specify which network packets are allowed to cross the firewall based on the network addresses or many other characteristics. Packets that do not match the specification are dropped. In contrast, blocklists only drop packets matching the specification, allowing everything else. Both allowlists and blocklists have pros and cons. Blocklists, such as ad blockers, are more frequently used because it is hard, if not impossible in some computing contexts, to enumerate all potential network hosts that a device may visit legitimately, especially when the device is a general-purpose device like a smartphone. Allowlists are far more restrictive, only allowing traffic from previously enumerated sources. With an allowlist deployed, a remote attacker typically cannot establish a connection with the IoT device as they would

typically be sending packets from a network host not included in the allowlist, resulting in their traffic being dropped at the home's gateway. Even if the victim's IoT device becomes compromised, a firewall allowlist could prevent the compromised device from sending data to the attacker because the attacker's endpoint hopefully would not be included in the deployed allowlist.

2.4. Threat Model

We imagine there is a user who has several home IoT devices connected to their home network, over which they have total control (e.g., for deploying an allowlist). In our threat model, a remote attacker aims to compromise these IoT devices, forcing them to contact arbitrary, remote endpoints outside the home network either to exfiltrate data (e.g., send private data about activities in the home to an endpoint the attacker controls) or to disrupt endpoints unrelated to the IoT device (e.g., as part of a broad DDoS attack). We assume the attacker does not have the ability to compromise the vendor's backend infrastructure (e.g., `meethue.com`), nor poison the victim's DNS. Local attacks (e.g., on WiFi or ZigBee) are out of scope.

Furthermore, we assume that most devices whose network data is being contributed to the crowdsourced dataset have not been compromised as of the time of data collection. We also assume that they are correctly labeled as a particular product. Employing a higher threshold ("dataset reliability" in Section 2.2) can mitigate this problem, but only for a limited number of compromised or mislabeled devices. For this reason, Sybil attacks in which a single attacker contributes data purporting to come from different households would be problematic. Furthermore, our approach would be vulnerable to supply chain attacks, which could change the behavior of many devices of some product type and which we consider out of scope.

3. Testbed

In this paper, we use the IoT Inspector dataset for crowdsourced measurements of home IoT device behavior in the wild; see Section 5.1 for details on this dataset. For clarity in presenting analyses and to make the cost of acquiring devices feasible, we selected 24 of the most frequent products in the dataset. All analyses, both on IoT Inspector and in our lab, use the following 24 devices: Amazon Echo Dot; Amazon Fire Stick; Amazon Ring; Belkin Wemo Plug; Chamberlain Garage; DLink Camera; Ecobee Thermostat; Google Chromecast; Google Home; Google Nest Thermostat; Honeywell Thermostat; iDevice Switch; iHome Switch; Lifx Light; Logitech Harmony; Lutron Bridge; Nintendo Switch; Philips Hue; Roku Streamer; Sonos One; Sony Console; TP-Link Plug; Wyze Camera; and Xiaomi Vacuum.

To test the devices in a way that paralleled how a real user might interact with them, we aimed to identify each device's key functionalities (see Appendix B). To this end, two researchers independently conducted cognitive walkthroughs for all 24 devices, their accompanying manuals, and key information online. The researchers met and merged the functionalities identified. For devices with third-party integrations, we randomly selected skills, apps, and media outlines hoping to capture both popular and unpopular examples. Unfortunately, it is intractable to try all possible combinations, limiting our comprehensiveness.

```
{
  "device_name": "Amazon Echo in Study",
  "device_mac": "XX:XX:XX:XX:XX:XX",
  "device_ip": "",                          # optional
  "product_name": "amazon_echo",
  "feature": "hostname",
  ...,
  "allowlist": {
    "ip": set([
      "52.94.229.122",
      "104.154.127.107",
      ...
    ]),
    "hostname": set([
      "avs-alexa-3-na.amazon.com",
      "prod.insights.comms.alexa.a2z.com",
      ...
    ]),
    "domain": set([
      "amazon.com",
      "amazonaws.com",
      ...
    ]),
    "patterns": {
      "guc3-accesspoint-a-p531.ap.spotify.com": "(guc|gae|gew)[0-9]+-
 accesspoint-.*\.ap\.spotify\.com",
      "spectrum.s3.amazonaws.com": "spectrum\.s3\.amazonaws\.com",
      ...
    }
  }
}
```

Figure 1: Example of our allowlist policy format.

Implementation of Network Allowlists (MUDs): In our lab, we used a Jetson Nano running Ubuntu 20.04 as an access point and to intercept home IoT devices' network traffic. The Jetson Nano serves as a NAT gateway. To manipulate network traffic packets, we implemented our own interception tool through `scapy`, a Python library that manipulates network packets, and `NetfilterQueue`, a Python library that intercepts packets using `iptables`, meaning it only naturally supports IP addresses and ports. To log hostnames, we use the router's default DNS resolver to resolve IP addresses encountered. We have (anonymously) open-sourced our implementation [2].

We also measure how transferring observations of one device's traffic to another in a simulated MUD allowlist impacts device functionalities. We enforce that the latter device may only contact endpoints the former device also contacted, implementing allowlists like those shown in Figure 1 via `NetfilterQueue`. To enforce allowlists that include hostnames or domains, we must take additional steps. All hostname rules are resolved to IP addresses and added to IP-based allowlists for enforcement. If the queried hostname in the DNS query is on the allowlist, then the firewall forwards the query to the DNS resolver. Otherwise, the firewall drops the DNS query. Once a DNS response is received, the firewall records the included IP addresses before forwarding it to the device.

Domain-based rules are handled similarly, with the difference that DNS queries are checked against domain-based allowlists instead of those based on hostnames. There is a cold-start issue for relying on DNS. Because the firewall can be activated at any time, some devices may have cached hostname-IP mappings. The firewall will not know about the cached mapping and will incorrectly reject some traffic until the device performs a DNS query.

4. Lab: Impact of Location

To explore how the passage of time and changes in geolocation impact home IoT devices' network traffic,

we conducted initial experiments in our lab. Specifically, we collected traffic from the aforementioned 24 popular home IoT devices in our US-based lab. We used the traffic to establish expected network behaviors (i.e., the endpoints the contacts). We then enforced MUD-style network policies both on the original network in the same lab, as well as on a VPN operating on a different continent to see how these allowlists impacted device functionality.

While prior work proposed generating MUDs from observations of a device's network traffic [14], [15] and reported that home IoT devices contact different services when placed in the UK versus the US [25], those studies did not examine the impact on device functionality like we do, nor perform measurements that were nearly as comprehensive. Another study investigated how *blocklists* of non-essential traffic impacted device functionality [21]. In contrast, our simulated MUDs are *allowlists* that must capture all of a product's necessary endpoints.

4.1. Methods

Before running any tests, two researchers collaboratively determined each of the 24 products' main functionalities by examining product manuals and companion apps. Appendix B gives a complete list of functionalities. We excluded functionalities outside the product's main purpose (e.g., changing the device name). We also excluded functionalities that do not require Internet connectivity, such as casting a webpage to a Google Chromecast.

Data Collection: With each device's functionality documented, we exercised all functionalities listed in Appendix B one by one, recording the resultant network traffic. No network policies (i.e., no MUDs) were applied.

Policy Enforcement: We then created allowlists based on the traffic observed, similar to the approach used in prior work [14], [15], [19]. We then waited a week and factory reset all the devices. Each device was re-initialized and registered as a device from a different country (Germany) at least one week later. We chose the US and Germany as example large markets on different continents with different regulatory regimes. Furthermore, the device's traffic was tunneled via a VPN through a Digital Ocean server in Germany after performing a factory reset on the device. Enforcing the allowlists we created, we again exercised all 24 devices' functionalities one by one, recording whether each functionality continued to work as before. We repeated this procedure without using the VPN to compare how the device's network traffic in our same lab compared to its traffic from at least a week previous. We recognize that this approach has limitations. Notably, a product may vary in its firmware or hardware across countries and thus may vary in multiple ways. Nonetheless, our experiments show that the same device produced for the US region can vary in network behaviors simply because it is registered in another country and its traffic is tunneled through that other country. An additional limitation is that we performed each function for each device only once, increasing the possibility of measurement errors.

Endpoint Representation: We tested allowlists representing endpoints by their second-level **domain** name (e.g., `amazon.com`) and by their **hostname**, or FQDN (e.g., `avs-alexa-6-na.amazon.com`). We also introduced and tested hostname **patterns**, which attempt to abstract similar hostnames into a regular expression. To generate patterns, we used DBSCAN with a precomputed distance matrix to group similar hostnames for each product. We clustered each hostname based on its lowest-level subdomain (i.e., the leftmost part of the FQDN), using the remaining part of the FQDN for grouping so that different domains or subdomains are not clustered together. We condense consecutive numbers to "[0-9]+" because digits rarely carry meaning in this context. Once a cluster was identified, we generated an ordered list of the longest non-overlapping common substrings for all hostnames in the cluster. If we could further group the remaining parts, we accepted these variations rather than using a wildcard to minimize the attack surface. If the remaining parts appeared random, a wildcard replaced them. For example, hostnames used by Apple notifications became the pattern `[0-9]+-courier.push.apple.com`. Spotify access-point hostnames became `guc3-accesspoint-a-.*.ap.spotify.com`.

4.2. Variability in the Same Lab

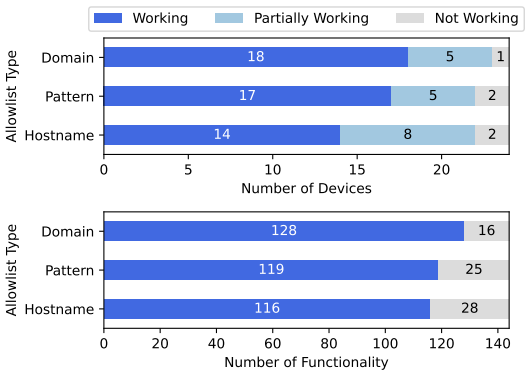
Surprisingly, 10 of the 24 devices lost at least some functionality when we enforced hostname-based allowlists created from each device's earlier network behaviors in the same US-based lab (Figure 2a). Even when we only enforced more permissive domain-based allowlists, 6 of the 24 devices were still affected. Even for the same devices in the same lab, some network behaviors changed between data collection and allowlist enforcement.

At one extreme, the Wyze Camera completely stopped working when enforcing allowlists because it uses IP addresses for live video feeds. We did not observe any DNS traffic relating to these IP addresses. Similarly, when resetting the Amazon Fire, some IP addresses were contacted without corresponding DNS traffic. We suspect these addresses were communicated in encrypted payloads.

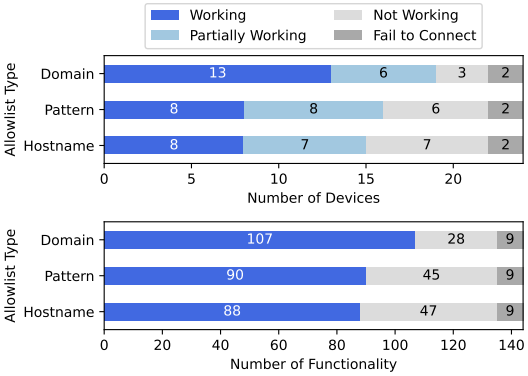
4.3. Variability When Changing Location

With the allowlists unchanged, we observed more failures when the device was instead registered as a European device and its traffic was tunneled over a VPN in Germany (Figure 2b). Of the 24 devices, 16 partially or fully stopped functioning for hostname-based allowlists, and 11 for domain-based allowlists. This result emphasizes that not only are home IoT devices' network behaviors region-specific [25], but attempting to generalize and enforce allowlists across regions will lead to functionality failures.

Comparing the results from the two networks let us isolate the variability caused by geolocation. The key reason was that some devices use different hostnames or domains across countries. For example, the Xiaomi vacuum contacted `us.ot.io.mi.com` in the US, but `de.ot.io.mi.com` in Germany. Amazon Echo tended to contact hosts like `avs-alexa-6-na.amazon.com` in North America ("NA"). In Europe ("EU"), it instead contacted hosts like `avs-alexa-7-eu.amazon.com`. These results showcase that observations made from a single device may not be transferable to another device of the same product, particularly (but not exclusively) if that



(a) Results testing on the same (US-based) network.



(b) Results tunneling through a German VPN.

Figure 2: Functionality degradation when enforcing the MUD-like allowlist created from prior traffic in our lab.

device is located elsewhere in the world. Consequently, small-scale, in-lab studies may fail to capture how home IoT devices' network behaviors vary. To explore factors beyond geolocation, we turn to the crowdsourced dataset.

5. Dataset: Transferability

As shown in our initial investigation in our lab, a home IoT device's network behaviors may change due to the passage of time or a change in location. Observations from one set of devices in one place may not showcase the full variability of that product's network behaviors.

We use the term **transferability** to refer to the capacity to transfer characterizations of observed network behavior from one home IoT device to another. In this section, we use the crowdsourced IoT Inspector dataset to further explore the variability of home IoT devices' network behaviors. For consistency, we used the same 24 products as for our in-lab measurements. We identified several sources of variability, including load balancing, content delivery networks (CDNs), geolocation, variable remote ports, and user-specific DNS resolvers. We also quantified how the considerations outlined in Section 2.2—host representation, sample size, and stability—impacted the transferability of network behaviors.

5.1. Dataset and Data Cleaning

To examine the generalizability of home IoT devices' network behaviors at a larger scale, we used the IoT Inspector dataset [17], which includes the network traffic from over 5,000 users' home IoT devices. Original data collection occurred between April 8 and July 24, 2019, documenting 38,164,870 network flows (combinations of device ID, local port, remote IP address, and remote port). That dataset was collected from human subjects with the approval of the original authors' IRB. They also obtained permission from their IRB to share that data with us.

We analyzed the subset of IoT Inspector containing only devices with product names and network traffic recorded. This subset consists of 5,423 unique devices, representing 424 unique products from 80 different vendors among 43 device types. These 5,423 unique devices came from 1,464 different households. For clarity, the body of the paper reports on the 24 products (3,461 devices total) appearing most frequently in the dataset, for which we also conducted in-lab experiments. The appendix presents analogous graphs with 52 products.

The dataset contains the following information, some of which involved post-processing by the original authors. Each flow contains the device's **product and vendor**, entered manually by participants on the IoT Inspector user interface. It also contains the **remote IP address and remote port** contacted by the device. The original authors attempted to compute additional information about the remote destination via a best-effort process detailed in the original paper [17] that included examining DNS traffic. Specifically, IoT Inspector attempted to identify the domain name corresponding to the IP addresses of endpoints using Server Name Indication (SNI) before resorting to reverse DNS lookup and passive DNS if SNI information was unavailable. Each flow is also associated with the household's **timezone**, as opposed to a more precise geo-location, to respect participant privacy.

While the original authors performed some data cleaning [17], we further cleaned the data in two ways. First, because product names were manually entered by participants, they may not be reliable. We regard a device's product name as likely mislabeled if the device never contacts *any* domain that 20%+ of the other devices with the same product name commonly contact. We found 16 potentially mislabeled devices across 10 products, excluding them from further analysis. In Section 5, we use thresholds to automate accounting for a small number of mislabeled devices in practice. We manually examined other outliers. One device labeled as a Belkin Wemo switch contacted 26,594 unique domains (very unusual relative to other Belkin Wemo switches), so we excluded it.

We also aimed to identify hostnames and domains missing from IoT Inspector. Although many traffic flows on the Internet are preceded by a corresponding DNS lookup, IoT devices are often already installed and operational when the user runs IoT Inspector, so the dataset may not capture the DNS lookup associated with a flow. In other words, as IoT Inspector was collected plug-and-play style not all contacted endpoints have a corresponding DNS request. Although the original authors of the dataset guessed missing hostnames (those without associated DNS traffic) using passive traffic monitoring and reverse DNS lookups, we determined these methods to be unreliable. For example, a reverse DNS lookup from an IP address often resolves to particular infrastructure, such as

a server, rather than a general hostname. About 62.1% of the collected flows contained these potentially unreliable hostnames. To further validate the data, we first compared the IP addresses of any two packets—one known to have the correct hostname based on associated DNS queries in the dataset and an IP address without an associated lookup—that shared the same hostname. If the second packet had the same IP address as the first, we concluded it had the correct hostname. To mitigate the effects of virtual hosting, we first performed this process at the product level, then repeated it at vendor level. This process reduced the fraction of flows with unreliable hostnames to 34.4%. We repeated this approach using ASNs, leaving us with 30.4% of flows with an unreliable hostname.

Limitations: Because the IoT Inspector dataset was crowdsourced from volunteers, some data that would have been interesting to analyze was intentionally not collected to protect participant privacy. Unlike traffic collected by researchers in a lab, IoT Inspector is aggregated on 5-second intervals and does not contain application layer data. It is thus impossible to perform analyses like identifying patterns in inter-packet arrival times. That said, measuring home IoT devices' variability requires a dataset that is as diverse as possible (e.g., collected worldwide from different network environments). IoT Inspector is the best available. These privacy-preserving limitations would likely apply to any crowdsourced dataset. We can nonetheless identify variability in home IoT devices' network traffic and determine to what extent observations transfer between devices. While we obtained two other IoT datasets from researchers at other institutions, both were collected in labs with only one or two devices per product, which made studying variations impossible. As we are unaware of any other large-scale, crowdsourced home IoT datasets, we proceed despite these limitations.

Ethics: This paper reports on analyses of new data we collected in our lab, as well as secondary analysis of IoT Inspector data. Data collection in our lab did not involve human subjects or sensitive data. The IoT Inspector dataset, on the other hand, was collected by its original creators from human subjects and thus required IRB approval, both to collect and to share with us. The original researchers discuss their IRB approval on their webpage [18]. We worked with their IRB to obtain express permission for the data to be shared with the specific researchers involved in this project for our analyses.

Availability: We have made our analysis code available in an anonymized GitHub repository [2]. Unfortunately, IoT Inspector data cannot be shared without approval from the original authors' IRB and proper training of data recipients [17]. We instead provide synthetic sample data in the correct format. That repository also contains our code for enforcing the MUD-style allowlists in our lab [2].

5.2. Methods

In this subsection, we first introduce our metric for measuring the transferability of network behaviors. We then list the five variables that can affect the transferability of network behaviors observed from one product.

5.2.1. Key Metric. We measure the transferability of network behaviors in part through retrospective simulations based on the IoT Inspector dataset. For a given product, we calculate a value we term the **median fraction of observed flows (MFOF)**. Intuitively, the MFOF captures the fraction of traffic previously observed from one set of devices that is still applicable to a different set of devices. A high MFOF is desirable since it reflects a product's network behaviors that are consistent across different devices. We take the median fraction instead of the mean because the distribution can be very skewed. Formally, we define MFOF as:

$$MFOF(D, D') = median(\{\frac{|F_D^d|}{|F^d|}, \forall d \in D'\}) \quad (1)$$

where D is the group of devices that are under observation, D' is the group of devices that have never been observed (but are the same product), d is a device in D', F^d is all the flows transmitted on d, and F_D^d is the fraction of d's flows that have already been known by observing D. We require $D \cap D' = \emptyset$ for a fair evaluation. A flow is a combination of the device ID, the local port, the IP address of the remote endpoint, and the remote port.

5.2.2. Measurement Methods. For each product, we randomly split the devices into five *experimental groups* as part of a cross validation process. The remaining 80% of devices in each fold are the *observation group*. We took the union of all traffic destinations contacted by devices in the observation group and considered them the given product's expected network behaviors, using MFOF to measure transferability to the experimental group.

5.2.3. Measurement Variables. We introduced five variables corresponding to the five challenges in Section 2:

1) **Host representation** is the level of abstraction for describing hosts. In addition to our focus on domains, hostnames, IP addresses, and hostname patterns (see Section 4.1), the appendix also compares other formats, such as subnet and BGP prefix. Ports can be added on top of any selected host representation.

2) **Dataset reliability** is captured by the threshold, or the minimum number of devices on which we need to observe an endpoint to add it to the allowlist. The threshold can be changed to adapt to a dataset's reliability.

3) **Geolocation representativeness** refers to the match between the location of devices in the observation and experimental groups. We use timezone as a proxy for approximate geolocation.

4) **Sample size** As discussed previously, it was an open question how many devices are needed to obtain a general picture of a product's network behaviors. Therefore, when randomly sampling devices from a product, we gradually increase the sample size from 5 to 200 with a step size of five. For each sample size, we performed five-fold cross validation five times and took the mean per product.

5) **Stability** references the length of time for which a previously collected network traffic dataset continues to capture a product's network behavior accurately. That IoT Inspector is three years old lets us measure the stability of a product's network behaviors. We calculated the lifespan of each destination by counting the days between the destination's first appearance in the dataset and the last.

5.3. Coverage of Network Behaviors

First, we took the union of the endpoints seen across all devices for each of the 24 products, analyzing the **coverage**, or the percentage of a product's remote destinations contacted by a given sample of devices of that product. If devices of a particular product behaved similarly to each other, coverage would approach 100%. For a number of products, however, this was not the case. Figure 11 in the appendix shows the coverage for different sample sizes for three key host representations: domains (e.g., `dropcam.com`), hostnames (e.g., `oculus1802-us1.dropcam.com`) and IPv4 addresses (e.g., `35.186.28.155`). For each sample size, we averaged the coverage over 100 random samples. For most products, the coverage for domain representations approached 100%, even for small samples. However, the coverage for hostnames and IP addresses was often substantially lower. For many products, a given device may contact different hostnames and IP addresses even when contacting the same domain.

While we expected to observe low coverage for IP addresses, we were surprised that hostnames followed a similar pattern for some products (e.g., Google Nest, Amazon Ring). Averaged across all 24 products, 95% coverage was only achieved with a sample of 54.9% of the devices of that product in the dataset when using hostnames, compared to 64.1% when using IP addresses. For 99% coverage, the percentages increased to 79.2% (hostname) and 83.8% (IP address). The low coverage in small samples underscores that observations made from one or two devices—as in prior work [11], [14], [15], [25]—miss key variability and thus fail to generalize.

5.4. Sources of Variability

Load Balancing and Content Delivery Networks: A major source of variability appeared to be the use of different, but related, hostnames to support load balancing and content delivery networks (**CDNs**). For example, Apple Push Notification uses multiple servers with hostnames like `27-courier.push.apple.com`, with the number (27) varying across devices. When many similar hostnames exhibit small variations, observations made based on precise hostnames are unlikely to generalize.

The good news is that most hostnames follow patterns. In the simple example above, only the number in a hostname changes, spurring the pattern representation we investigated. Some hostnames varied in more than just numbers. For example, we observed Spotify hostnames with forms like `guc3-accesspoint-a-f002.ap.spotify.com`, where `f002` varied. That said, we also observed seemingly random hostnames (e.g., `d37ju0xanoz6gh.cloudfront.net`), making it hard to know what hostnames a device should expect.

Regionalization: Although Section 4 already described some regional variation in endpoints, IoT Inspector showed more complicated cases. Streaming media destinations varied across regions, reflecting local interests. For example, we observed endpoints like `tf1.fr` (a French TV channel contacted by a Google Chromecast) and `bell.ca` (a Canadian internet service provider contacted by an Amazon Echo and a Chromecast).

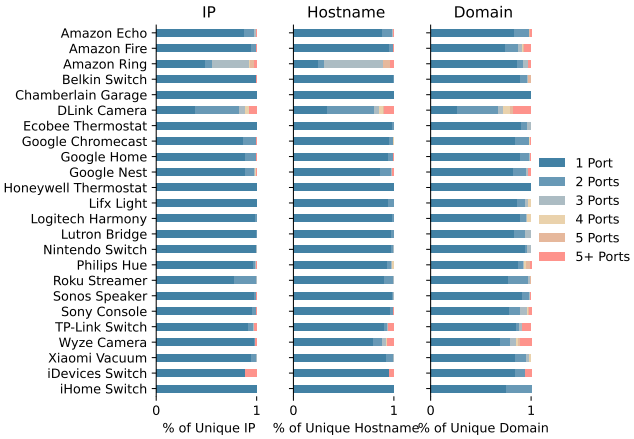


Figure 3: The number of ports used by remote hosts under different endpoint representations. Few remote hosts used multiple ports for incoming connections.

DNS: Because users or their ISPs assign a local DNS resolver, DNS traffic was an additional (expected) source of variability. We observed a long tail of DNS traffic.

Variable Remote Ports: Although most remote hosts used only a few dedicated ports (e.g., port 443 for HTTPS), as shown in Figure 3, some used a large number of remote ports. For example, Amazon Ring devices in our dataset contacted only 72 unique domains, but they contacted the domain `amazonaws.com` on 1,617 different remote ports. This was not an artifact from a single device; 38.3% ($N = 23$) Amazon Rings contacted 10+ remote ports, and 21.7% ($N = 13$) contacted 20+ ports. Some ISP-specific hosts also used a range of ports. For example, three Wyze cameras (all in one household) contacted `mycingular.net` through 2,399 unique remote ports. Two Amazon Echo devices from another household contacted `sbcglobal.net` through 3,858 remote ports.

Other Causes: Some variability may be artifacts of IoT Inspector's data collection process, which can occur to any real-world traffic dataset. For example, 26.7% of endpoints are IP addresses without any associated hostname or domain. Either IoT Inspector missed relevant DNS traffic, or these hosts were actually contacted using hardcoded IP addresses. In fact, we observed the latter in our own experimentation when the Wyze camera in our lab contacted some hosts without querying DNS. Prior work [10] also found that Belkin Wemo plugs transmitted IP addresses in payloads at the time of their data collection in 2020. We found they now transmit hostnames.

Limitations: Some products (e.g., Amazon Echo) have multiple generations that we grouped into a single product. Different generations may behave differently. Even for the same generation, behavior may change depending on the software or firmware version. As detailed in Section 6, a recent update to Belkin Wemo smart plugs caused them to contact completely different endpoints. Furthermore, product names were provided manually. Some volunteers gave only a general name, collapsing different models.

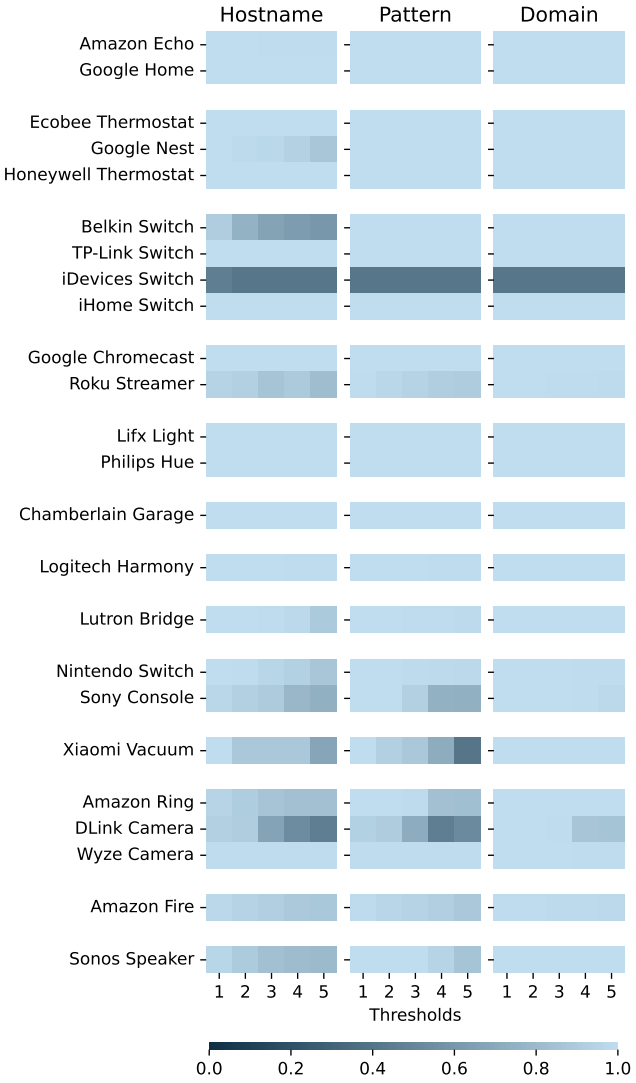


Figure 4: MFOF of behaviors observed from, and applied to, the same product (train-test split). Products (rows) are grouped by type. The color indicates the proportion of observation group flows captured.

5.5. Impact of Endpoint Representation

We tested seven possible representations of endpoints. In this section, we contrast representing endpoints by their domain, pattern, and hostname. Appendix E presents less successful alternatives (e.g., subnet, BGP prefix).

As shown in Figure 4, 23 out of the 24 products had an MFOF above 0.95 using (second-level) domain representations with the *threshold* set to 1. Thus, domains are very likely to be consistent across devices of a particular product. The only exception was the iDevices switch.

Observations abstracted to hostname and pattern represents also transferred across devices for most, but not all, products. With threshold=1, hostname representations achieved an MFOF greater than 0.95 ($mean = 0.96$) for 20 of the 24 products. Using pattern representations, 22 of the 24 products again achieved an MFOF greater than 0.95. Pattern representations outperformed hostnames at higher thresholds for products like Belkin switches.

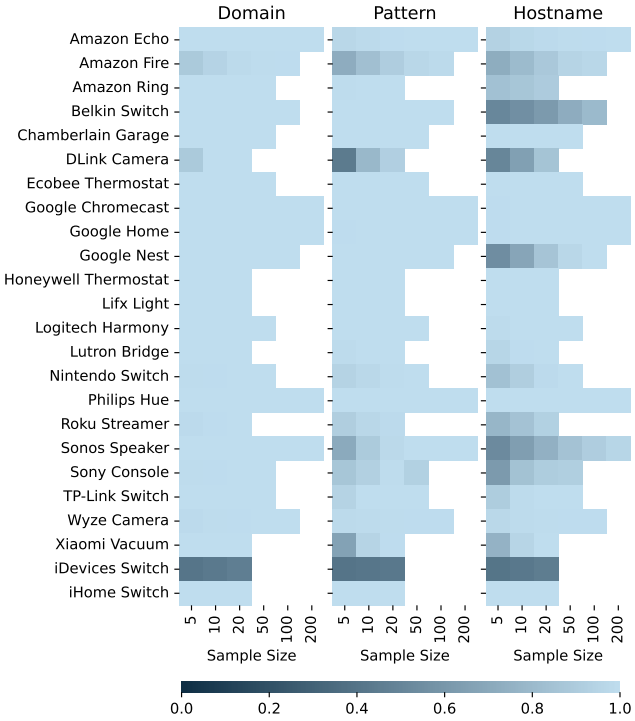


Figure 5: The MFOF increases with the training sample size ($threshold = 1$). Blanks indicate insufficient data.

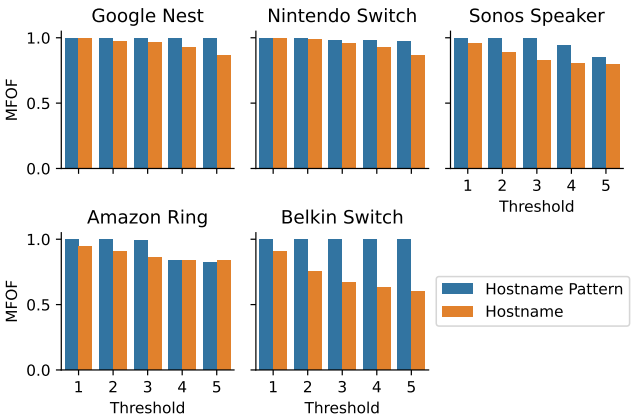


Figure 6: Comparison of MFOF for patterns and hostnames. While they performed similarly for some devices, patterns outperformed hostnames for others.

5.6. Sample Size Required

The more devices one observes, the less likely one will encounter unexpected behaviors in the future. However, it is impossible to observe every single home IoT device in the world, which makes it crucial to understand how many devices one needs to observe to achieve generalizable knowledge of a product's network behaviors. Thus, we calculated the MFOF for various sample sizes.

Figure 5 shows the results. In general, hostname representations necessitated a larger sample than domain and pattern representations. For hostname representations, only 18 of the 24 products ever achieved an MFOF of at least 0.95, and it took 25 devices on average to do so. In contrast, pattern-based allowlists enabled 22 products to achieve an MFOF of at least 0.95, requiring only

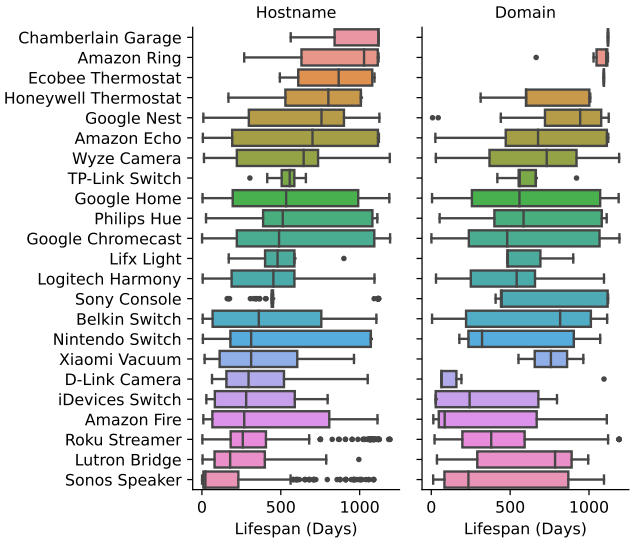


Figure 7: The lifespan of endpoints when *threshold=5*. Observations use the extended IoT Inspector dataset and can be considered lower bounds on lifespans.

10 devices on average. These numbers, however, varied significantly across products, particularly for hostname representations. While 5 Ecobee Thermostats were sufficient to map the hostnames used, 200 Sonos Speakers were needed to form an equivalent understanding.

5.7. Adapting to Unreliable Data

Network traffic datasets collected through crowdsourcing can be very noisy, which is why a threshold needs to be set. The threshold, specifying the minimum number of different devices on which an endpoint must be observed for it to be added to the allowlist, trades off generalizability for reliability. Increasing the threshold typically admits fewer endpoints, making observations more reliable, but may miss some crucial behaviors. Interestingly, we observed that increases in the threshold were not always proportional to the decrease in the amount of traffic allowed. For some products, it was possible to have a high threshold (better reliability) without losing much.

Crowdsourced datasets often contain noise. Setting higher thresholds to remove endpoints that only appear for one or two devices can reduce this noise. However, when the threshold was set to 5, only 12 products had an MFOF of at least 0.95 when using hostname representations. In contrast, 16 products had an MFOF of at least 0.95 when using pattern representations. Emphasizing these trends, Figure 6 shows the five products whose MFOF increases by more than 0.1 by switching from hostnames to patterns. Appendix E shows similar trends hold for a larger dataset of 52 products. In short, *many important hostnames appeared only a few times in the dataset.*

To further understand the relationship between thresholds and the MFOF, we sampled 50 devices from each product, calculating the MFOF under thresholds ranging from 1 to 36 with a step of 5. The process was repeated 20 times, taking the mean. As shown in Figure 12 in the appendix, some products effectively had plateaus in which increasing the threshold minimally impacted the MFOF. In one of the more extreme examples, the MFOF for Google Home was similar between thresholds of 1 and 26. One possible explanation is that, upon increasing the threshold, only the most fundamental endpoints contacted by nearly all devices of that product are kept, and most flows are to those endpoints. For Chamberlain Garage, most endpoints in the long tail were various DNS resolvers. Only two endpoints were contacted by the majority of those devices: `connect1.myqdevice.com` and `connect.myqdevice.com`. In contrast, products that stream user-specified content rarely exhibit this plateau.

5.8. Accounting For Regionalization

IoT Inspector primarily consists of devices from North and South America. It is common for datasets to be geographically biased; collecting data globally requires tremendous effort. Thus, it is important to understand how geographically biased data impacts the representativeness of observations. We created regression models that gauge how region impacts MFOF. Because IoT Inspector does not contain location information, a common privacy requirement for public dataset, we used timezones as a proxy, creating three regions: **Region A** (UTC-02:30 - UTC-10:00), **Region B** (UTC+01:00 - UTC+04:00), and **Region C** (UTC+5:30 - UTC+12:00). We selected the six products in the dataset that have at least 10 devices per region: Amazon Echo, Belkin Switch, Google Chromecast, Google Home, Philips Hue, and Sonos Speaker. We created linear regression models in which the MFOF was the dependent variable and the regions of the training and testing samples, the product, and the (log of the) sample size were the independent variables. Table 7 in Appendix D contains the regression tables. All coefficients of the observation group's region, the experimental group's region, and their interactions were significant. At a high level, network behaviors observed from one region cannot fully depict the network behaviors that appeared in another region. Regional differences in the MFOF can be observed even when keeping the sample size constant. For example, applying observations made from devices in Region A to devices in Region C caused a drop of around 0.1 in MFOF, which means around 10% of traffic is not expected using the observations made from Region A. Therefore, if the data is geographically diverse, care should be paid to region-specific observations.

5.9. Observation Stability and Lifespan

Home IoT products' network behavior might change over time. To better understand the stability of their network behaviors, we measured each endpoint's lifespan, or the time between that endpoint's first and last appearances (per product) in the IoT Inspector dataset. We used an extended version of the IoT Inspector dataset containing three years of traffic data (April 2019 to July 2022) shared with us by the original authors. This extended dataset is in the same format as the main dataset, except that the product information was inferred (mean precision=0.995). We excluded iHome switches because their first and last appearances were separated by less than a day.

Figure 7 shows the distribution of lifespans. Most products did not change endpoints frequently. Most end-

points were active for years in the dataset. Domains usually had a longer lifespan than hostnames. Of the 23 products in the extended dataset, 14 products had a median lifespan of over a year for hostnames, and 18 for domains. Note that these are lower bounds; many remain active.

6. Lab: Using Crowdsourced Data

Having further analyzed the variability of 24 home IoT devices' network traffic using the crowdsourced IoT Inspector dataset, our final experiments returned to the lab. Specifically, we tried applying the characterizations of each product's network behavior from this crowdsourced data to the 24 devices in our lab. We were again specifically interested in whether the devices would remain functional if only allowed to contact the endpoints learned from the crowdsourced data. Prior work aimed at blocking superfluous endpoints reported that many traffic destinations are not necessary for an IoT device to function [21]. Because our simulation of a MUD was an allowlist, not a blocklist, the completeness of our characterization was paramount. As in Section 4, we tested 144 functionalities across 24 devices in our lab when enforcing MUD-like allowlists. *In this final experiment, though, the allowlists were based on IoT Inspector data that was a few years old and from devices other than our own.*

6.1. Experimental Setup and Protocol

Our experiments resembled those from Section 4, except here we enforced different allowlists, only allowing a device to contact destinations determined from the IoT Inspector dataset. All other traffic was dropped. We ran experiments with key variations (e.g., sample size, endpoint representation) to test the real-world impact of the parameters whose abstract transferability we previously studied. After deploying the relevant allowlist, we again tried to exercise each of the product's functionalities. If one failed, we repeated the experiment to confirm.

Whenever possible, we tested each device from an external network (i.e., interacting with the companion app outside the lab's network) to maximize Internet exposure. Only when devices could not be controlled remotely (e.g., Google Chromecast) did we use the local network. We also used voice commands to control products and record the corresponding network traffic. Amazon Echo was used as the default receiver for voice commands.

The experiment consisted of two rounds of testing. The first was conducted in late 2021 on the nine most popular devices in the IoT Inspector dataset. In late 2022 and early 2023, we tested the 15 additional products.

6.2. Results Overview

As shown in Figure 8, domain representations of the MUD-like allowlists generated from crowdsourced data enabled most products to continue to work. In fact, 17 of the 24 products were fully functional, 5 were mostly or partially functional, whereas only 2 were completely nonfunctional. That 128 of the 144 total functionalities we tested (Figure 8) across the 24 products continued to work when enforcing domain-based allowlists is notable since the IoT Inspector data was collected years prior.

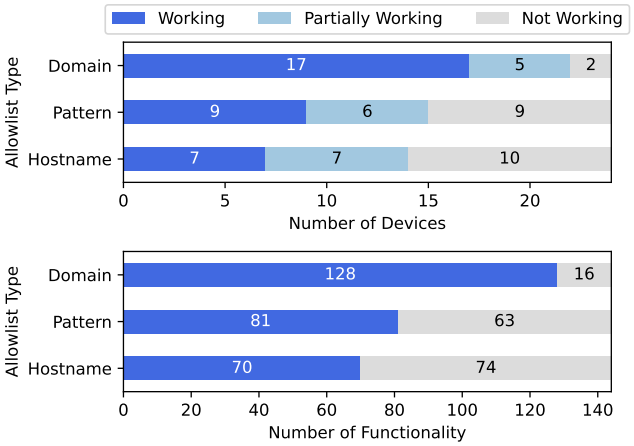


Figure 8: Number of devices (top) and functionalities (bottom) that work when enforcing crowdsourced allowlists.

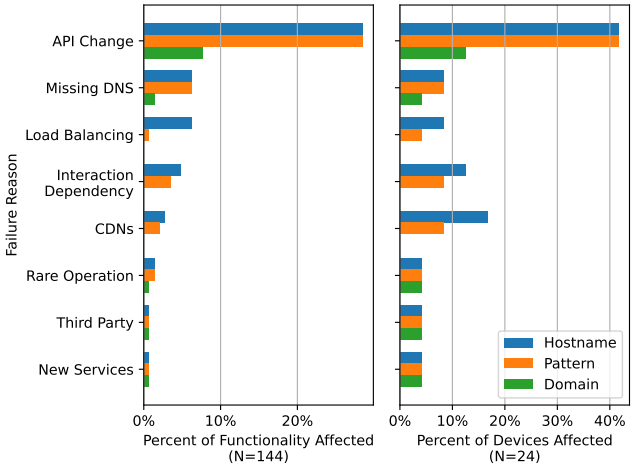


Figure 9: Reasons for failures caused by allowlists.

Conversely, hostname and pattern-based host representations were much more fragile. The devices in our lab were blocked from contacting a number of endpoints not previously seen in the IoT Inspector data, causing some devices to become non-functional. As a result, for pattern-based representations, only 9 products were fully functional, but 9 were completely nonfunctional. Even worse, for hostname-based representations, only 7 devices were fully functional, but 10 were completely nonfunctional.

Rare—But Impactful—New Domains: Although the IoT Inspector dataset was collected in 2019, both rounds of the experiment showed that a product's domains rarely change and even more rarely affect the product's functionality. As shown in Figure 9, during the past three years, only 3 of the 24 products changed the domain they used. The Belkin Wemo Switch changed its main domain name, while the iHome Switch and Honeywell thermostat changed the third-party services they adopted. Two out of the three devices completely stopped working.

Volatile Hostnames: We found that hostnames continued to be volatile. Due to load balancing and CDNs, hostnames not captured by even the pattern abstraction appeared for the first time in our lab. That said, 70 functionalities still worked after three years when using

hostnames, and 81 when using patterns. These functionalities were typically supported by dedicated hostnames, resulting in much cleaner traffic on the device side. For instance, Google Home only needs `www.google.com` for all functionalities other than media (e.g., playing music). Controlling other devices or querying external information were handled by server-to-server communication.

Streaming Media: Dedicated hostnames for streaming audio or video were rare, but our pattern representation better captured such endpoints. Of the 11 streaming functionalities tested, 7 did not work because of hostnames not seen in the dataset. Four of these seven functionalities used hostnames captured by our pattern representation. Nonetheless, streaming media for IoT TVs, smart speakers, and similar media-focused devices are a potentially insurmountable hurdle for MUD-like allowlists unless closely matched data is used to generate the allowlist.

6.3. Detailed Reasons for Failures

As shown in Figure 9, apparent API changes were the main reason for major changes in network behavior, but most such changes were only evident in hostname and pattern representations, not domains. Such changes impacted only 28.5% of functionalities, but 41.7% of the devices we tested. Tables 5–6 in Appendix C detail the types of functionalities that failed and the associated root causes we identified for each functionality failure.

New Services and API Changes: New services and API changes were the most frequent cause of failure. New domains caused functionality failure for 3 of the 24 devices in our testbed, but new hostnames caused functionality failures for 10 of the 24, as shown in Figure 9. Using Belkin Wemo plugs as an example, when the IoT Inspector dataset was collected in 2019, Belkin Wemo plugs used `api.xbcs.net` for their API, which was confirmed in previous papers [17], [21]. However, Belkin recently added a new hostname, `deviceapis.xwemo.com`, to their backend infrastructure. Blocking this previously unseen API endpoint put the device offline. Ironically, the original API endpoint was still live, and the switch in our lab still contacted it from time to time. Vendors may also add new features. In our case, the new Sonos Radio required a previously unobserved domain: `sonos.radio`.

Interaction Dependencies: Examining the relationship between devices' functionality and network traffic in our lab, we noticed an additional cause of failures: *interaction dependencies*. That is, some functionalities were dependent on the device completing a preceding functionality. For example, if streaming music is not available on Sonos speakers, then all other functionalities, such as changing the volume, failed. Connectivity checking was another common and important dependency. Many devices, such as the Google Home and Belkin Wemo Switch, performed connectivity checking. If such checks fail, then the device refuses to take any further commands until it believes it is online again. Google Home uses `www.google.com` for most functionalities, but connectivity checking is via `connectivitycheck.gstatic.com`. Blocking the latter leads to a Google Home that refuses to accept voice commands. Endpoints related to key interactions must be closely monitored for changes.

TABLE 1: Domains on ≥ 3 different products' domain-based allowlists. # indicates the number of products.

Domain	#	Domain	#
amazonaws.com	13	nflxso.net	4
google.com	10	spotify.com	4
amazon.com	6	akadns.net	3
cloudfront.net	6	akamai.net	3
akamaitechnologies.com	5	amcs-tachyon.com	3
googleusercontent.com	5	apple-dns.net	3
akamaiedge.net	5	dropcam.com	3
apple.com	5	google-analytics.com	3
comcast.net	5	googletagmanager.com	3
facebook.com	5	gstatic.com	3
fbcdn.net	4	nflxext.com	3
doubleclick.net	4	qwest.net	3
googleapis.com	4	sbcglobal.net	3
googlevideo.com	4	teksavvy.com	3
nest.com	4	yahoo.com	3
netflix.com	4	youtube.com	3

Load Balancing: CDN hostnames commonly included seemingly random numbers or letters. Even with a large dataset like IoT Inspector, not all CDN or load-balancing endpoints will be observed. For example, our Amazon Echo Dot tried to contact `avs-alexa-14-na.amazon.com`, which was not observed in the dataset, for Alexa services. Thus, it is hard to draw any transferable conclusions from these hostname observations. Our lab experiments verified that patterns sometimes mitigate this failure. Seven of the eight functionalities that failed due to load balancing can be fixed if we extract the patterns through clustering. For example, when we attempted to play a playlist, Sonos One contacted `guc3-accesspoint-a-vr15.ap.spotify.com`. This endpoint was blocked by hostnames, but allowed by patterns.

Third-Party Services: Integration with user-chosen, third-party services also caused failures. IoT Inspector captured popular integrations with YouTube, Netflix, or Spotify, but less popular services may encounter problems.

6.4. Domains Frequently on Allowlists

While the previous analyses focused on how allowlists impacted products' functionality, our final analysis turned back to their impact on security. Specifically, we examined the degree to which the 24 different home IoT products' domain-based allowlists had domains in common. The more domains they share in common, especially domains an attacker might try to DDoS or on which an attacker could establish an exfiltration sink, the worse for security. Regardless, though, MUD-style allowlists containing a relatively small number of domains still provide a vastly smaller attack surface than the current state of affairs even when they contain these shared domains.

We found that 90 domains appeared on at least two different products' domain-based allowlists. As Table 1 enumerates, 32 domains appeared on three or more products' domain-based allowlists. Some of these domains belong to major advertising networks like Facebook or Google and are presumably used for tracking and profiling users, or perhaps single-sign-on purposes.

Most notably, though, many of the most common domains in Table 1 are cloud providers and CDN hosts like Amazon Web Services (appearing on 13 allowlists), Akamai, CloudFront, and Heroku. For many of these types

of hosts, an attacker could simply pay to spin up a virtual machine on that domain to exfiltrate today. Furthermore, because these cloud providers and CDNs host so much content for other users, they can also become candidates for a DDoS attack, though they also tend to be very well-provisioned to resist DDoS attacks.

7. Related Work

Home IoT Network Measurements: Measurement studies often provide insights into a system's performance and behaviors. Home IoT is no exception. By collecting traffic from several home IoT products in the lab, prior work has assumed that the findings can be easily transferred across devices of the same product, which may not be true for all applications and products. For example, applications like anomalous flows detection [5], [9], [12], [13], [22], [23], [28], [29], [33], device fingerprinting [8], and blocklist (or allowlists) creation [14], [15], [21] all typically rely on training and evaluation data from dozens of devices in a lab environment. Whether these techniques would apply to a larger dataset remains unclear. Even for measurement studies, it is not uncommon to use tens of home IoT devices to draw conclusions [25]. Our use of a large-scale dataset of network traffic provides a new angle to understand to what extent observations can be generalized.

In addition to our novel combination of analyzing a crowdsourced dataset and in-lab measurements, the IoT Inspector dataset's realism is notable. It was collected from 5,439 volunteers around the world [17], reflecting devices deployed under real conditions. Much existing work, such as from DeMarinis et al. [7], assumes that IoT devices have predictable patterns in network traffic. Prior work, including from Dong et al. [8] and Mandalari et al. [21], is based on static snapshots of IoT device behaviors in the lab, missing changes from firmware updates.

Generating IoT network policies: Our work is largely motivated by the MUD [20] proposal, which in essence is a framework for deploying allowlists on home IoT devices. Network policies are often created, maintained, and shared by communities, but to enable crowd-sourcing IoT network policies, it is crucial that the observations made from one set of devices can be transferred to other devices for the same home IoT products. Identifying and quantifying the variations is merely a step in the exploration, which is different from all the prior work that set out for IoT network policy generation. There are two types of network policies: blocklists and allowlists. Blocklists have received more attention and are more widely used. In a different domain, many ad blockers use a blocklist strategy, gathering policies through community effort [24]. Prior research on IoT network blocklists instead creates broadly applicable blocklists by blocking observed hostnames one at a time on a real IoT device [21]. Blocklists do not have to be complete to maintain device functionality, so they can be deployed immediately.

Creating crowd-sourcing allowlists is a different story. Although several studies proposed creating allowlists based on observing a single device [4], [11], [14], [15], our analysis shows that due to load balancing, geolocation, and other network-specific settings, these allowlists will not generalize. Applications like allowlists are thus not suitable for in-lab studies. To enable such applications, it is crucial to acknowledge the variations of home IoT products' network behaviors first, and then think of ways to be adaptive. Unlike prior work, our study took a deep dive to understand the variations and quantify the transferability, showcasing what products' network behaviors are more consistent than others, in what aspect, and how to achieve a more stable dataset.

In-the-Wild Home IoT Network Measurements: Other than the IoT Inspector dataset, a few other network measurement studies focus on in-the-wild home IoT devices. Saidi et al. studied both IoT devices' backend infrastructure and IoT device detection through a major European ISP data [26], [27]. Their study focuses more on the cloud service providers instead of individual home IoT products, due to the nature of the dataset. In contrast, our study uncovers home IoT products' usage of cloud services on a large scale through the lens of individual devices, exploring how these services impact device functionality.

8. Discussion and Conclusions

Previous studies of the network behaviors of home Internet of Things (IoT) devices have typically been conducted in controlled lab settings, leaving open questions about how well these measurements generalize. We conducted the most thorough and comprehensive measurement to date of how home IoT products' network behaviors vary across different devices (physical instances of a given product), as well as the degree to which the devices would continue to function if these measurements were used to enforce MUD-like allowlists.

While one might have expected to find variations in the traffic, the variations we observed were significant enough to cause some products to stop working when trying to build an allowlist. More precisely, for the majority of the 24 products we tested, at least one type of functionality ceased to function when enforcing MUD-style allowlists that represented hosts with their full hostname (FQDN) or a pattern-based abstraction of the full hostname.

Allowlists That Work in Practice: Nonetheless, without manufacturers creating MUDs at scale, crowdsourcing MUDs is a possible way to improve IoT security. Even using arguably outdated (three-year-old) data, we showed functional MUD-like allowlists could be created successfully for some devices. Specifically, even using crowdsourced data from other people's devices collected a few years prior to generate allowlists, only 2 of the 24 products we tested completely ceased to function using the domain to represent endpoints. For both of those products—a Honeywell thermostat and an iHome switch—the vendor had changed a primary web domain with which those devices communicate. For the five additional products that had some functionality stop working even though most continued to work, we observed a variety of underlying reasons. Sonos One introduced a new streaming radio feature that launched after the IoT Inspector dataset had been collected [32]. Both the Lutron Bridge and Amazon Fire stick failed when performing a factory reset, a rare behavior that likely had not been captured in the IoT Inspector dataset and that used different API endpoints than the products' other operations. The Nintendo Switch failed when trying to watch YouTube, a third-party integration

that had not been exercised in the original dataset. The Amazon Ring live view and 2-way audio features failed because our system did not observe the DNS query necessary to map the IP addresses observed on the network to a domain in the allowlist, possibly due to caching.

As such, **we found evidence that one can construct a domain-based allowlist for many home IoT products based on a crowdsourced dataset like IoT Inspector, and that most products will continue to work normally**. Had we also collected more recent data from our own device, or likely just more recent from crowdsourced devices, the new API endpoints and new features that caused many of the failures above would not have been an issue. We can imagine an ecosystem in which new measurements are regularly taken by some crowd contributor for home IoT products to ensure the comprehensiveness of these types of allowlists, though it will be important to enforce the types of thresholds we investigated to prevent a small number of malicious actors from poisoning the allowlist.

A few types of home IoT products seem unlikely to work fully with an allowlist under any circumstances, however. Specifically, devices that rely on streaming media from potentially arbitrary media sources. Examples include certain features of the Sonos Radio, smart TVs (not investigated in our study), or using video game consoles for visiting arbitrary media platforms or websites. For home IoT products with more constrained purposes however, domain-based allowlists seem promising.

Security Implications: Prior work has demonstrated it is possible to create MUD files by measuring the traffic of one device in the lab [14], [15]. We, on the other hand, argue that these unofficial MUD files may only work on that same, single device for a limited period of time. This puts pressure on end users to maintain the MUD file, which can be very tedious. For hostnames that change due to load balancing and CDNs, we also show that they can sometimes be addressed by extracting patterns from these hostnames, or even better by instead simply using domain-based representations. However, this approach can have implications for security.

We observed a number of CDNs included as endpoints in the IoT Inspector data. Many CDNs are provided by public cloud service provider, which means anyone can register their own hostname under the same domain. Allowing a domain that is used for public cloud services is obviously problematic for security because it enables an attacker to create a new endpoint that is already contained in the allowlist. Using patterns may be another solution. Unfortunately, if the cloud service provider or the vendor uses random strings for their hostnames (e.g., CloudFront), then the end-user has to allow everything from that domain because the hostname is unpredictable. Even if the cloud service provider or the vendor has a particular naming convention for their services, cloud service providers may not provide a way to block off the whole naming space for the vendor, giving attackers an opportunity to exploit it. For example, `arlostatic-z1.s3.amazonaws.com` is one hostname used by Netgear's Arlo cameras. We observed related hostnames in the dataset with different final digits in the lowest-level subdomain. However, even with a large-scale dataset like the IoT Inspector dataset, we cannot be certain what is the range of these final digits. Thus, one may use *arlostatic-z[0-9]+\.s3\.amazonaws\.com* as the pattern. An attacker can exploit the fact there is no limitation placed on the numbers and try to register a hostname like *arlostatic-z111.s3.amazonaws.com* to match. We confirmed this to be feasible on Amazon AWS S3.

To reduce the attack service while still leveraging the benefits of cloud computing, home IoT vendors could use their own (vendor-controlled) second-level domains to point at relevant parts of cloud hosting services and CDNs, rather than doing so at the sub-sub-domain level as we observed. Using vendor-controlled second-level domains would prevent common domains like `amazonaws.com` from needing to appear in the allowlists. Alternatively, vendors would need to release their own comprehensive MUD files, which we earlier mentioned has not occurred despite the MUD format being standardized.

The threshold for deciding whether to include a particular endpoint in the allowlist also has important security implications. To provide robustness against a small number of previously compromised devices in the dataset, setting a threshold higher than the number of previously compromised devices can minimize the chance that illegitimate endpoints appear in the final allowlist. The same consideration applies to intentionally malicious devices that a bad actor submits to a crowdsourced dataset. Future work could consider more rigorously attempting to remove potentially compromised devices, which we did manually (Section 5.1) for a small number of IoT Inspector devices that seemed to be either mislabeled or possibly compromised based on how different their traffic looked from other devices of the same product.

Regional variability in the endpoints a product contacts can also have implications for security. If the endpoints a given product contacts differ substantially by region, the crowdsourced dataset must contain at least a threshold number of devices from each region, and likely more than that because not every device will necessary contact every endpoint. For regions in which few users have submitted data from a particular product, or even own a particular product, a lower threshold might be needed to ensure functionality, yet this negatively impacts security.

Lifespan of Network Measurement Studies: As discussed in Section 5.9, most products' network behaviors may need to be updated yearly or bi-yearly. Our lab experiments (Section 6) further showcased that a three-year-old dataset does not fully reflect all current behaviors of home IoT devices even though they captured many. Based on our use of a three-year-old dataset, we estimate that home IoT measurements' "shelf life" may be on the order of a few years. If one uses network measurements for creating MUD files or for anomaly detection, it may be expedient to collect new data after a year or two. That said, our measurement can only provide a lower bound on lifespans. There could be endpoints that are still active, but were not captured by IoT Inspector.

Data Availability

Our code for conducting the analyses reported in this paper is available [31]. Unfortunately, IoT Inspector data cannot be shared without explicit permission from the original authors' IRB and required training [17]. We instead provide correctly formatted synthetic sample data.

Acknowledgments

We thank Christine Jacinto, Julio Ramirez, Lefan Zhang, Olivia Morkved, Paula Martinez-Garcia, and Valerie Zhao. This material is based upon work supported by the National Science Foundation under Grants CNS-1756011, CNS-1955805, CNS-2219867, and DGE-2140001, and by a Consumer Reports Digital Fellowship.

References

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

[2] Anonymous. Code and artifacts, 2023. https://github.com/research0269/iot-firewalls-2023.

[3] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the mirai botnet. In *26th USENIX Security Symposium, USENIX Security 2017*, pages 1093–1110, 2017.

[4] David Barrera, Ian Molloy, and Heqing Huang. Standardizing iot network security policy enforcement. In *Workshop on Decentralized IoT Security and Standards (DISS)*, page 6, 2018.

[5] Suman Sankar Bhunia and Mohan Gurusamy. Dynamic attack detection and mitigation in iot using sdn. In *2017 27th International telecommunication networks and applications conference (ITNAC)*, pages 1–6. IEEE, 2017.

[6] Ezra Caltum and Ory Segal. Sshowdown - exploitation of iot devices for launching mass-scale attack campaigns, 2016. https://tinyurl.com/3tbe358x.

[7] Nicholas DeMarinis and Rodrigo Fonseca. Toward usable network traffic policies for iot devices in consumer networks. In *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*, pages 43–48. ACM, 2017.

[8] Shuaike Dong, Zhou Li, Di Tang, Jiongyi Chen, Menghan Sun, and Kehuan Zhang. Your smart home can't keep a secret: Towards automated fingerprinting of iot traffic. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pages 47–59, 2020.

[9] Tomer Golomb, Yisroel Mirsky, and Yuval Elovici. Ciota: Collaborative iot anomaly detection via blockchain. *arXiv preprint arXiv:1803.03807*, 2018.

[10] H. Guo and J. Heidemann. Iotsteed: Bot-side defense to iot-based ddos attacks (extended). USC/ISI Technical Report ISI-TR-738, 2020.

[11] Javid Habibi, Daniele Midi, Anand Mudgerikar, and Elisa Bertino. Heimdall: Mitigating the internet of insecure things. *IEEE Internet of Things Journal*, 4(4):968–978, 2017.

[12] Ibbad Hafeez, Aaron Yi Ding, Lauri Suomalainen, Alexey Kirichenko, and Sasu Tarkoma. Securebox: Toward safer and smarter iot networks. In *Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking*, pages 55–60, 2016.

[13] Ibbad Hafeez, Aaron Yi Ding, and Sasu Tarkoma. Ioturva: Securing device-to-device (d2d) communication in iot networks. In *Proceedings of the 12th Workshop on Challenged Networks*, pages 1–6, 2017.

[14] Ayyoob Hamza, Dinesha Ranathunga, Hassan Habibi Gharakheili, Matthew Roughan, and Vijay Sivaraman. Clear as MUD: generating, validating and applying iot behavioral profiles. In *Proceedings of the 2018 Workshop on IoT Security and Privacy*, pages 8–14. ACM, 2018.

[15] Ayyoob Hamza, Dinesha Ranathunga, Hassan Habibi Gharakheili, Theophilus A. Benson, Matthew Roughan, and Vijay Sivaraman. Verifying and monitoring iots network behavior using mud profiles. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[16] Stephen Herwig, Katura Harvey, George Hughey, Richard Roberts, and Dave Levin. Measurement and analysis of hajime, a peer-to-peer iot botnet. In *Proc. NDSS*, 2019.

[17] Danny Yuxing Huang, Noah Apthorpe, Frank Li, Gunes Acar, and Nick Feamster. 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):1–21, 2020.

[18] IoT Inspector Team. IRB considerations. https://inspector.engineering.nyu.edu/irb, 2023.

[19] Eliot Lear, Ralph Droms, and Dan Romascanu. Manufacturer usage description specification. https://tools.ietf.org/html/rfc8520.

[20] Eliot Lear, Ralph Droms, and Dan Romascanu. Manufacturer usage description specification, 2020. https://datatracker.ietf.org/doc/rfc8520/.

[21] Anna Maria Mandalari, Daniel J. Dubois, Roman Kolcun, Muhammad Talha Paracha, Hamed Haddadi, and David R. Choffnes. Blocking without breaking: Identification and mitigation of non-essential iot traffic. *PETS*, 2021.

[22] Samuel Mergendahl, Devkishen Sisodia, Jun Li, and Hasan Cam. Source-end ddos defense in iot environments. In *Proceedings of the 2017 workshop on internet of things security and privacy*, pages 63–64, 2017.

[23] Mehdi Nobakht, Vijay Sivaraman, and Roksana Boreli. A host-based intrusion detection and mitigation framework for smart home iot using openflow. In *2016 11th International conference on availability, reliability and security (ARES)*, pages 147–156. IEEE, 2016.

[24] Pi-hole. Pi-hole - network-wide ad blocking, 2022. https://pi-hole.net/.

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

[26] Said Jawad Saidi, Anna Maria Mandalari, Roman Kolcun, Hamed Haddadi, Daniel J. Dubois, David R. Choffnes, Georgios Smaragdakis, and Anja Feldmann. A haystack full of needles: Scalable detection of iot devices in the wild. In *Proceedings of the ACM Internet Measurement Conference*, pages 87–100. ACM, 2020.

[27] Said Jawad Saidi, Srdjan Matic, Georgios Smaragdakis, Oliver Gasser, and Anja Feldmann. Deep dive into the iot backend ecosystem. In *Proceedings of the 22nd ACM Internet Measurement Conference*, pages 488–503. ACM, 2022.

[28] Arunan Sivanathan, Daniel Sherratt, Hassan Habibi Gharakheili, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Characterizing and classifying iot traffic in smart cities and campuses. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 559–564. IEEE, 2017.

[29] Arunan Sivanathan, Daniel Sherratt, Hassan Habibi Gharakheili, Vijay Sivaraman, and Arun Vishwanath. Low-cost flow-based security solutions for smart-home iot devices. In *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6. IEEE, 2016.

[30] Statista. Do you own smart home devices - i.e. devices that you can control via a smartphone / an internet connection?, Jul 2021. https://www.statista.com/forecasts/1097129/smart-home-device-ownership-in-selected-countries.

[31] UChicago SUPERgroup. Code and artifacts, 2024. https://github.com/UChicagoSUPERgroup/eurosp-24-code.

[32] Chris Welch. Sonos launches its own streaming radio service, 2020. https://www.theverge.com/2020/4/21/21228460/sonos-radio-announced-features-streaming-music-date-price.

[33] Tianlong Yu, Vyas Sekar, Srinivasan Seshan, Yuvraj Agarwal, and Chenren Xu. Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, pages 1–7, 2015.

[34] Nan Zhang, Soteris Demetriou, Xianghang Mi, Wenrui Diao, Kan Yuan, Peiyuan Zong, Feng Qian, XiaoFeng Wang, Kai Chen, Yuan Tian, et al. Understanding iot security through the data crystal ball: Where we are now and where we are going to be. *arXiv:1703.09809*, 2017.

Appendix A.
Product Popularity and Approximate Prices

Device Name	# Amazon Reviews	Price
Amazon Echo Dot	101.6K	$49.99
Amazon Fire Stick	468.1K	$39.99
Amazon Ring	15.3K	$59.99
Belkin Wemo Plug	19K	$38.68
Chamberlain Garage	305	$199.00
DLink Camera	84	$69.99
Ecobee Thermostat	11.8K	$149.99
Google Chromecast	20.5K	$49.99
Google Home	411	$49.99
Google Nest Thermostat	13.8K	$111.90
Honeywell Thermostat	2.5K	$159.99
iDevice Switch	1.2K	$67.60
iHome Switch	234	—
Lifx Light	2.3K	$39.98
Logitech Harmony	3.4K	$219.00
Lutron Bridge	4.9K	$99.95
Nintendo Switch	35.7K	$349.99
Philips Hue	35.2K	$39.95
Roku Streamer	76.9K	$29.00
Sonos One	7.7K	$230.85
Sony Console	1.6K	$227.00
TP-Link Plug	29K	$13.99
Wyze Camera	84.8K	$32.50
Xiaomi Vacuum	412	—

TABLE 2: A reference for the market popularity (measured using the number of reviews as a proxy) and current prices of the products in our testbed according to Amazon.com. Please note that the figures in this table were recorded in May 2024, which may represent newer versions of products compared to the ones we purchased in 2020 and 2021. Most products now have a newer model available, and some have even been discontinued. "—" denotes products that no longer have a price on Amazon.com.

Appendix B.
Functionalities Tested

TABLE 3: Product functionalities tested in our lab.

Functionality	Description
Amazon Echo Dot	
Productivity	"Alexa, what's the time?"
Entertainment	"Alexa, play music."
Device control	"Alexa, turn on/off all Hue lights."
Communication	"Alexa, Call Alice."
Shopping	"Alexa, search for dog toys."
Skills	"Alexa, play thunderstorm sounds."
Change volume	Change the volume of the Echo Dot.
Factory Reset	Reset and initialize the device.
Amazon Fire Stick	
Built-in Alexa	"Alexa, find action movies"
Streaming	Streaming a movie that is free (with ads).
Download apps	Download Spotify in the App Market on Amazon Fire.
Factory Reset	Reset and initialize the device.
Amazon Ring	
Live view	Watch live video feed.
Notification	Receive notification.
Two-way audio	Two-way audio communication.
Motion detection	Motion detection.
Belkin Wemo Plug	
On	Turn on the plug.
Off	Turn off the plug.
Scheduling	Schedule the plug to turn on/off at a specific time.
Auto-off	Set up a one-minute countdown to turn off the plug.
Away mode	Activate the away mode.
Factory Reset	Reset and initialize the device.
Chamberlain Garage	
Control Garage Door	Open/close garage door.
Schedule	Schedule an event.
Alerts	Alerts of door opening or closing.
DLink Camera	
Live view	Watch live video feed.
Move camera	Move the view of camera.
Two-way audio	Two-way audio communication.
Motion detection	Motion detection.
Recording	Recording the video feed.
Schedule	Schedule an event.
Change scene	Change to a preset scene.
Automation	Execute an automation.
Factory reset	Reset and initialize the device.
Ecobee Thermostat	
Change temperature	Change target temperature.
Select modes	Change thermostat's mode.
Schedule	Schedule an event.
Comfort setting	Change one's comfort setting.
Vacation	Change the thermostat to vacation mode.
Reminder	Set up a reminder.
Fan setting	Change the fan's setting.
Follow me	Turn on/off the "Follow me" feature.
Built-in Alexa	Use the built-in Alexa to control the device.
Factory reset	Reset and initialize the device.
Google Chromecast	
Watch Live TV	"Hey Google, play Youtube on the Hallway TV"
Cast tab to TV	
Cast screen to TV	
Cast media to TV	
Factory Reset	Reset and initialize the device.
Google Home	
Media	"Google, play some music."
Control Chromecast	"Hey Google, play Youtube on the Hallway TV"
Find your phone	"Hey Google, find my phone"
Manage tasks	"Hey Google, set a timer for 1 min."
Control your home	"Hey Google, turn on the wemo plug."
Plan	"Hey Google, how's the weather?"
Change volume	
Factory Reset	Reset and initialize the device.
Google Nest Thermostat	
Change temperature	Change target temperature.
Change mode	Change thermostat's mode.
Change preset	Change the preset.
Hold temperature	Hold temperature.
Turn on/off the fan	Turn on/off the fan.
Change schedule	Change the schedule.
Honeywell Thermostat	
Adjust temperature	Change target temperature.
Schedule	Schedule an event.
Vacation	Change the thermostat to vacation mode.
Mode	Change thermostat's mode.
Adaptive recovery	Turn on/off the "Adaptive recovery" feature.
Voice control	Control the thermostat using Alexa.
iDevice Switch	
Turn on/off	Turn on/off the switch.
Turn on/off the night light	Turn on/off the night light on the switch.
Change the night light's color	Change the night light's color.
Schedule	Schedule an event.
Factory Reset	Reset and initialize the device.
iHome Switch	
On/off	Turn on/off the switch.
Schedule	Schedule an event.
Scene	Change a scene.
Voice control	Control the switch (on/off) using Alexa.

TABLE 4: (Continuation) Functionalities tested.

Functionality	Description
Lifx Light	
Schedule	Schedule an event.
Voice control	Control the light (on/off) using Alexa.
Change color	Change the light's color.
Change scene	Change to a preset scene.
Change effects	Change to a preset effects.
Logitech Harmony	
Select activity	Perform an action on a device
Turn on/off device	Turn device on/off.
Help	Open help page.
Set up device	Connect to a new device.
Lutron Bridge	
Turn on/off	Turn on/off the switch.
Add/Change scene	Add and change the scene.
Add schedule	Add a scheduled event.
Factory Reset	Reset and initialize the device.
Nintendo Switch	
eShop (first time)	Open eShop for the first time (need password)
eShop (not first time)	Open eShop not for the first time (do not need password)
News	Open News.
Switch Online	Open Switch Online.
Link account	Link Nintendo account when it is used for the first time
Download app	Download app.
Friends	Add friends.
Use an app	Open the YouTube app.
Purchase an app	Purchase a free game/app.
Factory Reset	Reset the device.
Initialization	Initialize the device.
Philips Hue	
On/Off	Turn lights on/off.
Brightness	Change the brightness level of a light.
Voice control	Control the device (on/off) using Alexa.
Timer	Activate a timer that would turn lights on/off in one minute.
Routine	Change time to fit for the experiment
Factory Reset	Reset and initialize the device.
Roku Streamer	
Stream video	Stream a video.
Adjust volume	Change volume on a remote.
Voice control	Using voice command to change volume and make searches.
Remote control (from phone)	Using Roku app on a smartphone as a remote controller.
Search	Search for a show.
Sonos One	
Radio	Play Sonos Radio.
Streaming	Play songs from Spotify.
Change volume	Change volume.
Pause	Pause the music.
Play songs from phone	Play a song from a phone under the same network.
Voice control (streaming)	Using Alexa to play songs.
Voice control (volume)	Using Alexa to change volume.
Playlists	Edit playlists.
Factory Reset	Reset and initialize the device.
Sony Console	
Buy games	Purchase a game.
Download games	Download a game.
Family control	Change family settings.
Browse store	Open and browse the store page.
Share screenshots	Share screenshots.
Connect to spotify	Connect to the user's Spotify account.
TP-Link Plug	
On	Turn on the plug.
Off	Turn off the plug.
Scheduling	Schedule the plug to turn on/off at a specific time.
Timer	Set up a one-minute countdown to turn off the plug.
Away mode	Activate the away mode.
Factory Reset	Reset and initialize the device.
Wyze Camera	
Live streaming	Watch the live stream from the camera.
Event recording	Record the live stream.
Motion Tagging	Tag motions.
Night vision	Switch the camera to night vision.
2-way audio	Two-way communication through the camera.
Sharing	Share a video clip to other family members.
Rules	
On	Turn on the camera.
Off	Turn off the camera.
Factory Reset	Reset and initialize the device.
Xiaomi Vacuum	
Clean	Begin cleaning.
Level control	Change clean level.
Schedule	Schedule an event.
Automation	Execute an automation.
Remote control	Control the vacuum's direction through phone.
Find the vacuum	Make the vacuum beep.
Factory reset	Reset and initialize the device.

Appendix C.
Tables Indicating Failures Due to Network Policy Enforcement

TABLE 5: The highest tested thresholds (i.e., the strictest allowlist) and the number of allowed hostnames/patterns/-domains that keep devices functioning in our in-lab experiment. × [Failure Reason] means none of our generated allowlists maintained functionality; the "failure reason" explains why. To force the tested devices to use the Internet instead of the local network, the companion app is used on a smartphone connected to a different network by default, unless the functionality under test requires the smartphone to be on the same network (e.g., Google Chromecast).

Device	Functionality	Hostname		Hostname Pattern		Domain	
		# of Allowed Hostnames	Threshold	# of Allowed Patterns	Threshold	# of Allowed Domains	Threshold
Amazon Echo Dot (N=576)	Productivity	× Load balancing		22	115	2	518
	Entertainment	× Load balancing		× CDNs		4	460
	Device control	× Load balancing		22	115	2	518
	Communication	× Load balancing		× CDNs		11	57
	Shopping	× Load balancing		22	115	2	518
	Skills	× Load balancing		1982	1	91	4
	Factory reset	× Load balancing		22	115	4	460
Amazon Fire Stick (N=136)	Built-in Alexa	77	13	74	13	6	54
	Streaming	10486	1	5481	1	17	27
	Download apps	210	5	187	5	33	13
	Factory reset	× API change		× API change		× API change	
Amazon Ring (N=60)	Live view	× Missing DNS		× Missing DNS		× Missing DNS	
	Notification	× Missing DNS		× Missing DNS		2	48
	2-way audio	× Missing DNS		× Missing DNS		× Missing DNS	
	Motion detection	× Missing DNS		× Missing DNS		2	48
Belkin Wemo (N=220)	On/Off	× API change		× API changes		13	5
	Scheduling	× API changes		× API changes		13	5
	Auto-off	× API changes		× API changes		13	5
	Away mode	× API changes		× API changes		13	5
	Factory Reset	2	110	2	110	1	198
Chamberlain Garage (N=68)	Control Garage Door	× API change		× API change		1	56
	Scheduling	× API change		× API change		1	56
	Alerts	× API change		× API change		1	56
DLink Camera (N=40)	Live view	× API Change		× API Change		1	36
	Move camera	× Interaction dependency		× Interaction dependency		1	36
	Motion/People detection	× API Change		× API Change		1	36
	Two-way audio	× Interaction dependency		× Interaction dependency		1	36
	Recording	× Interaction dependency		× Interaction dependency		1	36
	Schedule	× API Change		× API Change		1	36
	Change scene	× API Change		× API Change		1	36
	Automation	× API Change		× API Change		1	36
	Initial setup	× API Change		× API Change		1	36
Ecobee Thermostat (N=116)	Change Temperature	5	11	5	11	1	104
	Select modes	5	11	5	11	1	104
	Schedule	5	11	5	11	1	104
	Comfort setting	5	11	5	11	1	104
	Vacation	5	11	5	11	1	104
	Reminder	5	11	5	11	1	104
	Fan setting	5	11	5	11	1	104
	Follow me	5	11	5	11	1	104
	Built-in Alexa	5	11	5	11	3	11
	Factory Reset	3	26	3	26	1	104
Google Chromecast (N=288)	Watch Live TV	× CDNs		230	2	47	5
	Factory reset	13	115	5	201	2	201
Google Home (N=490)	Media	× CDNs		151	3	2	392
	Control Chromecast	8	245	7	245	2	392
	Find your phone	8	245	7	245	2	392
	Manage tasks	8	245	7	245	2	392
	Control your home	8	245	7	245	2	392
	Plan	8	245	7	245	2	392
	Factory reset	26	49	25	49	2	392
Google Nest (N=163)	Change Temperature	× API change		× API changes		20	4
	Change mode	× API changes		× API changes		20	4
	Change preset	× API changes		× API changes		20	4
	Hold temperature	× API changes		× API changes		20	4
	Fan on/off	× API changes		× API changes		20	4
	Change schedule	× API changes		× API changes		20	4
Honeywell Thermostat (N=28)	Change temperature	× API change		× API change		× API change	
	Schedule	× API change		× API change		× API change	
	Vacation	× API change		× API change		× API change	
	Mode	× API change		× API change		× API change	
	Adaptive recovery	× API change		× API change		× API change	
	Voice control	× API change		× API change		× API change	
iDevice Switch (N=40)	On/off	× Missing DNS		× Missing DNS		1	28
	Night light on/off	× Missing DNS		× Missing DNS		1	28
	Night light color	× Missing DNS		× Missing DNS		1	28
	Scheduling	× Missing DNS		× Missing DNS		1	28
	Factory reset	× Missing DNS		× Missing DNS		1	28

TABLE 6: This table is a continuation of Table 5.

Device	Functionality	Hostname		Hostname Pattern		Domain	
		# of Allowed Hostnames	Threshold	# of Allowed Patterns	Threshold	# of Allowed Domains	Threshold
iHome Switch (N=31)	On/off	× API change		× API change		× API change	
	Schedule	× API change		× API change		× API change	
	Scene	× API change		× API change		× API change	
	Voice control	× API change		× API change		× API change	
Lifx Lights (N=50)	Scheduling	1	45	1	45	1	45
	Voice control	1	45	1	45	1	45
	Change color	1	45	1	45	1	45
	Scene	1	45	1	45	1	45
	Effects	1	45	1	45	1	45
Logitech Harmony (N=112)	Select activity	1	100	1	100	1	100
	Device on/off	1	100	1	100	1	100
	Help	1	100	1	100	1	100
	Set device	1	100	1	100	1	100
Lutron Bridge (N=31)	On/Off	× API change		× API changes		4	15
	Scheduling	× Interaction dependency		× Interaction dependency		4	15
	Scene	× Interaction dependency		× Interaction dependency		4	15
	Factory Reset	× API change		× API change		× Rare operation	
Nintendo Switch (N=62)	eShop (first time)	× Rare Operation		× Rare Operation		× Rare Operation	
	eShop (not first time)	54	2	54	2	2	31
	News	9	18	9	18	1	55
	Switch Online	× API Change		× API Change		2	31
	Link account	× Rare Operation		× Rare Operation		62	3
	Download app/game	31	4	31	4	2	31
	Friends	22	6	9	18	2	31
	Factory reset	22	6	22	6	2	31
	Initial setup	13	12	9	18	1	55
Philips Hue (N=295)	On/Off	1	236	1	236	1	236
	Brightness	1	236	1	236	1	236
	Voice control	1	236	1	236	1	236
	Timer	1	236	1	236	1	236
	Routine	1	236	1	236	1	236
	Factory reset	6	59	6	59	3	59
Roku Streamer (N=52)	Stream	× CDNs		× CDNs		32	5
	Search	124	3	124	3	2	46
	Voice Control	826	1	826	1	16	10
	Remote Control	3	41	3	41	2	46
	Adjust volume	× API change		× API change		× Rare operation	
Sonos One (N=261)	Radio	× New services		× New services		× New services	
	Streaming	× CDNs		119	3	4	52
	Change volume	× Interaction dependencies		119	3	4	52
	Pause	× Interaction dependencies		119	3	4	52
	Voice control (streaming)	× Load balancing		× Load balancing		62	3
	Voice control (volume)	× Load balancing		521	1	62	3
	Factory Reset	411	2	223	2	3	78
Sony Console (N=62)	Buy Games	814	1	814	1	12	5
	Download Games	1	55	1	55	1	55
	Family Control	814	1	814	1	12	5
	Share Screenshots	1	55	1	55	1	55
	Connect to Spotify	814	1	814	1	47	2
TP-Link Switch (N=88)	On/Off	3	26	3	26	1	70
	Scheduling	3	26	3	26	1	70
	Timer	3	26	3	26	1	70
	Away mode	3	26	3	26	1	70
	Factory Reset	3	26	3	26	1	70
Wyze Camera* (N=167)	Live streaming	77	2	68	2	4	100
	Event recording	77	2	68	2	4	100
	Motion Tagging	77	2	68	2	4	100
	Night vision	77	2	68	2	4	100
	2-way audio	77	2	68	2	4	100
	Sharing	77	2	68	2	4	100
	Rules	77	2	68	2	4	100
	On/Off	77	2	68	2	4	100
	Factory Reset	18	16	17	16	4	100
Xiaomi Vacuum (N=30)	Clean	× API Change		× API Change		1	27
	Level Control	× API Change		× API Change		1	27
	Schedule	× API Change		× API Change		1	27
	Automation	× API Change		× API Change		1	27
	Remote control	× API Change		× API Change		1	27
	Find the vacuum	× API Change		× API Change		1	27
	Initial setup	× API Change		× API Change		1	27

* The Wyze Camera only worked with additional manually allowed IP addresses. The IP addresses were the main contributors to the Wyze Camera's traffic and can be easily observed; no associated DNS lookup was observed.

Appendix D.
Regionalization Regressions

TABLE 7: Linear regressions modeling the impact on transferability of the geographic region in which devices were located, whether the source and target regions were the same (* represents an interaction term), the amount of sample data, and the product. As the baseline for categorical variables, we use the largest category: *A* for region and *Amazon Echo* for product.

(a) Hostname-based allowlists

Factor	β	SE	t	p
(Intercept)	0.828	0.004	194.915	**<.001**
Source Region: B	-0.025	0.001	-19.322	**<.001**
Source Region: C	-0.036	0.001	-26.541	**<.001**
Target Region: B	-0.041	0.002	-22.423	**<.001**
Target Region: C	-0.091	0.004	-24.806	**<.001**
Product: Belkin Switch	-0.363	0.002	-170.291	**<.001**
Product: Google Chromecast	0.086	0.002	51.714	**<.001**
Product: Google Home	0.089	0.001	65.493	**<.001**
Product: Philips Hue	0.074	0.002	43.744	**<.001**
Product: Sonos Speaker	-0.321	0.002	-194.669	**<.001**
log(Train Sample Size)	0.008	<.001	18.014	**<.001**
Source Region: B * Target Region: B	0.060	0.003	23.276	**<.001**
Source Region: B * Target Region: C	0.063	0.005	12.131	**<.001**
Source Region: C * Target Region: B	0.054	0.003	21.339	**<.001**
Source Region: C * Target Region: C	0.128	0.007	19.644	**<.001**

(b) Hostname-pattern-based allowlists

Factor	β	SE	t	p
(Intercept)	0.792	0.004	213.467	**<.001**
Source Region: B	-0.031	0.001	-25.069	**<.001**
Source Region: C	-0.032	0.001	-25.500	**<.001**
Target Region: B	-0.036	0.002	-21.423	**<.001**
Target Region: C	-0.100	0.003	-29.007	**<.001**
Product: Belkin Switch	-0.044	0.002	-22.248	**<.001**
Product: Google Chromecast	0.063	0.002	40.579	**<.001**
Product: Google Home	0.066	0.001	51.954	**<.001**
Product: Philips Hue	0.048	0.002	30.074	**<.001**
Product: Sonos Speaker	-0.187	0.002	-121.330	**<.001**
log(Train Sample Size)	0.008	0.000	18.412	**<.001**
Source Region: B * Target Region: B	0.074	0.002	30.675	**<.001**
Source Region: B * Target Region: C	0.055	0.005	11.369	**<.001**
Source Region: C * Target Region: B	0.064	0.002	27.024	**<.001**
Source Region: C * Target Region: C	0.121	0.006	19.944	**<.001**

(c) Domain-based allowlists

Factor	β	SE	t	p
(Intercept)	0.915	0.004	225.965	**<.001**
Source Region: C	-0.034	0.001	-30.047	**<.001**
Source Region: B	-0.013	0.001	-11.484	**<.001**
Target Region: C	-0.053	0.003	-17.047	**<.001**
Target Region: B	-0.021	0.002	-13.376	**<.001**
Product: Belkin Switch	-0.093	0.002	-54.276	**<.001**
Product: Google Chromecast	-0.002	0.001	-1.303	0.193
Product: Google Home	0.014	0.001	12.889	**<.001**
Product: Philips Hue	0.044	0.001	31.927	**<.001**
Product: Sonos Speaker	-0.097	0.001	-68.704	**<.001**
log(Train Sample Size)	0.004	<.001	11.531	**<.001**
Source Region: C * Target Region: C	0.092	0.006	16.720	**<.001**
Source Region: B * Target Region: C	0.036	0.004	8.231	**<.001**
Source Region: C * Target Region: B	0.044	0.002	20.333	**<.001**
Source Region: B * Target Region: B	0.022	0.002	10.072	**<.001**

Appendix E.
Additional Figures

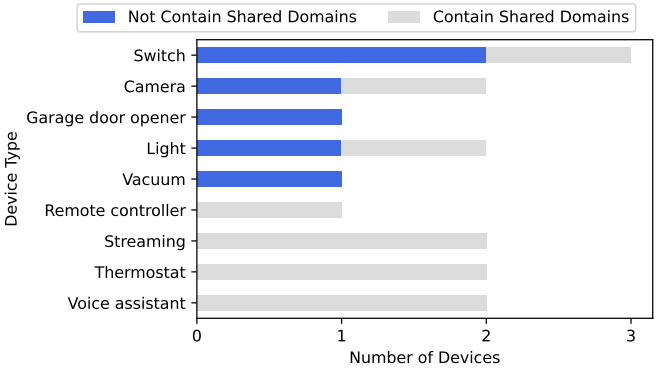


Figure 10: Types of devices whose traffic contains domains that are shared with other vendors. The observations are made in the lab. Devices whose functionality is impaired after enforcing the network policies are omitted.

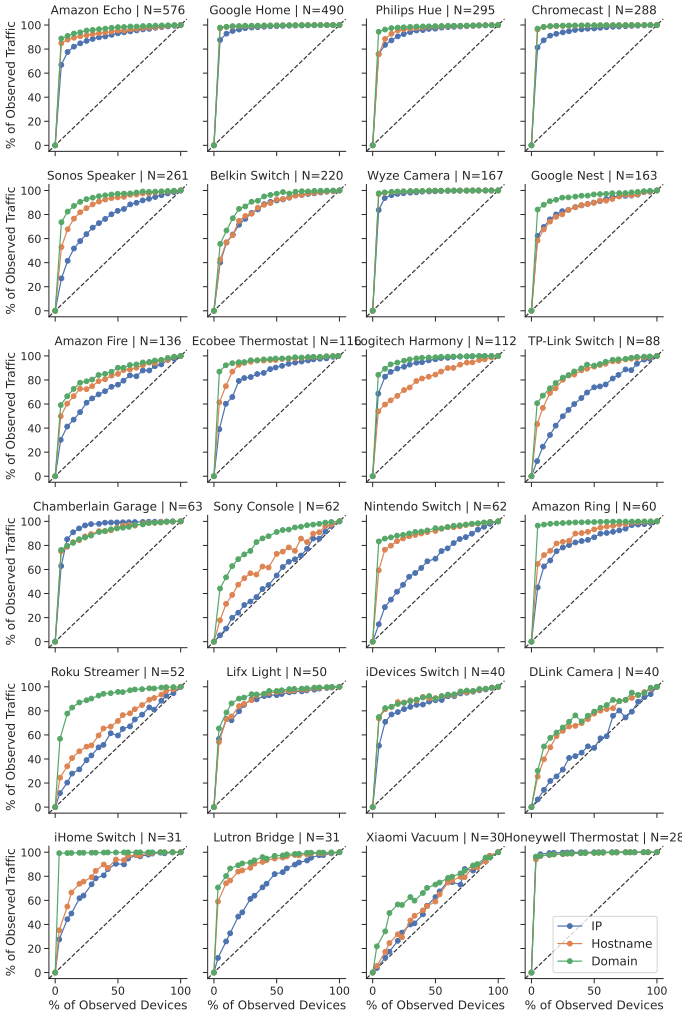


Figure 11: Flow coverage (averaged across 100 runs) for three ways of characterizing endpoints (domain, hostname, IP address) for the 24 most popular products. Coverage far below 100% indicates high variability across devices.

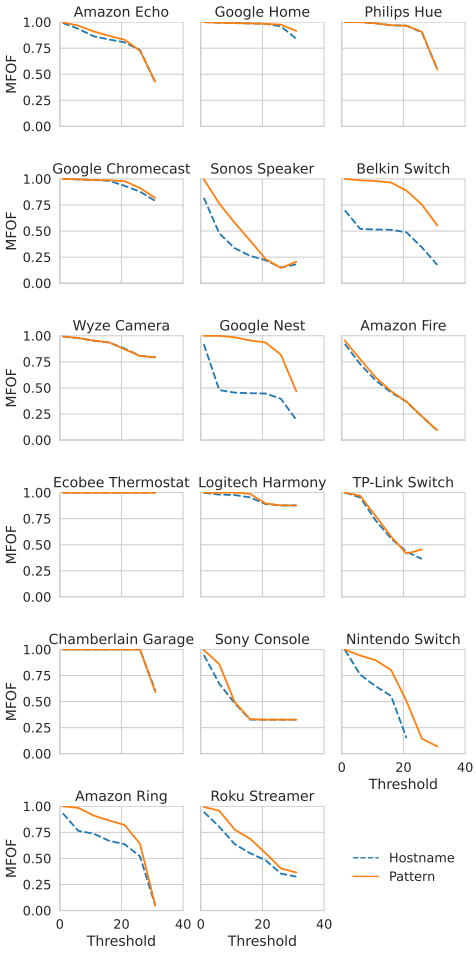


Figure 12: Although the MFOF generally decreases with increasing thresholds, many products experience plateaus where the thresholds increase, but the corresponding MFOF changes little. Lines end when the allowlist is blank above that threshold.

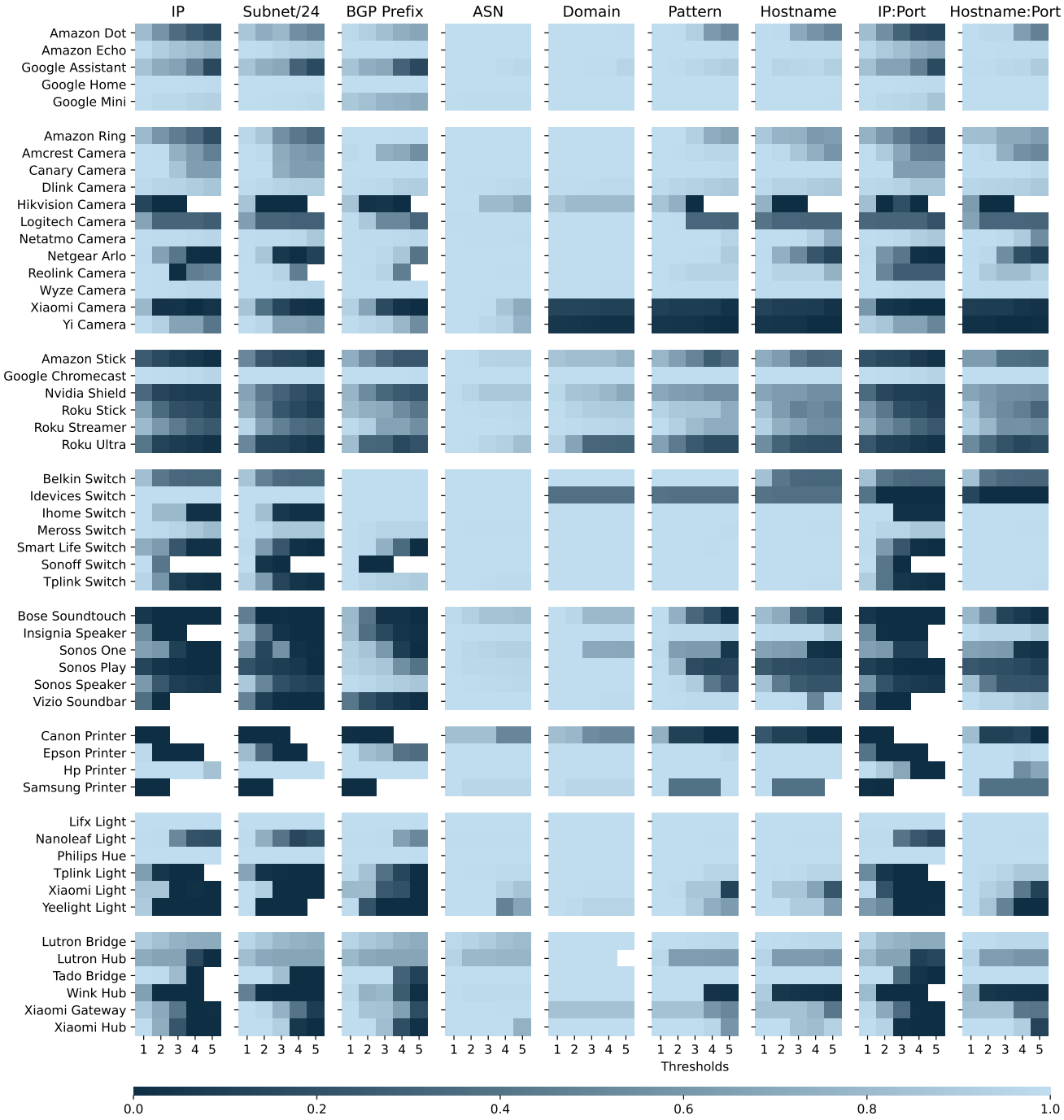


Figure 13: Transferability of device network behavior using different host representations (columns). Rows represent the source product whose traffic we assume is known or observed, and the color indicates the proportion of flows expected when applying such observations to devices of **the same product**, with a threshold of between 1 and 5s (sub-columns).

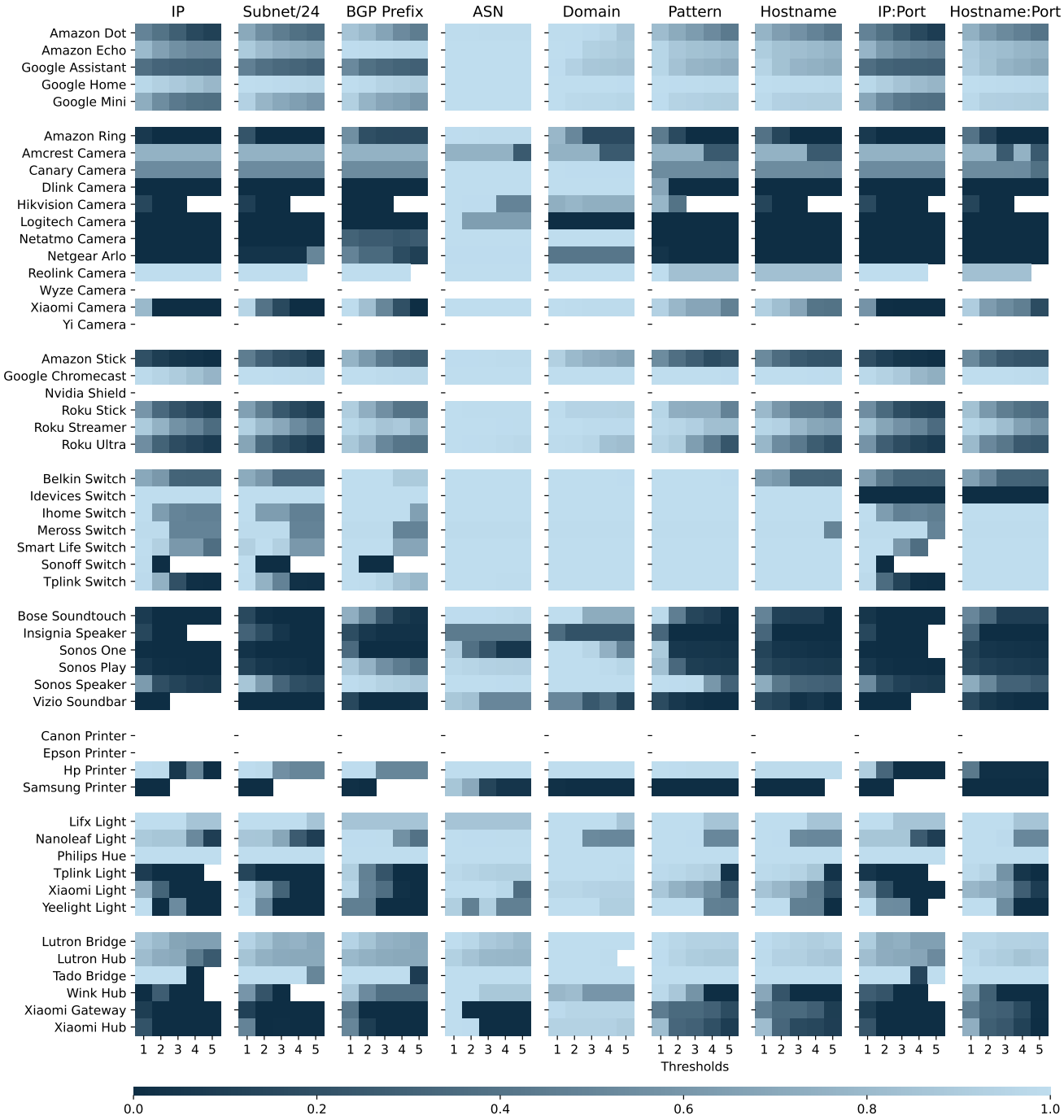


Figure 14: Transferability of device network behavior using different host representations (columns). Rows represent the source product whose traffic we assume is known or observed, and the color indicates the proportion of flows expected when applying such observations to devices of **the same vendor**, with a threshold of between 1 and 5s (sub-columns).

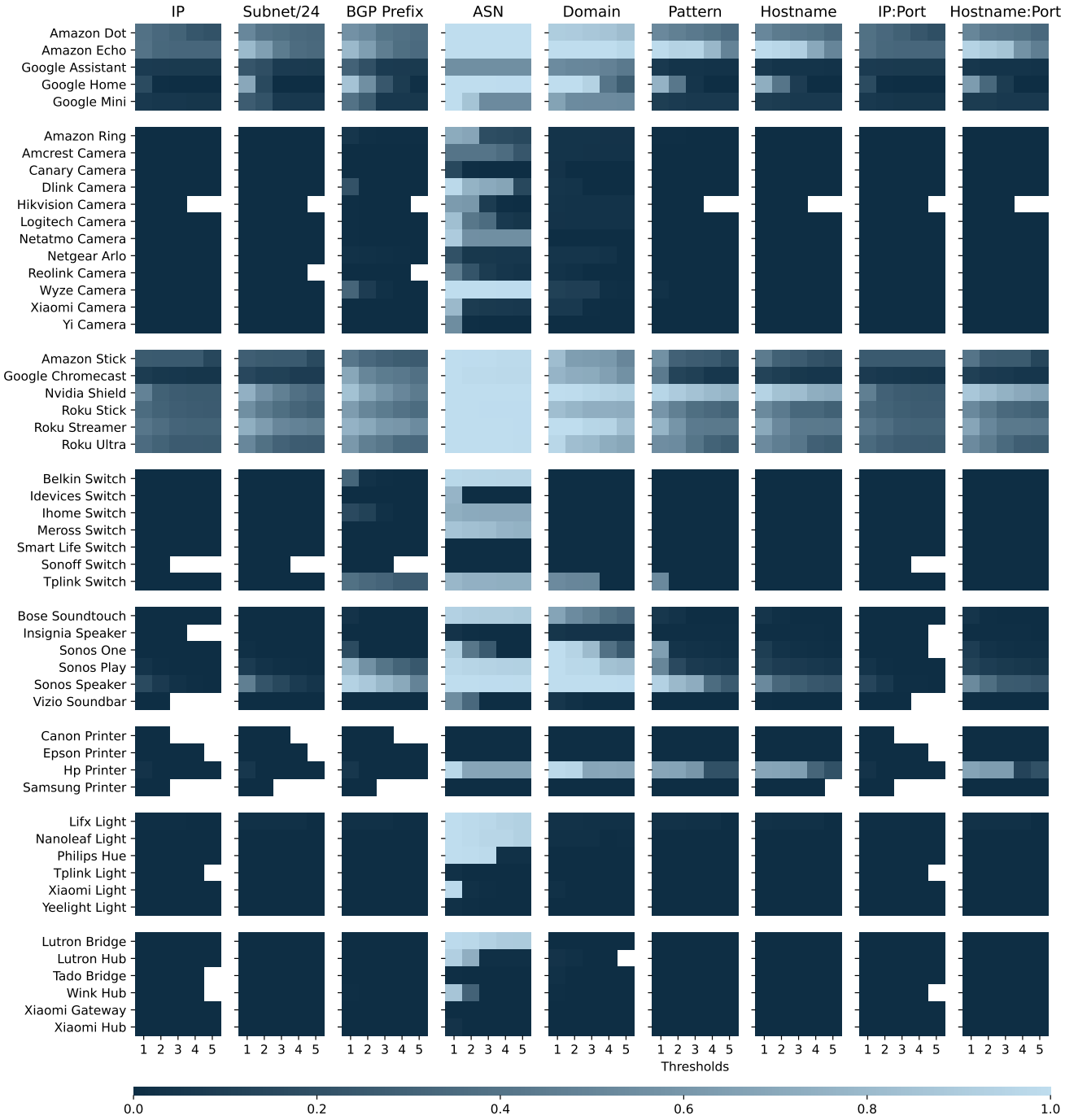


Figure 15: Transferability of device network behavior using different host representations (columns). Rows represent the source product whose traffic we assume is known or observed, and the color indicates the proportion of flows expected when applying such observations to devices of **the same type**, with a threshold of between 1 and 5s (sub-columns).